EAR 99'ER

East Anglia Region User's Group

VOLUME 1 — ISSUE 2 — JUNE '87

Great Britain:
Scott & JoAnn Copeland
13 Elm Walk
Lakenheath
Suffolk, England IP27 9QR

United States:
Scott & JoAnn Copeland
PCS Box 5927
APO New York
09179-5379

Contents:

WELCOME! to EAR 99'er, Volume 1, Issue 2 † † We trust everyone reading this Newsletter finds something useful &/or informative! Our main intent is to support the TI-99/4A and its' Users in any way we can. If you have any contributions, or ideas, please let us know! We look forward to hearing from you!

PLEASE give credit where credit is due, and notate the Club and the Author. Any user assumes sole liability for use and please note any information by authors' does not guarantee it to be error-free.

Now, turn the page for an EAR-full of information.....

The front cover has a request to give author and source if reusing anything in this magazine. The program below has been printed without author or source. The reference (see Issue 5) is a mystery. This item has been traced to COMPUTE Vol 5 No11, November 1983 and the program is by DAVE MILLER. The magazine is available for download from archive.org and includes many notes including code for keyboard operation. 2021 bb

ALPHA-BLAST

(see Issue 5)

```
100 GOSUB 510
110 RANDOMIZE
120 DIM N(3)
130 CALL CLEAR :: CALL SCREEN(16)
140 CALL HCHAR(8,5,120,24):: DISPLAY AT(10,4):"A L P H A -- B L A
S T" :: CALL HCHAR(12,5,120,24)
150 CALL MAGNIFY(2):: FOR L=1 TO 28
160 CALL SPRITE(#L,INT(RND)+65,INT(RND)+3,INT(RND)*8+1,IN
T(RND)*8+1,INT(RND)-30,INT(RND)-30)
170 IF L=25 THEN DISPLAY AT(21,10):"GET READY!"
180 NEXT L :: CALL DELSPRITE(ALL):: CALL CLEAR :: HS=0
190 CALL COLOR(12,6,1)
200 DISPLAY AT(1,6):"HIGH SCORE:";HS :: U=0 :: R=0 :: SC=0
210 U=H+.03*SGN(1-U):: R=R+1 :: DISPLAY AT(5,14):"ROUND #";R :: DI
SPLAY AT(2,6):"SCORE:     ";SC
220 FOR I=6 TO 21 :: CALL HCHAR(I,6,128):: NEXT I
230 FOR I=5 TO 7 STEP 2 :: CALL VCHAR(5,I,95,17):: NEXT I
240 FOR I=3 TO 9 STEP 6 :: CALL VCHAR(4,I,120,20):: NEXT I :: CALL
HCHAR(4,4,120,5):: CALL HCHAR(23,4,120,5)
250 FOR I=0 TO 3
260 N(I)=INT(RND)+65
270 FOR J=0 TO I-1 :: IF N(J)=N(I)THEN 260
280 NEXT J :: NEXT I
290 CALL SPRITE(#6,42,3,97,153)
300 CALL SPRITE(#2,N(0),14,57,153):: CALL SPRITE(#3,N(1),14,97,201
):: CALL SPRITE(#4,N(2),14,137,153):: CALL SPRITE(#5,N(3),14,97,10
5)
310 ROW=21 :: A=-1 :: B=-1 :: C=-1 :: D=-1
320 T=0
330 CALL JOYST(1,X,Y):: IF ABS(X)-ABS(Y)<>4 THEN CALL HCHAR(ROW,6,
32):: ROW=ROW-U :: IF ROW <5 THEN 400 ELSE 330
340 IF (X=0)*(Y=4)*(A)THEN CALL PATTERN(#2,32,#6,43):: V(T)=0 :: A
=0 :: GOTO 390
350 IF (X=4)*(Y=0)*(B)THEN CALL PATTERN(#3,32,#6,43):: V(T)=1 :: B
=0 :: GOTO 390
360 IF (X=0)*(Y=-4)*(C)THEN CALL PATTERN(#4,32,#6,43):: V(T)=2 ::
C=0 :: GOTO 390
370 IF (X=-4)*(Y=0)*(D)THEN CALL PATTERN(#5,32,#6,43):: V(T)=3 ::
D=0 :: GOTO 390
380 CALL HCHAR(ROW,6,32):: ROW=ROW-U :: IF ROW<5 THEN 400 ELSE 330
390 CALL SOUND(-10,200,2):: CALL PATTERN(#6,42):: T=T+1 :: IF T=4
THEN 450 ELSE 330
400 DISPLAY AT(22,11):"YOUR TIME IS UP !"
410 CALL SOUND(800,110,5,120,5):: FOR I=1 TO 200 :: NEXT I
420 DISPLAY AT(24,10):"PLAY AGAIN(Y/N)?" :: IF SC>HS THEN HS=SC
430 CALL KEY(0,KEY,ST):: IF ST=0 THEN 430
440 IF (KEY=89)+(KEY=121)THEN CALL CLEAR :: CALL DELSPRITE(ALL)::
GOTO 200 ELSE 560
450 REM
460 FOR T=0 TO 2 :: IF N(V(T))<N(V(T+1))THEN 480
470 SC=SC-INT(1.5*R*ROW):: GOTO 490
480 SC=SC+INT(R*ROW)
490 NEXT T
500 CALL DELSPRITE(ALL):: GOTO 210
510 REM
```

```
520 CALL COLOR(14,9,1)
530 CALL CHAR(120,"007E7E7E7E7E7E00"):: CALL CHAR(128,"")
540 CALL COLOR(12,6,10):: CALL COLOR(13,1,9)
550 RETURN
560 CALL CHARSET :: CALL CLEAR :: CALL DELSPRITE(ALL):: CALL SCREE
N(16)
```

