VOLUME 1 ISSUE 3 NOVEMBER 1984

SUPER 99 MONTHLY

Our theme for this month is business and home finance. We hope you enjoy it as much as we've enjoyed bringing it to you.

This month has been a month with a lot of news to cover. We are keeping an eye on several stories for which inadequate details were available for this issue. We will keep you posted on what we know - but not the rumors that we hear! We've heard some really incredible rumors and such disinformation is not beneficial to you or the producers of products!

We are pleased to have included in this issue articles on Mini Memory. We've had a number of requests for information relating to this useful TI Module.

If you own a disk system and the Editor/Assembler, we hope you have tried FORTH. It is fun to work with and packs a lot of power! Beginner projects are easy in FORTH - even with no prior programming knowledge!

Days Away : The Backbone of an Aging Program

STANDARD: 1A 2A 9A

Keeping a close watch on amounts owed to vendors and amounts due from customers is one of the primary controls to be monitored by successful businesses. The age of the payable or receivable is of great importance. Computers should be used for agings if more than a few invoices are to be evaluated.

At the end of this article is an example program for determining the number of days since a date (one of the more difficult to develop sections of an aging program). The program is valid for any dates in the 20th Century and takes into account leap dates.

The program includes our first example of subprograms - an important tool for the TI-99/4A programmer. The subprogram passes values of variables through a parameter list. Other variables do not affect the program or other subprograms. This is an important concept that is being increasingly accepted as the proper approach to structuring a program. Next month we'll look at the future of microcomputer BASIC as it relates to TI Extended BASIC and will bring together the various concepts we have covered into a format that conforms to modern practices to the degree that is allowed by TI Extended BASIC. We will show that TI Extended BASIC is a much better form of BASIC than is currently available for most microcomputers!

There are many uses for the days away routine. It can be used for calculating the average age of a group of employees (for insurance purposes, etc.). Or, you could determine whether equipment or equipment parts have exceeded their expected life, which could also be useful around the house for auto maintenance, etc.

->

Here is the days away routine:

```
>100 DIM DAYS_AWAY(5)
>110 DISPLAY AT(4,1)ERASE ALL
 :"TODAY'S DATE (MMDDYY)"
>120 ACCEPT AT(4,23)BEEP VALI
 DATE(DIGIT)SIZE(6):DATE$
>130 CALL CONVERT(DATE$,Z)
>140 DAYS_AWAY(0)=Z
>150 FOR I=1 TO 5
>160 DISPLAY AT(I+4,1):"TEST
 DATE";I;"(MMDDYY)"
>170 ACCEPT AT(I+4,23)BEEP VA
 LIDATE(DIGIT)SIZE(6):DATE$
>180 CALL CONVERT(DATE$,Z)
>190 DAYS_AWAY(I)=DAYS_AWAY(0
 )-Z
>200 NEXT I
>210 FOR I=1 TO 5
>220 DISPLAY AT(I+10,1):"DAYS
  AWAY";I;DAYS_AWAY(I)
>230 NEXT I
>999 END
>1000 SUB CONVERT(DATE$,Z)
>1010 MO=VAL(SEG$(DATE$,1,2))
>1020 DA=VAL(SEG$(DATE$,3,2))
>1030 YR=VAL(SEG$(DATE$,5,2))
>1040 YR=YR+1900
>1050 CE=(YR-1)-1582
>1060 KO=INT(CE/100)
>1070 VR=INT(CE/4)
>1080 Z=((CE+KO-VR)*365)+((VR
 -KO)*366)
>1090 RESTORE 1120 :: FOR LOO
 P=1 TO MO-1  :: READ ADD
>1100 Z=Z+ADD
>1110 NEXT LOOP
>1120 DATA 31,28,31,30,31,30,
 31,31,30,31,30
>1130 IF YR/4=INT(YR/4)AND YR
 /100<>INT(YR/100)AND MO>2 TH
 EN Z=Z+1
>1140 Z=Z+DA
>1150 SUBEND
```

Mini Memory Manual Correction

STANDARD: 1A 2B 9A

The Mini Memory Manual includes an excellent assembly language program that has a few misprints in the version we have. The program allows rapid displays of strings to any place on the screen - the equivalent of the Extended BASIC statement DISPLAY AT.

