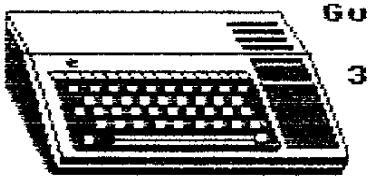
GUILFORD 99'ERS NEWSLETTER

SUPPORTING THE TEXAS INSTRUMENTS TI-99/4A COMPUTER



GUILFORD 99'ERS UG
3292 CANTERBURY DR
GREENSBORO NC
27408

TO:

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: June 4, 1991 Time: 7:30 PM. Place: Glenwood Recreation Center, 2010 S. Chapman Street.

Program for this meeting will be a program swap from the newly refurbished Users' Group library. Currently, there are well in excess of 450 programs consisting of several thousand individual files. Bring some blank disks!!

MINUTES

The May meeting of the Guilford 99er Users' Group was held on Tuesday the 7th, at the Glenwood Recreation Center on Chapman Street in Greensboro, N.C. There were six members present, eventually. Some of us had Little League ball games to attend earlier in the evening and, therefore, arrived later during the meeting. Summer evenings can be mighty hectic!

The Secretary/Treasurer's report was approved as written. As of 05/12/91 the club has $181.24 on deposit.

Mac Jones contributed a program to the club. This program presents a puzzle containing several (12?) pieces that are somehow supposed to be put together to form a solid block. Since the program uses a joystick (and Mac FORGOT to bring a joystick), we were unable to fully demonstrate the program. Just kidding, Mac. Thanks for the program.

Bob Camany presented the results of the re-cataloguing project. Sixteen disks have been created and catalogued by Bob using his PR-BASE screen and reports. Four reports were presented to the group: PROGRAM INDEX, DISK INDEX, PROGRAM ANALYSIS, and DISK CATEGORY. I'll let Bob explain the library further. I imagine he'll want to include an article later. How's that for a hint, Bob?

Next month's meeting will be a disk swap. Summer time's hectic and no one could be certain that they'd have time to prepare a presentation.

Respectfully Submitted,

Tony Kleen
Sec/Treasurer

LIBRARY NOTES

After a month-and-a-half, the UG library has been re-organized and catalogued. The completed library and indices will be presented to the UG at the June meeting. There are currently some 497 programs in the UG library which consist of several thousand individula programs. Some applications take up an entire disk and others are but a single program and there are many that fit somewhere in-between.

In the course of updating and re-cataloguing the library, a number of obsolete programs were deleted and replaced by more current versions. For example, F'WEB Vn 4.0 (in the library) was replaced by F'WEB Vn 4.31, etc. The result is that the library is more up-to-date as a whole.

The library index now consists of four indices. The first two, PROGRAM INDEX and PROGRAM ANALYSIS, are arranged alphabetically by program name. PROGRAM INDEX includes the program name, diskname, load information, environment, and a brief

description of the program. PROGRAM ANALYSIS is similar except that in place of the program description it lists the various files that comprise the application. This one is extremely valuable if you want to check out a disk and want to know which files make up a particular programming application.

DISK INDEX is a listing of the disks by name in ascending alphabetic order. It also includes load information, environment and a listing of the individual files that make up each programming application.

CATAGORY INDEX is an alphabetic listing by program application category (ie. UTILITY, GAMES, LANGUAGE, etc.). A description of the program is included along with load and other information.

Of some interest is the MUSIC category which now contains some sing-along music program that integrate the speech synthesizer and music. There are some other fine classical pieces amongst the lot as well.

Another interesting category is CARTRIDG(e) which contains programs that were previously released as TI and third-party cartridges. Some of them have been altered to support additional capabilities that weren't in the original releases. If there is enough interest, some unreleased cartridges or original unreleased version could be included at a later date.

The library is available to ALL UG members and I would encourage each and every member to make use of it. You would be surprised at some of the programs that are contained amongst the 60+ disks!!!