STRANGE FIGURES

So how accurate is it? Huh?? When all is said and done computers are really nothing more than elaborate, expensive, number manipulators. All of our programming and visual results on the screen are nothing more than the results of "number crunching" (the very fast switching of ones and zeroes).

Creative Computing Magazine gave the results of 170 tests of a benchmark program involving 140 different computers: mainframes, mini's, micros and one TI SR-50 calculator. These tests were designed to determine the speed, accuracy, and ability of the random number generator.

The results are "strange figures":

Speed: fastest was the Cray 1 in 0.1 seconds, the slowest was the TI SR-50 in 12.7 days. Five computers were under 1 second, 58 under 1 minute, 39 between 1-2 minutes, 15 in 2-4 minutes, and 23 over 4 minutes. The TI-99/4A finished in 3 minutes and 46 seconds.

Accuracy: the best came in at .0000000000160298 and the worst was .32959. The TI-99/4A had an error of only .00000011 (only 22 computers were better and none were the large mainframe types).

Random: the TI-99/4A ranked 5th with a 2.7 - remember these rankings are against 140 different computers, including the Cray 1, IBM mainframes, DEC Vax's, etc.

Let's compare the TI with the "home" computers (remember, the smaller the number the more accurate the computer):

COMPUTER	ACCURACY	RANDOM
TI-99/4A	.00000011	2.7
Timex Sinclair	.0041294098	8.7
Coleco Adam	.000426292419	6.2
RS Color Computer	.000596284867	7.3
Commodore 64	.0010414235	8.9
VIC 20	.0010414235	23.7
Apple IIe	.0010414235	12.0
DEC Rainbow 100	.005859375	7.2
IBM PC	.01159668	6.3
Atari 400/800	.012959	23.8
TRS-80 Model III	.0338745	5.8
Heath/Zenith H-98A	.187805	7.4
TI SR-50 (12.7 days later)	.193704289	16.4

◆◆ PROGRAMMING TIPS ◆◆

1◆ Never comment your code; it only helps people to find redundancys in it.

2◆ Remember - if it ain't a game, it ain't worth programming. (Unless someone pays you.)

3◆ Never work on a program for more than four hours. If it takes that long, save it on a disk, place it in your disk box (preferably at the back) and forget about it. Plan on finishing it between now and the death of your great grandchildren.

