CEDAR VALLEY 99er UG MAY 1993

PRESIDENT: Jack Johns (319) 366-4541
VICE PRES: Bob Wahlstrom (319) 393-6042
TREASURER: Bruce Winter (319) 393-0610
SECRETARY: Wayne Betts (319) 377-2493
NL EDITOR: Gary Bishop (319) 377-9574
LIBRARIAN: AVAILABLE

CEDAR RAPIDS/MARION

Supporting the TI-99/4A and 9640 in Eastern Iowa for over 10 years!

NEXT MEETING: 7:00 PM MAY 11, 1993

WEST MUSIC, COLLINS ROAD PLAZA

CONTENTS Page

RUN A VARIBLE NAME AS A PROGRAM

by Tom Freeman, LA Topics

(Reprinted from HOCUS Milwaukee, WI)

OK, boys and girls, here is the "definitive" load program for xb which will run a variable as a program. It can be edited, resequenced, placed in any part of the program, smashed, crunched, merged, whatever you want to do. I have used "unusual" variable names so as not to interfere with any you are using. All the lines are self-contained subs except line 100 which is the line your program must eventually wind up in, with FN$ being the name of the program you want to run (device name first).

How does it work? address -31954 (>832E) contains a pointer to the CURRENT line in the line number table, i.e. a pointer to a pointer. By the middle of subroutine 120 we have gotten to the actual tokenized line. Line 120 finds the "RUN" token (of line 100) and advances to the length byte of the quoted string. The rest is like the previous versions, with a byte by byte replacement of the original string by the one you want, then a return to thhe line that does it.

BTW, if you are using TI's XB, you need a CALL INIT too.

Enjoy, Tom Freeman

```
100 CALL PEEK(-31954,PP,QQ) :: GOSUB 110 ::: RUN
"123456789A123456789B123456789C123456789D"

110 GOSUB 140 :: CALL PEEK(RR,PP,QQ) ::: GOSUB 140 :: GOSUB 120
:: GOSUB 130 :: RETURN

120 CALL PEEK(RR,PP) :: RR=RR+1 :: IF PP=169 THEN RR=RR+1 ::
RETURN ELSE 120

130 FN$=FN$&CHR$(0) :: PP=LEN(FN$) :: CALL LOAD(RR,PP-1) :: FOR
QQ=1 TO PP :: QQ$=SEG$(FN$,QQ,1) :: CALL LOAD(RR+QQ,ASC(QQ$)) ::
NEXT QQ :: RETURN

140 RR=256*PP+QQ-65536 :: RETURN
```

FOR SALE:

Microstar SG10 printer, calble for TI included, $100 or best offer. Jim Bishop, 1716 Texas Ave, NE Cedar Rapids, IA 52402 Home: 319-393-6172 Work: 319-395-0095.

Not for sale: In past issues, an ad for John DeCook of East Moline, IL. He has sold all his equipment, and has nothing left.

TI EQUIPMENT

John Green
1702 Iron City Ave.
Nichols, Ia. 52766
(319)627-4284

2 - TI 99/4A
1 - Speech Synthesizer
1 - Peripheral Expansion Box
1 - Disk Drive & Control Card
1 - 32K Expanded Memory
1 - RS 232 Interface & Cable
2 - Dual Cassette Cables
1 - Joy Sticks
1 - Joy Stick Y-Adapter
2 - RF Modulators
2 - Power Transformers

Early Learning Fun
Minus Mission
Number Magic
Multiplication
Beginning Grammar
Alligator Mix
Percents
Hangman
Addition & Subtraction

Disc Manager II
Teach Yourself Basic
Terminal Emulator II
TI Writer
Editor/Assembler
TI Logo II

Hunt The Wumpus
Pac Man
Camelot
Flack Attack
Tombstone City

Personal Real Estate
Personal Report Generator
Personal Record Keeping
Household Budget Management
Home Financial Decisions
Tax Investment Record Keeping

Introduction to TI Basic (Hayden)
Fundamentals of TI-99/4A Assembly Language (Tab) (M.S.Morley)
Introduction to Assembly Language for the TI Home Computer (Ralph Molesworth)
Assembly Language Primer (John T. Dow)
Easy Programming with the TI 99/4A
36 Texas Instrument TI 99/4A Programs
Texas Instrument Home Computer Graphics Programs

with borders, too. Since a two drive system will eventually need repair in one drive, I have a backup program disk I can use with a one drive system for emergencies, as I would not want to copy an original at a time when malfunctioning is present and perhaps undefined. I only do that with the programs we use daily.

