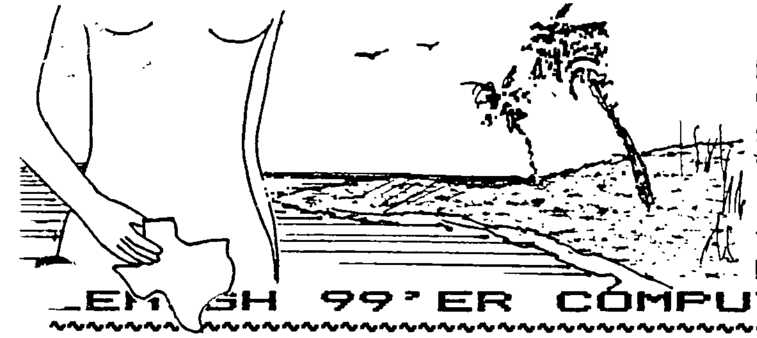


President Rowland Buckwalter 264-5790
Vice-pres Joe Birchak 759-4052
Secretary Alan Hewko 262-6347
Treasurer Mark DeNardo 791-1015
Ah! The Polar Bare Club —
vol. II, no. 12 December 1984
Editor Frederick Hawkins 432-5913

LEHIGH 99'ER COMPUTER GROUP

Next meeting: Monday, Jan 21 7:30 PM as usual. Community Room, First Nat'l Bank 7th and Hamilton, Allentown

XBASIC: BARE ESSENTIALS

Longtime XBASIC users remember how welcome the text-file-to-MERGE-format (TRANSL, et al) program was when it first appeared. Like the shot heard 'round the world, the conversion generated programs, press, and abounding joy among TI people worldwide at near light speed.

Well, I've a comeback to TRANSL that will stop hearts, make faint men weak, little old ladies giggle and strip out unwanted code, all from XBASIC's immediate mode.

Tha's right, while your program is standing there, you can gut the beast and hang the parts out to dry, and leave the rest for dog food and garbage collection routines (XMLLNK >0036). The last is a red herring, so on with it. We'll start out with the whole program, all three lines and then decompose it a little, ending up with an in-the-head arithmetic problem.

nekkid (power) an XBASIC utility program by Fred Hawkins (SAVE in MERGE format)

```
1 D=-31952 :: PRINT "CALL LO
AD(D,B,C)":"1st ";:: ACCEPT
A :: CALL PEEK(D+2,B,C):: C=
1+C-A*4 :: A=C<0 :: B=B+A ::
C=C-256*A :: STOP !@P-

2 D=-31950 :: PRINT "CALL LO
AD(D,B,C)":"last ";:: ACCEPT
A :: CALL PEEK(D-2,B,C) :: C
=C-1+A*4 :: A=C>256 :: B=B-A
:: C=C+256*A :: STOP !@P-

3 !@P-
```

In all actuality, we've three programs here: line 1, line 2, and your program. To use Nekkid, you'll MERGE and either RUN 1 or RUN 2. If you RUN 1, you'll be able to RUN 2 but you can't do it the other way around -- there won't be a line 1 to RUN.

While we're muddying the waters, let's simplify some more: Two things prevent RUN 1, RUN 2 from executing your pro-

(continued on page 5)

at the io port

The December crow's nest survey of the IO Port reveals a positive flotilla of small observations and tiny programs. The Elusive M. DeNardo, no Scarlet Pimpernel he, navigates a fourfold path through the Reefs of Disorder. A Great Journey 'tis, as M. has embarked on a series bridging 84 to 85.

XEDITOR Dave Hendricks takes to the seas in a small dinghy. His catch is small because his boat is slow, lanquishing in the backwaters of page ten.

Near at hand, we brave the cold and join a foolhardy polar bare swim -- skinny dipping at the IO Port! Then, stranger than crows at sea, your erstwhile editor shares an albatross to hang about the neck of the unwashed, and later inventories the ship's maps. Ever on the lookout for the visual he reveals five scrawlings, some that may perchance lead to treasure. But forsooth, they're really an Elephant's ribs, littering the shore. Likely only sufficient to 'Mark Twain' or deep six. Finally, further off, flying fish. Three, no, four! A curious sight, isn't it?