4◆ Never use structure programming techniques. All programs should look sloppy, confusing, and completely unreadable. When programming in BASIC, every third statement should be a GOTO. This, when others look at your listings, will make you look like a genius. (Which, in most cases, is a sharp contrast to reality!)

5◆ Never organize your disks. Remember - a tidy disk box is the sign of an exceptionally weak mind.

6◆ Debugging is to programs as decaffeinating is to coffee! Never keep your source code after you've compiled it. Only an idiot puts out later "bug-free" versions.

7◆ Always use global variables. Programmers have enough problems without passing parameters.

8◆ Never make flowcharts or pseudocode. A program should come from the top (or in some cases the bottom) of your head. It should never do what was planned for it and should seldom, if ever, do what it does completely correctly.

9◆ Try not to use descriptive variable names. In languages that require you to predefine your variables, predefine all possible combinations of letters and save it as a template for all your programs. This allows you to use whatever variables you want without having to go back and insert.

10◆ Never spill Coke on your keyboard more than once a week.

11◆ Only use disks as coasters under cold drinks.

12◆ Always have an excuse if things don't work properly. Some good examples are: "It's not supposed to do that!" "The instructions were unclear/unreadable/completely wrong!" And of course, "This computer is broke!"

XX◆ Never use the number 13 in your programs; it's unlucky. Define it as a constant, such as "XX".

14◆ Do not hit your computer with objects like wooden baseball bats. Use aluminum ones - you'll have less problems with static.

15◆ Never, but never, read the instructions. If it doesn't work, plug it in, fix it, or bang it against the wall.

PS: By the way, if your computing teacher says anything contrary to these basic rules, calmly point it out to them. If they still disagree, then ask them why they make their living teaching instead of programming, if they're so smart!

SPACESTATION PHETA

A challenging game of skill where you explore a mysterious abandoned Spacestation. Written in Wycove Forth this game includes graphics; color; sound; realistic animation; 79 built-in screens; create your own screen designs; built-in solutions for boards you get stuck on; joystick or keyboard control; game speed control; and more.

Thus begins an introduction to Spacestation Pheta! I'll admit now that I spent many hours on this and prove I'm as much a kid as anyone else! (High score to date 151,850 and practicing! - and I scored this without a FCTN Z!) If I had seen this in a store before purchasing it, I still would have brought it home!

Spacestation Pheta requires 32K Memory Expansion, Disk Controller and Disk Drive. Use either Extended Basic, Editor/Assembler, Mini-Memory, or TI-Writer to load. After loading, you find a cute little guy standing on a platform!

You start with 4 spacemen and gain extras at 10000, 20000, 40000, etc. The object of the game being to exit through the appropriate doors farthest right on the screen to gain entrance to the next level. Not as easy as it states, though. You carry one key at a time to find out which door it opens. If it happens to open the wrong door - you're stuck! You pick up oxygen to help make it through the screen in time (if you run out of oxygen you have to do the whole screen again!) and energy packs and keys award points. Bonus points are gained depending on the amount of time left when you finish a screen.

So, here you are running around collecting a key to see what door it opens... you can jump 1 level without killing yourself, and pick up oxygen and energy packs. You need to climb stairs; jump ledges; keep from falling off of temporary floors; go down chutes (WHEEEE!); go through materializers and transporters; get spit out of cannons (UP, UP, UP WE GO!!!!); use falling floors to make a pit-stop; use anti-gravity fields; use booters to fly across the screen; walk on conveyor belts; get fried on an electrical floor; find the key to the secret ladders; and walk on top of copyright symbols. Yes, folks, and more!

On top of the 79 built-in screens (level 80 starts off on 1 again), you can build your own screens with your own solutions and problem solving! Can slightly be compared to Gravity Master - but this offers SO much more your imagination is left open! You can also start off on different levels (for practice maybe?) and give yourself extra men any time you want by pressing FCTN Z! This comes in helpful more than once, but it takes any High Score achievement away from you...so use it only if you have to. Three years ago this went on the drawing board, and at the time Tad (the author) was 13 years old. Released three years later after being refined, it was worth the wait! Spacestation Pheta is interesting, challenging, and FUN!

