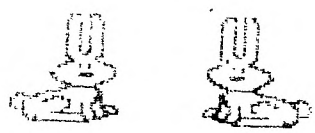
VOL 5–2 NOVEMBER 1986 $2.50

CLUBLINE

AIR WOLF PAGE 14

SIX SUPER PROGRAMS



MORE BIGGEST PROGRAM PAGE 4

"WHO'S-WHO"

CLUBLINE-99 Magazine

EDITOR in CHIEF
Malcolm W. Johnson

ASSOICIATE EDITOR & RESEARCH
Stephen W. Johnson

RESEARCH & LAYOUT
Andrew Johnson B.A.

FAST LANE EDITOR
Iain M. Johnson

EDITORIAL ASSISTANT
Jean Johnson

BASIC EDITOR
Tor Hansen

ADVERTISING MANAGER
Mike Towers

ILLUSTRATIONS
Tom Phalen

COMPUTER GRAPHICS
Stephen W. Johnson

CLUBLINE-99 Magazine is published 12 times a year by Wentworth Supplies. Any and all written communication should be addressed to:

Clubline-99
P.O. Box 1005, Station A
Hamilton,Ontario
L8N 3R1

CONTENTS

VOL 5-2 NOVEMBER 1986

NEXT MONTH

BUILD YOUR OWN 32K RAM

X→stream

EDITED BY Tor Hansen

HIGH-LEVEL LANGUAGE ENCOUNTERS

HOW ONE USES [illegible] COMPUTER

By Frank Butler

This month [illegible] the program [illegible] help me [illegible]

After I run these programs, I take all the [illegible] from all the utilities and combine the data in one program I wrote that handles all the information and formats it for print out.

But first, a look at the Hydro utility.

Before I save any of the programs I write, I first run that program, and, as you may have noticed from previous samples, either the first or second line of the program contains the name I want to use for the disk filename. This has saved this writer from losing many program files!

In a quick summary, line 190 to line 710 cover the screen display option on the menu as well as the program logic to format the numbers for display.

Lines 750 to 820 open the Printer file, then control is sent back to line [illegible] to allow the use [illegible] the program logic.

After the program logic has executed, control is directed according to the value of F$. The information is sent to the output device where the program then ends after the print out.

As these are fairly simple routines, feel free to do what you will with them, and adapt them any way you like. [illegible] be printed, and [illegible]

(I am including one program file per month for this segment of this column. The programs are indeed fairly easy to follow and adapt. Let's see some ingenuity from the membership that [illegible] the time [illegible] read this column.

In lieu of saying start from scratch and show me something, I am saying here is a starting point [illegible]

Is there anyone out there up to it? Ed.)

```
100 CALL CLEAR
110 DISPLAY AT(6,8):"DSK1.HY
DRO"
120 DISPLAY AT(7,8):"-------
[illegible]
130 DISPLAY AT(8,5):"1 = SCR
EEN"
140 DISPLAY AT(9,8):"2 = PRI
NTER"
150 PRINT "LIST # from above
"
160 INPUT F$
170 IF F$="1" THEN 190
180 IF F$="2" THEN 750
190 PRINT "[illegible] ONTARIO HY
DRO [illegible]"
200 PRINT
210 PRINT "DATE";TAB(8);"#";
TAB(12);"Read";TAB(18);"Kwh"
;TAB(24);"COST"
220 PRINT TAB(7);"Days";TAB(
12);"-ing";TAB(18);"used"
230 FOR X=1 TO 4
240 REM DATE.....=D$
250 REM Days.....=E
260 REM Reading..=R
270 REM Kwh......=K
280 REM Cost.....=C
290 READ D$
300 READ E
310 READ R
320 READ K
330 READ C
340 IF F$="2" THEN 840
350 PRINT D$;TAB(7);E;TAB(11
);R;TAB(17);K;TAB(23);C
[illegible]
[illegible]
370 [illegible]WH=TKWH+K
380 REM TOTAL cost last 4 re
adings=TC
390 TC=TC+C
400 REM Total Days=TE
410 TE=TE+E
420 REM Kwh per day=KD
430 KD=TKWH/TE
440 [illegible]=INT(KD*100+.5)/100
450 REM est ANNUAL Kwh=AKWA
460 AKWA=KD*366
470 REM EST. Yearly Cost=YC
480 REM est.Daily cost=DC
490 DC=YC/365
500 DC=INT(DC*1000+.5)/1000
510 YA=YA+C
520 YB=YA/TE
```


Continued on page 8

THE BIGGEST PROGRAM IN THE WORLD

by Stephen W. Johnson

If you have sent in a program and have not seen it published yet, don't panic! We have a back-log of programs which will all eventually be published.

Here are another two excellent short programs. But first, if you remember last month we added a line to the Score Card program to return the colours and characters back to normal. Well there is one other thing that should also be returned to normal, the sprite magnification. Simply add a call magnify and a delsprite at the end of that line so it looks like this.

```
175 CALL CHARSET :: FOR L=0
TO 14 :: CALL COLOR(L,2,1)::
 NEXT L :: CALL DELSPRITE(AL
L):: CALL MAGNIFY(1)
```

Also, do not forget to change line 350 to load the new programs. Change it from:

```
350 X=INT(RND*3+1)
```

to:

```
350 X=INT(RND*5+1)
```

GAME/5

Here is a fun little game that everyone should recognize, Frogger. In this version you are some bits of information that have to get to the CPU. But watch out, you have to avoid the Power Surges and then make your way across the chips. Once you successfully cross you'll continue onto the next board.

Many people have given up trying to write Frogger in Extended because they can not get their man to move with the Logs. There really is a simple solution over this problem, so simple that most people over look it. The speed of the logs in each row are put into an array. Then to find out what horizontal speed the man should have, you find what row he is in and then look in the corresponding element of the array.

Although the man moves with the chips (logs) in this version, he will slowly slide off them. This can be easily changed by multiplying the speed by 1.8 in line 400. Then the line would like this:

```
400 IF L<5 THEN XSP(L)=SPD*1
.8
```

You will also notice that it only has one game loop for both the top and the bottom of the screen. A clever little memory saver.

For all those experimenters out there, you might want to modify the program so that it uses magnification #4. This will allow you to make the logs longer. They are a bit short with magnification #3.

If you want to play this game without a disk drive simply rem all the lines that have something to do with one. They are 150 to 170 and 540 to 570. You will also have to add a line to give you some men.

```
145 men=10
```

```
100 REM GAME BOARD #5
110 REM BIT CROSS
120 REM BY STEPHEN W.JOHNSON
130 REM VOL 5-2 NOV 1986
140 REM FOR THE BIGGEST PROG
RAM IN THE WORLD
150 OPEN #1:"DSK1.SCORE/REC"
160 INPUT #1:HSE,SCORE,MEN,N
AME$
170 CLOSE #1
180 DIM XSP(12)
190 CALL CHAR(120,"92AA92004
AAA4A00"&RPT$("0",48))! YOUR
 MAN
200 CALL CHAR(124,"88888884
422118844221188442211884422
18844444444")!MIDDLE OF SCRE
EN
210 CALL CHAR(128,"00006D49F
F88DADADADBFFFF496D00000000B
624FE22AEA2BAA2FEFE24B60000"
)! CHIP TMS
220 CALL CHAR(132,"00006D49F
F8BBB8AEA88FFFF496D00000000B
624FEBABAAAAA22FEFE24B60000"
)! CHIP SWJ
230 CALL CHAR(136,RPT$("55AA
",16))!BOTTOM THING
240 CALL CHAR(140,RPT$("55AA
",16))!BOTTOM THING 2
250 CALL CLEAR :: RANDOMIZE
260 FOR L=2 TO 11 :: CALL CO
LOR(L,2,10):: NEXT L
270 CALL COLOR(1,2,10,12,4,1
3):: CALL SCREEN(7)
280 CALL MAGNIFY(3)
290 HRD=HRD+1
300 FOR L=1 TO 6 :: READ X,Y
 :: CALL HCHAR(X,1,Y,32):: N
EXT L
310 DATA 1,125,2,126,11,124,
12,126,21,124,22,125
320 CALL HCHAR(13,1,121,256)
330 FOR L=1 TO 8
```

```
340 SPD=INT(RND*3+7):: IF L
5 THEN SPD=SPD-3
350 IF L/2=INT(L/2)THEN SPD=
-SPD
360 FOR L2=1 TO 3
370 DFN=128+INT(RND*2)*4 ::
IF L<5 THEN CL=2 :: INC=1 EL
SE DFN=DFN+8 :: CL=16 :: INC
=17
380 CALL SPRITE(#L*3+L2,DFN,
CL,L*16+INC,L2*85,0,SPD)
390 NEXT L2
400 IF L<5 THEN XSP(L)=SPD
410 NEXT L
420 REM START OF GAME
430 TRY=TRY+1 :: GOSUB 590 :
: CALL SPRITE(#1,120,16,169,
128)
440 CALL JOYST(1,X,Y):: CALL
 MOTION(#1,-Y*13,XSP(INT((YP
+3)/16))+X*13)
450 CALL MOTION(#1,0,0):: CA
LL COINC(ALL,C):: CALL POSIT
ION(#1,YP,XP):: CALL MOTION(
#1,0,XSP(INT((YP+3)/16))):: 
IF YP<77 THEN C=-C-1 :: IF Y
P<13 THEN 500
460 IF C=0 THEN 440
470 MEN=MEN-1
480 IF MEN>0 THEN 430
490 DISPLAY AT(5,3)BEEP:"SOR
RY, YOU HAVE LOST ALL";TAB(3
);"YOUR MEN"(1)"
500 CALL DELSPRITE(ALL):: GO
SUB 590 :: DISPLAY AT(8,1)BE
EP:" HOLD ON FOR THE SCORE C
ARD"
510 IF MEN>0 THEN SCR=110-TR
Y*10 :: IF SCR<10 THEN SCR=1
0
520 SCORE=SCORE+SCR
530 CALL SOUND(-1,110,30)
540 OPEN #1:"DSK1.SCORE/REC"
550 PRINT #1:HSE:SCORE:MEN:N
AME$
560 CLOSE #1
570 RUN "DSK1.SCCARD"
580 STOP
590 DISPLAY AT(24,1)BEEP:"ME
N";MEN;"SCORE";SCORE;"GO#";T
RY :: RETURN
600 END
```

GAME/6

Here is a very clever program that is great fun. You have to destroy five tanks by hitting them with artillery. You lose a man every time a tank reaches you.