The more one writes, the less is right:
Jim Peterson, the mage of TIgercub Software, 156 Collingwood Ave, Caolumbus, OH 43213, is owed a correction. His disk of 102 XBASIC SUBs in MERGE format is priced at 19.95. That's twenty cents a program, you simply can't get anything for less. Jim's disk deserves your support (in order$) partly because the routines are useful, entertaining and enlightening. More importantly, his disk ("NUTS&BOLTS") is first systematic and fully matured collection of MERGEd SUBs available (and no pun is intended). The techniques demonstrated are positively essential to the maturation of XBASIC. That it is needed is proved by the simple fact that NUT&BOLTS is the first such compilation ever.
The date algorithm in October was incorrect; the mistake was in line 100. It should have read 100 D=(A=2)*(B>27-(INT(C/4)=C/4))+...... the =C/4 was missing.
TI WRITER's option #3 can not RUN PROGRAM FILES (AL) that assume ANY standard UTLTAB vectors in spite of what I implied in "Assembled Guilt". If a file does execute correctly, it is because all of the utilities are in the program itself.
In November's WINDOWING, the third case diagram was mislabeled: switch RSPILL and LLEN. The code was OK.

BRINGING IN THE NEW CHEER

As TI users head into our second year adrift, it becomes more important to band together. We need to $upport some companies. We also need to prod some others. We at the Lehigh 99'ers and the IO Port could just hand out our Bums-of-the-Year awards, but we're short of unpillarized villains at the moment.

So, as a Public Service and confident that appropriate targets will turn up, we're democratizing the Dead Turkey Awards! Every user can use one or two. Been burned lately? Read a lousy manual or bought a bum steer? Got a disk that won't format, a drive that don't or, heaven protect me, another newsletter that runs on and on?

Well, fight back! Send 'em all their very own personalized DTA! Keep a couple copies on hand -- winners can turn up at any moment.

Further down, you'll find an alternative malediction and an blank turkey for the colorists among the nominating committees. Remember to tell your recipients why they're so singled out. A typical example dating from May '83 (no reply yet):

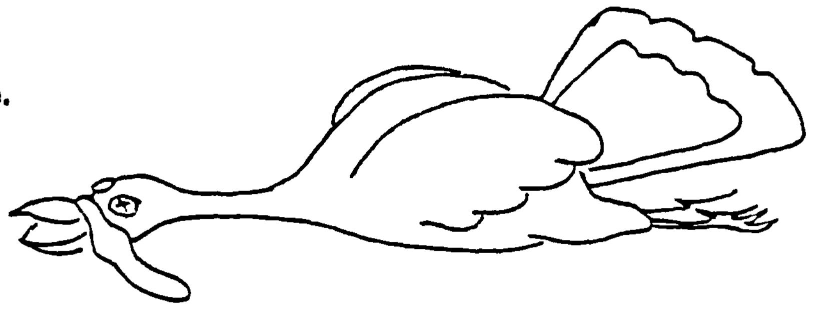
Awarded to

the staff of the

TEXAS INSTRUMENTS LEARNING CENTER

for aiding and abetting the publication of the astonishingly bad and inept EDITOR/ASSEMBLER owner's manual, most likely by putting all of the dunderheads in one project.

Coloring guide:
Beak, feet: chrome yellow.
Wattle: crimson.
Body: medium brown.
Tail: orange brown to yellow orange, red tips.



Oh! By the way, in the trade this is known as a User Response Form.

Alternative malediction:

A valedictory malediction:

May armadillos devour your dog, your horse develop ingrown toenails, your children never comprehend any part of your job, your car run great for years at two miles per gallon and you henceforth be laughingly called to your face, 'DT the first', by your coworkers who will xerox, distribute and post in their cubicles this award.

shucking your duds

(nekkid power, continued)

gram. Firstly, you STOP, which is handy for not RUNning 2 after 1. Second, you'll note that the prescan is turned off in all three lines. Line three makes certain that it is indeed off, as I don't quite trust its behavior with longish lines. Prescan off (!@P-) permits you to have defined any of Nekkid variables as DIMs, DEFs or whatever and still RUN Nekkid. There's a slim possiblity that some programs might turn the prescan back ON (!@P+), but if you're smart enough to do that, you're smart enough to deal with it.

Both of them, 1 and 2, work the same. Firstly they remind you of what you'll have to ENTER in the immediate mode after they STOP. (That's the trick: really YOU do the work. Nekkid is an idiot savant -- great with numbers but doesn't DO anything.) Then they demand a number of lines that you want to save. RUN 1 will clip the end of your program, saving the '1st A' lines, 2 will drop the start and save the 'last A' lines. Get it? An example (diagram at bottom):

```
RUN 1 and get the reminders:
   CALL LOAD(D,B,C)
   1st ?
```

If you reply 6, Nekkid calculates the values of D, B and C that will adjust the BASIC system's pointers for your program. When you ENTER 'CALL LOAD(D,B,C)', your program will suddenly consist of lines 1, 2, 3, 100, 110 and 120. For most ordinary purposes, the others are gone.

