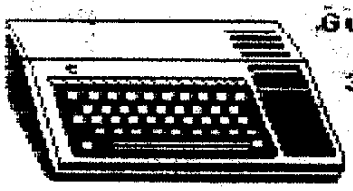
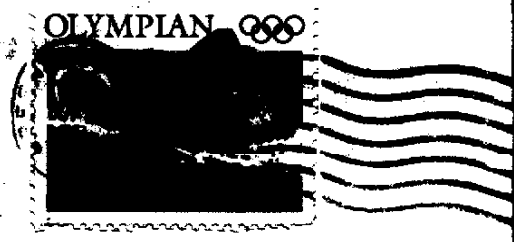
GUILFORD 99'ERS NEWSLETTER

SUPPORTING THE TEXAS INSTRUMENTS TI-99/4A COMPUTER



GUILFORD 99'ERS UG
3202 CANTERBURY DR
GREENSBORO NC
27408



TO:

VOLUME 7 NUMBER 11 NOVEMBER 1990

Bob Carmany, Pres. (855-1538) Emmett Hughes, Vice Pres. (584-5108)
Mack Jones, Sec/Treas (288-4280) 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:LF Jones, 3202 Canterbury Dr., Greensboro, NC 27408. The Software Library is for dues paying members only. (George von Seth, Ed.: 292-2035)

OUR NEXT MEETING

DATE: Nov 13, 1990 Time: 7:30 PM. Place: Glenwood Recreation Center, 2010 S. Chapman Street.

Program for this meeting will be a demonstration of the Hunter Valley Eprommer. This device enables you to modify GROM and reproduce DSR's. You can also create your own customized modules with it. It makes the TI immortal!!

OUR MEETING IS NOV 13TH!!

MINUTES

The October 2nd meeting of the Guilford 99er Users' Group met at the Glenwood Recreation Center on Chapman St. in Greensboro, N.C. There were 5 members present.

The meeting was called to order at 7:40 P.M. by President Bob Carmany. The September minutes were read and accepted as read. The treasury report was given.

There was no old business to discuss.

New business: Bob told members that he has cleared it with the Center to hold our November meeting on the second Tuesday in November which will be the 13th due to the election. Before, we always had to wait for the voting machines to be removed from our meeting room before we could start the meeting. This will prevent this from happening this November. Please make a note of this so you will not get it mixed up.

The members decided to drop the Vice President office from the roster and instead just have a CEO which George von Seth has agreed to take for the 1991 term. Tony Kleen has agreed to handle my Secretary/ Treasury office, and Bob Carmany will handle the newsletter. Hopefully Bill Woodruff will handle the Library for another year.

Bob has offered to do the program next month which will be on his new E-Prommer from OZ.

The book auction netted the club $15.00 and a donation from Tony made it $25.00. Tony gave out of the kindness of his heart and we thank you for it Tony! This brings us back over the $100. amount for the treasury, $105.84 to be exact.

The program was on PagePro and was very well handled by Bob. He showed members some of the many TIPS files and how they could be converted to TI incidences and in turn to PagePro pictures. Very good demo Bob and thanks.

Respectfully Submitted,
L.F. "Mac" Jones, Sec/Treas.
Guilford 99er Users' Group

RAMBLING BYTES

by "Mac"

Hello sunshine, goodbye rain!! Even tho we needed the rain, I hate to see it all come at one time! There is always something a little foreboding for me when the skies become dark and dreary. It is the time for snuggling down in front of my buddy the 99/4A and letting it take away the gloom which it can do if you let it.

We didn't have a very large crowd at the meeting, but we did have a lot of fun. It was great to get back with the gang and enjoy the fellowship I always look forward to. Hospitals might be a cozy place for some, but not this dude! It does seem funny tho, being able to wear the 29/30 waist lines that I used to wear many years ago! I sure am glad Jo missed throwing some of them out. (Grin)

I am looking forward to the next month meet because I have wanted to know exactly how those E-Prommers worked. Bob had a little trouble with his right after Ron sent it to him, but after changing a few non named chips, it now works great. Bob says he doesn't know just what those chips were, but they sure wasn't the right ones! If I understand correctly, you can take an EA cart or Car Wars cart and remove the chip and replace it with a socket. Then you can use the E/Prommer and make you about any program that you have the source code for and plug it into the cart and 'presto' you have the plug in program you need. Wouldn't it be nice if someone came up with something like that for the Nintendo? (chuckle chuckle)! Even if I don't have one my grandson does and if I had the money that those games cost when I was his age, I would have thought I had died and gone to Heaven!!