If you want to play this game without a disk drive simply rem all the lines that have something to do with one. They are 150 to 170 and 810 to 840. You will also have to add two lines to give you men and display the score.

```
145 MEN=10
805 DISPLAY AT(11,1):"SCORE
";SCORE
```

```
100 REM GAME BOARD #6
110 REM TANK
120 REM BY MIKE TOWERS
130 REM VOL 5-2 NOV 1986
140 REM FOR THE BIGGEST PROG
RAM IN THE WORLD
150 OPEN #1:"DSK1.SCORE/REC"
160 INPUT #1:HSE,SCORE,MEN,N
AME$
170 CLOSE #1
180 CALL CHAR(42,"0000183C18
1824")
190 CALL CHAR(64,"00183C7E3C
3C4242")
200 CALL CHAR(33,"18DB3CDBBC
FF3CC3")
210 CALL CHAR(62,"006666663C
2418")
220 CALL CHAR(103,"00CC5522
00CC5522")
230 CALL CHAR(123,"00CC5522
00CC5522")
240 CALL COLOR(12,7,1)
250 CALL CLEAR :: CALL SCREE
N(4):: RANDOMIZE :: TRAJ=60
:: XCOL=100 :: YCOL=176
260 CALL SPRITE(#1,33,2,YCOL
,XCOL)
270 DISPLAY AT(24,1):"TRAJEC
TORY IN DEGREES:"
280 XMAN=INT(RND(1)*44)+4
290 YMAN=1
300 CALL SPRITE(#4,[illegible]
N,XMAN)
310 CALL MOTION(#4,[illegible]
320 [illegible]
330 CN=0
340 IF [illegible]
350 IF [illegible]
360 CALL MOTION(#1,0,5)
370 GOTO 500
380 IF [illegible]
390 CALL MOTION(#1,0,-5)
400 GOTO 500
410 IF Y [illegible] 4 THEN 440
420 TRAJ=TRAJ-1 :: IF [illegible]
 THEN TRAJ=0
430 CALL MOTION(#1,0,0 :: GO
TO 500
440 IF Y [illegible]-4 THEN 47
450 TRAJ=TRAJ+1 :: IF TRAJ>9
0 THEN TRAJ=90
460 CALL MOTION(#1,0,0):: GO
TO 500
470 CALL KEY(1,K,S)
480 CALL MOTION(#1,0,0)
490 IF K=18 THEN GOSUB 540
500 CALL POSITION(#4,Y,X) ::
IF Y>170 THEN 740
510 DISPLAY AT(24,24):TRAJ
520 IF CN=(-1)THEN 280
530 GOTO 320
540 EXPL=ABS(RND*45 [illegible]
,9+175 :: CALL POSITION(#1,Y
,X)
550 CALL SPRITE(#2,64,15,Y+4
,X
560 CALL [illegible]
X
570 CALL MOTION(#2,-10,1,#3,
-8,0)
580 CALL POSITION(#3,Y,X)::
IF Y<EXPL+175-EXPL THEN 5
30
590 CALL POSITION(#4,
IF Y>170 THEN 740
600 CALL MOTION(#2,-8,-1)
610 CALL POSITION(#2,Y1,X1)
: IF Y1>176-EXPL THEN 610
620 CALL MOTION(#2,0,0,#3,0,
0):: CALL COINC(#2,#4,10,N)
[illegible]
640 CALL POSITION(#4,Y,X)
IF Y>170 THEN 740
650 DISPLAY AT(INT(Y1/8)+1,I
NT(X1/8)-1):"CC"
660 DISPLAY AT(INT(Y1/8)+2,I
NT(X1/8)-1):"CC"
670 CALL DELSPRITE(#2,#3)::
CALL SOUND(-200,-8,0)
680 RETURN
690 DISPLAY AT(INT(Y1/8)+1,I
NT(X1/8)-1):"mm"
700 DISPLAY AT(INT(Y1/8)+2,I
NT(X1/8)-1):"mm"
```


Continued on page 26

CHECK IT OUT

by

Ron Marissen

This program will allow you to track up to 9 different accounts per file. The accounts may be any kind you like. The main reason for developing this particular program, of which millions exist, was for several reasons:

-Being able to store and retrieve information about each account saves you the trouble of entering parameters each time the program is run...although you could if you wanted.

d -Allows you to update an account and also to reconcile your bank statement when you receive it.

-Does NOT ask you to enter a "zero" when leaving a function. Boy, I hate it when that happens. Just pressing "enter" will return you to the previous menu or bring up the next field depending on the particular function you are in.

-Menu driven. My wife even went through the program, updated an account and reconciled a statement. Just follow the prompts and just remember the enter key.

When you first run the program, you will be prompted for the date. If you have a real time clock on line, you could alter the program to fetch the date automatically. Just pushing enter at this point will generate an "N/A" as the date for that particular file.

The next menu has four options:

1. Update Account
2. Reconcile Account
3. QUIT (save data)
4. "HELP"

If you select help, a screen with a couple of pointers will come up. It isn't really a lot of help but may get someone going.

QUITting will allow you to save your data. You don't have to quit, but once this route has been taken, data will be lost if not saved.

Reconcile Account will enable you to, when you receive your bank statement, perform those strange calculations on the back of it with your computer. You should update your account (interest, service charges, etc.) before selecting this option.

Update Account will bring up another menu:

1. Load Data (both disk and tape are supported)
2. Add or Delete Account
3. Update Account
4. Main Menu

Select 2, Add or Delete Account if setting up for the first time or if adding or deleting an account. Enter the account number and current balance when prompted.

I think that the program is fairly user friendly, (according to Malcolm's definition), and you should be able to find your way through it without too much trouble. As with any public domain program, alter it as you like.

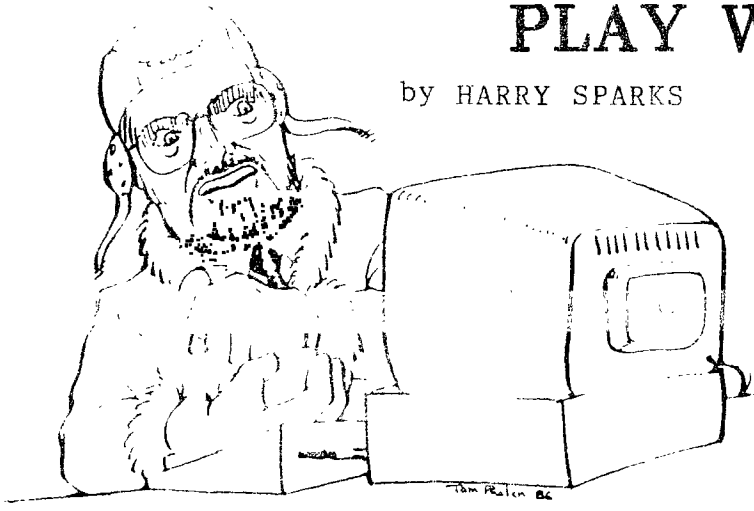
Though the program really jumps around a lot, (due to the piece by piece way in which I wrote it), it does run well. However, if you do modify it, some detective work will have to be done in order to find everything you may have to alter. The word "detective" was not, incidentally, chosen arbitrarily either. There is a program called XB DETECTIVE which, if you have memory expansion, enables you to perform a multitude of searches of your program. It's great! Anyway, use this program and see if you like it. I think you will.

```
100 !************************
110 !*                      *
120 !* check it out  V 1.0  *
130 !*         by           *
140 !*     Ron Marissen     *
150 !*    Channel 99 u.g.   *
160 !*  in Extended Basic   *
170 !*                      *
180 !************************
190 GOTO 210 :: CALL COLOR
: CALL HCHAR :: CALL SCREEN
:: CALL CLEAR :: A,C,CBAL,DE
LAY,I,J,K,L,M,N,NR,R,S,T,W,X
,Y,Z,A$,C$,D$,FN$,N$,Q$,Y$
200 CALL ERR :: D,E,MODE,NAM
T,SB,CB,DIF
210 ON ERROR 1240 :: OPTION
BASE 1 :: DIM SC$(2),CBAL$(9
),DATE$(9),ACN$(9),AMT$(2,20
),AMT(2,20):: F$="0" :: CALL
 KEY(3,K,S)
220 Z$="0123456789.-" :: V$=
SEG$(Z$,1,11):: SC$(1)="depo
sits" :: SC$(2)="checks"
230 !@P-
240 CALL CLEAR :: CALL SCREE
N(5):: FOR I=0 TO 12 :: CALL
 COLOR(I,16,5):: NEXT I
250 DISPLAY AT(1,9)ERASE ALL
:"CHECK IT OUT":TAB(9);"----
--------" :: DISPLAY AT(24,1
0):"version 1.0"
```


continued on page 23

PLAY WITH SPARKY

by HARRY SPARKS



MINER 2049er.