CONTROL U
VS.
OTHER COMMANDS

We're going to investigate Margin and Page Length Commands this article as we continue our examination of CTRL U Special Command Mode vs. programming &/or transliteration. As I stated last article, each way works - it just depends on what mode the User is most accustomed to using, and what is more comfortable for the User. Let's see what we can do with a Left Margin Command in a program:

```
10 OPEN #1:"PIO"          ↓
20 ESC$=CHR$(27)          ↓
25 LH$=CHR$(77)           ↓
30 PRINT #1:ESC$;LH$;CHR$(15)
40 PRINT #1:"THIS IS A TEST"
50 CLOSE #1
60 END
```

Line 30, where it states CHR$(15) informs the printer you want a left margin at 15. If you were to enter 35, your left margin would be at 35, entering 20 gives a left margin of 20, etc. So, typing this program in Extended Basic with the printer on, you type RUN (Enter) and the line THIS IS A TEST types out with your left margin set at the number you entered. The printer holds the command until turned off.

What if you needed to change the left margin several times? Needless to say, we wouldn't want to type out several changes to keep running the program. Another way is using the Margin Commands in the TI-Writer Manual (covered in previous articles). Such a use would be:

eg: .FI;AD;LM 10;RM 65 where the left margin is at 10. Several lines down in your text file you could change this to:

eg: .FI;AD;LM 5;RM 65 where your left margin is now at 5. Typing your text through the Formatter would then make these changes. CTRL U can do this also.

'ьMn

FCTN R (in CTRL U Mode) followed by a Capital M and a value sets your Left Margin (n representing a numerical value). If you wanted your Left Margin at 5 the command would be:

'ьM'ᴇ

(FCTN R in CTRL U mode, a Capital M (out of CTRL U mode), followed by Shift E in CTRL U mode).

Likewise, if you wanted to change the Left Margin to 30, the command would be:

'ьM_

(FCTN R in CTRL U mode, a Capital M (out of CTRL U mode), followed by Shift 6 in CTRL U mode).

In each case, the Left Margin is set and the command will be held until another command is encountered changing it.

If you wanted a Left Margin of 40, use Character Code 40 which is the (sign. The ! would give you a left margin of 33. If you follow your printer manual, you can change your left margin to any desired number.

The same rule applies for the Right Margin. You can change this margin with the TI-Writer Command:

eg: .FI;AD;LM 10;RM 65 where Right Margin is 65. To change it to 70 just type in the following:

eg: .FI;AD;LM 10;RM 70. The Formatter will recognize these commands and your text will print out as such. Again, CTRL U mode looks like:

'bQn (where n represents a numerical value).

Instead of using a Capital M for the Left Margin, we now use a Capital Q for the Right Margin. Again, n represents a numerical value and your printer manual helps with the character codes:

'bQF

(F = character code 70, or Right Margin at 70). Replace F with K for a Right Margin of 75, as K equals character code 75, etc. Easy, huh?

Now, on to Form Length... Most of us have had the occasion to acquire paper, whether perforated computer paper or singularly fed sheets, that was another length besides the 66 lines (normal 8 1/2" X 11" size). A problem with the Formatter is that it will page after the 66 lines. Ever told the Formatter via your text to change that? .PL 70 should advise the Formatter to page at 70 lines per page. Every time I did this, it would print a sheet, and then wildly skip a sheet to start printing again - only off 1/4 inch. As each page went on, that 1/4 inch grew to where it was printing half on one page and halfway into the next page! Investigating the CTRL U commands a group of us found the command: 'bC

'bCF

Again, FCTN R in CTRL U mode, followed by a Capital C (out of CTRL U mode) changes the Form Length to *n lines.* Here, character code 70 (F) sets the form length to 70 lines per page. Changing the F to K sets form length to 75 lines per page. Change K to P for 80 lines per page, etc. You'll now find the Formatter doesn't skip or otherwise mess up on paging!