TECH NOTES

By Bob Carmany

As most of you know, I have been carrying on an ongoing correspondence with the "Wizards of Oz" --my Aussie 'mates'. The correspondence (and friendship) has been responsible for a lot of things over the years, the most recent was the Epromaer from Ron Kleinschafer. With it, I could make up my own cartridges with just a little time and effort. It worked great on 21 and 25V eprom chips. The proliferation of the 12.5V variety caused a bit of a problem, though. After a long bit of whining and carping, Ron sent along a modification to handle the 12.5V variety. I haven't analyzed what he did but it is a collection of resistors and a voltage regulator in an insulated package with some wires sticking out of it. All I had to do was follow his instructions and wire it into a SPDT switch and I could select the programming voltage. The big advantage is that now I can reproduce virtually and DSR ROM or PROM chip. This makes the TI just about immortal and it opens the doors for a reliable and cheap source of eprom chips as well. The 21V 2764's were becoming increasingly difficult to find!!

Ron also included a DSR killer for my Quest. Basically, it "flushes" the Quest DSR without trashing the programs. A simple push switch, a couple of wires and instructions are all that is needed. I anticipate installing it as soon as I can. It will solve those dilemmas where the RAMdisk DSR gets stuffed up and you don't want to have to initialize the whole thing and lose everything you have on the RAMdisk. I imagine a similar modification could be made to the Horizon and other RAMdisks. All you are doing to interrupting the power from the power trace to pin #28 of the DSR chip which effectively wipes it clean without distrubing anyhting else. A very neat trick, indeed!!!

On another note, Peter Smith the Hunter Valley UG President, recently underwent quintuple heart bypass surgery. He is now at home and doing well I understand. Our best wishes for a speedy and successful recovery!!

SIDE*PRINT

Those of us who use Multiplan know that one of the most frustrating aspects of the program comes when you print out your spreadsheet. Not only does Multiplan print slow (it's no faster with a buffer), but it prints in pages. There is no way to print out your spreadsheet in the manner the computer sees it (unless you cut and paste the pages together).

Jim Swedlow has just released Version 2.2 of his SIDE*PRINT program. This program prints a Multiplan spreadsheet sideways on a piece of paper. Now you can finally see what your spreadsheet really looks like in one big piece. First, you have to print your spreadsheet to disk using Multiplan's Print File option. The SIDE*PRINT docs explain how to configure the margins and page length so that the program works best.

Then you run the SIDE*PRINT program and follow the prompts, that's all there is to it. The program reads a page at a time and then shoots it to the printer, so it takes a little time to read the file, convert the characters, etc. But you have to realize that there's a lot of calculating going on here.

I would suggest reading the docs first, so you can alter the program to suit your printer. SIDE*PRINT works with parallel printers and the docs tell how to customize the program for Epson double density graphics, Gemini 10X, Epson FX80, Panasonic KX-P1090, and the GP 550TI. By customizing the program for your printer, the program may be able to print faster and take advantage of some of your printer's features.

It's really something to see your spreadsheet all laid out in one big piece!! Also included on the disk is a sample file

for printing sideways and a LOAD program which not only catalogs your disk, but lets you dump it to the printer and load any of the runable programs displayed. You can also look at DISPLAY files and peek at INTERNAL files.

The program is distributed under the FAIRWARE concept, and for your copy, send a SSSD disk and SASE return mailer to: Jim Swedlow 7301 Kirby Way Stanton, CA 90680

I showed a sideways printout to my accountant (who has an Atari) and he said that they can't do that with their version of Multiplan. Ha Ha!!

MINI-MEM NOTES

MINI-MEMORY BATTERY REPLACEMENT

Everybody makes a big deal about replacing the battery in the Mini- Memory cartridge. The Home Computer Lampoon is even going to give us a "tech tip" on how to do it since it is so technically difficult.

Well, first you need a Phillips screwdriver to remove the single screw in the middle of the cartridge. Then, use a flat-bladed jeweler's screwdriver to pry the catches on the side.

The battery is a #CR2430 3 volt Lithium battery which is soldered in place. This is NOT a weird battery and any competent camera store or watch dealer should be able to get them.

When you solder it back in, use a small-wattage iron, not one of those things that plumbers use to seal up pipes.

That's all there is to it, pretty big deal, eh?