RUN 2 is similar, except now you need to count backwards. A reply of 2 (after 1, above) will calculate the limits for 110 and 120. A simple diagram:

```
1   NEKKID 1       ]
2   NEKKID 2       ] RUN 1
3   NEKKID 3       ]  will save the 1st A lines
100 ...            ]
110 ... your    } ]
120 ...program  } ]
140 ...is here,} ]
150 ...chewing } ]
160 ...its cud }
170 ... and    }
180 ...minding } RUN 2
190 ...its own }  will save the last A lines
200 ...business}
```

Remember: RES 1,1 will make counting much, much easier.

How does it work? That's simple arithmetic and a little research. First of all, we know that BASIC programs consist of a table of line numbers. These are made up of your program's numbers and their respective pointers to where the code of the line is stashed. Each line has 4 bytes, 2 for the number and 2 for the location. And secondly, we know that hex 8330 and 8332 in PAD are pointers to the beginning and ending bytes of the table. >8330 is the pointer to the highest line and >8332 points to the lowest line. Changing them over to BASIC two's complement, we get -31952 and -31950.

So, without going into the absolute details, Nekkid looks at the value of the low line number pointer, calculates how many four byte blocks are needed for 'A' lines and puts the calculations into B and C. (That's RUN 1.) If there is a trick to all of this, it's that you can't just finish the program with CALL LOAD(D,B,C). Ordinarily -- or more accurately, every time I tried-- when you do adjust the line number table the system must forget where B and C are. In short, a fancy way to crash. BUT BASIC remembers in the immediate mode.

A simple immediate version:

```
   >CALL PEEK(-31952,A,B,C,D)::PRINT A,B,C,D
   >PRINT D-((number of lines)*4-1)
```

If the calculation goes negative, add 256 and subtract one from C. CALL LOAD(-31952 with the two numbers. This is equivalent to Nekkid's RUN 1.

If you have a routine you need moved, just MERGE Nekkid, strip the routine out, RES it and SAVE it back in MERGE format. OLD your original and MERGE your new code. Or if you've a bunch to get rid of, strip off the front and SAVE that, then OLD to get the tail end. The middle disappears. MERGE the front back in.

Lastly, consider: Nekkid is fast! It puts TRANSL in the old age home, relegated to its proper role of TRANSLating downloaded files. So, let's get nekkid! (There goes a real hit single streaking by.)

>Frederick Hawkins

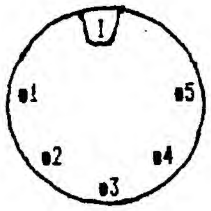
bad algorithm

CHRISTMAS DINNER DANCE
Lehigh 99'er Computer Group
Green Pond Country Club
December 22, 1984
6:30 P.M.
Prime Rib
$20.00 Per Person
Cash Bar

PROGRAMMER'S POTPOURRI, IN THE POT,

It's very likely that most TI users don't even miss the two or three characters we seem to lose to the sides of the screen. But for the few, there is a $250 solution. With a horizontal adjust that allows you to put a CALL HCHAR(1,1,X) nearly an inch into the viewing area. Of course, there's a catch: you have to make up the cable. The solution is the standard COMMODORE 64 monitor (model 1702). The cable consists of two RCA plugs (for the front end of the monitor), a five pin DIN connector (the video circuit plug) and a four wire cord. The connections are as follows:

5-pin DIN (Radio Shack 42-2151)
- pin 1: 12VDC (no connection)
- pin 2: video to RCA #1 center
- pin 3: shield to both RCAs' cases
- pin 4: ground (no connection)
- pin 5: audio to RCA #2 center (TI signals out, take care!)



TI 99/4A VIDEO/AUDIO to COMMODORE MONITOR 1702

'I' represents the indexing key. Diagram shows plug as viewed from the computer. Mind the 12 volts; short #1 to one of the other pins and you risk owning a doorstop with a keyboard.

The video is quite clear. The monitor has a visible grid, caused partly by the 9918 video chip's composite signal and partly by the monitor's resolution. Each pixel is very precise. However I frankly prefer a good quality tube -- the 'blending' hides the color phantoms. Some users may be bothered by the monitor's own grid, which probably has a good technical name, but put ignorantly looks like an black outline around a pixel. The monitor's audio isn't very good and the speaker talks to the ceiling. That might be a godsend for the errant BASIC programmer, though. Let it HONK. If you're looking to upgrade from the black and white exile (or worse) that competition with TV puts many users, it's worth considering.