We find the Line-by-Line Assembler to be tough enough to use with a correct listing (the Assembler included with the Editor/Assembler is much easier to use). However, Assembly Language is often worth a lot of effort! Here are the corrections we have noted:

LOCATION	TYPO	CORRECTION
>7E60	MOVE	MOV
>7E90	A1	AI
>7E94	L1	LI
>7EC4	A1	AI
>7EC8	A1	AI

If you had difficulty in keying the listing, now you know that you were not the only one!

The Real Power of Mini Memory

STANDARD: 1A 2B 9A (o: 4A 5A 6A 7A)

Many times we have been asked to evaluate the usefulness of Mini Memory. To be quite honest, Mini Memory does not have the obvious capabilities of XBASIC or the powerful Assembler of the Editor/Assembler. Closer observation shows a myriad of capabilities.

Consider the following program:

```
>100 X$="DSK1.INFO"
>110 REM
>120 OPEN #1:X$,RELATIVE,INTE
 RNAL,UPDATE,FIXED
>130 FOR A=1 TO 100
>140 PRINT #1:A
>150 NEXT A
>160 CLOSE #1
```

Using a TI Drive and TI Disk Controller, the program executes in approximately 34.7 seconds in Extended BASIC.

Switching to the Mini Memory Module, the following changes can be made to the program:

```
>100 X$="EXPMEM2"
>110 CALL LOAD(-24574,8)
```

->

-2-

Execution time is now 20.4 seconds! Similar time factors can be obtained using the MINIMEM space, but not as many records can be stored.

To give some credit back to the disk drives, much of the access time is for naming the file in the disk directory. Subsequent accesses were timed at approximately 27.4 seconds. Of course, Mini Memory's data areas do not require a directory listing.

This amazing power of Mini Memory is what is sometimes referred to as "RAM disk" - the storage of disk-type files in RAM. As shown, a time savings of 25% to 41% was accomplished in the example program.

Sorting is one of Mini Memory's strong points. Disk files created by any Module (or no Module) can be sorted, large files can be sorted in EXPMEM2 faster than using the TI Disk System, and the program that created the file does not have to be used to sort it, since the data can be chained from a previous Mini Memory program! With 32K memory, an Assembly sort would make the process even faster. Sorting in Mini Memory also reduces long-term wear on your disk drive(s).

Cassette files must be SEQUENTIAL files, but can be converted by a BASIC program to RELATIVE format for use in MINIMEM or EXPMEM2. This offers a relatively inexpensive alternative for obtaining disk file power. After the file has been manipulated, it can be saved to tape using the SEQUENTIAL format.

TI did not provide a method of storing Assembly programs created by the Line-by-Line Assembler to disk. However, programs created for Mini Memory using the Editor/Assembler can be loaded from disk. Such programs can reside in Expansion Memory by using AORG.

Mini Memory is especially useful for advanced users and persons operating on a limited budget.

FORTH Phone List

STANDARD: 1A 2C 4B 5A 6B 7A 9A

A good phone list is a must for active people and busy programmers and is very easy to create in FORTH.

Start by finding an unused screen and define FORTH words that will be the initials of the persons on your list:

```
: JS ." John Smith 555-1843 " CR ;
```

If you need to continue on the next screen, use the --> symbol to continue to the following screen. Next, define a word to initialize the screen:

```
: PHONEINIT  TEXT CLS 0 0 GOTOXY
             23 7 VWTR ;
```

If you do not BLOAD your menu options, you'll need to have -SYNONYMS -TEXT at the beginning of the screen. The code 23 7 VWTR sets the screen to BLACK on CYAN and is not required.

The next step is to build as many different phone lists as you need. A single phone listing can be on more than one list. Here is an example:

```
: PROGPHONE   PHONEINIT RH RR ;
: COMPUPHONE  PHONEINIT RH JS SM RR ;
```

The word PROGPHONE is for computer programmers and COMPUPHONE is for all computer users. Of course, you can create your own words that would correspond to your lists. FLUSH the screen and EDIT screen 3. Put in a definition for a new word "PHONE":

```
: PHONE  6 LOAD :
```

Our definition of PHONE assumes the phone list begins at screen 6, so change 6 to the screen you used.