DISKS AND SUCH

By Jim Ness

It's funny (at least to me), but there are lots of people who seem to know lots of stuff about their computers, and all those tiny chips, and how the bits and bytes are handled. And there seems to be next to nobody that knows anything about disk drives, and how they work. Sensing this huge gap in man's knowledge, I decided to figure out what makes them tick.

The great thing about disk drives is that they can find files buried randomly within a huge field of data, and they do it pretty fast. Actually, they can do it so fast because it's not at all random.

The mechanical concept is not all that complicated. A small motor spins at 300 rpm (at least in this country, with its 60 hz power supply), and there is a tiny stepping motor attached to a read/write head. A stepping motor is a common item in indexing applications, where you want a motor to move a precise distance and stop on a dime. The read/write head is just a smaller version of what you have on a cassette recorder.

The stepping motor "steps" the head from track to track on a diskette. The tracks are concentric circles, not a long spiral as you would have on an album.

All of this is ultimately controlled by the disk software provided with your computer. Usually this is located in ROM within the machine. In most machines, the ROM is only sophisticated enough to load in the official Disk Operating System (DOS) which is located on the disk in the drive when the machine is turned on. The DOS contains all the file handling software, copying software, etc, and because it is on disk, it can be easily modified and/or updated as time goes by.

Our friends at TI decided to put the whole thing in ROM, which has a few bad side effects. First, it makes it hard to update and improve the software, which is located in the Disk Controller Card. Second, although the machine is a 64k machine, just like all the others, TI has set aside so much memory for special purposes, that there is only 32k left to play with. They set aside 8k for cartridges, 4k for disk drive, 4k for RS232/PIO cards, 4k for the Operating System (can't complain about that one), and 8k for various interfaces (speech, sound, VDP). Ok those are all good applications to have, but if you don't use them, you still can't use that memory for other things.

Anyway, all of the controlling software for the TI99/4A is located in the ROM card, as I said. This software tells the step motor when to step to the next track, when to return to the beginning, etc.

There is no standard for how a computer keeps track of data. In the case of TI, there is a directory of existing files, and a map of where they are located, at the beginning of each disk. These files are not necessarily all in complete groups. If you delete a 12 sector file from a disk, there is a 12 sector gap recorded in the map. Then if you add a 20 sector file, the software will put the first 12 sectors in the gap, and put the rest in the first available spot. When you ask for a file that is broken up this way, you can hear the disk head scooting along to read each individual segment.

Because the disk drives themselves are pretty standard, there are a few things that don't change. For instance, there are 48 tracks per inch in most 5 1/4" systems (There is a new 96 TPI system around, not TI compatible). And most systems only use 35 or 40 of the available 48 tracks. There are either 9 or 18 sectors per track (single or double density). Each sector holds 256 bytes of data. And the standard design allows 250,000 bits per second to be written.

Now, you say, 250k! That is about 25k bytes per second, right? How come I can not load a 25k pgm in one second, then?

Oh, yes, I mentioned that most drives are capable of transferring data at about 250,000 bits per second. And you were asking how come your programs don't transfer that fast.

Two reasons. First, as I said, the transfer of data is actually controlled by the ROM software in the TI99/4A. And to be as good as it is, it had to be a little bit slow. Not REAL slow (anyone ever use a C64 disk drive?), but not as fast as it could be. The second reason also has to do with software, but it is a universal problem associated with single density storage.

The major difference between single and double density storage is the way in which the data is coded. In order for the software to keep track of where the read head is located on a particular track, there are clock or synch bits laid down with the data bits. In the old fashioned single density format, a synch bit was laid down ahead of each "0" bit, so there were never two "0" bits in a row. That kept the software from getting lost if there were a lot of "0" bits in series. Putting all those synch bits on the disk took up a tremendous amount of space that should be used for data.

So, some genius came up with a way of encoding the clock bits in with the data bits, so that no unnecessary space was lost. Voila, double density storage was born! And double density, as used with the Corcomp software, is said to increase transfer speed by at least 80%, mostly because the number of bits to transfer is cut way down.