Well, this month I'm going to tell you the story of Bounty Bob. This little fellow is the hero of "Miner 2049er" by TIGERVISION. I have the command module which inserts in the peripheral outlet. With my PE box, this could be a real problem. Fortunately, it is also on disk at Wentworth Supplies and loads from Extended Basic. I have only seen one of the modules--mine. Thank heavens it is on disk.

Bounty Bob has tracked Yukon Yohan to an abandoned uranium mine but is trapped inside by a cave-in. There are 8 sections of the mine with cute but deadly mutant organisms. If they touch you-death. There are also various articles lost by previous miners.

You can escape death by jumping over the mutants or by touching one of the lost articles, the mutants can in turn be killed. Points are awarded for lost articles and killing mutants. All walkways in each section will turn solid as you walk over them. Untravelled sections are obvious at all stages. ALL walkways solid move you to the next section.

When loaded, the title screen appears. The first few times bear with Bob as he will fill in the title screen and go into a short demo of all 8 mine sections. Look them over carefully as there is no pause option so you must move fast and keep moving. Why ?. Well, in the upper right corner is a count-down. When (or if) you complete that section, the time remaining is added to your score. The faster you complete the section, the more bonus points.

Scoring is as follows--each piece of walk-way is 5 points. Each mutant made edible by first touching a lost article is 80 points. Articles are worth 100 - 900 points. Also added are the remaining points on the clock.

As mentioned, there are 8 levels. They are: Slides, Transporters, Lillipads, Advanced Lillipads, Radioactive Waste, Advanced Transporters, Pulverizers and the Cannon. The Transporters (sections 2 6) and Cannon (section 8) could turn your hair as grey as mine if you tried to use them without an explanation. Transporters are simple enough-put Bob in the middle of one on any level and push a number for the level you want to go to. He will flash off and on several seconds at each end of the operation and is still vulnerable to the mutants. The Cannon is a real stinker. Bob can be fired up several levels. First load the Cannon with enough TNT. Go into the TNT hut and touch the cannisters that you want to load into the Cannon. Each cannister equals 10 tons of TNT. You need 10 tons for each level. If you want to go up 2 levels, load 2 cannisters in the Cannon. Now for the fun part if you think that loading is easy. Jump to the level picked, climb the ladder and walk off the walkway falling into the mouth of the Cannon. Push the joystick right or left to aim. Press the fire button to blast Bob to that level. Real easy to do with the nasty little mutants after you. Overload the Cannon and Bob dies.

Most people will not have too much difficulty with the Cannon because it will probably be "lonnnnng" time before this level is reached. THEN you will have difficulty as there are no extra men awarded for completing levels.

You start with three men and that is it. Very realistic in that when a men dies--he dies and is forever gone. You start the game off at first by hitting the space bar. As each level is completed the clock adds bonus points and then an Alert Mode comes on screen warning you of the start of the next level and how many men are left. At game end you will be shown you current score and high score.

The first 2 levels are quite easy. The next levels are NOT quite easy as the jumps and moves require very exact moves with a lot of these moves taking a fair bit of practice to get correctly. If (and I do mean if) you can get through the 8 levels you deserve a nice cup of tea. On level 8 there are no lost articles so the mutants must be jumped over. They are always alive on this level. The first levels are called Zone 1. If they are all complete the difficulty will be increased. Bob will follow the direction of your joystick and jump when the fire button is pressed.

A few hints might be in order so you won't age too quickly trying to get Bob through the mine sections. Bob can drop off the pads or walkways as long as there is another one under him, but can only survive short falls. A fall one pixel too far and he will die. Memorize the distance he can fall. When you use the Transporters, time your move to avoid arriving at the new level with a mutant too close. Some of the sections have areas that you can enter but can't leave unless that area has the last walkway to be walked over. Make sure the Alpha Lock is up. As mentioned earlier, most of the jumps must be made at the very end of the pad area or you will fall short and die. There are a few on some levels where the

Continued on page 25

WHERE ARE THEY NOW

by
Tor Hansen

There was a time, in very recent memory, that a certain Don Cook used to grace these pages with his wisdom and knowledge in assembly language.

You, the reader, may have noticed his recent lack of input. There is a reason for this.

I have KIDNAPPED him and I'm holding him HOSTAGE so I can get some input for my column.

Actually, Don has himself an IBM clone and is in the throes of wrestling his way through its assembly language. He is, however, hanging on to his 99/4A, so we may yet be able to get further input from him and benefit from his knowledge.

But ONLY WHEN I LET HIM ESCAPE!!

Now that I have him, here is a sampling of some high level language work Don has done.

PAINT

By Don Cook

This program was my first effort at writing a Basic program on the TI-99/4A and is, therefore, very inefficient.

The program checks each joystick to find out which direction to move, and then moves a blinking cursor in the direction selected, leaving a painted trail behind. Pressing the fire button will change the colour of the paintbrush.

Alternatively, colours can be changed directly by pushing one of the keys on the keyboard. A sound is made with a frequency which varies with the colour selected.

```
100 REM PAINT
110 A$="FF0000FF0000FF00"
120 CALL CHAR(159,A$)
130 X=15
140 X1=16
150 Y=13
160 Y1=13
170 C=2
180 C1=2
190 CALL CLEAR
200 CALL COLOR(1,11,11)
210 FOR C=2 TO 16
220 CALL COLOR(C,C,C)
230 NEXT C
240 CALL JOYST(1,DX,DY)
250 CALL JOYST(2,DX1,DY1)
260 CALL KEY(1,K1,S1)
270 CALL KEY(2,K2,S2)
280 IF (K1>18)+(K1<2)THEN 35
0
290 IF K1=18 THEN 310
300 C=K1-1
310 C=C+1
320 CALL SOUND(500,252+5*C,2
330 IF K1<>18 THEN 350
340 C=2
350 X=X+DX/4
360 Y=Y-DY/4
370 X=INT(32*((X-1)/32-INT((
X-1)/32)))+1
380 Y=INT(24*((Y-1)/24-INT((
Y-1)/24)))+1
390 CALL HCHAR(Y,X,C*8+16)
400 CALL HCHAR(Y,X,30)
410 CALL HCHAR(Y,X,C*8+16)
420 IF (K2>18)+(K2<2)THEN 49
0
430 IF K2=18 THEN 450
440 C1=C2-1
450 C1=C1+1
460 CALL SOUND(500,252+5*C1,
0)
470 IF C1<>18 THEN 490
480 C1=2
490 X1=X1+DX1/4
500 Y1=Y1-DY1/4
510 X1=INT(32*((X1-1)/32-INT
((X1-1)/32)))+1
520 Y1=INT(24*((Y1-1)/24-INT
((Y1-1)/24)))+1
530 CALL HCHAR(Y1,X1,C1*8+16
)
540 CALL HCHAR(Y1,X1,159)
550 CALL HCHAR(Y1,X1,C1*8+16
)
560 CALL KEY(5,K2,S3)
570 IF K3<>179 THEN 240
580 GOTO 130
590 END
```

Hydro from page 3

```
530 YC=YB*365
540 YC=INT(YC*100+.5)/100
550 REM cost per Kwh=KC
560 KC=TC/TKWH
570 KC=INT(KC*100+.5)/100
580 NEXT X
590 PRINT
600 IF F$="2" THEN 860
610 PRINT "    --- SUMMARY -
--"
620 PRINT "Kwh used(last 4 r
eads)";TKWH
630 PRINT "# DAYS from last
          4 readings......"
;TE
640 PRINT "COST (last 4 rdgs
..)$";TC
650 PRINT "..... CONCLUSION
....."
660 PRINT "Annual Kwh used .
...";AKWA
670 PRINT "Av. daily Kwh fro
m          last 4 readings .
.....";KD
680 PRINT "COST per Kwh.....
...$";" ";KC
690 PRINT "DAILY COST.......
...$";DC
700 PRINT "ANNUAL COST......
.$";YC
710 PRINT "Suggested GOAL..6
100 Kwh..'!"
720 FOR PAUSE=1 TO 1000
730 NEXT PAUSE
740 END
750 OPEN #1:"PIO",OUTPUT,SEQ
UENTIAL,VARIABLE
760 PRINT #1:CHR$(27);CHR$(6
6);CHR$(3);CHR$(27);CHR$(65)
;CHR$(9)
```


Continued on page 18

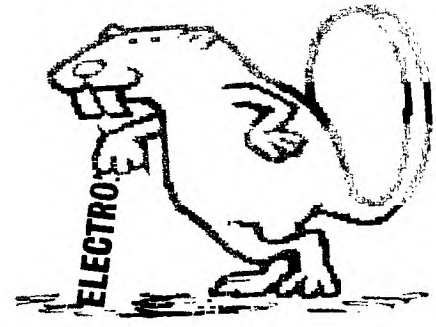
[illegible] number 13
by Jean Johnson

P	1 DAPPLE	2	3	G
	4	5	6	
	7	8	9	
	10	11	12	
	13	14	15	

INSTRUCTIONS FOR PLAYING

Using the clues for the numbered areas, enter your answers, making sure you comply with the rules:- Remove one letter from the first word, placing it in the box on the left. The centre word must be an anagram of the remaining letters. Remove one more letter and place it in the box on the right. The third word must then be an anagram of the remaining letters. When you have finished the left and right columns will spell something to do with your hobby. The first word is done.