I hope you all will read the newsletters that I bring each meet. They are full of goodies that will be to your advantage. After all, this is one reason for our exchanges, the other being the feeling of closeness with other 99ers across the country. John Willforth of the West Penn newsletter has had a series of articles on printers that is a good thing to have a knowledge about. The series has taught me quite a bit about print heads that I really didn't know and needed to. This is not to slight the other good writers of other newsletters who give us their knowledge about most any subject you could ask for on the TI.

Hope to see everyone at the next meeting. If you have been staying away because you were afraid you would have to serve in office, that's all been taken care of for 1991, so come on down and enjoy the fellowship. You're safe for another year! Until then, enjoy the good TImes.

CALL PEEK(PRES)

Just a quick reminder! Our Meeting this month is the second Tuesday of the month (Nov. 13th). We have run into our annual conflict with the elections so we had to reschedule. We will be back to normal next month.

A couple of months ago, Ron Kleinschafter sent me one of his eprommers. After some initial problems dure to a bad lot of chips, I finally got it up and running. It was just too good not to share with the rest of the UG so it has become our program for this month.

The obvious question is "why bother with an eprommer to begin with?". Well, there are really a couple of reasons to have one or have access to an eprommer. As we are all painfully aware, TI is no longer producing anything for the 99/4A and how much longer they will keep up the repair support is a good question. What happens if the DSR on your disk controller gets trashed or the same happens to your RS232 card? Or what happens if a GROM chip blows up? Right now, it isn't much of a problem --just ship the item off to TI and they will replace it for a price. It usually takes a few weeks to get your card or console back and the cost is more than it would be if you were able to fix it yourself. That's where the eprommer comes in!

All of the chips in the console and the cards for the PE-Box are "off the shelf". You can walk into an electronics store or open up a catalog and buy any of them. Any of them, that is, except the DSR chips and the console GROM chips. With an eprommer, you can reproduce some of these unique chips by copying a good chip, saving the file to disk, and burning a replacement for the defective chip. The neat thing is that copies of the DSR's for the RS232 and disk controller are already available. As it turns out, the TMS 2532 eprom is an exact replacement for the DSR's and high byte and low byte ROM chips. Just put one into the eprommer and burn a replacement.

F'WEB in a cartridge? Or how about ARCHIVER? With a little practice and ingenuity it isn't very difficult to do that as well. Best of all, if you get tired of that particular application you can erase the eprom and burn another one.

The Eprommer and the software than comes with it opens up a new realm of possibilities. If you want to see what you can do with it, come to this meeting!

HYPHENATOR

A TI-Writer Utility
By Herman Geschwind

One of the nice features of TI-Writer is the ability to type in word-wrap mode, which speeds up typing by allowing you to concentrate on text without having to worry about exceeding the right margin.

There is a draw-back, though, to word-wrap in that longer words which would exceed the right margin are scrolled to the next line in their entirety.

The disadvantage of this system becomes obvious when text is printed out using the FORMATTER when there is a tendency for

the right margin to have the "jaggies".

Using the right-margin-flush feature (.AD) of FORMATTER provides only a partial cure since now FORMATTER inserts blank spaces between words to fill up the line. The amount of white space inserted varies with the number of characters that need to be filled with the result that text can be rather blotchy in appearance.

The only sure way to improve the appearance is to re-edit word-wrap text and to hyphenate as much as possible where lines break.

Unfortunately, the EDITOR of TI-Writer is not quite up to that task:

At the most, the EDITOR can display 80 characters per line whereas the FORMATTER and most printers can handle Elite (up to 96 characters) or Compressed (up to 132 characters) per line. In such a case the EDITOR is of no help.

A further hindrance is that the EDITOR will display imbedded print commands which is helpful in creating text but a serious obstacle in fine tuning right margins. Typical examples are string commands to turn super- or subscript on or off or the "ampersand" or "at" commands of TI-Writer for underlining and double strike.

Quite often for ease in typing and editing users elect to fix the right margin at 40 characters to do away with horizontal scrolling. An attempt to judge the final appearance of text by resetting tabs to final form and using the "Reformat" command can be misleading since previously entered indentations are then ignored.

HYPHENATOR is an editing utility for TI-writer that succeeds in addressing all these problems:

HYPHENATOR can handle print widths , or right margins of up to 160 characters.

HYPHENATOR properly accounts for imbedded print commands, be they the TI-Writer "at" or "ampersand" type or special character mode (CTRL U) transliterate symbols.