In Page Pro format, I also have files topically arranged; here I have fonts and graphics all on the same disks. It avoids a lot of disk swapping.

One other aid I have found where we have many disks is color coding. When color disks were on sale in bulk, I bought them so at a glance I know red is Christmas, dark blue, patriotic; light blue, special events; orange, Halloween or Thanksgiving; brown, Jiffycards; yellow, fonts; signs, lavendar etc. I started with colored labels until I could build up colored disks. I also know I am not working an original if I have color in most cases.

A last help is the physical arrangement. My disk holders have 2 sides. 1-ARTIST format is in one disk holder. It uses both sides for all my topics and programs; below it on the next shelf is CSGD format and all frequently used utilities such as disk managers, Funnelweb etc. on the second side; on the top shelf is Page Pro with music and games on the second side with programs and utilities I use less often.

Since fractured file takes a lot of loading time, after my working disk is finished, I do a file by file copy with a disk manager. Fractured files are indicated with an * on our disk manager. I figure it must be harder on the drives to go looking for parts of files too and since we use our computers so much, time saving in the long run saves us too. Then I get out a good mystery and read while my final version is being copied!

It is my hope that one of these ideas may help you, too! Perhaps you have an idea that would be even better?

Sister Pat Taylor, BVM

As long as the spare PIO input bit on pin 13 is low, this program will forever check the line. As soon as the spare input is pulled high, the JMP will not occur, and the program will end (to where??)

Now I haven't actually typed these programs in to try them. They are "dry" programs to explain the fundamental operations involved. There may indeed be some other fatal syntax flaw with the programs, but I hope you get the idea of how to use the spare bits. I am not, nor claim to be, an assembly language programmer, at least on the TI.

Now, on to more lofty considerations: There are many problems with just using the above pieces of code to control the spare bits. Some of them are: what if the CRU base is something other than >1300? This could certainly occur if we wanted to get at the second RS232 card. Also, how about the Parallax, Axiom, Multicom, etc. stand alone printer ports? Do they have the same CRU address? Nothing other than convention says they have to. Just our luck they probably don't. But not to worry; there is a way to find out where the port is.

How about going on a scavenger hunt, turning on each CRU base address in succession, and checking for the device name "PIO"? Once the device was found, the CRU base could be stored and referred back to when needed. This would permit the printer device to have any CRU base address. A much better scheme than hard wiring CRU to >1300.

Well, the next layer of problems must be handled: What to do if the printer isn't named "PIO"? I remember my Hamsoft adapter had the name "PP" so there is at least one instance of a nonstandard priter port name. A safe way to do this would be to simply ask for the name of the printer device. The default could be PIO, but make it so that it can be covered up or typed over with the proper name, or simply accept the default by tying enter.

Finally, the coupe de grace would be to make a couple of simple AL programs that could be accessed from extended basic. The linkages and calling conventions are stock programs from many sources. I would envision the following types of calls:

```
1 CALL INIT
2 CALL LOAD("DSK1.SPO")
3 CALL LINK("SPO","PIO",0) !TURN OFF PIO SPARE BIT TO ZERO
4 CALL LINK("SPO","PIO",1) !GUESS WHAT THIS ONE DOES?
```

Or perhaps the companion:

```
1 CALL INIT
2 CALL LOAD("DSK1.SPI")
3 CALL LINK("SPI","PIO",A)
4 PRINT STR$(A);
5 GOTO 3 ! CONTINUOUSLY PRINT STATUS OF THE SPARE INPUT LINE
```

If you wanted the second printer, try:

```
3 CALL LINK("SPI","PIO/2",A)
```

OR

```
3 CALL LINK("SPI","PP",A)
```

Or whatever your print device calls itself.