To use the list, just key PHONE and the name of the list you want, such as PROGPHONE. Use FORGET or COLD after using the list to free up memory and reduce the vocabulary entries.

->

Dual-column Right-adjustment Using TI-Writer

STANDARD: 1A 2E 3B 4B 5A 6B 7A 9A 10A

Many of you have probably heard a lot of sad tales about what TI-Writer will not do. Nearly all of those rumors are false! We are so comfortable in using TI-Writer that we use it almost exclusively for this publication (sort of a practice what we preach approach).

One claim has been that TI-Writer cannot be used to create a page such as the one you are reading - what we call dual-column right-adjusted in one pass of the page so that the left and right columns align with one another.

One of the tricks is that we use a parallel printer set for PIO.LF, but select PIO.CR for printing through TEXT FORMATTER. We recommend that you test only a few lines by the following guidelines to be sure the method works with your configuration.

Upon entering TEXT EDITOR, you must always turn WORD WRAP off by pressing <CTRL> <O>. Use a left margin of 0 and a right margin of 79. We have our page length set to 300. The FILL and ADJUST commands are used.

Essentially everything else is done using the caret for forced spaces. It takes some time to get accustomed to all those carets instead of blanks.

It is important that we point out now that we reduce our printout before it is published, so don't expect to be able to use as many characters as we do and still have margins without reduction!

Determine the size of your center margin (we use 3 spaces). For the first line of the left column, use forced spaces from the beginning of the center margin to the right. Only the last word needs to be set to the center margin - spaces will be properly inserted throughout the line, giving a look similar to typesetting.

Proceeding to the next TEXT EDITOR line, the first line of the right column is done in reverse of the left column. We use a caret as the last character (see why below).

We use the third line for a transliterated carriage return. This prevents some oddities from creeping in, such as occasional uneven lines. Keep in mind that any area in which you are not writing should be filled with carets.

If you want to underscore with a solid line, insert a line below the item to be underscored and use transliterated underscore printer commands. Although such transliterate commands are not printed, they do take up space on a TEXT EDITOR line so that the printout will not line up evenly on the right unless you allow at least one caret at the end of each line (see above).

The easiest way to set up the first page is to use LINES COPY from edit mode. Once you have one page built, use it as a master for future pages, changing your page numbers and writing over your previous words. Be sure to remember to save the new page with a new name.

We warn you that this method uses a lot of disk sectors! We don't mind as we use DS/DD diskettes, but on SS/SD it might present problems.

You'll also find that using dual columns is not very forgiving when you need to make massive changes.

Even if you don't intend to use this method of printing, we recommend that you advanced TI-Writer users try a few lines of this to see if you can pick up on any new tricks through modifying our system.

One final note is that you'll discover that a few tricks are needed here and there (nothing that is really difficult), but TI-Writer can handle almost anything using this method!

->

Using the CorComp 9900 Disk Controller: Instant Character Sets

STANDARD: 1A 2A 4B 5A 6B (no) 7A 9A

Nothing seems to fascinate old computer hands and novices alike as much as fancy screen displays. The new BASIC and XBASIC statements provided with the CorComp 9900 Disk Controller Card will allow programmers of an intermediate level to write software that a teen who is an avid video arcade enthusiast will take a second and third look at!

Let's see how to switch out all or a portion of a character set many times in less than 1 second!