HYPHENATOR makes it possible to change margin settings within a document for quoted text that needs to be indented further.

HYPHENATOR recognizes a double "ampersand" or "at" symbol as a character to be printed rather than as a non-printing control character.

HYPHENATOR allows for the FORMATTER idiosyncracy of inserting two blanks following a period even though only one space might have been keyed in.

The program is a stand-alone utility that can be loaded using the LOAD AND RUN option of the Editor/Assembler or Mini-Memory cartridge. After loading, HYPHENATOR will prompt for the name of the input file (the name of the document created with TI- Writer EDITOR) and a name for an output file which HYPHENATOR will create in TI-Writer format. The use of either a single disk drive or two disk drives is supported. The original text file will not be altered in any way.

Once the proper files are set up, HYPHENATOR will read in a paragraph of text which can be up to 5280 characters long (a full page, single-spaced).

According to the margin and indentation information for which HYPHENATOR has prompted, the first block of text will be displayed (five lines) with an end-of-line marker exactly on that character which would be the last character to be printed by FORMATTER, with all non-printing characters, extra spaces, etc. already accounted for.

If the end-of-line marker points to a space or the last character of a word, no further action is necessary except for pressing the <ENTER> key to bring up the next line.

If the EOL marker points to the middle of a word, a decision needs to be made whether hyphenation is possible. If yes the editing cursor (FCTN S) should be moved to the last character prior to the hyphen and a hyphen symbol keyed in. HYPHENATOR will supply the necessary prompts to complete the job.

If hyphenation should not be possible, moving the cursor to the first blank and pressing <ENTER> would complete the job.

Once all the lines of a particular block have been edited a screen message will prompt for writing the block out to the disk file.

For speed and convenience, HYPHENATOR has a number of imbedded defaults. Thus empty lines or lines with only format control characters are written to the output file without user intervention.

An "Oops" feature can be invoked at any time by pressing <CTRL 1> to go back to the beginning of the paragraph. This comes in handy if there should be any second thoughts about a line just completed.

<CTRL 3> and <CTRL 4> toggle the screen display color which make it possible to display many combinations of screen and text color.

<CTRL 2> invokes the margin/indentation set option to change these values at any time.

<CTRL 9> writes out the remainder of an input file without further editing to the output file. This comes in handy where only a portion of text needs that final touch.

Any time a line of text is displayed on the screen, minor editing is possible. Thus "recieve" can be changed to "receive". The limitation is that the new text must have the same length as the original text.

HYPHENATOR is written in Assembler and thus is very fast. A test with a 59 sector compressed print document could be "fine-tuned" in under twenty minutes.

The use of a pocket dictionary in conjunction with HYPHENATOR is strongly recommended. Due to the memory limitations of the 99/4A system, HYPHENATOR can only show WHERE to hyphenate. The "IF" and "HOW" is up to the user. This is where a pocket dictionary comes in handy.

All-in-all, HYPHENATOR is an excellent utility along the lines of Tom Kirk's AUTO SPELL CHECK to make a good product, such as TI-Writer, even better.

HYPHENATOR complete with four and one half pages of documentation on disk is available from the author:

Wayne L.Stith
715 Timken Drive
Richmond, VA 23229

for $10.00. Source code in addition to object code and documentation is available for $15.00.

Wayne is a good friend of mine and thus I was available to test a number of HYPHENATOR versions, starting with one in Extended Basic. It turned out that XB just did not offer the speed that is necessary to accomplish this editing task. Over many months, Wayne wrote several versions of an Assembler HYPHENATOR with the present release being the best blend of speed and user convenience.

MINI—MEMORY

The other Assembler

Another item in the 99/4A arsenal that has been overshadowed by the PEB memory expansion, Extended Basic and the Editor/Assembler is the Mini Memory module (PHM 3058).

This module adds a whole arsenal of features to a console/cassette system: (1) 4k of non-volatile memory, (2) an extension to Basic with just about all the features of Extended Basic except for multiple statements per line, and (3) all the tools needed to create Assembler programs or Assembler subroutines to be linked with the MM version of Basic.

While MM will support diskette input/output and let Basic access the additional 32K if installed, MM was primarily designed for those without the PEB and will work equally as well with cassette.