1 Mott[illegible]
2 Beseech
3 Metal
4 Flowers
5 Small horses
6 Backbone
7 Man's name
8 Kin
9 Bash
10 Past tense of keep
11 Cat or dog
12 Physical Training
13 Cuddle
14 Frighten
15 Concern

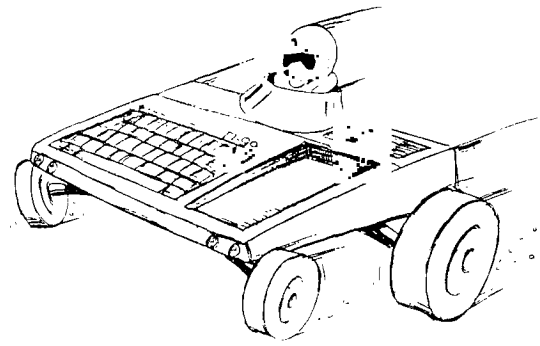


Answers to PUZZEL number 12
by Jean Johnson

M	1 GREMLIN	2 LINGER	3 LINER	C
Y	4 YEARN	5 EARN	6 RAN	F
A	7 NAME	8 MEN	9 ME	G
R	10 EASTER	11 TEASE	12 SEAT	E
C	13 SCARVES	14 SAVERS	15 SEARS	V
S	16 BEAST	17 BEAT	18 TAB	E

EDITED BY
Iain Johnson

LEARNING ASSEMBLY LANGUAGE



GAME #11

```
001 ****************************
002 * GRAVITY BALL WITH PADDLE *
003 *           by             *
004 *     Stephen Johnson      *
005 ****************************
006         DEF  START
007         REF  VSBW,VMBW,VSBR,VMBR,VWTR
008         REF  KSCAN,GPLLNK
009 *********************************
010 *R3 & R4 - BALL'S Y & X POSITION *
011 *R5 & R6 - BALL'S Y & X SPEED    *
012 *R7 - NUMBER OF BALLS            *
013 *R8 - PADDLES X POSITION         *
014 *R9 - PADDLES EFFECT ON BALL     *
015 *********************************
016 STATUS EQU  >837C   *ADDRESS OF STATUS REGISTER
017 KEYNUM EQU  >8374   *ADDRESS OF KEYBOARD CODE NUM
018 KEYVAL EQU  >8375   *ADDRESS OF KEY VALUE RETURND
019 NUM    EQU  >837A   *ADDRESS OF NUMBER OF SPRITES
020 MASK   DATA >2000   *MASK TO TEST EQUAL STATUS BIT
021 ZERO   DATA 0
022 SLIST  DATA >6060,>800F
023        DATA >BD77,>8401,>D000
024 SPDEF  DATA >3C7E,>FFFF,>FFFF,>7E3C
025        DATA >0000,>0000,>0000,>0000
026        DATA >0000,>0000,>0000,>0000
027        DATA >0000,>0000,>0000,>0000
028        DATA >FFFF,>0000,>0000,>0000
029        DATA >0000,>0000,>0000,>0000
030        DATA >FFFF,>0000,>0000,>0000
031        DATA >0000,>0000,>0000,>0000
032 NUMBAL DATA 10
033 YPOS   BYTE 50      *Y STARTING POSITION
034 XPOS   BYTE 128     *X STARTING POSITION
035 YSPD   DATA 0       *Y SPEED
036 XSPD   DATA 60      *X SPEED
037 DELSP  BYTE >D0     *Y POSITION STOPS ALL SPRITES
038 BBL    BYTE >80
039 SPACE  BYTE >20     *ASCII CODE FOR SPACE (32)
040 YBT    BYTE 182     *BOTTOM OF SCREEN
041 YBTT   BYTE 191
042 YTP    BYTE 1       *TOP OF SCREEN
043 XLT    BYTE >0A     *LEFT
044 XRT    BYTE >EF     *RIGHT
045        BYTE 40
046        EVEN
047 TEXEND TEXT '     ** THE GAME IS OVER **     '
048        EVEN
049 PADSTR DATA >BE00
050 ***********************
051 * PROGRAM BEGINS HERE *
052 ***********************
053 START  LWPI >8300   *SET WORKSPACE AREA TO
054 *                   *   START AT >8300
055        BLWP @SOUNDS *SET UP SOUND EFFECTS
056 ************************************
057 * LOOP TO CLEAR SCREEN WITH SPACES *
058 ************************************
059 MAIN   MOV  @NUMBAL,R7 *NUMBER OF BALLS
060        BL   @BEEP
061 CLEAR  LI   R0,767   *LAST SCREEN POSITION
062        LI   R1,>2000 *MSB=SPACE(32)
063 CLRMOR BLWP @VSBW    *PUT CHARACTER ON SCREEN
064        DEC  R0       *DECREASE SCREEN POSITION
065        JLT  INIT     *IF SCRN POS<0 THEN LEAVE LOOP
066        JMP  CLRMOR   *REPEAT LOOP
067 ***********************
068 * TEST FOR END OF GAME *
069 ***********************
070 ENDOFG DEC  R7        *DECREASE BALL COUNTER
071        JNE  INIT     *IF MEN NOT 0 CONTINUE GAME
072        LI   R0,>300  *Y-POSITION SPRITE #0
073        MOVB @DELSP,R1    *LOAD DELETE SPRITE DATA
074        BLWP @VSBW    *DELETE ALL SPRITES NOW
075        LI   R0,>180  *MID SCREEN ROW
076        LI   R1,TEXEND *POINT END GAME TEXT
077        LI   R2,32    *32 BYTES FOR ONE ROW TEXT
078        BLWP @VMBW    *WRITE END OF GAME
079        LI   R2,4000
080 DELAYT BL   @DELAY   *DELAY BY 32 DELAY TIMES
081        DEC  R2       *DECREASE UNTIL 0
082        JNE  DELAYT   *IF NOT 0 DELAY AGAIN
083        JMP  MAIN
084 ***********************
085 * INITIALIZE VARIABLES *
```

```
086 ************************
087 LOOP    MOVB @YPOS,R3  *SET Y POSITION
088         MOVB @XPOS,R4  *SET X POSITION
089         MOV  @YSPD,R5  *SET Y VELOCITY
090         MOV  @XSPD,R6  *SET X VELOCITY
091         LI   R0,>01E2  *DOUBLE SIZE SPRITES
092         BLWP @VWTR     *WRITE TO VDP REGISTER 1
093 ************************
094 * DEFINE SPRITES *
095 ************************
096         LI   R0,>400   *SPRITE CHARACTER >80
097         LI   R1,SPDEF  *DEFINITION OF CHARACTER.
098         LI   R2,64     *2 DOUBLE SIZED SPRITE DEFS
099         BLWP @VMBW     *SEND DEFINITION TO TABLE
100 *
101         LI   R0,>300   *ADDR OF SPRITE ATTR LIST
102         LI   R1,SLIST  *POINTER TO THE LIST
103         LI   R2,10     *10 BYTES IN LIST
104         BLWP @VMBW     *SEND SPRITE ATTRIBUTES
105 ************************
106 * BOUNCE BALL *

107
108 * KEY SCAN FOR PADDLE AND RETURN TO E/A
109 LOOP    LI   R0,>300   *READ Y POSION
110         A    R5,R3     *ADD Y SPEED TO X POSITION
111         MOV  R3,R1     *MOVE Y POS TO R1
112         BLWP @VSBW     *WRITE NEW Y POS TO VDP
113         INC  R5        *INCREASE VERTICAL SPEED
114 YTPCHK  CB   R1,@YTP   *CHECK FOR TOP OF SCREEN
115         JH   YBTCHK    *IF HIGHER JMP TO Y BOT CHEK
116         ABS  R5        *ABSOLUTE R5, REVERSES DIR
117         BLWP @SOUND1   *EXECUTE SOUND EFFECT
118 *
119 YBTCHK  CB   R1,@YBT   *CHECK FOR BOTTOM OF SCREEN
120         JL   XCHNG     *IF < BOTTOM JUMP TO X CHK
121         MOV  R4,R2     *PUT BALL'S X POS INTO R1
122         AI   R2,1024   *ADD 4 TO COMPARE TO PADDLE
123         CB   R2,R8     *CHECK LEFT SIDE OF PADDLE
124         JL   CHKBOT    *IF LOGICALY LOW-NO BNCE-CHECK
125 *                      *          FOR BOTTOM
126         AI   R2,-4096  *ADD NEGATIVE >1000
127         CB   R2,R8     *COMPARE TO PADDLE POSITION
128         JH   CHKBOT    *IF GREATER, NO BOUNCE;
129 *                      *     CHECK FOR BOTTOM
130         ABS  R5        *MAKE R5 POSITVE
131         NEG  R5        *MUST HAVE HIT TO REACH HERE
132         A    R9,R6     *PADS EFFECT ON BALLS X-SPEED
133         MOV  R6,R6     *COMPARE R6 TO ZERO
134         BNE  EXSND     *IF NOT 0 VALUE IS OK ELSE
135         MOV  @XSPD,R6  *BAL WILL NEVER HAVE 0 X-SPEED
136 EXSND   BLWP @SOUND2   *EXECUTE SOUND EFFECT
137         JMP  XCHNG
138 CHKBOT  CB   R1,@YBTT  *CHECK FOR BOTTOM OF SCREEN
139         JLE  XCHNG     *IF NOT JMP TO CHCK X-
140         BL   @HONK     *     ELSE MUST HAVE-
141         JMP  ENDOFG    *     LOST BALL-RESTART PROG
142 XCHNG   LI   R0,>300   *READ X POSITION
143         A    R6,R4     *ADD X SPEED TO X POSITION
144         MOV  R4,R1     *MOVE NEW X POSITION TO R1
145         BLWP @VSBW     *WRITE NEW X POSITION TO VDP
146 *
147         CB   R1,@XLT   *CHECK FOR LEFT OF SCREEN
148         JEQ  XREV      *IF EQUAL REVERSE X DIR
149         CB   R1,@XRT   *CHECK FOR RIGHT OF SCREEN
150         JEQ  XREV      *IF EQUAL REVERSE X DIR
151 ************************
152 * SCAN KEYBOARD *
153 ************************
154 KEYCHK  CLR  R0
155         LI   R1,>0300
156         MOVB R1,@KEYNUM
157         CLR  R1
158         MOVB R1,@STATUS
159         BLWP @KSCAN
160         LIMI 2
161         LIMI 0
162         MOVB @>8375,R1
163         SWPB R1
164         CI   R1,15     *FCTNBACK HASS BEEN PRESSED?
165         JEQ  RETURN    *IF BACK PRESS, RETURN TO E/A
166         CLR  R9        *SET PADLE SPEED EFFECT TO 0
167 *
168         CI   R1,83     *CKECK IF S WAS PRESSED
169         JNE  CHKA      *IF NOT JMP CHKA
170         AI   R8,-100   *MOVE PADDLE LEFT SLOW
171         LI   R9,-10    *X-SPEED EFECT ON BALL IF BNCE
172 CHKA    CI   R1,65     *CHECK IF A WAS PRESSED
173         JNE  CHKD      *IF NOT JMP TO D CHECK
174         AI   R8,-200   *MOVE PADDLE LEFT FAST
175         LI   R9,-20    *PADDLES EFECT ON BALL
176 CHKD    CI   R1,68     *CHECK IF D WAS PRESSED
177         JNE  CHKF      *IF NOT JMP TO F CHECK
178         AI   R8,100    *MOVE PADDLE RIGHT SLOW
179         LI   R9,10     *PADDLES EFFECT ON BALL
180 CHKF    CI   R1,70     *CHECK IF F WAS PRESSED
181         JNE  MOVPAD    *IF NOT JMP TO MOVE PADDLE
182         AI   R8,200    *MOVE PADDLE RIGHT FAST
183         LI   R9,20     *PADDLES EFFECT ON BALL
184 MOVPAD  LI   R0,>01E2   *DOUBLE SIZE PADDLE
185         BLWP @VWTR     *WRITE TO VDP REG 1
186         LI   R0,>305   *MOVE PADDLE
187         MOV  R8,R1     *MOVE PADDLE POSITION FOR VSBW
188         BLWP @VSBW     *WRITE PADDLE POSTION TO VDP
189         BL   @DELAY    *CAUSE TIME DELAY
190         JMP  LOOP      *BEGIN GAME LOOP AGAIN
191 *******
192 XREV    BLWP @SOUND2   *EXECUTE SOUND EFFECT
193         NEG  R6        *REVERSES X DIRETION
194         JMP  KEYCHK    *RETURN TO KEY SCAN
195 ******************************
196 **DELAY TO SLOW PROGRAM DOWN**
197 ******************************
198 DELAY   LIMI 2         *ENABLE VDP INTERUPTS TO OCCUR
199         LI   R0,180    *SETS DELAY AT 300
200 DELAYZ  DEC  R0        *DECREMENTS DELAY
201         JNE  DELAYZ    *IF ZERO THEN EXIT DELAY
202         LIMI 0         *DISABLE VDP INTERRUPTS
203         RT             *     ELSE CONTINUE DELAY
204 **********
205 * RETURN *
```

```
206 **********
207 RETURN LI   R0,>300  *SET Y POS TO D0
208        LI   R1,DELSP *DATA TO DELETE SPRITE
209        BLWP @VSBW    *DELETE SPRITES
210 *
211        CLR  R0
212        MOVB R0,@>837C *CLEAR STATUS BYTE
213        LWPI >83E0    *SET WORKSPACE
214        B    @>70     *EXIT PROGRAM ALTOGETHER
215 ** GOOD/BAD RESPONSE TONES **
216 ** BL  @BEEP **OR** BL  @HONK **
217 BEEP   MOVB @ZERO,@>837C
218        BLWP @GPLLNK
219        DATA >34
220        RT
221 HONK   MOVB @ZERO,@>837C
222        BLWP @GPLLNK
223        DATA >36
224        RT
225 ***************************
226 * AUTOMATIC SOUND EFFECTS *
227 ***************************
228 SNDWSP BSS  32     SOUND WORKSPACE
229 *CALL SOUND(50,-7,0)
230 SND1   BYTE 2,>E6,>F0,3,1,>FF,0,0
231 *CALL SOUND(1,110,0)
232 SND2   BYTE 3,>89,>3F,>90,2,1,>9F,0
233 *CALL SOUND(1,-5,5)
234 SND3   BYTE 2,>E4,>F2,2,1,>FF,0,0
235 *
236 SOUNDS DATA SNDWSP,$+2
237        LIMI 0
238        LI   R0,>1000
239        LI   R1,SND1
240        LI   R2,>08
241        BLWP @VMBW
242        RTWP
243 SOUND1 DATA SNDWSP,$+2
244        LI   R0,>1000
245        BL   @AUTO
246        RTWP
247 SOUND2 DATA SNDWSP,$+2
248        LI   R0,>1008
249        BL   @AUTO
250        RTWP
251 SOUND3 DATA SNDWSP,$+2
252        LI   R0,>1010
253        BL   @AUTO
254        RTWP
255 WAIT   MOVB @>83CE,@>83CE    *IS SOUND FINISHED?
256        JNE  WAIT
257        RT                 *RETURN IF SOUND FINISHED
258 ************************
259 * AUTO SOUND PROCESSING *
260 ************************
261 SET    DATA >0100
262 AUTO   LIMI 0
263        MOV  R0,@>83CC
264        MOVB @SET,@>83CE
265        SOCB @SET,@>83FD
266        RT
267        END
```