Page 81 of the CorComp Manual covers a "CALL MOVEM" statement which allows memory to be moved between any combination of CPU and VDP (except to ROM, of course). Appendix C shows a VDP Memory Map that includes the location of the Pattern Descriptor Table (character set in use). Here is an XBASIC program that includes the REM's necessary for understanding what is being accomplished:

```
>100 CALL INIT :: DELETE "LD-
 CMDS"
>110 ! X=FIRST CHARACTER TO
       MOVE TO CPU RAM
>120 ! Y=LAST CHARACTER TO
       MOVE TO CPU RAM
>130 ! Z=NUMBER OF BYTES TO
       MOVE
>140 X=128
>150 Y=143
>160 ! A=VDP ADDRESS OF FIRST
       CHARACTER TO MOVE
>170 ! B=VDP ADDRESS OF END
       OF LAST CHARACTER TO
       MOVE
>180 A=768+8*X
>190 B=768+8*Y+7
>200 Z=B-A+1
>210 ! C=ADDRESS OF FIRST
       FREE ADDRESS IN LOW
       MEMORY AFTER ASSEMBLY
       ROUTINES LOADED
>220 CALL PEEK(8194,I,J):: C=
 I*256+J
>230 ! 2 SIGNIFIES VDP TO CPU
>240 CALL LINK("MOVEM")(2,A,C
 Z)
>250 ! 3 SIGNIFIES CPU TO VDP
>260 CALL LINK("MOVEM")(3,C,A
 Z)
>999 END
```

But, you didn't see a thing when you ran that one! So, add the lines that follow to prove that it works:

```
>105 GOSUB 2000 :: GOSUB 1000
>245 GOSUB 3000
>1000 ! SET CHARACTERS 128 TO
 143 TO SOLID SQUARE
>1010 FOR I=128 TO 143
>1020 CALL CHAR(I,"FFFFFFFFFF
 FFFFFF")
>1030 NEXT I
>1040 RETURN
>2000 ! PRINT CHARACTERS 128
       TO 143
>2010 FOR I=128 TO 143
>2020 PRINT CHR$(I);
>2030 NEXT I
>2040 RETURN
>3000 ! SET CHARACTERS 128 TO
       143 TO ONE PIXEL
>3010 FOR I=128 TO 143
>3020 CALL CHAR(I,"1")
>3030 NEXT I
>3040 RETURN
```

Of course, we realize that we had a lot of lines that would normally be calculated outside the program. This was an example.

We now know it takes 128 bytes to store characters 128 to 143. So, if we want another set of 128 to 143, we should begin at C+Z. If no Assembly routines have been LOAD'ed, the next CPU address to use would be 9568 plus 128, or 9696. Be sure to not go too high in Low Memory, as the REF/DEF Table for Assembly routines works its way down from 16374. Using 9568 to 16000 gives room for 804 characters or 8 full screens (Screen Image Table) to be stored in Low Memory alone!

We are working on an article for the really advanced CorComp user for making full use of the CorComp statements (maybe next month).

->

Household Consumer Price Index
Using Multiplan

STANDARD: 1A 2D 4B 5A 6B 7A 9A

Federal authorities periodically release figures known as the Consumer Price Index. If investments are your interest, the index may help you. For household management, typical prices nationwide are rather meaningless.

So, we decided to create a spreadsheet to relate prices to your household's economic structure. Our intention is to avoid an excessively scientific approach.

The idea is to consider as many variable costs as possible. Each item considered will be expressed in terms of a typical quantity you would consume in one month. Then, each month you enter the current cost of that quantity of consumable. The total for the first month will be the base amount. Totals for subsequent months will be compared to the base period for an index. The index will indicate the degree to which your household budget is controllable. For instance, if there is a ten percent increase in the index, your costs can be expected to be ten percent over budget. Any other variance deals with your money management decisions.

HOUSEHOLD CONSUMER
PRICE INDEX

	JAN	FEB
UTILITIES		
electr. (3000 u)	200.00	190.00
natural gas (20 c)	17.00	19.00
gasoline (150 gal)	150.00	175.00
TOTAL UTILITIES	367.00	384.00
FOOD		
eggs (4 doz)	4.00	3.86
milk (4 gal)	12.00	12.40
round steak (6 lb)	18.00	12.00
ham (4 lb)	16.00	14.00
potatoes (10 lb)	5.00	6.00
TOTAL FOOD	55.00	48.26
TOTAL INDEXED ITEMS	422.00	432.26
INDEX	100.00	102.43

Of course, our model was designed to conserve space and yours may include many more items, especially if you do a lot of entertaining, etc.

The formulas should all be easy if you followed our September model or the examples in the Multiplan manual, except the one for the INDEX:

(R[-2]C/R19C2)*100

Remember to use COPY RIGHT 11 to build February through December.