As with so many other things, if TI had not stopped half-way with some of the MM concepts, the 99/4A would still be around as the number 1 home computer. A case in point: The 4K non-volatile RAM is really a pleasure to work with. Programs can be saved to MM and brought back with the OLD command just like disk or cassette input or output.....only what a difference in speed! The added RAM is battery refreshed so that nothing is lost when the module is removed from the system. The though of having 90K as nonvolatile RAM disk in the PEB is a tantalizing thought, The idea of loading MultiPlan into a RAM disk where it could page at solid state speed is truly intriguing. Even so, 4k is not bad for small program segments or small data files.

In some ways the extensions to Basic provided by MM are more powerful than what XB has. MM has the facility to write directly to screen memory (POKEV and PEEKV) which can make for lightning fast screen updates in game programs. Unlike XB, MM Basic will allow the linking of Assembler routines that have been compiled in compressed format. XB does not allow Assembler programs to be compressed which can make for a lengthy loading process.

The Assembler that comes with MM is a comprehensive subset of the TMS 9900 Assembler. The unusual thing about this Assembler is that compilation takes place as soon as a line of Assembler program has been entered. While this cuts out the separate compilation run that is necessary with the "big" Assembler, the trade off is that all programs can only be saved in object form, there is no way to save the source input.

The worst part about MM is that the manual that comes with the module was written by "techies" and is addressed to "techies". There is not even a reference manual to the Assembler language but the user is referred to consult the "big" Assembler manual, sold separately naturally and of very little help to the beginner.

To get anything out of the Assembler part of MM, Compute!'s Beginner's Guide to Assembly Language on the TI-99/4A by Peter M. L. Lottrup (Compute Publications, PO Box 5406, Greensboro, NC 27403) is the book that TI should have packed with MM. This book is an excellent guide to the Line by Line Assembler and in 259 pages features an excellent tutorial with many programming examples. Unlike the official TI manual the author makes no assumptions about previous Assembler experience and provides an excellent step by step guide into this difficult but also very rewarding world.

= by Tony Kleen; Guilford Users Group =


```
ACTIVITY    DATE(09/27/90) CNT( 00427 )
--------    CREATED(01/26/90) CHG(09/15/90)

  FIELD  DESCRIPTOR  TYPE  WIDTH  DEC
  -----  ----------  ----  -----  ---
      1  KEY           C    013
      2  D             C    001
      3  MBR           C    003
      4  TRANSDATE     D    008
      5  TRANSAMNT     N    008    02
      6  TRANSDESC     C    036
      7  BLANK         C    011
  -----  ----------  ----  -----  ---


DEFAULTS    DATE(09/27/90) CNT( 00001 )
--------    CREATED(02/09/90) CHG(06/02/90)

  FIELD  DESCRIPTOR  TYPE  WIDTH  DEC
  -----  ----------  ----  -----  ---
      1  OWNER         C    001
      2  MEMBER        C    003
      3  YEAR          C    003
      4  MONTH         C    003
      5  DAY           C    003
      6  DATE          D    008
      7  KEY1          C    013
      8  KEY2          C    007
      9  CB0           C    078
     10  BLANK         C    100
  -----  ----------  ----  -----  ---


DB_LIST     DATE(09/27/90) CNT( 00008 )
--------    CREATED(09/27/90) CHG(09/27/90)

  FIELD  DESCRIPTOR  TYPE  WIDTH  DEC
  -----  ----------  ----  -----  ---
      1  SEQ           C    002
      2  DB_NAME       C    010
  -----  ----------  ----  -----  ---
```

What I'm presenting this session is a little routine that reports any or all database structures for a TIBase data disk, in any sequence you desire. The purpose of this process was to present a listing of any one of my DATDISKs, in the format I wanted, and with as little intervention as possible. During development, I'm creating/changing right and left. Whenever I want to re-document, I just need to re-run a command file. I don't have to edit a thing to get a new report of all my current database.

To get from nowhere, to where I wanted; I figured the tasks before me were to create command files to ()list the directory, ()review this directory list for databases, and ()for each database in this list, list the attributes. As I got going on this process, I then challenged myself to compress the code to fit into TIBase's INSTALL area (which I refer to as VDPram). This was quite a challenge as my original code was over 5K bytes. What I ended up with fit the VDPram with 41 bytes to spare. I'll mention some of my techniques as I explain each command file.

Now then, one could readily document the databases using the command set given on TIBase; using DISPLAY STRUCTURE, and SNAPshot'ing to the printer. My problem is that I want my report to look different from what is provided. An example of what I wanted, and also the results of my process, are to the immediate left of this page. All the information, except the current date, is provided by TIBase's DISPLAY STRUCTURE command, but you'll notice a somewhat different presentation.