by Stephen Johnson

A few new additions have been made to last week's fine program. Sound effects occur when the ball hits the top, sides, or paddle. There is a paddle with which to hit the ball. Stephen has given us the option of a fast paddle or a slow paddle simply by using S or D for slow or press A or F for a fast paddle. Another wonderful improvement is the ball counter (register 7). You get ten balls and then the game ends with a message. There is a fixed delay and the game starts all over again.

1-15....Program header.
6.......DEFined entry point into program.
7-8.....REFerence for built-in utilities.
16-19...EQUate values for easier (and better) programming technique.
22......Data to send to sprite attribute list (sprite row location, sprite column location, character, colour). The fifth byte is >D0 (208 decimal). This value is placed in the y location after the last sprite to prevent any higher numbered sprites showing up.
24-31...Character definition of the ball and paddle.
32......Data to represent number of balls.
33-36...Variables for Y position, X position, Y direction, X direction.
37......>D0 in sprite attribute list as Y position will delete all sprites after it.
39......Ascii code of space character 32.
40-44...Edge of screen variables.
47......End of game message
48...... The "EVEN" directive tells the assembler to assume an even word boundary so that the TEXT and BYTE directives don't screw it up.
53......Set up workspace area in the RAM PAD at address >8300. Any instruction using registers will execute at a much higher speed if the registers are located in the area >8300 to >83FF.
55......Executes routine to put all [illegible] ram ready for instant access at any time.
59......Load R7 with number of balls.
60......Execute a GPL routine to call a sound.
61-66...Clear screen by putting space character in every screen position.
70-71...Tests if balls are gone.
72-74...If balls are gone all sprites are deleted.
75-78...Writes a neat little message accross centre screen.
80-83...Delays for a fixed length of time before jumping to the main program to start all over again.
87-90...INITialize registers 3 to 6. R3 will now be a variable containing the row position of the

ball. R4 is a variable for column location. Register 5 contains the Y direction and speed. R6 contains the X direction and speed.

91-92...Writes the value for double sized sprites (you will see next month why double-size is used) to VDP Register 1.

96-100..Using Vdp Mulitple Byte Write utility the 2 32 byte character definitions are defined as the first sprite definitions in the table. The ball sprite character definition will be refered to as charcter >80 while the paddle will be character >84. More sprites can be defined and placed in the list. You can calculate the address by multiplying the sprite number by the number of bytes in the definition and adding it to >400 i.e. the address for sprite #0 is >400+0=>400 or for sprite #1 is >400+1=>420 or sprite #2 is >400+2=>440. The character definitions would be called >80,>84,>88,etc.

101-104.Sprites will appear on the screen according to the list in line 15. This list contains the y-dot, x-dot, character, and colour of the sprites and is written to the VDP RAM.

109.....Loads R0 with the base address of the sprite attribute list i.e. R0 points to the Y-position of sprite number zero (sprite #0).