So much for the exciting story of double density versus single density. How about double sided versus single sided? Well, obviously, it requires two read/write heads in the drive. Did you know that when reading a disk, the software reads, first, a track from side one, then the opposing track from side two, and continues back and forth? You didn't know that? There is a simple reason for doing it that way.

The disk head needs something to keep the disk stationary against it. In a single sided drive, there is a small arm holding the back side of the disk against the head. In a double sided drive, that arm would be in the way of the back side read/write head, so the solution was to use the two heads, directly across from one another, to hold the disk in place. In order to keep them across from one another, they alternate reading or writing as I said above. Very interesting, right? So if you wreck one side of a dbl sided disk, you can kiss the whole thing goodbye.

TEXT NOTES

I have recently come across a series of articles (published in an old defunct newsletter) that comprise an excellent tutorial on MULTIPLAN. I had planned to excerpt them and publish the lot as monthly installments in our UG newsletter. However, the series is rather lengthy and the space might be better used for something else. So, what I have decided to do is excerpt the lot and link them together on disk for inclusion into the UG library.

The series of files is probably the best tutorial for MULTIPLAN that I have seen short of the book itself. Besides being just about as complete as the manual, it is certainly a whole lote easier to read and filled with examples to help you over the rough spots. Look for the tutorial's appearance in the library in the next couple of weeks!

TONY'S CORNER

After a slight absence, Tony Kleen is back with his excellent on-going TI-BASE tutorials. This month's topic is VDP paging. I'm not sure that I understand everything that Tony writes about in the article but it sure is good stuff. Without further delay, H-E-R-E-'-S TONY!!!

```
========================================
TIBase Topic - VDPram */I Paging Example
by Tony Kleen, Guilford TI99er Users Grp
----------------------------------------

Article 07: Copyrighted April 1991
Reproduction, for gain, is prohibited.
========================================

  Two articles ago, I discussed command
file paging. Last article's topic was
install file paging. I stated in that
last article that I would give you an ex
ample of how I was using install paging.
Well, here goes...

  First off, let me list my PB:001/C
file. This is nothing more than an out-
line of all the command files, showing
'what' file calls 'whatever' files; and
vice-versa. An example - the first cou-
ple lines - PB:005 calls PB:045, PB:006,
and PB:007. In turn, PB:007 calls
PB:006. On my outline, I drop the 'PB:'
portion of the command file name. This
saves typing and gives me more room to
describe the purpose of the command
file. To the right of each command file
name, I give a brief description of the
command file's purpose. If the cf is re
peated, the description is simply RPT.

* ---------------------------------- *
* TIB910423.PB:001/C          V27    *
* ---------------------------------- *
* PB:publisher processing outline    *
* ---------------------------------- *
*
* --- --- --- --- --- Install Load BOOT-
* 005                 master control.
* --- --- --- --- --- ----------------
*     045             USE DSK?.PB:MENU
*     006             screen display
*     007             help
*         006                     RPT
*
*     --- --- --- --- --- Install Load B
*     060B                Name Change
*     --- --- --- --- --- --------------
*         060             Control proces
*             006                 RPT
*             007                 RPT
*                 006             RPT
*             053         Find tbl entry
*             063         'working' msg.
*             062         right adjust F
*             056         Control
*                 057     upd PB:MENU
*             036         parm file?
*                 042     Control
*                     043 select/go
*                     048 print/set/sele
*                     DO  PB:WORK
*             042                 RPT
*                 043             RPT
*                 048             RPT
*                 DO      PB:WORK RPT
*             --- --- --- --- --- Instal
*             005             BOOT
*             --- --- --- --- --- ------
* 005                 master control con
* --- --- --- --- --- ----------------
*     042                         RPT
*         043                     RPT
*         048                     RPT
*         exe                     RPT
*     --- --- --- --- --- Install Load-B
*     013B                PB:EDITOR
*     --- --- --- --- --- --------------
*         013             control
*             014         disp 10 lines
*             015         disp filler
*             007                 RPT
*                 006             RPT
*             020         select control
*                 016     row?
*                 017     form?
*                 018     lines?
*                 019     insert
*                 021     purge
*             --- --- --- --- --- Install Lo
*             005                 RPT
*             --- --- --- --- --- ---------
*     --- --- --- --- --- Install Load--
*     008B                BOOT3
*     --- --- --- --- --- --------------
*         034             file cleanup
*             006                 RPT
*             007                 RPT
*                 006             RPT
*             035         sure?
*                 042             RPT
*                     043         RPT
*                     048         RPT
*                     exe         RPT
*             --- --- --- --- --- Install Lo
*             005                 BOOT   RPT
*             --- --- --- --- --- ---------
*     009                 publish query
*         006                         RPT
*         007                         RPT
*             006                     RPT
*     --- --- --- --- --- Install Load--
*     022B                BOOT2
*     --- --- --- --- --- --------------
*         042                         RPT
*             043                     RPT
*         048                         RPT
*         exe                         RPT
*     024                 PGNUM pos
*     022                 1st PB:publish
*         023             publish pr
*             042                     RPT
*                 043                 RPT
*                 048                 RPT
*                 exe                 RPT
*             040                     RPT
*             006                     RPT
*             007                     RPT
*                 006                 RPT
*     --- --- --- --- --- Install Lo
*     028B                BOOT5
*     --- --- --- --- --- ----------
*         028             2nd PB:pub
*             037         4/80 PGNUM
*             --- --- --- --- --- Instal
*             029B            BOOT4
*             --- --- --- --- --- ------
*                 038         calc (
*                 029         publis
*                     032     check
*                     030     SLW pr
*                     046     build
*                         043     RPT
*                         044     se
*                         047     de
*                         050     RPT
*                     031         444 pr
*                         exe     PB:WOR
*                     033         48  pr
*                         032     RPT
*                 --- --- --- --- --- In
*                 005             BO
*                 --- --- --- --- --- --
*     --- --- --- --- --- Install Load--
*     051B                BOOT6 - Utilit
*     --- --- --- --- --- --------------
*         051             Utility Menu
*             006                 RPT
*             007                 RPT
*                 006             RPT
*             052         LM,CM,DLM
*                 053     Find in table
*                 056     Updte PB:MENU
*                     057 Updte LINES
*             055         BP#,EP#
*                 053             RPT
*                 056             RPT
*                     057         RPT
*         053                     RPT
*             058         FORM LINE
*                 059     FORM chrs
*                     057         RPT
*         --- --- --- --- --- Install Lo
*         005             BOOT   RPT
```