A little background: I have a RAMdisk configured with DSK1 being my PRGDISK, and three different DATDISKs assigned on DSK4. One for development, one for accounting, and the other for special use. The Ramdisk allows me to 'toggle' directly via RDS (Ram Doer Sys), or I can toggle while in TIBase by using the CATALOG DSK.xxxx, where xxxx is my diskname.

Let me explain how I use this process. First, one must INSTALL LOAD BAK051 to load the VDPram with the command files. Then I select one of the three DATDISKs using the CATALOG 'toggle' technique explained above. The third step is to enter the macro command BAK051 (or DO BAK051). From here on, it's all menu driven; answer the questions asked, and follow the instructions requested.

The best way to explain the processing is to review the command files. The next page begins the listings. I have a documentation technique I use to keep track of my command files. As you look at the first three lines of each command file, you'll notice each has a TIBxxxxxx.BAKyyy/C Vn. TIB is the diskname. xxxxxx is the date of the last revisions. 'yyy' is the program number. 'n' is the version number for the revision date. I will usually print each version I've tested, just to keep track of what has worked, and what doesn't. I find myself going in circles, otherwise!

```
* ------------------------------ *
* TIB900926.GAK051/C             *
* ----------------------V3------ *
*
* GAK051:-) GAK056
*       :
*       :-) GAK057:-) GAK060 -) GAK056
*       :         :
*       :         :-) GAK062 -) GAK052
*       :         :
*       :         :-) GAK061
*       :
*       :-) GAK059:-) L400
*                 :
*                 :-) GAK060 -) GAK056
*                 :
*                 :-) GAK063 -) GAK054
*                 :
*                 :-) GAK061
* [initialize]
SET TALK OFF
SET RECNUM OFF
SET HEADING OFF
 T PAGE=0
SET CRLF OFF
SET DATDISK=DSK1.
LOCAL C C 1
LOCAL A C 1
* [user menu]
WHILE C()"E"
* [display the screen]
  REPLACE C WITH " "
  DO GAK056
  REPLACE C WITH A
*
* [execute the choice]
  IF C="E"
    SET PRINTER=PIO.CR.LF
    SET CRLF ON
    SET DATDISK=DSK4.
    RETURN
  ENDIF
  DO GAK056
  IF (C="S").AND.(A="Y")
    DO GAK057
    REPLACE C WITH "U"
    DO GAK056
  ENDIF
  IF (C="U").AND.(A="Y")
    DO GAK059
  ENDIF
ENDWHILE
```

Let's look at GAK051/C! The first thing you'll notice is a bunch of GAKs, arrows, and 'overbars'. From left to right, top to bottom, these are the calling and called command files. GAK051 calls (actually, the DO directive) GAK056, GAK057, and GAK059. The command file GAK056 has an 'overbar' above it. This signifies that this cf is also called by some other cf. If there are any changes to this multiply called cf, one had better be certain that it works for all calling cf's. I've found this flowchart of the calling/called cf's to be quite valuable, especially during development.

Another thing you might notice. I use plenty of 'comment' lines. When we INSTALL cf's into the VDPram area, TIBase ignores loading comments. It also ignores leading and trailing blanks. Be advised that imbedded blanks are loaded into VDPram, though. You'll want to 'squish' the words / variables / whatever as close together as possible, in order to save more bytes!

GAK051's purpose is to initialize, present the master menu, and execute the user's selection. Initialization consists SETting the status, and defining some variables. You'll notice that all my LOCAL variables are one character in length. This saves bytes, but makes it harder to know what the varible represents.

WHILE the user does not select the (E)xit selection, the master menu will be displayed, and the user's selection executed. The master menu is displayed by REPLACEing the variable C with a specific value, and then DOing GAK056; the cf that does the screen displaying. The only other purpose for GAK056 is to receive the user's response. Let's now look at this re-entrant cf. One thing you might question are the two COLOR directives; one at the front, one at the end. At first design, I didn't care to have the menu scrolled onto the screen. Setting the foreground and background colors the same made the LISTing function 'invisible'. When the foreground color was reset to white, then it looked like the whole screen was printed at the same time. Now, I have more of a hang-up with the screen turning blue for a few seconds. It appears that the computer 'locks' up, or something like that. I'd rather see the lines scroll than the system turn blue for a few moments..

I save 'moocho' code by LISTing 40 column by 21 line'd disk files. If we filled every character on the screen, and were using the WRITE directive; it'd take (40*21)+(8*21)=1008 bytes. The 8 is for 'DISPLAY ' on 21 lines. Wouldn't take long to fill VDPram's 2.5K space!