Magic markers make it simple. There's no reason on earth why using a computer to design character shapes should be either easy or even fast. After all, we've spent decades pushing pencils, using an unduplicatable analog interface between head, muscle, hand, eye and so on. And just because you've a computer, it don't make sense to settle for yet another joystick-sensing routine that don't work so hot or insists on twelve separate keypresses to put down two dots. Back in the real world of kids and crayons, magic markers and graph paper (1/4 inch ruled, cheap by the tablet) the ordinary slob (me) can design an entire alphabet in 20 minutes or so. Try that on your whizbang program. Keying it in is another matter.

A fat, juicy magic marker bleeds just right on graph paper, simulating to a 'T' just what you're going to end up with on the screen. One thing for certain, when it just isn't right, becoming violent with paper is a whole lot cheaper than putting your console through the tube a hour before the superbowl. You can do it as much as you like, i.e. more than just once. Some examples follow, NOT worked up just for this item.

A pictograph that helps. An ofttimes overlooked means of keeping a bit of information clear is to draw its image. A great example can be found in most descriptions of how the 'fracturing' information about a disk file is stored.

("Fracturing" describes what happens when your concept of what a file looks like is actually stored on a real bit of magnetic surface. Two more terms pop up at this juncture: "logical record" and "physical record". These correspond to the break between what we think we're doing and what the system does. It's important to stress that there is little irony in this dichotomy. TI's system went to great lengths to conceal from the user the how of much that goes on in, say, storing a record, reading a file. This "hiding" extends down to the AL level. So, in BASIC you INPUT and PRINT, in AL you DSRLNK using PABs, as you do in FORTH as well. This by the way is a GOOD THING. There probably isn't another home computer before TI or since that so carefully engineered the software links to devices as the 99/4A. TI deserves some brickbats but they did the interfaces the right way. 'Ray, somebody!

Digging a little more parenthetically deeper, the only reason why TI users try to get past the logical to the physical is because as a group we are 'code short'. Unlike the Apple-Pet-Commodore-Atari train, the only 9900 code you're likely to run across is written for the 99/4A. And there's damn little of it. So, we're -- as a group -- ready

NINE DAYS OLD

to bust a disk's protection just to read how-it's-done and to break all of the rules "hiding information" to find out more. This explains part of the popularity of reading fracturing. The other, alas, is probably simple theft.

"Hiding information" is a technique by which a programmer and his programs live or die. Although my BASIC programs are written in A$(#)'s and TEMP$'s, when the program RUNs, I can write letters, make lists of names and do my taxes. When I'm RUNning the BASIC interpreter, that is, writing a program, I don't ever have to worry about where A$ is or (mostly) how big it is, or what's next to it anyway? One of the cold shocks of the BASIC-to-AL plunge is just how much BASIC does for you. Let's see someone out there trot out the source code for simultaneously using 9 disk files.

And if someone were to have just that on hand, you can be certain they don't bother to not DSRLNK. The odds are great that they don't fiddle with the fracturing directly; the disk DSR (Device Service Routine, the other is DSR LINK) does all of that. Unless, I suppose, they're bent on copying or moving a file from one disk to another. Quick, given all of the cloners, name two besides the Disk Manager cartridge, that can copy just one file.)

Anyway, TI did a little extra hiding, maybe. The fracturing information is organized in a curious fashion. Instead of being simply a direct representation of the starting sector and the amount written onto the last, the three bytes are mixed together. These bytes are about 10 past the file specifiers in the file header sector. Instead of printing all of those specifications (an entire article, plus), I'll just diagram the mix/unmix of starting sector and end of record offset:

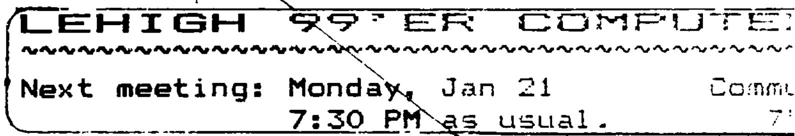
```
              |byte1|byte2|byte3|
              |_____|_____|_____|
as stored:    |s2 s3|e3 s1|e1 e2|

interpreted:  s1 s2 s3      e1 e2 e3
```

Each number is made of 3 nybbles -- a byte and a half; FFF max. This makes sense for the starting sector -- 357 in hex is >165 and we need three nybbles (1 hex digit= 1 nybble) -- but it doesn't for the end record offset. Since each sector for a disk consists of 256 bytes, 0 through FF would be sufficient, unless the size in bytes per sector changes when a double density disk/drive is used. Apparently the controller's format is more flexible than what is either used or needed. Good programming practice!