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

Notice that PB:005, PB:060B, and PB:013B are preceeded and succeeded by dashed lines. The dashed lines symbolize to me that these are 'renamed' command files and are 'paged' into VDPram instead of 'called' within VDPram by the DO directive. You'll see what I mean by 'renamed' when I present PB:BOOT/C and PB:BOOT2/C command files. The dashed lines also remind me that I still have five levels of called command files available. When PB:005 'pages' PB:060B for execution, I still have all five levels available. If you look further in my outline (PB:001), you'll notice that PB:005 'pages' PB:022B, which 'pages' PB:028B, which 'pages' PB:029B, which 'calls' PB:029, which 'calls' PB:046, which 'calls' PB:043. If you count the number of command files we've stepped through in that last sentence, you'll see that we're on our 7th level. There is no limitation on the number of command files that can be 'paged'. There is still the limitation of 5 levels that can be 'called' by the DO directive, however. You are no longer bound to the 5 level limitation of 'called' command files. If you need more than five levels, simply utilize the 'paging' technique.

Let's review that 7 level process. I'm going to take each command file and explain what's necessary to make the process work. As you TI-BASErs are aware, the first command file referenced is the SETUP/C file. Here is mine!

```
SET TALK   OFF
* ------------------------------------ *
* BOOTDISK910207.SETUP/C    V2         *
* ------------------------------------ *
* [ RAMdisk SETUP ]
CLEAR
WRITE 23,2 ;
"Initializing the parameters.          "
SET PRGDISK DSK1
SET DATDISK DSK4
SET LSPACE=512
SET CURSOR 2
CHANGE F84C 06A0
CHANGE F84E FFD8
CHANGE FFD8 0420
CHANGE FFDA 32E0
CHANGE FFDC 2912
CHANGE FFDE 045B
SET PRINTER=RS232.BA=9600.DA=8.PA=N;
.CR.LF.TW
*
CLEAR
INSTALL LOAD DSK3.PB:BOOT
* ------------------------------------ *
```