Further work with the model could lead to an automatic multiplication of the unit cost times the units, one or more dependent sheets for summary information, automatic adjustment of your household budget, seasonal adjustments, etc. The model could also be modified for variable business expenses.

Program Testing

by Jim Peterson, Tigercub Software,
156 Collingwood Ave., Columbus, OH
43213

STANDARD: 1A 9A

No programmer should test his own programs, for two very good reasons:

1. By the time you have the program debugged, you are so sick of looking at it that you test it all too briefly, while your mind is on the next program you just thought of.

2. You wrote the program, you know which keys to hit, so you subconsciously avoid hitting the wrong ones.

The best way to get a program tested is to go out and catch a 6 year old, wipe the jam from his fingers, set him down at the keyboard and turn him loose. He will hit all the right keys in the wrong order, all the wrong keys in the wrong order, and all the keys at once. He will make your program do things you never dreamed

->

-6-

of, and produce error messages that you didn't know were in the computer's vocabulary!

I have one program which will begin to print the alphabet across the top of the screen after you deliberately press the wrong key 28 times. A 6 year old showed me that - and I still haven't found anything in the program logic to account for it!

It is amazing how easy it is to overlook basic flaws in your own programs. I recently received the disk of programs from the Home Computer Magazine Vol. 4 No. 3, after a long wait. One of the programs is Elementary Addition and Subtraction - quite good, although the response is slow. It generates random numbers between 1 and 5, for addition and subtraction practice.

The first time I tried it, it asked me for the answer to 1 + 1. When I answered correctly, it produced another random problem - 1 + 1 again! This is known as the idiotic computer syndrome, and it helps us to remember that computers are still no smarter than their programmers. However, this bit of idiocy is easy to cure.

In line 1170 you will find LEFT= INT(5*RND)+1. Just add two lines:

```
>1172 IF LEFT=LEFT2 THEN 1170
>1174 LEFT2=LEFT
```

You will have to make similar changes after lines 1780 and 1800.

Do you see how it works? The first time you get a number, LEFT2 will equal 0 because it has never been given a value. LEFT will have been selected as a number between 1 and 5 - let's suppose it's 2, which does not equal LEFT2, so the program continues to line 1174, where LEFT2 picks up the same value as LEFT. So, the next time around, if line 1170 picks 2 again, line 1172 will match it with LEFT2 and go back for another number.

If you want to avoid a repeat until after two times, change to:

```
>1172 IF (LEFT=LEFT2)+(LEFT=L
 EFT3) THEN 1170
>1173 LEFT3=LEFT2
```

Make sure you get that in the right sequence. For a test, change to:

```
>1170 LEFT=INT(3*RND)+1
>1175 PRINT LEFT
>1176 GOTO 1170
```

With only 3 numbers to choose from, the same sequence is repeated endlessly.

Peeking and Pokeing Bits

STANDARD: 1A 2A 5A 7A 9A

Many of you have probably worked with PEEK and LOAD for accessing bytes. However, you may not have considered the possibilities available in accessing bits. While many possibilities may come to mind for you, we have an idea that may not come to mind so easily.

Our project is to write a program to simulate 800 coin tosses, store the results of each toss, total the heads, tails, and total tosses, and display the totals. Here is a solution that might immediately come to mind:

```
>100 DIM A(800):: RANDOMIZE
>110 FOR I=1 TO 800
>120 A(I)=INT(RND*2)
>130 NEXT I
>140 FOR I=1 TO 800
>150 IF A(I)=1 THEN H=H+1
>160 TL=TL+1
>170 NEXT I
>180 PRINT "TOTAL";TL
>190 PRINT "HEADS";H
>200 PRINT "TAILS";TL-H
```

The program RUN's in about 82 seconds, requires 39 bytes of stack space and a whopping 6610 bytes of program space.

Our solution is on the next page. We store the results of 8 coin tosses in one byte - making the new solution over 8 times as byte efficient!