To use the LISTing feature, just remember to SET the PRINTER=DISPLAY.

One last comment about GAK056. I use the DOCASE construct. If you are conserving bytes, use the DOCASE construct for more than three comparisons, otherwise, the IF construct is more efficient on space.

Lines 1 through 21 are used for LISTing; line 22 is left blank; and line 23 is used for the user's response.

Now, it's back to GAK051; the section where we [execute the choice]. The 1st response tested for is (E)xit. If the request is (E)xit, then several of the statuses are reSET, and control is passed to the .DOT prompt. If the response is otherwise, then we're back to DOing GAK056. Our re-entrant variable 'C' has been previously REPLACEd with what the user's response was to the master menu. Soooo, this time through GAK056, we display the screen for the user's previous selection!

```
* ------------------------------------ *
* TIBS@0926.GAK056/C                   *
* ----------------------------V1------ *
* COLOR LIGHT-BLUE LIGHT-BLUE
CLEAR
SET PRINTER=DISPLAY
*
DOCASE
  CASE C=" "
    LIST GAK055:1/H GO
  CASE C="S"
    LIST GAK055:2/H GO
  CASE C="H"
    LIST GAK056:1/H GO
  CASE C="@"
    LIST GAK060:1/H GO
  CASE C="U"
    LIST GAK059:1/H GO
ENDCASE
*
* COLOR WHITE LIGHT-BLUE
*
WRITE 23,5 "_ (entry to continue.)"
*
READCHAR 23,5 A
* ------------------------------------ *
```

Let's back up just a little bit. The first time through GAK056, our re-entrant variable 'C' was equal to a space. You'll notice that we LIST a file called GAK055:1/H. This guy is listed on this page, in the lower left. As you can readily see, it is the master menu.

Now then, let's suppose you enter (H)elp as a response to the master menu selection. A value of 'H' for GAK056's variable 'C' will list a file called GAK055:2/H. This is directly below in the middle.

Entering an (S) to the master menu selection will list the file in the lower right. This is actually a 'help' screen display for the next process in our progression through GAK051; that is, listing the directory. Take a moment to read the lower right display. After this file is listed to the screen you are given a chance to exit (by answering (N)o to the proceed?). This gives you the 'are you really sure?', kinda' like when you request to delete a file. The main reason for have an answer given is that our re-entrant GAK056 expects us to give a selection; that last READCHAR directive. Sooo, I had to have some kind of question/selection. This can be a drawback of using re-entrant cf's, ie., conforming to prior standards or requirements!

Now, let's say you've entered (S) to the master menu, and (Y)es; GAK051/C now calls GAK057/C to do the following: ()delete the prior DB_LIST records, () list the directory, () convert he directory listing to a database for the next step, and () create DB_LIST records using the just converted 'directory list' database.

```
----------------------------GAK055:1/H-
      Database Structure Reporting
      ============================

      (S)ystem generated list/report

      (U)ser edited list/report

      (H)elp

      (E)xit
```

```
----------------------------GAK056:1/H-
(S)ystem generated list/report.
===============================
  The process will display the disk di-
rectory and generate a list of all data
bases contained on the data disk.  If a
list previously exists, it will be de-
leted prior to the disk being cataloged.
You will be given the opportunity to
edit the list.  The database report is
then generated from this list.

(U)ser generated list/report.
=============================
  The option allows you to edit your own
list of databases.  Upon completion of
the edit, the report is then generated.

(E)xit.
=======
  Returns to the calling command file or
the .DOT prompt.
```

```
----------------------------GAK055:2/H-
 This next step is a repetitive process.
You will need to answer Yes to the MORE
DIRECTORY SNAPshots? question until all
screen snapshots are taken.  TIBase lists
the directory to the screen and this pro
cess prints a SNAPshot of the screen to
disk.  It may take several screen dis-
plays and SNAPshots to list the entire
directory.

 To take a SNAPshot of the screen dis-
play; enter (F9), ie. function-9.



          Do you want to proceed?
     (Y)es
     (N)o
```

```
* ------------------------------------ *
* TIB900927.GAK057/C                   *
* ----------------------------V1------ *
*
* [initialize]
SET PRINTER=DSK1.L40A
WHILE A="Y"
  CATALOG DSK4.
  SNAP
  WRITE 23,1 "More SNAPshots(Y/N)?"
  READCHAR 23,31 A
ENDWHILE
*
* [delete DB_LIST records]
SELECT 2
SET TALK ON
USE DB_LIST
DELETE RECORD ;FOR 1=1
SORT ON DB_NAME
SET TALK OFF
*
* [convert to a database]
DO GAK060
*
DISPLAY "-->Build DB_LIST"
LOCAL B C 10
LOCAL N N 3
LOCAL D C 3
LOCAL E C 3
DO GAK062
*
* [delete the temporary files]
DO GAK061
* ------------------------------------ *


* ------------------------------------ *
* TIB900926.GAK060/C                   *
* ----------------------------V1------ *
*
* [convert to a database]
REPLACE C WITH "0"
DO GAK056
SELECT 1
CONVERT L40A L40B GO
USE L40B
*
SET TALK ON
RECOVER
TALK OFF
QUIP
```