You can also change Form Length to *n inches,* instead of n lines. This command would be:

'bCO

'bCOcr

(FCTN R in CTRL U Mode, Exit CTRL U Mode, Capital C, Zero, Enter CTRL U Mode for Shift M) advises the printer to form length 13 inches per page, rather than the normal 11. This can be changed to however many inches you require by typing: FCTN R in CTRL U mode, exit CTRL U mode for a Capital C, a Zero, and then the character that represents the number of inches you want per page. I usually use the *n lines* command as it is easier representing character codes in that command. Following my printer manual I check out the Standard ASCII character set page and go from there. You'll find it is easier using higher numbers for *n lines* rather than *n inches*.

Little bits left to cover involve Line Feed Length, Reset Commands, and Buzzer complaints (yes - buzzer). Then maybe all you lucky people will finally be able to get rid of me! Until then, happy word processing! .....

C
T
R
L

U

G
R
A
P
H
I
C
S

E N D

[STARCROSS is a text based adventure game sold on disk by INFOCOM]-2021 bb

STARCROSS

TO: All Spacecraft Owners and Operators
FROM: Bureau of Extra-Solar Intelligence
SUBJECT: Encounters with Aliens
DATE: June 1, 2130

We understand you are having difficulty in obtaining your destination. **SHAME!** All you need are the three coordinates! If you examine your mass detector, read it, look at screen, your display shows the appropriate MASS UM and number. EASY! You'll need a code designation and appropriate R, Theta, and Phi coordinates. After you have your designated UM number, such as UM70, type the following:

```
>Computer, R is 100.Theta is 120.Phi is 101
>Computer, confirm new course.
```

Or, for UM 31:

```
>Computer, R is 150.Theta is 105.Phi is 067
>Computer, confirm new course.
```

If you get a response from your shipboard computer saying "There's nothing out there" - you got it wrong!

Once your ship lands (and keep the seat belt on until you do!) exit your ship (with your space suit on). Here you begin your journey to find the following rods (yes, rods):

Black (arrive at Red Air Lock), Red (Rat-Ant Nest), Yellow (Blue Spherical Ship), Silver (Weapons Deck), Gold (Computer), Pink (Among Debris at Yellow Dock), Blue (Laboratory), Green (Maintenance Garage), Brown (Village Center), Violet (Control Room, Green Dock), Clear (Observatory), White (Drive Bubble).

Some of these will be put into appropriate items and some will help you get into appropriate locations. (Huh?) You should also find the Red Hall, Blue Hall, Yellow Hall, and Green Hall, (and the Warren).

Each Rod is 25 Points. To equal 400, you also need to get through the Red Airlock; Enter the Control Bubble (each for 25 Points); and Return to Earth (for 50 Points)! And you don't return the way you came either...

After you exit your ship, you find yourself at the Red Air Lock area. You have to push an appropriate column (number) which in turn makes something happen. Try pushing again and see what happens. Go up.

The Yellow Rod may help you gain Emergency Lighting if you put it in the appropriate place.

The Red Rod can give you a fresh air supply. Go to the Repair Room for an appropriate slot.

The Chief helps you get a Brown Rod. He's kinda' greedy, though. He also helps you get to the Warren if you know what to do!

That funny looking Ceramic Square you find is not all it seems. There's a panel just waiting for it somewhere!

When you fix the computer, you need to see a *READY* sign, or you didn't do it correctly. Likely to die of coal gas poisoning while you're at it...

The projector is likely to blind you (take my word for it). Try looking at it through something...

The Rat-Ant Nest is easier than you think. Instead of getting bitten, try throwing something at it!

Did you know the Blue and Red Disks are transporters? No? Well, they are...

Your basket helps with the sphere. So do the transporters.

That cute little mouse is quite helpful. Letting him pick up something you thought you needed might help.

That stinky old skeleton is better than you thought. Most card players hold something up their sleeve...Wonder if this guy ever played cards?