Faced with the simplicity of the above diagram, many readers might wonder why bother? The point is that you've got it. Doubting Thomases and Freds might try this bit of expository prose: "Pointer Blocks - 6 nibble, 3 byte, cluster that point to the Start Sector numbers and the highest logical Record Offset in the cluster. Change the nibble order from |ss2:ss1| |ro1:ss3| |ro3:ro2| to |ss3:ss2:ss1| |ro3:ro2:ro1|" Pretty sturdy stuff, huh? Almost 100% nondigestible. From Craig Miller's Smart Programmer; his is the best text explanation I've seen.

Something that almost works. Killing off elephants is simpler than you might imagine. My Elephant (aka Carter, both by Dennison) has faded to the point that I just know it won't reproduce well. I can still read it OK, but the copying machines can't. The datum is: bought Oct 14, RIP Dec 14. Twixt, two issues of the IO Port (and about 1/5 of this), and a middling amount of programming and text use. Seat of the pants estimates are likely the best guide; ~~neither~~ * time or paper through and into the used-once-box (2? inches, about). Anyway, they seem to fade just sitting, and particularly quickly in the second month. Compare these two bits of newsletter, one month apart:

*to quality

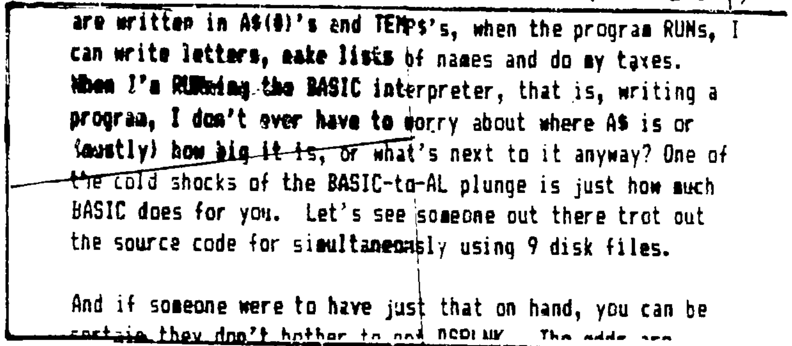

LEHIGH 99'ER COMPUTE
~~~~~~
Next meeting: Monday, Jan 21 — Commu
7:30 PM as usual. — 7

A careful inspection of the ribbons yields the following subjective observations: (Elephant versus Epson)

1) Coarser weave than the Epson (holds less ink)
2) A dryer feel when new, often not at all messy.
3) The vehicle seems to be more volatile, drys out quickly. (No vehicle equals no print.)
4) The ribbon isn't as saturated as the Epson.

Item 4 was proved by the almost-remedy, WD-40. I've read about it, heard about it and tried it in September. Back then, my second Carter was on it last legs. However, I bought the Elephant only after prying off the lids of the first Carter, a (get this!) Sears and the two-year-old Epson and soaked them with WD-40. I waited two weeks to use any, and all made the printer look like a sloppy mimeograph: lower case in compressed mode needed context to be decipherable. Well three months after the soaking, the results are in. The Epson and the Carter still smear -- see below but you're reading the Sears. I suppose it can't last but we'll see. By the way, none of the renewed ribbons can clearly print in double strike mode, and you can expect to paint horizontal tracks if you overdo the WD-40. Otherwise, some samples of what you can expect should you try it:

EPSON | CARTER (single strike only)


are written in A$(#)'s and TEMP$'s, when the program RUNs, I can write letters, make lists of names and do my taxes. When I'm RUNning the BASIC interpreter, that is, writing a program, I don't ever have to worry about where A$ is or (mostly) how big it is, or what's next to it anyway? One of the cold shocks of the BASIC-to-AL plunge is just how much BASIC does for you. Let's see someone out there trot out the source code for simultaneously using 9 disk files.

And if someone were to have just that on hand, you can be

(continued, page 8)
~~~~~~