110.....Adds register 5 to register 3 or adds the vertical speed to the Y position.

THE FOLLOWING EXPLANATION OF LINE 59 IS IMPORTANT

111.....Moves the variable Y position to R1 ready for a Vdp Single Byte Write utility. NOTE THAT in a vdp write only the left byte is sent but the whole register of R3 contains the Y position. The Y position is remembered in the left byte. The speed is added to the whole register. When the right byte reaches the value >FF and a speed of 1 is added the right byte overflows into the left byte and the left byte is incremented by one thus the sprite will be moved one pixel. The right byte is now 0. Another way of explaining this method of controlling the sprites speed is to say that if the speed were >10 (16 in deciaml) it would take 16 additions of this speed to move the sprite one pixel (>0100 + >10 X 16 = >0100 + >0100 = >0200). In extended basic this is impossible but because of the impressive speed of assembly language this method works very well indeed.

112.....Y position is sent to VDP RAM as sprite moves up or down.

113.....Verticle speed has one added to it each time it bounces.

114.....The Y position now remembered in R1 as well as R3 is compared to the top of the screen and if high program control is sent to Y bottom check.

116.....If control passes to this line then the ball has reached the top of the screen and the speed is made positive so the ball will g down.

117.....Call a sound effect when ball hits top.

119.....Compare Y position to bottom screen limit.

120.....If the comparison is low (JL, JH, JHE are use for byte comparisons) control is passed to th code at line 142.

121-122.Puts the ball X position in R2 and adjusts i so that a comparison can be made to determine if the paddle has been struck by the ball.

123.....Ball's adjusted X position is compared to paddle position.

124.....If comparison is low control is passed t checjk for bottom of screen else the othe side of the paddle is checked to see if bal hit paddle (lines 126-128).

130.....Y velocity of ball is made positive to prevent modification of value.

131.....Ball has definately hit paddle. Negates spee so that the ball will go up.

132.....If paddle was moving when it hit the ball speed will be transferes from R9 to R6 t modify the balls speed.

133.....It is undesirable for the ball not to have a X speed however so MOV R6,R6 tests it to see if the balls X speed is 0.

134-135.If balls X speed is not 0 then execute line 136 else (135) load R6 with a new speed.

136.....Execute sound effect as ball hits paddle.

137.....Jump over Y bottom check.

138-139.Compares balls Y position to very bottom of screen. If the comparison is high then the ball has passed off the bottom of the scree and you loase a ball.

140.....Execute bad response tone.

141.....Jump to END OF Game tester.

142-145.Calculates and writes the X position the same way as the Y position. 147.....The X position now contained in R1 as well as R4 is compared to the left screen limit and if it is equal control is passed to code that will reverse the X direction.

149-150.The X position now contained in R1 as well as R4 is compared to the right screen limit and if it is equal control is passed to code that will reverse the X direction.

154-159.These lines of code should seem quite familiar by now. They are a standard way of accessing the keyboard.

162.....Gets the ascii code in the word value of R1.

164.....Compares the ascii code with function back and if it is equal jumps to the code at line 20 and returns back to the editor/assembler.

168-171.There are four such comparisons and subsequent loading of registers 8 and 9. According to which one of four keys is pressed R8 and R9 will be loaded with the paddle speed and paddle's effect on ball respectively.

184-185.Resends double-sized sprite information again if you save the >E2 part at location >83D4 the

sprite size will not change when a key is pressed.

186-188.Move paddle on screen.

189.....Branch and link to delay time routine.

190.....Jump back to beginning of game loop.

195-203.This delay routine can be called at any time with the BL instruction which is much like GOSUB in BASIC.

207-209.Deletes all sprites in preparation for a safe return from this program.

211-214.Returns to editor/assembler.

225-267.Will be covered in detail in a section on sound.

HAVE FUN!!!!