The first activity GAK057/C pursues is creating the directory lising. CATALOGing displays to the screen. SNAPshot'ing prints to a file called L40A. Then the user is queried as to whether there are more fils to be catalogued. NOTE: CATALOG is itself a WHILE loop. While you do not enter a (F9), it lists the next 20 files of your directory. If you enter (F9), it drops out of he CATALOG function and back into my cf. What the user has to do is enter (F9) for each 20 lines CATALOG displays and therefore allow my GAK057/C to snapshot each 20 line. (I'm researching a better way, ..later..).

Once we've got our directory list, the next section will delete the DB_LIST records. I set the talk on/off to let the user know what's happening. After deleting, it's into the conversion and GAK060/C. The user will be creating a new database, so the 1st step is to inform him/her of that. Setting C=0 and DOing GAK056/C will display the GAK060:1/H screen (page 4). We enter a field-name L40 of 40 chracters, and enter (F8) to proceed with he CONVERT directive. After conversion of L40A to L40B, we USE L40B and RECOVER the index.

```
-----------------------------GAK060:1/H-

  Your next entry will be as follows:



   field  type  width
     ---    -    ---          ------------
     L40    C     40    then (Function 8)
     ---    -    ---          ------------
```

```
* ------------------------------------ *
* TIB900925.GAK052/C                   *
* ----------------------------V1------ *
WHILE (E()"255") .AND. (.NOT.(EOF))
  MOVE
  REPLACE E WITH SUBSTR(L40,30,3)
ENDWHILE
* ------------------------------------ *

* ------------------------------------ *
* TIB900925.GAK061/C                   *
* ----------------------------V1------ *
*
* [delete the temporary files]
SELECT 1
DELETE DATABASE
DELETE FILE L40A
CLOSE ALL
```

```
* ------------------------------------ *
* TIB900925.GAK062/C                   *
* ----------------------------V2------ *
*
* [create the DB_LIST database]
WHILE .NOT. (EOF)
  REPLACE D WITH SUBSTR(L40,12,3)
  REPLACE E WITH SUBSTR(L40,30,3)
  IF (D="I/F") .AND. (E="255")
    REPLACE D WITH L40
    REPLACE N WITH LEN(B)
    REPLACE N WITH N - 1
    REPLACE D WITH SUBSTR(L40,N,2)
    IF D="/S"
      REPLACE N WITH N - 1
      REPLACE B WITH SUBSTR(L40,1,N)
      SELECT 2
      FIND B
      IF (EOF)
        APPEND BLANK
        REPLACE DB_NAME WITH B
      ENDIF
      SELECT 1
    ENDIF
  ENDIF
  REPLACE E WITH " "
  DO GAK052
ENDWHILE
```

```
* ------------------------------------ *
* TIB900927.GAK059/C                   *
* ----------------------------V1------ *
LOCAL P N 4
LOCAL F C 40
*
* [create the cf for db structures]
USE DB_LIST
SET PRINTER=DSK1.L40B/C
WHILE .NOT. (EOF)
  REPLACE F WITH "USE DSK4."|DB_NAME
  PRINT F
  PRINT "DISPLAY STRUCTURE"
  PRINT "SNAP"
  PRINT "CLOSE"
  MOVE
ENDWHILE
CLOSE
*
* [execute the cf just created]
SET PRINTER=DSK1.L40A
DO L40B
*
* [ convert to a database]
DO GAK060
*
DISPLAY "}L40 reporting{"
SET PRINTER=DSK1.L40
REPLACE P WITH 1
GO 1
*
* [generate the report]
DO GAK063
*
* [delete the temporary files]
DO GAK061
DELETE FILE L40B/C
* ------------------------------------ *
```

```
                        :L40B/C example:
USE DSK4.ACTIVITY
DISPLAY STRUCTURE
SNAP
CLOSE
```

```
* ------------------------------------ *
* TIB900924.GAK054/C                   *
* ----------------------------V1------ *
WHILE L40 () "      "
  PRINT L40
  MOVE
ENDWHILE
```