WHEEEEE! Remember that in most adventures? Helps again here. Try "JUMP".

Don't shoot the gun where it isn't needed. And, remember, interplanetary rules state "do not kill" another human being (or alien). Think this might help out at the Drive Bubble? No? Okay...

Another pentagon? With five slots and five rods left? Hmmm... the slots are colored and so are the rods. A display shows nearby space until you press one of the spots (yes, press a spot). Where were we headed, anyway?

Now, is it a parabola looping around Earth? A line terminating in the Center of Earth? A line terminating in an ellipse surrounding Earth? Or a line terminating in a circle around Earth? Your guess is as good as mine... By the way, SAVE before you do anything further...

If you complete this correctly you should see the remarks:
"Congratulations, you who have passed our test. You have succeeded where others failed. Your race shall benefit thereby."
"I expect to see you in person, someday."

Your score would be 400 (total of 400 points), in ??? moves. This score gives you the rank of Galactic_Overlord.

Good luck and Happy Hunting!

PROGRAMMING TECHNIQUES

by: Scott Copeland

Did you know that you can load an assembly language program into the console memory or the memory expansion unit directly from Extended Basic? You can use EXBASIC's Call Load subroutine to load the values to be written to the CPU RAM or the Memory Expansion Unit. Letting ExBasic poke the assembly code values into memory will allow the user to load an assembly based program with his ExBasic program, using normal ExBasic commands. You are probably wondering how this can be done. Well, let's look at it.

An assembly language program, when assembled, is a list of binary values that are expressed in a hexidecimal notation form. These hexidecimal codes can be loaded directly into memory from ExBasic. Let's look at an assembly language instruction: B *R11 is translated to hexidecimal as : >045B but when translating to ExBasic it is equal to two values. The first value >04 is = to 4 and the second value >5B = 91. So if you load a memory address with the values 4 and 91, you would in reality, be entering B *R11. From an ExBasic standpoint it would look something like this::

CALL LOAD(xxxxx,4,91)(will place the instruction B *R11 at memory address xxxxx)

Most ExBasic programs that place an assembly language program in memory are called ExBasic loaders.

To load a complete assembly language program directly into memory from ExBasic, you will be required to load its decimal equivalents one byte at a time. To get the necessary decimal equivalents you should look at an assembled source listing and convert the hex codes to decimal. The next page gives a listing.

>>>>>>>>>>>>>>>>>>>>>>>>>>>M>O>R>E>>>>>>>>>>>>>>>>>>>>>>>>>>>

```
LINE MEM  HEX  LABEL  OP   OPERAND     COMMENTS
NUM  ADDR CODE        CODE


0001                  DEF TRYIT       (Put the word TRYIT into
                                      the REF/DEF table)
0002            TRYIT                 (The start of the program)
0003 0000 0200        LI     R0,298   (Loads register 0 with a
     0002 012A                        value of decimal 298,
                                      which will be the
                                      position on the screen
                                      where text is displayed)
0004 0004 0201        LI     R1,CD    (Loads register 1 with
     0006 0016                        the memory address where
                                      the text is to be dis-
                                      played is located.)
0005 0008 0202        LI     R2,12    (Loads register 2 with
     000A 000C                        the length of the text
                                      to be displayed.)
0006 000C 0420        BLWP   @>2024   (Loads the subroutine
     000E 2024                        that will display the
                                      text on the screen.)
0007 0010 04E0        CLR    @>837C   (This will clear the
     0012 837C                        status byte so as to
                                      avoid false errors upon
                                      return to ExBasic.)
0008 0014 045B        B      *R11     (This will return you to
                                      ExBasic upon completion
                                      of the assembly program.)
0009 0016 A8A5  CD    DATA   >A8A5,>ACAC,>AF80,>B4A8,>A5B2,>A580
     0018 ACAC
     001A AF80                        (This is the text to be
     001C B4A8                        displayed. It is added
     001E A5B2                        with the DATA directive
     0020 A580                        because the screen bias
                                      of >60 has to be added
                                      to the ASCII code of each
                                      character.)
0010 2004             AORG   >2004    (This updates the LFAL (Last
0011 2004 3FF8        DATA   >3FF8    Free Address in Low Memory),
                                      and puts the address of the
                                      REF/DEF table here).
0012 3FF8             AORG   >3FF8    (Start of the REF/DEF Table
                                      where we have the name and
                                      starting address of the
                                      program.)
0013 3FF8   53        TEXT   'TRYIT'  (The name of the program.)
0014 3FFE 3F86        DATA   >3F86    (The starting address.)
0015                  END             (Closing statement.)
```