->

-7-

```
>100 CALL INIT
>110 FOR I=9568 TO 9666 STEP
 2
>120 RANDOMIZE :: CALL PEEK(-
 31808,A,B)
>130 CALL LOAD(I,A):: CALL LO
 AD(I+1,B)
>140 NEXT I
>150 FOR I=9568 TO 9666 STEP
 2
>160 CALL PEEK(I,A,B)
>170 FOR J=0 TO 7
>180 X=((2^J)AND A)/(2^J)
>190 Y=((2^J)AND B)/(2^J)
>200 TL=TL+2
>210 H=H-((X=1)+(Y=1))
>220 NEXT J
>230 NEXT I
>240 PRINT "TOTAL";TL
>250 PRINT "HEADS";H
>260 PRINT "TAILS";TL-H
```

Our solution RUN's in about 47 seconds and uses only 109 bytes of stack and 401 bytes of program space! This is a good example of why there are so many users who unneccessarily complain of not having enough memory!

Many of you will probably need an explanation of what in the world we did in our example, so here we go.

Our first loop increments through the first 100 bytes available in low memory (we've allowed for loading of CorComp Controller routines). Since there are 8 bits in a byte, 800 bits (with values of "1" or "0") can store our 800 results in 100 bytes.

In the first loop, we access the random number seed in scratch pad RAM. This address is identified on page 406 of the Editor/Assembler Manual. For a complete explanation of how this address works to produce random numbers, you should purchase Millers Graphics' excellent book, "Smart Programming Guide for Sprites" (see MG address on p. 10). We then LOAD our random numbers into low memory.

In our second loop, we return to the addresses in low memory. Using a nested loop, we "peek" each bit. The weights of bits can be expressed as 2 to the power of 7 minus the bit being accessed (in other words, we count the bits in reverse sequence, with the Least Significant Bit being numbered as 0 and the Most Significant Bit being numbered as 7 and use the new reverse sequence as the exponent). By AND'ing with the reversed bit sequence, we determine whether the bit is on or off. By dividing by the bit weight, the value becomes either a "1" or a "0". Here is a chart to help in understanding this principle:

Normal sequence	Reversed sequence	Bit Weight	Power of 2
0 (Msb)	7	128	7
1	6	64	6
2	5	32	5
3	4	16	4
4	3	8	3
5	2	4	2
6	1	2	1
7 (Lsb)	0	1	0

We do this for each of the two numbers produced by the random number seed.

Finally, at line 210 we use logical addition to track the count of heads.

If you still do not understand, the best thing to do is to insert PRINT statements in the program to see what values are being produced. We should add that this is a nifty way of debugging a program. When you get into advanced programming techniques, you'll occassionally produce numbers without knowing how they were generated. Printing all of the values leading up to the final value may disclose the program bug. You can also often access variables after the program has run - from immediate mode! For instance, you can RUN our example program and then key:

```
>PRINT I;A;B;J;X;Y;TL;H
```

So, what can we do with all this other than flip coins? Well, we can obviously sample a population randomly for one. Or, large numbers of "yes" or "no" ("1" or "0") flags could be stored. Or, any bit could be peeked or poked for whatever purpose.

->

Hardware Maintenance:
The Navarone Cartridge Expander

STANDARD: 1A 8A 9A

The Navarone Cartridge Expander has been highly touted as a tool of convenience. No doubt, the ability to plug in 3 cartridges at once and change cartridges with the flick of a switch and reset the title screen with the push of a button, as is possible with the Navarone unit, is extremely convenient. But, we find it even more important that the Expander reduces wear on the cartridge port and the cartridges - both in the form of physical contact damage and heat damage (expansion and contraction). The Expander offers the bonus that if one of the Expander's ports suffers damage, two more usable ports remain. When not using the Expander, damage to the cartridge port is a potential (and almost inevitable) problem. Cartridge port problems are very frustrating - one moment the cartridge is working, the next moment it ceases to work and all your work may be gone! We highly recommend the Navarone Cartridge Expander for hardware maintenance - or should that be for convenience?