Back to GAK057/C where we now build the DB_LIST database by DOing GAK062/C. This is a repetitive process where we look at every line listed by the CATALOG cf. TIBASE databases are unique; they are Integer/Fixed of 255 bytes, and end in a '/S' suffix. If we find such a match, we add the DB_NAME to the DB_LIST database. Duplicates are ignored! GAK052/C is DOne as the last step of the WHILE. It speeds the search for the next database name!

```
* ------------------------------------ *
* TIB900925.GAK063/C                   *
* ----------------------------V3------ *
*
* [generate the report]
WHILE .NOT. (EOF)
*
* [line 1]
  MOVE 22
  REPLACE F WITH SUBSTR(L40,15,8);
)"    DATE("|.DATE.|") CNT( ";
|SUBSTR(L40,29,5)|" )"
  PRINT F
*
* [line 2]
  GO P
  REPLACE F WITH "-------- CREATED(";
|SUBSTR(L40,12,8)|") CHG(";
|SUBSTR(L40,25,8)|")"
  PRINT F
*
* [line 3,4,5]
  PRINT " "
----------------------GAK059:1/H------
```

The process to edit the DB_LIST does not yet exist. You would need to exit to the .DOT prompt, USE DB_LIST, and EDIT. The database is sorted on SEQ / DB_NAME. If the prior process created DB_LIST, then SEQ=' ' for all entries, and the DB_LIST database is in DB_NAME sequence, only.

If you wish to list the current DB_LIST as it exists, answer (Y)es.

(Y)es, to continue the report
(N)o , to exit to the main menu.

```
* GAK063/C continued.
  MOVE
  PRINT L40
  PRINT (2 )(5-)(2 )(10-)(2 )(4-)(2 );
(5-)(2 )(3-)
*
* [detail lines]
  MOVE 2
  DO GAK054
*
* [last lines]
  PRINT (2 )(5-)(2 )(10-)(2 )(4-)(2 );
(5-)(2 )(3-)
  PRINT " "
  PRINT " "
  PRINT " "
* FINALIZE
  REPLACE P WITH P + 24
  GO P
ENDWHILE
* ------------------------------------ *
```

```
* ------------------------------------ *
* TIB900925.GAK051I/C                  *
* ----------------------------V1------ *
*
* [INSTALL the cf's into VDPRAM]
CLEAR
INSTALL CLEAR
INSTALL ADD GAK051
INSTALL ADD GAK056
INSTALL ADD GAK057
INSTALL ADD GAK060
INSTALL ADD GAK062
INSTALL ADD GAK052
INSTALL ADD GAK061
INSTALL ADD GAK059
INSTALL ADD GAK063
INSTALL ADD GAK054
INSTALL SAVE GAK051
INSTALL CATALOG
SNAP
EJECT
RETURN
```

Again, back to GAK057/C where the last step is to delete the temporary files. This process is done by GAK061/C (listed on page 4).

We're done with GAK057/C and the 'system generated list/report'. Go back to GAK061/C on page 2; you'll notice I set C=U and DO GAK056 before leaving the IF statement. The user is automatically primed for the 'user edited list/report' process. The screen displayed, GAK059/M is on page 5. If one answers (Y)es, then GAK051 will DO the GAK059/C processing (edit/report).

GAK056/C is listed on page 5. It creates the cf which will DISPLAY the db STRUCTURES. An example of L40B/C is shown immediately following GAK059/C. After it is created (L40B/C), it is executed (DO L40B). Then we CONVERT the listing (produced by L40B) by DOing GAK060. We discussed GAK060 earlier. The PRINTER is set to our report file L40, and the report is generated by DOing GAK063 (listed across 2 columns on page 5). This cf reformats the DISPLAY STRUCTURE listing to my format.

After the report is generated, the temporary files are deleted (reusing GAK061/C), as well as L40B/C.

WE'RE DONE! L40 is the report. It can be printed directly. I usually print the report using my three-column reporting function.

NOTE: Don't forget to create the DB_LIST database. Its layout is on page one at the bottom left!

Creating this process was a lot of fun, and a MAJOR learning experience. I hope you can use these listings to come up with your own cf's and techniques. Thre are better/shorter cf's than what I have done, I'm sure. HAVE FUN!!!!