Notice that the last command issued is an INSTALL LOAD. This 'pages' my initial VDPram page into memory. When I'm ready to execute my PB:publisher process, I simply type BOOT on the .DOT prompt, and TI-Base executes my BOOT/C command file (located in VDPram). Now to show you how DSK3.PB:BOOT/I is created. Here's the command file that creates the initial 'page'.

```
* ------------------------------------ *
* TIB910405.PB:BOOT/C         V16      *
* ------------------------------------ *
*
* [INSTALL the cf's into VDPRAM]
CLEAR
INSTALL CLEAR
COPY PB:005/C BOOT/C GO
INSTALL ADD BOOT
INSTALL ADD PB:006
INSTALL ADD PB:007
INSTALL ADD PB:009
INSTALL ADD PB:040
INSTALL ADD PB:042
INSTALL ADD PB:043
INSTALL ADD PB:045
INSTALL ADD PB:048
INSTALL ADD PB:049
INSTALL SAVE DSK3.PB:BOOT
INSTALL CATALOG
* SNAP
* EJECT
RETURN
* ------------------------------------ *
```

Points of interest about PB:BOOT/C.

(*) CLEAR and INSTALL CLEAR. Clean the display screen and the VDPram area.

(*) COPY. Here is where I'm COPYing the controlling cf to a commonly named cf; in this case, the command name is BOOT.

(*) INSTALL ADD BOOT. Add the commonly named, controlling cf to the VDPram area. All the other cf's used by PB:005 (and others) are ADDed, also.

(*) INSTALL SAVE DSK3.PB:BOOT. Save the VDPram area to disk for later 'paging'.

(*) INSTALL CATALOG. Simply document how much VDPram has been used, and how much remains.

Notice that I copy PB:005/C to BOOT/C, in essence 'renaming' the command file. I'm using BOOT/C as my 'common' command file. Also, remember that anytime you ADD a command file to VDPram, you can access that command file by a macro, ie. by simply entering the command file's name (without the '/C' suffix), and the command file is executed.

Now then, when I enter BOOT at the .DOT prompt, TI-Base searches VDPram for the command file BOOT/C, finds it, and executes it. Since I copied PB:005/C to BOOT/C prior to creating the PB:BOOT/I page, I am in essence executing PB:005/C. Hope that's as clear as mud!

PB:005/C is listed next. As I have stated, this is the FIRST command file executed when I enter BOOT at the .DOT prompt.

```
* ------------------------------------ *
* TIB910423.PB:005/C          V50      *
* ------------------------------------ *
* [ TIBASE PUBLISHER ]
* [ initialize]
*
LOCAL A C 1
REPLACE A WITH " "
ENDCASE
ENDWHILE
*                          (initialization)
SET TALK OFF
SELECT 5
DO PB:045
GO 16
REPLACE LINES WITH " "
LOCAL Y N 5
LOCAL Z N 5
SET SPACES=0
SET RECNUM OFF
SET HEADING OFF
SET PAGE=0
*                        (master menu loop)
WHILE A<>"E"
  REPLACE Z WITH 1
  DO PB:006
  DOCASE
    CASE A="E"
      CLOSE ALL
```

```
    CLEAR
    RETURN
  CASE A="H"
    REPLACE Z WITH 1
    DO PB:007
  CASE A="1"
    INSTALL LOAD DSK3.PB:BOOT7
  CASE A="2"
    REPLACE Z WITH 0
    REPLACE Y WITH 1
    DO PB:042
    INSTALL LOAD DSK3.PB:BOOT1
  CASE A="3"
    DO PB:009
  CASE A="4"
    INSTALL LOAD DSK3.PB:BOOT3
  CASE A="P"
    INSTALL LOAD DSK3.PB:BOOT2
  CASE A="5"
    INSTALL LOAD DSK3.PB:BOOT6
 ENDCASE
ENDWHILE
* ---------------------------------- *
```

What's unique about this command file? (*) This is the 'controlling' command file. This guy has the master menu loop. You make your selection off the master menu, and this cf will 'page' the next set of cf's into VDPram.