Statistics

STANDARD: 1A 9A

Here are a few elementary programs for common statistical needs:

```
>100 REM SIMPLE AVERAGE
          ENTER 9999 TO END
>110 INPUT "NUMBER":A
>120 IF A=9999 THEN 150
>130 TOTAL=TOTAL+A
>140 COUNT=COUNT+1
>150 GOTO 110
>160 AVERAGE=TOTAL/COUNT
>160 PRINT "AVERAGE";AVERAGE

>100 REM STANDARD DEVIATION
          ENTER 9999 TO END
>110 INPUT "NUMBER":A
>120 IF A=9999 THEN 170
>130 TOTAL=TOTAL+A
>140 SIGMA=SIGMA+A^2
>150 COUNT=COUNT+1
>160 GOTO 110
>170 MEAN=TOTAL/COUNT
>180 VARIANCE=SIGMA/COUNT-MEA
 N^2
>190 DEVIATION=SQR(VARIANCE)
>200 PRINT "TOTAL    ";TOTAL
>210 PRINT "MEAN     ";MEAN
>220 PRINT "VARIANCE";VARIANC
 E
>230 PRINT "STANDARD DEVIATIO
 N";DEVIATION

>100 COMPOUND INTEREST
>110 INPUT "PRINCIPAL    ":PRI
 NCIPAL
>120 INPUT "ANNUAL PCT   ":IRA
 TE
>130 INPUT "MONTHS       ";MON
 THS
>140 IRATE=IRATE/100
>150 FOR I=1 TO MONTHS
>160 COMPOUND=(PRINCIPAL*IRAT
 E)/12
>170 PRINCIPAL=PRINCIPAL+COMP
 OUND
>180 NEXT I
>190 PRINT "COMPOUND AMOUNT =
  ";PRINCIPAL

>100 REM MARK-UP AS PERCENT
          OF RETAIL
>110 INPUT "COST         ":COST
>120 INPUT "PCT MARKUP ":PCT
>130 PCT=PCT/100
>140 RETAIL=COST/(1-PCT)
>150 PRINT RETAIL

>100 REM  MARKUP AS PCT OF
          COST
>110 INPUT "COST         ":COST
>120 INPUT "PCT MARKUP ":PCT
>130 PCT=PCT/100
>140 RETAIL=COST+COST*PCT
>150 PRINT "RETAIL";RETAIL

>100 REM SUM
          ENTER 9999 TO END
>110 INPUT "NUMBER   ":A
>120 IF A=9999 THEN 150
>130 TOTAL=TOTAL+A
>140 GOTO 110
>150 PRINT "TOTAL";TOTAL
```

See - writing useful programs can be very easy!!!

->

-9-

SUPER 99 MONTHLY

99 POTPOURRI

News, Corrections, Updates, Editorials, Kudos, and Come-what-may

CORRECTIONS TO OCTOBER:
PAGE 9: Purists prefer tuner/contact cleaner over alcohol. Though many persons have informed us that they use a pure form of alcohol without any problems, we do advise using the same product as the pro's use for safer results.

NEW PRODUCT NEWS!

Foundation has announced the availability of an improvement to their 128K card, allowing for a 96K disk file emulator (a 64 byte record file that is vaguely similar to Mini Memory RAM files). We have not tested the card, but our opinion of the information available to us is that it is the best memory expansion card available. However, due to its high price, it is a must that you write Foundation Computing, 74 Claire Way, Tiburon, CA 94920, before buying! The card does not do what beginners think it will, so consider the documentation they send to you very carefully before buying. Foundation's 80 column card is being revised and is not yet available.

Myarc, Inc. is now producing a disk controller card. It is DS/DD capable and is managed by a TI Disk Manager II (included). Myarc is also offering an RS-232 card that features an extra serial port. Both cards are for the TI Peripheral Box and drawings we've seen indicate the cards may use TI cases (a nice plus - we'll try to confirm this). Myarc also produces full expansion systems, as shown in the Triton catalog. If you are not familiar with Myarc, they have been producing very high quality Winchester hard disks for the TI for several years and have a good reputation. Don't be surprised if you see more products from Myarc!

Many of you have asked us for memory maps. We attempt to avoid duplicating the efforts of other publications. Therefore, for detailed (and we mean excellent and very detailed) memory maps, you should subscribe to The Smart Programmer. It is available from Millers Graphics, 1475 W. Cypress Ave., San Dimas, CA 91773. The firm advises us that all back issues are currently available, as of October 16, 1984, so that you can specify at which issue you would like to begin. If you use other than console BASIC and are at all serious about programming, the MG newsletter is for you!