POTPOURRI, GETTING COLD

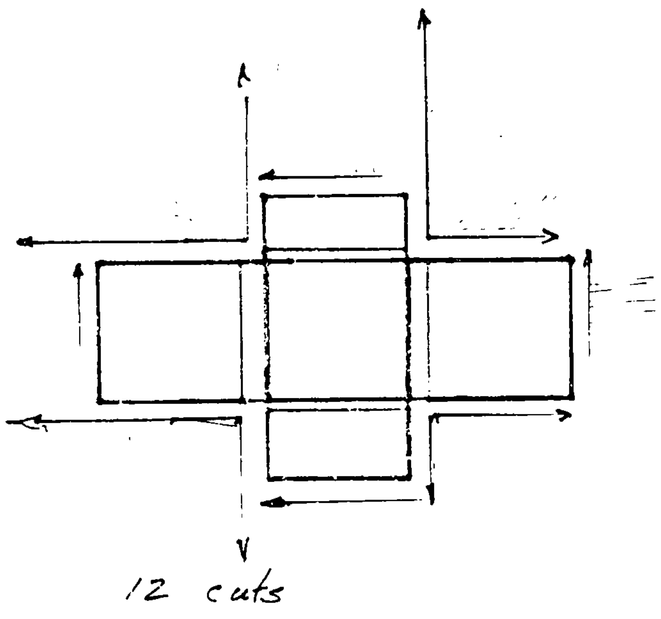
The Epson's ink quality is a rich blue-black, the Carter looks both lighter and more towards brown. Moral: It's cheap for a reason. I'm looking for an Epson dealer, and waiting for the WD-40 to percolate.

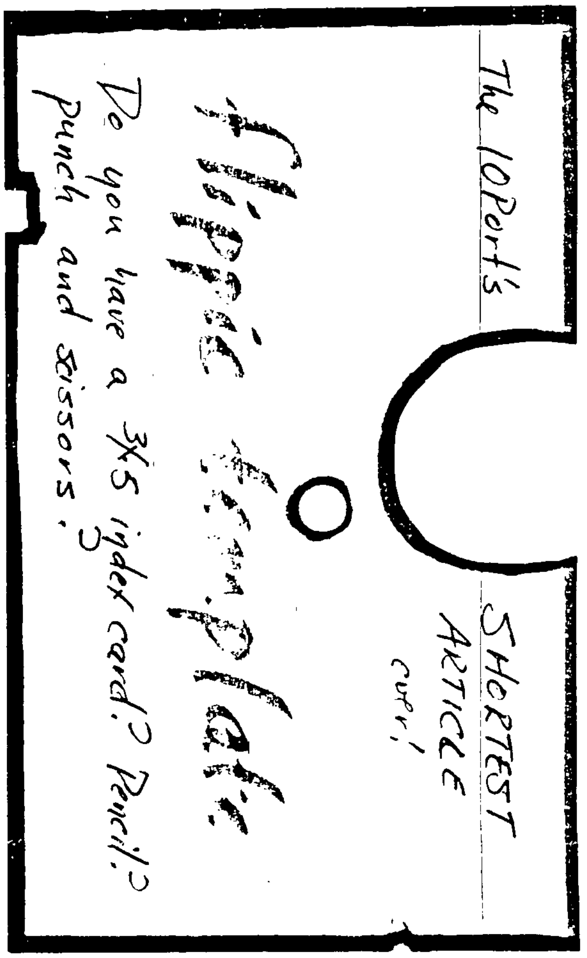
Multicolor TI MULTIPLAN. Know-it-alls can't read or won't, experts read only what they need to. But back on page 12 of MULTIPLAN manual there's a tip for all. The page looks like the typical idiot's reminder and if you scan the page, it seems to say turn it all on, select option 2 and press ENTER when you have the disk in place.

But wait! Lookit number two. READ number two. (Thank Pat Leibensperger; she didn't read it either but she's a hunt and peck typist and bumped the space bar on the way to the ENTER.)

Five-minute disk mailer distructions: It could take less time if you remember where the utility knife is. Besides the knife you need some corrugated card-board, a pencil, a breadknife, a straight edge and a disk. Put the disk in the middle of the 'board and loosely trace it three times, making a row of boxes. Add to one end the thickness required by the number of disks you're sending plus one 'board thickness. Do the same at the other end but allow for two boards. Now extend the middle box about 2 inches on both top and bottom. It should look like the sketch below. Whip out your knife and whack off the corners, using the straight edge to do just that. Switch to the dull knife and score the board so the short ears fold in first, and so on. Put the disk(s) in and now you can waste your time looking for tape, stamps, address....

The finished mailer is nearly bullet proof, reversible (put your address inside for a return trip!), and probably won't wear out; you'll replace it because it's gotten scuzzy.





RESequencing index for BASIC. A program's size is a major handicap to the printer-less user. As the program gets bigger lines get harder to find. RESequencing is often dreaded as much as it's needed: in 300 lines one can spend twenty minutes searching the LISTing just find one routine. Well, here's a way to get elbow room by RESing, yet rapidly find the important lines:

```
10 GOTO 100
11 GOTO 230 REM first routine
12 GOTO 400 REM second
24 GOTO 534 REM third and so on
100 REM this is the first line
that executes.
```

Line 10 lets your program RUN correctly. REM statements can be added in console BASIC because they tell the interpreter to ignore what follows; and mostly it does. The XBASIC version is simpler:

```
10 GOTO 100 !    auto index
11 GO 230 ! name or description
```

The GO is a variant syntax for GOTO. It correct form is GO TO, which gets tokenized as hex 85 and B1 (85 & 177). GO by itself is an error but RES isn't bugged at all. GOTO has a single token, >E which saves a byte. Console BASIC's editor won't let you ENTER just a GO but XBASIC is a little more resiliant.