If you are not careful, you could end up expanding this concept into a global, general purpose CRU bit twiddler. The term for that is called creeping elegance. What say all you AL slingers out there? Can this code be rolled into a concise package, as I have tried to outline here? If you tinker with this, please let me know how you are doing. Hearing reports of attempts and successes are what keeps me involved with the old 16-bit beast. -Gary Bishop, 3270 28th Ave, Marion, IA 52302-1355

REVIEW OF RECENTLY RECEIVED NEWSLETTERS

Rocky Mountain Tic Toc April '93: Jim Peterson's news and views (reprint for Dec 92), using the CC40 as a portable data book, First Draft/Final Copy review by Harry Ledyard, TI supplementary handouts for CC40 and TI 99/4A, disk of the month review.

LA TopIcs April/May 93: XB miscellaneous # 20 and #21 by Earl Raguse with "Listman" program; accessing another 96 characters from a Gemini printer, an improved XB screen dump called Shorty D; hardwiring a load interrupt switch; Logo fun by Johh Bolda; more on Logo by Earl Raguse; Logo Summary by John Bolda and Earl Raguse; review of the Panda expansion box.

(Above submitted by Jeff Craft)

(Note to Bob Wahlstrom: I have misplaced your newsletter review received via packet. I will look for it, and if found, I will include it in the next Newsletter. -Ed)

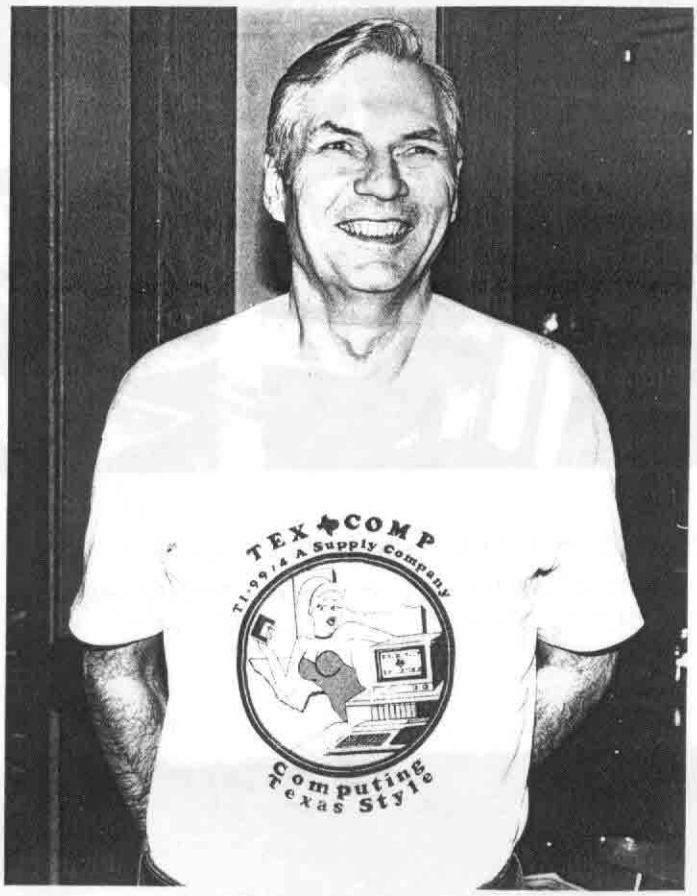
I wish to say thanks to C. S. Stringer of Decatur, IL for providing me with information that I had requested in a past newsletter. It is heartening to know someone out there actually reads this. -Gary Bishop.