AIRWOLF

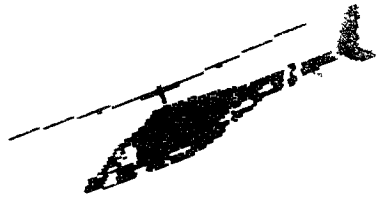
by Iain Johnson

The following program is written in the bit-map-mode and will be added to in an article next month to make the helicopter move. You will notice that most of the lines contain subroutines, in only the last few lines (lines 331-332) is Airwolf actually drawn. There are two main lists of data. One for the character definitions and one for the colour definitions. If you feel like it you can look over the code or just type it for fun!!!!!

```
001 ****************
002 *    AIRWOLF   *
003 *      by      *
004 * Iain Johnson *
005 ****************
006 ***************************************
007 * AIRWOLF SPECIFICATIONS FOR DRAWING *
008 *   GRAPHICS MODE=BIT MAPPED MODE     *
009 *   SPRITES=1/WHITE UNDERSIDE         *
010 *           2/ROTOR                   *
011 *   SPRITE SIZE=DOUBLE UNMAGNIFIED    *
012 ***************************************
013         DEF  START
014         REF  KSCAN,GPLLNK,XMLLNK,DSRLNK
015 ** EQUATES **
016 STATUS EQU  >837C        CPU STATUS REGISTER
017 VDPSTA EQU  >837B        VDP STATUS REGISTER
018 NUMSPR EQU  >837A        NUMBER OF SPRITES IN MOTION
019 *
020 KEYNUM EQU  >8374
021 KEYVAL EQU  >8375
022 JOYY   EQU  >8376
023 JOYX   EQU  >8377
024 *
025 SUBWS  EQU  >2700
026 SUBWS2 EQU  >2720
027 HEXWS  EQU  >2740
028 UTILWS EQU  >2760
029 MYREG  EQU  >8320
030 *
031 EQUMSK DATA >2000
032 SPACE  DATA >2020        SPACE CHARCTERS
033 ** AIRWOLF DATA FOR DRAWING **
034 * CHAR DEF STARTING AT ADDRESS >0000
035 CHRDEF DATA 5,>0000,16*8    *5 LISTS,ADDR 0
036         DATA >0000,>00FF,>0000,>0000 *1
037         DATA >0000,>00F0,>0F00,>0000 *2
038         DATA >0000,>0000,>FF00,>0000 *3
039         DATA >0000,>0000,>00FF,>0100 *4
040         DATA >0000,>0000,>00F0,>8F00 *5
041         DATA >0000,>000C,>0000,>FFC3 *6
042         DATA >0000,>0000,>0000,>FFFF *7
043         DATA >0000,>0000,>0000,>FFF0 *8
044         DATA >0000,>0000,>0007,>F800 *9
045         DATA >0000,>0000,>00FF,>6000 *10
046         DATA >0000,>0000,>3FC0,>0000 *11
047         DATA >0000,>0003,>FC00,>0000 *12
048         DATA >0000,>00FF,>0000,>0000 *13
049         DATA >0000,>00C0,>0000,>0000 *14
050         DATA >000C,>0000,>0103,>0707 *15
051         DATA >0000,>00F0,>F0E0,>E0E0 *16
052 * CHAR DEF >0128
053         DATA >0128,11*8
054         DATA >0000,>0000,>0000,>0107 *1
055         DATA >0C0C,>0C0C,>0CFF,>FFFF *2
056         DATA >0000,>0000,>00FF,>FFFF *3
057         DATA >0000,>0000,>000C,>FFFF *4
058         DATA >0000,>0000,>0000,>00F8 *5
059         DATA >000C,>0000,>0000,>0000 *6
060         DATA >0000,>0000,>0000,>0001 *7
061         DATA >0000,>0000,>00C0,>00C0 *8
062         DATA >0000,>0000,>0000,>0000 *9
063         DATA >0F0F,>1F1F,>3F7F,>7FFF *10
064         DATA >E0C0,>C0C0,>8080,>8080 *11
065 * CHAR DEF >0218
066         DATA >0218,13*8
067         DATA >0000,>0000,>0000,>0107 *1
```

```
068         DATA >0000,>010F,>3FFF,>2361 *2
069         DATA >0F3F,>FFFF,>FFFF,>FFFF *3
070         DATA >FFFF,>FFFF,>FFFF,>FFFF *4
071         DATA >FFFF,>FFFF,>FFFF,>FFFF *5
072         DATA >FFFF,>FFFF,>FFFF,>FFFF *6
073         DATA >FFFF,>FFFF,>FFFF,>F8C0 *7
074         DATA >FFFF,>FFFC,>E0FF,>FFF0 *8
075         DATA >FDFD,>FBFB,>F7F7,>E700 *9
076         DATA >DFDF,>BFBF,>BF82,>0100 *10
077         DATA >FEFD,>FEFF,>8000,>0080 *11
078         DATA >FFFF,>FF7F,>3F1F,>0F03 *12
079         DATA >0000,>0080,>C0E0,>E0F0 *13
080 * CHAR DEF >0310
081         DATA >0310,9*8
082         DATA >0000,>C000,>0F0F,>1F3F *1
083         DATA >0F1F,>3FF1,>7FF3,>E3C7 *2
084         DATA >C173,>8383,>FFFF,>FFF0 *3
085         DATA >FFFF,>FFFF,>FFFF,>FFFF *4
086         DATA >FFFF,>FFFF,>FFFF,>80FF *5
087         DATA >FFFF,>FFFF,>FFFF,>FFFF *6
088         DATA >FEF0,>FFFE,>F8F8,>E000 *7
089         DATA >FFFF,>F0C0,>0000,>0000 *8
090         DATA >C000,>0000,>0000,>0000 *9
091 * CHAR DEF >0410
092         DATA >410,6*8
093         DATA >3F00,>0000,>0000,>0000 *1
094         DATA >87FF,>0000,>0000,>0000 *2
095         DATA >FFF0,>0000,>0000,>0000 *3
096         DATA >FEFF,>0000,>0000,>0000 *4
097         DATA >FFFF,>0000,>0000,>0000 *5
098         DATA >F880,>000C,>0000,>0000 *6
099 * COLOUR DEFINITIONS STARTING ADRESS >0000
100 COLDEF DATA >0000,16*8
101         DATA >F0F0,>F0F0,>F0F0,>F0F0 *1
102         DATA >F0F0,>F0F0,>F0F0,>F0F0 *2
103         DATA >F0F0,>F0F0,>F0F0,>F0F0 *3
104         DATA >F0F0,>F0F0,>F0F0,>F0F0 *4
105         DATA >F0F0,>F0F0,>F0F0,>F0F0 *5
106         DATA >F0F0,>F0F0,>F0F0,>F0F0 *6
107         DATA >F0F0,>F0F0,>F0F0,>F0F0 *7
108         DATA >F0F0,>F0F0,>F0F0,>F0F0 *8
109         DATA >F0F0,>F0F0,>F0F0,>F0F0 *9
110         DATA >F0F0,>F0F0,>F0F0,>F0F0 *10
111         DATA >F0F0,>F0F0,>F0F0,>F0F0 *11
112         DATA >F0F0,>F0F0,>F0F0,>F0F0 *12
113         DATA >F0F0,>F0F0,>F0F0,>F0F0 *13
114         DATA >F0F0,>F0F0,>F0F0,>F0F0 *14
115         DATA >F0F0,>F0F0,>F0F0,>F0F0 *15
116         DATA >F0F0,>F0F0,>F0F0,>F0F0 *16
117 * COLOUR DEF >0128
118         DATA >0128,11*8
119         DATA >1010,>1010,>1010,>1010 *1
120         DATA >E0E0,>F0F0,>F010,>1010 *2
121         DATA >1010,>1010,>1010,>1010 *3
122         DATA >1010,>1010,>1010,>1010 *4
123         DATA >1010,>1010,>1010,>1010 *5
124         DATA >1010,>1010,>1010,>1010 *6
125         DATA >1010,>1010,>1010,>1010 *7
126         DATA >1010,>1010,>1010,>1010 *8
127         DATA >1010,>1010,>1010,>1010 *9
128         DATA >1010,>1010,>1010,>1010 *10
129         DATA >1010,>1010,>1010,>1010 *11
130 * COLOUR DEF >218
131         DATA >218,13*8
132         DATA >B0B0,>B0B0,>B0B0,>B0B0 *1
133         DATA >1010,>1010,>1010,>1B1B *2
134         DATA >1010,>1010,>1010,>1010 *3
135         DATA >1010,>1010,>1010,>1010 *4
136         DATA >1010,>1010,>1010,>1010 *5
137         DATA >1010,>1010,>1010,>1010 *6
138         DATA >1010,>1010,>1010,>1F1F *7
139         DATA >1F1F,>1F1F,>1FF0,>F0F0 *8
140         DATA >1F1F,>1FF1,>FFF1,>F0F0 *9
141         DATA >1F1F,>1FF1,>F1F0,>F0F0 *10
142         DATA >1F1F,>1FF1,>F0F0,>F0F0 *11
143         DATA >1F1F,>1FF1,>F0F0,>F0F0 *12
144         DATA >F0F0,>F0F0,>F0F0,>F0F0 *13
145 * COLOUR DEF >310
146         DATA >310,9*8
147         DATA >0000,>0000,>E010,>1010 *1
148         DATA >B0B0,>B01B,>1E1B,>1B1B *2
149         DATA >1B1B,>1B1B,>1F1F,>1F1F *3
150         DATA >1F1F,>1F1F,>1F1F,>1F1F *4
151         DATA >1F1F,>1F1F,>1F1F,>1FF1 *5
152         DATA >1F1F,>1F1F,>1F1F,>F1F1 *6
153         DATA >1F1F,>1F1F,>1F10,>F0F0 *7
154         DATA >F0F0,>F0F0,>F0F0,>F0F0 *8
155         DATA >F0F0,>F0F0,>F0F0,>F0F0 *9
156 * COLOUR DEF >410
157         DATA >410,6*8
158         DATA >1010,>0000,>0000,>0000 *1
159         DATA >1B10,>1000,>0000,>0000 *2
160         DATA >1F1F,>1F00,>0000,>0000 *3
161         DATA >1FF0,>0000,>0000,>0000 *4
162         DATA >F0F0,>0000,>0000,>0000 *5
163         DATA >F0F0,>0000,>0000,>0000 *6
164 ********************
165 * BIT-MAP-MODE SET *
166 ********************
167 VDPREG BYTE >02   0  Bit map mode
168         BYTE >80   1  16K, interupt off, screen off
169         BYTE >06   2  Screen image table >1800->1AFF
170         BYTE >FF   3  Colour table >2000->37FF
171         BYTE >03   4  Pattern table >0000->17FF
172         BYTE >36   5  Sprite attributes >1B00->1B7F
173         BYTE >03   6  Sprite descriptors >1800->2FFF
174         BYTE >03   7  Foregnd trans ! backgnd black
175 * Routine to change to bit mapped mode
176 BMMODE DATA HEXWS,$+2
177         LI   R1,VDPREG    VDP bit map register data
178         LI   R2,>7F00
179 REGLP  MOVB *R1+,@>8C02
180         AI   R2,>0100
181         MOVB R2,@>8C02
182         CI   R2,>8700
183         JL   REGLP
184 * Routine to set up VRAM tables
185 TABLES BLWP @VSBDUP       Clear VRAM
186         DATA 0,0,>2000
187         LI   R2,>300      # of values in screen table
```

```
188         LI   R0,>1800      Screen table location in VRAM
189         BL   @S1VWRT
190         CLR  R1           Start at pattern 0
191 LOOPS   MOVB R1,@>8C00    Put value in VRAM
192         AI   R1,>0100     Screen pattern location
193         DEC  R2
194         JNE  LOOPS
195         BLWP @VSBDUP
196         DATA >2000        Colour table VRAM location
197         DATA >1000        black/transp
198         DATA >1800        # of colour table values
199         LI   R0,>01E2     Screen on, double size sprite
200         BLWP @VWTR
201         LI   R0,>0706     DARK RED SCREEN
202         BLWP @VWTR
203         RTWP
204 *+++++++++++++++++
205 * BLWP UTILITIES +
206 *+++++++++++++++++
207 ** VMBW **
208 VMBW    DATA UTILWS,$+2
209         BL   @S1RCLL
210         JMP  VMBWW
211 ** VMBWD **
212 VMBWD   DATA UTILWS,S1WRTE
213 S1WRTE  BL   @S1DCLL      Get data from call routine
214 VMBWW   BL   @S1VWRT      Set to write in VDP RAM
215 S1WBYT  MOVB *R1+,@>8C00 Write byte in VDP RAM
216         NOP
217         DEC  R2           Next byte
218         JNE  S1WBYT       Last byte?
219         LIMI 2
220         RTWP
221 S1VWRT  ORI  R0,>4000     Set to write
222 S1VRD   LIMI 0
223         SWPB R0           Swap for MSB
224 S1VLOC  MOVB R0,@>8C02    :
225         SWPB R0           :  Set VDP RAM
226         MOVB R0,@>8C02    :  write location
227         NOP
228         RT                :
229 ** VSBW **
230 VSBWD   DATA UTILWS,$+2
231         MOV  *R14+,R0
232         MOV  *R14+,R1
233         JMP  VSBWW
234 VSBW    DATA UTILWS,$+2
235         MOV  @2(R13),R1
236 VSBWX   MOV  *R13,R0
237 VSBWW   BL   @S1VWRT
238         MOVB R1,@>8C00
239         NOP
240         LIMI 2
241         RTWP
242 ** VSBW8 ** SEND LSB R1 TO VDP
243 VSBW8   DATA UTILWS,$+2
244         MOV  @2(R13),R1
245         SLA  R1,8
246         JMP  VSBWX
247 ** VWTR **
248 BYT1    DATA >0100
249 VWTR    DATA UTILWS,$+2
250         MOV  *R13,R0
251         CB   @BYT1,R1
252         JNE  VWTRZX
253         SWPB R1
254         MOVB R1,@>83D4
255 VWTRZX  MOV  *R13,R0
256         ORI  R0,>8000
257         BL   @S1VRD
258         RTWP
259 * REGISTER CALL
260 S1RCLL  MOV  *R13,R0      SAME AS @0(R13)
261         MOV  @2(R13),R1
262         MOV  @4(R13),R2
263         RT
264 * DATA CALL
265 S1DCLL  MOV  *R14+,R0     Get VDP location
266         MOV  *R14+,R1     Get value to write to VDP R
267         MOV  *R14+,R2     Get Number of bytes
268         RT
269 ** HCHAR **
270 HCHAR   DATA UTILWS,$+2
271         BL   @S1RCLL
272         LI   R3,1
273         JMP  HCHARZ
274 VSBDUP
275 HCHARB  DATA UTILWS,$+2
276         LI   R3,1
277         JMP  HCHARB
278 HCHARM  DATA UTILWS,$+2
279         MOV  *R14+,R3     *#HCHAR'S
280 HCHARB  BL   @S1DCLL      *ADDRESS
281         BL   @S1VWRT
282 HCHARZ  MOVB R1,@>8C00
283         NOP
284         DEC  R2
285         JNE  HCHARZ
286         DEC  R3
287         JNE  HCHARB
288         RTWP
289 ******************************************
290 * DRAW AIRWOLF FROM DOT ROW DOT COLUMN IN R1 *
291 ******************************************
292 WOLF    DATA SUBWS,WOLFY
293 WOLFY   LI   R2,>2000     colour base
294         LI   R4,>FF00     mask left byte
295         LI   R5,256       num bytes in a row
296         LI   R7,8         num bytes in char
297         MOV  @2(R13),R12  R12=loc
298         MOVB R12,R8
299         SRL  R8,8
300         MPY  R5,R8        R9=ADDRESS OF ROW
301         MOV  R12,R2
302         SZC  R4,R2        R2=COLUM NUMBER
303         MPY  R7,R2        R3=ADDRESS COLUMN
304         A    R3,R9        R9=ADDRESS
305         LI   R10,CHRDEF   R10=DATA CHARACTER
306         LI   R6,COLDEF    R6=DATA COLOUR
307 * DRAW CHARACTERS
```



```
308         MOV  *R10+,R4      R4=NUMBER OF LISTS
309         MOV  *R10+,R0      R0=ADDRESS
310         A    R9,R0         MOD ADDRESS
311         MOV  *R10+,R2      R2=NUMBER BYTES
312         MOV  R10,R1
313         BLWP @VMBW         DRAW ONE CHARACTER LINE
314         A    R2,R10
315 * DRAW COLOURS
316         MOV  *R6+,R0       R0=ADDRESS
317         A    R9,R0         MOD ADDRESS
318         AI   R0,>2000      ADJUST TO COLOUR TABLE
319         MOV  *R6+,R2       R2=NUMBER BYTES
320         MOV  R6,R1
321         BLWP @VMBW         DRAW ONE CHARACTER LINE
322         A    R2,R6
323         DEC  R4
324         JNE  DAWULF
325         RTWP
326 *****************
327 * PROGRAM START *
328 *****************
329 START   LWPI MYREG        Load workspace
330         BLWP @BMMODE       Set bit mapped mode
331         LI   R1,>0B0A      Y/X CHAR LOCATION
332         BLWP @WOLF
333 LOOPY   LIMI 2
334         LIMI 0
335         JMP  LOOPY
336         END
```