>Frederick Hawkins

xperimenter's XBASIC:
WHEW! WHAT was that? (#'s 2,3,4 & 6)

In keeping with the festive spirit of the approaching holidays and their attendant free time, I offer these sugar-coated hollow calories. They may not be useful, but they sure are spiffy. Suggestions for use:
a) The idle evening's entertainment; just fiddle with numbers.
b) Should an 'expert' visit, sheepishly ask his opinion. (Tip, RUN one first and then set the hook: ask an innocent question before LISTing.
c) Put either of the third or fourth on a new disk as a LOAD. Turn your would-be genius kids, nephews, grandchildren loose.
d) Be real scientific: YOU (I could but won't-- just yet) figure out exactly what to CALL LOAD and PRINT to set any given sprite going.

Overview: These quick and dirties (Boy are they ever!) meddle with the system's PRINTing screen location -- WHEW2, WHEW3, WHEW6. WHEW4, on the other hand, trys to tell the computer there's up to 255 sprites buzzing around. There isn't of course; you can only have 32. Since WHEW4 doesn't fit with the others, we'll start out with him:

WHEW4: Alters location >837A, "number of sprites in motion". This is part of the console's GPL status block. The block is a set of 16 consecutive memory addresses in PAD (aka cheapo TI's fastRAM). A little-known 4A 'secret' is that:
1) Sprites don't move because the video chip is moving them.
2) Sprite motion is directed by the console ROM (yeah, sprites OUGHT to come with plain BASIC, the code is minimal, about like a CALL CHAR.)
3) The interrupt processing routine doesn't do much error checking; it just calculates each sprite's increment and writes the new location back out. The interrupt occurs every 60th second, and because this routine reads from the VDP, then calculates, and finally writes, it is SLOW. The more sprites, the slower things get; try setting only the 28th sprite into motion and see how much XBASIC slows down. This routine is also the reason we have the lag when you change sprites direction. (FORTH and AL aren't immune, either. If you use auto-motion you're gonna crawl-- you're much better off keeping the locations in RAM someplace and writing the updates without looking first.)

Back to WHEW4: This routine sets the base number to a range from 32 to 255. That is, it tries to. As the console interrupt calculates the apparent sprite positions it eventually the writes past the character color table, past the character descriptions and into BASIC's number area, either the floating point or maybe the variables themselves. The loop's limit, increment and the loop counter get clobbered and BASIC figures it is finished 'long about 125. In the meantime, you get to watch the auto-motion routine move through the VDP. This variation crashes, SAVE first!

WHEW2 (and 3): This simplicity changes where the system stores the pending print (column) postition. Limit yourself to 0 to 31 to get results just like a DISPLAY AT(24,X). Again there isn't any error checking. After about 100 you start to see sprites, without CALLing 'em! The sprite's motion is directly caused by the PRINT, not the interrupt routine. WHEW3 proves that the sprite is made of what you're printing, in this case "A;". The memory location is >837F.

WHEW6: Adds an some embellishments to WHEW3, including the direct sound LOAD. There's a couple intersting sounds in here, especially a crytal-clear chime. This is one of those programs to get the creative juices going, rattle the structured programmer's cage and open sleepy eyes. Fiddle with it.

>Frederick Hawkins

```
1 ! WHEW4
2 CALL CLEAR :: PRINT "SET C
OLORS" :: FOR A=1 TO 12 :: C
ALL COLOR(A,16,4):: NEXT A
3 PRINT "BUILD THE STR$" ::
FOR A=0 TO 254 :: A$=A$&CHR$
(A):: NEXT A
4 PRINT "SHOW ALL CHARS" ::
FOR A=0 TO 31 STEP 2 :: B=1+
(A*8):: DISPLAY AT(A/2+3,3):
SEG$(A$,B,8);"      ";SEG$(A$
,B+8,8):: NEXT A
5 PRINT "ANY KEY to test CAL
L LOAD"
6 CALL KEY(0,K,S):: IF S=0 T
HEN 6/7 FOR B=31 TO 125 :: C
ALL LOAD(-31878,B)
8 DISPLAY AT(14,5)SIZE(-5)BE
EP:B :: NEXT B
9 ! call load(-31878,0)
10 ACCEPT AT(14,13)SIZE(-1):
A$
11 CALL LOAD(-31878,0)
```