RECENTLY RECEIVED NEWSLETTERS

West Penn Apr 93, Cleveland Area User Groups Apr 93, Chicago Times Apr 93, LA Topics Apr/May 93, Tic Toc Apr 93, K-Town 99ers Apr 93, Net99er HCUG Dec, Jan, Feb, Mar, Apr 93, Cedar Valley Computer Association Apr 93.

LEFT TO RIGHT BY HEAD POSITION: RAY NOVEY, BILL [illegible]TH, ED [illegible], BOB HENDERSTADT (FOREGROUND), BOB WAHLSTROM, BOB'S WIFE (JENNY & DON'T GET HER NAME), SIS SCHMOLLING (FOREGROUND) JOHN JOHNSON, JIM GREEN.

JIM GREEN, GARY BISHOP



Our fearless leader, John Johnson
AKA Jack Johns

SOURCE CODE

By Ray Weiss

Evading the tiger

Contrary to popular belief, the human mind wasn't created to sit and philosophize or to solve engineering problems. It was designed to recognize a tiger lurking in the underbrush and leap out of the way. That's one reason we humans do so well at pattern recognition and so poorly at logic.

Generally, we rely on easily recognized patterns, rather than reasoning things out each time. We build up recognition/action patterns for future actions. For example, take driving a stick shift car: At first it's very jerky, eventually smoothing out as shifting becomes second nature—using automatic patterns.

The good news, of course, is that we automate routine chores; the bad news is that we tend to do poorly at new situations.

Another consequence is that we tend to use generalizations for decision making. And that can be very iffy, especially when we rely on "common knowledge."

And so, some words of warning: Many of the technical truisms accepted as gospel may indeed be flawed. Trusting to them can land us in deep yogurt. Here are a few examples that bear watching.

- *Iconic, windowing systems solve the user interface problem.* Wrong on at least three counts. One, icons start to fail for large sets of items; two, running through mouse-driven, windowed menus is distracting and takes time away from the problem at hand; and three, the interface problem may be more complex than first realized—that is, convoluted problem environments (like CAE) end up generating intermediate data structures, which will need to be managed through a higher level view or environment.
- *RISC is simple and small.* RISC started out as a minimal design with a tight ALU cycle (minimal control logic), software deferred actions and a load-/store architecture. It enabled systems houses to turn out micros using ASICs in half the time as standard CISC micros. But today's RISC chips are exceeding 1 million to 2 million transistors, matching or passing the current CISCs. And complexity is creeping back in, especially for multiple instruction issuance (superscalar) CPUs.
- *The waterfall model—specification, architecture, design, code/debug, integration/test, maintenance—defines the software cycle.* Nope. It describes the activity relationships, but software maintenance NEVER ends. Enhancement is continual, ranging from increased functionality to porting to new platforms.
- *Object-oriented programming systems solve the software mess.* Yes, OOPS incorporate a lot of good ideas and are emerging as the next software technology. But don't expect a silver bullet solution. Code reuse—a major OOPS selling point—ain't easy, especially with multiple inheritance. Code maintenance and configuration management costs will go through the roof.
- *Relational databases are good enough.* The theory behind relational databases is fairly straightforward, but there's a snag. Remember the old joke: "If all you have is a hammer, everything looks like a nail." With relational database management systems we try to treat everything like a relation. Unfortunately, much is hierarchical, including company organizations and system designs.
- *Hierarchical decomposition works.* In the early days, hierarchical decomposition worked like a charm: Carve up a hardware or software design into pieces and form the parts into a hierarchy for easy lookup and integration. For small to medium-size designs it still works. For large systems, you end up creating a large hierarchical sewer. We need an alternate approach to paper-and-pencil era approaches.

NEXT MEETING: Tuesday

May 11, 1993 7:00 PM

WEST MUSIC, COLLINS ROAD PLAZA, MARION
ACROSS FROM LINDALE MALL

10

Cedar Valley 99'er Users Group
c/o Jim Green
377 Cambridge Dr. NE
Cedar Rapids, Iowa 52402-1446

FIRST CLASS

Send To:

GARY BISHOP
124-222
3270 28TH AVE
MARION, IA 52302