(*) The first two directives LOCAL and REPLACE. The first time executing this cf, the local character A is allocated and replaced with a blank. All recurring entries into this cf, the LOCAL directive is ignored, but the variable A is REPLACEd with a blank.

(*) The next two directives, ENDCASE and ENDWHILE. The first time executing this cf, these two directives are ignored. Subsequent entries into PB:005 (after an INSTALL LOAD DSK3.PB:BOOT) will execute these directives, I explained this process last article. Briefly, on the second entry, when the ENDWHILE directive is processed, TIBASE rereads this PB:005/C cf, line by line, looking for the WHILE directive. When it finds the WHILE directive, TIBASE is happy and continues executing this cf at that directive, the WHILE. This WHILE construct contains the master menu, so that, to the user, we're back at the master menu selection screen.

(*) The initializing directives, SET TALK through SET PAGE. These directives are only executed ONCE, during the first initial execution of PB:005. For an explanation of why, read the previous paragraph!

(*) DOCASE construct. Look at the second directive where CASE A="3". We page in PB:BOOT2 by executing the directive: INSTALL LOAD DSK3.PB:BOOT2. At this point, we're instructing TIBASE to reload VDPram with the contents of PB:BOOT2/I. After TIBASE has reloaded VDPram, it will then try to continue executing the command file it currently is/was executing, namely, BOOT/C (the copy of PB:005/C). Remember from last article's discussion; to continue processing form one 'page' to the next 'page', one must have a commonly named command file.

Remember back to our discussion of PB:005, which 'pages' PB:BOOT2 into the VDPram area. Let's now look at how this PB:BOOT/I file is created, ie., let's look at PB:BOOT2/C.

```
* ---------------------------------- *
* TIB910407.PB:BOOT2/C        V15
* ---------------------------------- *
*
* [INSTALL the cf's into VDPRAM]
CLEAR
INSTALL CLEAR
COPY PB:022B/C BOOT/C GO
INSTALL ADD BOOT
INSTALL ADD PB:006
INSTALL ADD PB:007
INSTALL ADD PB:022
INSTALL ADD PB:023
INSTALL ADD PB:024
INSTALL ADD PB:040
INSTALL ADD PB:042
INSTALL ADD PB:043
INSTALL ADD PB:048
INSTALL SAVE DSK3.PB:BOOT2
INSTALL CATALOG
* SNAP
* EJECT
RETURN
* ---------------------------------- *
```

(*) COPY PB:022B/C BOOT/C, INSTALL ADD BOOT. Again, copy the ontrolling cf into our commonly named cf, and then add this to the VDPram area.

Now we need to somehow get back to the main menu process, PB:005/C, which is the controlling commonly named BOOT/C file in DSK3.PB:BOOT/I. You should notice that the last directive in PB:060B is an INSTALL LOAD DSK3.PB:BOOT. This directive 'pages' in the maste menu process.

```
* ---------------------------------- *
* TIB910423.PB:060B/C         V02    *
* ---------------------------------- *
* [Name change - BOOT7]
*
DO PB:060
*
INSTALL LOAD DSK3.PB:BOOT
* ---------------------------------- *
```

SO, let's see if I can reiterate the above discussion. I initially page in PB:BOOT/I via the SETUP/C cf execution. I enter 'BOOT' at the .DOT prompt when I want to execute the process that is in VDPram. This is because I have given BOOT/C as my COMMONly named cf. Whenever I want to transfer control to my next 'page', my current BOOT/C cf in VDPram memory simply INSTALL LOADs the next 'page' of cf's into VDPram memory. The controlling cf in this page has been previously COPYed, ADDed and SAVEd the commonly named cf, BOOT/C. TIBA goes about its business and continues executing the commonly named cf, BOOT. This can go on, adinfinitum.

END OF EXAMPLE: One should reread my prior article explaining how to do install file paging. That article explains all the requirements necessary to properly do */I paging. This article is simply an example, not a 'how to'.

If you need help on paging, give me a call! (919)924-6344.