WHEW2 and WHEW3

```
1 CALL CLEAR :: CALL MAGNIFY
(4)
2 FOR B=1 TO 10 :: FOR A=100
TO 255 :: PRINT A;:: CALL LO
AD(-31873,-A):: PRINT CHR$ (
A);:: NEXT A :: NEXT B
```

same except CALL MAGNIFY(2) and PRINT "A;"

WHEW6

```
1 CALL CLEAR :: FOR A=1 TO 1
4 :: CALL COLOR(A,16,2):: CA
LL SCREEN(5):: NEXT A
2 CALL MAGNIFY(4)
3 FOR B=10 TO -10 STEP -1 ::
FOR A=100 TO 255 STEP ABS(B-
(B=0)*14):: C=64+(A AND 31):
: D=SGN(B)*(A*B)
4 PRINT B;A;"Ok";CHR$(C);CHR
$(D);:: CALL LOAD(-31873,-A)
:: PRINT CHR$(A);CHR$(C);
5 CALL LOAD(-31744,-C,-A,A,-
D,C):: NEXT A
6 CALL VCHAR(1,1,32,464):: N
EXT B ! CALL DELSPRITE(ALL)
:: NEXT B
7 CALL SOUND(110,444,3)
```

the asynchronous sieve

First of all I would like to thank the readers from other user groups who have written me for information and copies of TE-III. The kind words of praise about this newsletter are appreciated by both Fred Hawkins and myself. We hope to keep up the standards that you have grown used to in the last several months. Again thank you all.

S.O.S. Publishers is preparing a new mini magazine called "Mini-Mag 99". Exclusively for the TI-99/4A users, Mini-Mag will include feature articles, "new" product reviews, book reviews, news items, etc.
To receive your first FREE issue, write to:

S.O.S. Publishers
21777 Ventura Bl. #203
Woodland Hills, CA 91364
(818) 704-0145

Don't miss out! The first Issue will be out January 20, 1985. Looks like we're always gaining new sources of information for the TI-99/4A (and you thought it was dead!).

More good news for the TI user comes from The SOURCE. SUBFILE 99, lost for several months, is now back online in a new place. This online newsletter can now be accessed by typing PUBLIC 181 DIRECT at command level. I got a brief look at it and it looks pretty good. There was quite a variety of information presented and at least 4 sections were devoted to FORTH. I recommend using one of the "large buffer" terminal emulator programs when accessing and downloading after signing off to save connect time. If some one else is downloading SUBFILE 99 maybe we can share files and cut down both our costs. Let me know! Additionally, SUBFILE 99 will be published online bi-monthly.

A quick note on the software offer by I.S.S. in last month's issue. I sent for 2 cassettes and received them in short order and found there to be 4 programs on each cassette. The "Phantom of Blackmore" adventure is by far the best but it takes up three programs. I'm not too worried about it as the 2 cassettes only cost me $4, a bargain any way you look at it!

SST Software has released an "add-on utility package" for their expanded compiler. The new commands focus on the TI's High Res Graphics mode and include PLOTLINE, SCREENDUMP, INPUT AND PRINTAT in HRG, and the availability of CALL SOUND IN HRG. Also the new commands include the ability to set the screen-to-text mode and use all the normal screen I/O routines in the original SST Compiler.

SST is also offering updates to the earlier version of the compiler and a back-up copy. Contact them at:

SST Software
PO BOX 26
Cedarburg, WI 53012

M + T Utilityware has just released a disk-reading utility called DISK MAPPER. Also in the works are DISK-MANIPULATOR, DISASSEMBLER, and TE-128K. (That's what I heard, 128K!!!!)
For more information contact them at:

M + T Utilityware
3907 Murl Avenue
Muskegon, MI 49422
(616) 773-4504

Write your own adventure!!!!
The following is a reference for a program that can be used to write an adventure file to be used with the TI Adventure command module. I've not had my hands on the program yet but rumors are that it's great! For more info write to:

Markus Weiand
Friedrichstrasse 49
D-5300 BONN 1
West Germany

This follows the rumor of a new adventure for the module called "IRON HEART" that was written by a West German author. Has anyone seen it yet? Let me know how it is.

>Dave Hendricks

~~~~~~

P.O.Box 4837 * 1501 Lehigh St.
Allentown, Penna. 18103

```
------------------
! stamp target   !
!   put it       !
!   here, pal    !
Allentown, PA.18102
!  PERMIT NO.2018 !
------------------
```

Real Programmers CN RD THS ND NT VN NTC.
~~~~~~