The fields are broken down into several areas, the first being the line numbers, the second being the memory address to be loaded, the third is the hex code which will be loaded, the fourth being the op code or directive, the fifth being the operand field, and the sixth being the comments field. The area that we will be discussing is the third area. This area is the data that is actually placed into memory during program application.

Now looking at the Hex Code Area the first hex code is >0200. To convert the hex code to decimal we must seperate the MSB(most significant byte) and the LSB(least significant byte). Look at the first code:

HEX CODE	MSB	LSB
>0200	02	00

Now we have two hex codes to convert. The first code >02 equals a value of 2 in decimal, and the second >00 equals 0 in decimal. Look at the next hex code in the listing - >012A. The MSB = 01 which equals 1 in decimal and the LSB = 2A which equals 42 in decimal. Now let's take it from the top. We will start with the first hex code and work our way down to hex >A580 as this is the bulk of our program.

HEX			DECIMAL	EQUIVALENTS
CODE	MSB	LSB	MSB	LSB
0200	02	00	2	0
012A	01	2A	1	42
0201	02	01	2	1
3F9C	3F	9C	63	156
0202	02	02	2	2
000C	00	00	0	12
0420	04	20	4	32
2024	20	24	32	36
04E0	04	E0	4	224
837C	83	7C	131	124
045B	04	5B	4	91
A8A5	A8	A5	168	165
ACAC	AC	AC	172	172
AF80	AF	80	175	128
B4A8	B4	A8	180	168
A5B2	A5	B2	165	178
A580	A5	80	165	128

Now let's look at line 10. It has a memory address of >2004 but so does line 11. The reason for this is because this is where the LFAL(Last Free Address in Low memory) is loaded. Line 10 is used to establish this memory address and line 11 is used to place the hex code of the LFAL.

Both the memory address and the hex code need to be converted. First the memory address is converted, it must be converted as is, not as MSB and LSB. So the Hex Code is converted as follows:

HEX CODE	DECIMAL EQUIVALENT
2004	8196

Let's see how it was derived. We must convert this number starting from the right, let's look at it this way:

```
    HEX CODE

    2    0    0    4
                  x1
  x16  x16  x16  ---
  x16  x16  ---
  x16  ---
  ---
 8192 + 0 + 0 + 4 = 8196
```

The memory address is equal to 8196, now we find out what values o load into it. The hex code is >3FF8 which converts to a decimal of 3 and 248.

Now on to the REF/DEF table. The above table starts at HEX >3FF8 nd converts to a decimal value of 16376. This is the memory address o load the ASCII decimal values of the word TRYIT.

On the last line the memory address 3FFE equates to a value of 16382 and the hex code 3F86 equates to the values 63 and 134.

Now let's look at it in an ExBasic program format. First we will chose an address to load these decimal values into, I have chosen 3F86, which equates to a decimal value of 16262:

```
130 CALL LOAD(16262,2,0,1,42)
(this line equates to LI R0,298)

140 CALL LOAD(16266,2,1,63,156)
(this line equates to LI R1,CD)

150 CALL LOAD(16270,2,2,0,12)
(this line equates to LI R2,12)
```