We have had a number of requests for information about TMS 99000 Family processors (TI-99/4A uses a TMS 9900 processor - not what we are discussing). They use multiplexed 16 bit address and data bus, capable of addressing 1 megabyte. Capable of up to 24 Mhz, internally divided by 4. Minimum cycle time is 167 ns. The processors include multiprocessor interlock signal and multiprocessor support instructions. A "macrostore" (similar to XOP) capability uses high-speed memory and appears very impressive. CRU instructions are expanded from the TMS9900's 4K bits to 32K bits (split as 16K serial and 16K parallel). Double precision 32 bit instructions and signed multiply and divide are included. All TMS9900 instructions are supported. And, there's lots more! All this can be summarized as nothing short of amazing (A whole new world for people into counting benchmarks!)! If a company can build a quality computer at a reasonable price based on this type chip, we'll buy one and we bet you will, too!

TI is still giving away to users groups. An Editor/Assembler Debugger (different from the original E/A debugger) is now available. It allows ->

disassembling, memory dump to hard copy, single-step program execution, and several other nice features. The two debuggers both have good points.

We will not be publishing the article we promised last month on how to double FORTH. The information has already been covered elsewhere. For a copy of the article, send $1.00 to 99'ers Users Group Association, 3535 So. H St., #93, Bakersfield, CA 93304. Ask for the April 1984 issue. The publication is issued about 10 times per year and has contributions by some very highly regarded TI-99/4A programmers. The primary focus of the publication is on product news and current events, though programming tips are a regular feature. The subscription rate is $10.00 per year.

COMING IN DECEMBER: Music and Sound Effects - for your December holiday season.

CorComp, Inc. has announced filing for financial protection under Chapter XI statutes. The firm plans to attempt to attract new investors, pay creditors, and continue to provide third party hardware and software.

CorComp has been in business for about one year. Growth pains are not uncommon for rapidly growing companies such as CorComp. The firm is expected to release several new products soon and Chapter XI status may allow the firm the opportunity to have revenues catch up with research and development costs. The firm's CorComp 9900 Disk Controller Card is widely regarded as one of the best products since the TI-99/4A itself. Our sources tell us that CorComp's chances for full recovery are rather good.

Computer products will be in heavy demand again this Christmas season. However, the products being purchased may be somewhat different from recent years. Expected to be popular this Holiday season are peripherals, as many households now have computers, but are looking to expand the capabilities of their equipment. If you have not yet done your Christmas shopping, you may already be too late for some items and should shop early. Support TI-99/4A third party vendors and give a computer gift this Holiday season!

Extended BASIC may soon be more readily available and/or at a reduced price. Full details are incomplete as of our deadline.

Good luck to Paul Powers, new President of CONNI Users Group in Columbus, Ohio.

->

Super 99 Monthly is published monthly by Bytemaster Computer Services, 171 Mustang Street, Sulphur, LA 70663. Subscription rate in U.S. and possessions is $12.00 per year; all other countries $16.00 U.S. funds for surface mail. All correspondence received will be considered unconditionally assigned for publication and copyright and subject to editing and comments by the editors of Super 99 Monthly. Each contribution to this issue and the issue as a whole Copyright 1984 by Bytemaster Computer Services. All rights reserved. Copying done for other than personal archival or internal reference use without the permission of Bytemaster Computer Services is prohibited. Bytemaster Computer Servies assumes no liability for errors in articles.

STANDARD KEY

1	Computer	A	TI-99/4A
2	Cartridge	A	Extended BASIC
		B	Mini Memory
		C	Editor/Assembler
		D	Multiplan
		E	TI-Writer
3	RS-232	B	TI
4	Disk Drive	A	TI
		B	TEAC 55B
5	Expansion Box	A	TI
6	Disk Controller	A	TI
		B	CorComp
7	32K Card	A	TI
8	Cartridge Unit	A	Navarone Cart. Expander
9	Monitor or TV	A	TV & RF Modulator
10	Printer	A	Gemini 15-X PC

Note: List adjusted monthly.

Bytemaster Computer Services
171 Mustang Street
Sulphur, LA 70663