Four lines start slightly off-page in the original document 2021 bb

```
160 CALL LOAD(16274,4,32,32,36)
(this line equates to BLWP @>2024)

170 CALL LOAD(16278,4,224,131,124)
(this line equates to CLR @>837C)

180 CALL LOAD(16282,4,91)
(this line equates to B *R11)

190 CALL LOAD(16284,168,165,172,172,175,128,180,168,165,178
,165,128)
(this line equates to CD DATA >A8A5,>ACAC,>AF80,>B4A8,>A5B2
,>A580)

200 CALL LOAD(16376,84,82,89,73,84,32,63,134)
(this line equates to TEXT 'TRYIT' and DATA >3F86)

210 CALL LOAD(8196,63,248)
(this line equates to DATA >3FF8)

220 CALL LINK("TRYIT")
(this calls the program name from the REF/DEF table.)

230 END
```

Now we add three additional lines at the beginning of the program:

```
100 CALL CLEAR
(this clears the screen.)

110 CALL INIT
(this initializes the memory space.)

120 CALL SCREEN(11)
(this changes the screen color to dark yellow.)
```

Now Load up the program and run it. When run, it should display the words HELLO THERE in the middle of the screen. If it does not work please check your program for errors and try again.

I have tried to make this article as easy to understand as possible. If you should have any questions about the article you may write or call me for assistance. My home address is 13 Elm Walk, Lakenheath, Suffolk England, IP27 9QR. Or phone Eriswell (063881) 3457.

>>>>>>>>>>>>>>>>>>>>>>>>>>E>N>D>>>>>>>>>>>>>>>>>>>>>>>>>>>>

ADDITIONAL COMMENTS:

This month we welcome a Newsletter Exchange with the following User's Groups - WELCOME ABOARD!

1† LA 99ers Computer Group
TopIcs
Gardena, California

2† New Hampshire 99'ers UG
Manchester
New Hampshire

3† Delaware Valley UG
Wilmington
Delaware

4† Winnipeg 99/4 Users Group
Winnipeg, Manitoba
Canada

5† TI-D-BITS
Camden, New Jersey

6† P.A.T.I.U.G.
Philadelphia, Pennsylvania

Last month we listed some Stateside companies for purchasing/acquiring software/hardware for the TI - this month we're listing companies in Great Britain. I haven't dealt personally with each of these, but those that I have dealt with are highly recommended and marked with a (†) sign:

†† 4-Front
New Day Computing
17 Jerrard Close
Honiton, Devon EX14 8EF

Stainless Software
10 Alstone Road
Stockport Cheshire SK4 5AH

Parco Electrics
1 Manor Close
Weston, Honiton
Devon EX14 OPE
(0404) 44425

†† Arcade Hardware (closing out)
211 Horton Road
Fallowfield,
Manchester M14 7QE
(061) 225 2248

See PAGE 18 for an Advertisement regarding 4-Front - Newsletter and Magazine ON DISK! Exceptional quality!!

2021- No page 18. Probably a two sided flyer
for product enclosed loose with the newsletter.

ENDING NOTES:

Well - our second issue is done - We still trust most of you found something interesting in this issue! We're trying!! And, Yes, you did receive an original printed copy last issue! Due to production costs, at this time it is inconceivable to pay the amount for copying when EAR 99'er can be printed by möi! With more subscriptions coming in every day, we look forward to shortly sending this to the printers! (My personal THANKS to all the subscribers!)

†† For interested parties ††

Issues are monthly with a subscription price of £10.00 or U.S.A./equivalent. If you are inside the Great Britain area please use the Suffolk, England address. In the United States, you can use the APO New York address. If you have any questions † Or Contributions † please write in! Don't worry about any particular format - we'll worry about that for you!

Thank you for all the support we're still getting! See you next issue!

Contributions & Items used:

Programming Tips †by Jim Beck†
Edmonton 99'er Computer User's Society (April '87)

Strange Figures †by Keith G. Koch†
New Hampshire 99'ers (April 1987)

THANK YOU!

Have something you want to sell? Looking for a particular item? Why not use this space to advertise? It doesn't cost a thing - and you just might come out better for it! Just write in, or call, and let us know the particulars!

As these pages were printed both sides, page 18 would have been on the back of this sheet- which was actually blank, so no page 18. 2010. bb