SUPER GRAPHICS MODE

NORTHERN PIKE

```
110 ***************************
111 *                         *
112 *    + NORTHERN PIKE +    *
113 *                         *
114 ***************************
115 *VDP WRITE ONLY REGISTERS
116 NOR     DATA 0
117         DATA >01E2
118         DATA >0707
119 *HCHARS
120         DATA 2
121         DATA 32          *NUM BYTES
122         DATA 0           *ADDR
123         DATA >E300       *CHAR
124         DATA 768
125         DATA >40
126         DATA >E000
127 *CHARACTER DEFINITIONS
128 *DARK GREEN ON LIGHT YELLOW
129         DATA 6,112,>0D40
130         DATA >FFFF,>FFFF,>FFFF,>FFFF *A8
131         DATA >FF7B,>DE7F,>EFBA,>EFBB *A9
132         DATA >FFFF,>EEFF,>BAFF,>EABF *AA
133         DATA >FFFD,>D7FF,>B7FD,>EFBB *AB
134         DATA >FF5F,>F5DF,>7ADF,>FA5F *AC
135         DATA >6AFF,>5DFF,>77DD,>77DD *AD
136         DATA >FFF1,>C6C6,>0087,>0FFF *AE
137         DATA >DF9F,>3F7F,>FCFE,>F0FF *AF
138         DATA >EEFF,>EFCB,>0000,>00FF *B0
139         DATA >EAFF,>CFCF,>0000,>00FF *B1
140         DATA >EFFF,>CDD7,>0000,>00FF *B2
141         DATA >7FDB,>F601,>013E,>C000 *B3
142         DATA >77DE,>767E,>0000,>0FFF *B4
143         DATA >FFFF,>FFFF,>E3E3,>E3FF *B5
144 *DARK GREEN ON DARK BLUE
145         DATA 56,>0DC0
146         DATA >0000,>0000,>010F,>1F3F *B8
147         DATA >0001,>073F,>FFFF,>FFFF *B9
148         DATA >0000,>0000,>0000,>00FF *BA
149         DATA >0000,>0000,>0000,>FFFF *BB
150         DATA >FF7F,>07FF,>FFFF,>7F00 *BC
151         DATA >0000,>0000,>0000,>C0FF *BD
152         DATA >0100,>F87F,>3F07,>0000 *BE
153 *BLACK ON DARK BLUE
154         DATA 32,>0E00
155         DATA >0404,>0002,>0200,>0000 *C0
156         DATA >0080,>8020,>2080,>8828 *C1
157         DATA >2000,>0000,>0000,>0000 *C2
158         DATA >80A0,>2808,>0405,>0100 *C3
159 *******DATA >0000,>000C,>0118,>C318 *C4
160 *******DATA >0000,>D803,>9800,>3000 *C5
161 *BLACK ON ORANGE
162         DATA 24,>0E40
163         DATA >0000,>0000,>0018,>0300 *C8
164         DATA >0000,>3001,>CC00,>3300 *C9
165         DATA >0060,>0098,>0000,>6000 *CA
166 *ORANGE ON DARK BLUE
167         DATA 88,>0E80
168         DATA >0000,>0107,>1F3F,>7FFF *D0
169         DATA >0000,>0001,>071F,>7FFF *D1
170         DATA >0000,>70F8,>FCFC,>F8F8 *D2
171         DATA >FFFF,>FFFF,>FFFF,>FFFF *D3
```

```
       DATA >F0F0,>C000,>E0F8,>FCFE *D4
       DATA >7F3F,>1F03,>0000,>0000 *D5
       DATA >FEFE,>FCF8,>F000,>0000 *D6
       DATA >0F07,>0000,>0000,>0000 *D7
       DATA >E0C0,>0000,>0000,>0000 *D8
       DATA >007E,>FFFF,>FFFF,>FFFC *D9
       DATA >0000,>F0F0,>F0E0,>0000 *DA
*DARK BLUE ON CYAN
       DATA 32,>0F00
       DATA >FFFF,>FFFF,>FFFF,>FFFF *E0
       DATA >0000,>0000,>0000,>0007 *E1
       DATA >0000,>0000,>0010,>78FC *E2
       DATA >0000,>0000,>0000,>0000 *E3
*PUT CHARACTERS THERE
       DATA 6,32,>20           *ROW #
       DATA >E1E2,>E1E2,>E1E2,>E1E2 *2
       DATA >E1E2,>E1E2,>E1E2,>E1E2
       DATA >E1E2,>E1E2,>E1E2,>E1E2
       DATA >E1E2,>E1E2,>E1E2,>E1E2
       DATA 10,>8E,>BABB            *5
       DATA >BBBD,>D0D9,>DAD1,>D2E0
       DATA 14,>A9                  *6
       DATA >B8B9,>B5A8,>A9AA
       DATA >ABAD,>ADAC,>ACAA,>D3D4
       DATA 14,>C9                  *7
       DATA >BEBC,>AEAF,>B0B1
       DATA >B2B4,>B3C8,>C9CA,>D5D6
       DATA 10,>EC,>C0C1            *8
       DATA >E0C3,>E0E0,>E0D7,>D8E0
       DATA 2,>10D,>C2E0            *9
*SPRITE DESCRIPTOR BLOCKS   *SPRITE #
       DATA 1,256,>0400
       DATA >1F07,>0303,>0307,>070F *1
       DATA >1F3F,>FFFF,>FFFF,>FFFF
       DATA >80C0,>E0FC,>FFFF,>FFFF
       DATA >FFFF,>FFFF,>FFFF,>FFFF
       DATA >0000,>0000,>0000,>0000 *2
       DAT [illegible]
       DATA >0000,>0000,>0000,>0000
       DATA [illegible]
       DATA [illegible]
       DATA >0100,>0000,>0000,>0000
       DATA >0080,>6070,>D0D8,>7874
       DATA >D4D8,>7830,>0000,>0000
       DATA >38F8,>D85E,>5735,>1D0E *4
       DATA >0601,>0000,>0000,>0000
       DATA >0000,>0000,>0080,>C0C0
       DATA >A0A0,>E060,>0000,>0000
       DATA >000F,>FFFF,>FFFF,>FFFF *5
       DATA >FFFF,>FFFF,>FFFF,>FFFF
       DATA >40F8,>FFFF,>FFFF,>FFFF
       DATA >FFFF,>FFFF,>FFFF,>FFFF
       DATA >0000,>80FF,>5FF5,>DF7A *6
       DATA >DFFA,>5FFF,>FFFF,>FEE0
       DATA >0000,>00EF,>5FF5,>DF7A
       DATA >DFFA,>5FFF,>F[illegible],>0000
       DATA >0000,>0000,>0000,>0000 *7
       DATA [illegible]
       DATA >0000,>0000,>0000,>0000
       DATA >0000,>D803,>9C00,>3000
       DATA >0003,>0030,>03C0,>0007 *8
       DATA >3EC0,>0019,>0003.>0000
       DATA >3000,[illegible]
       DATA [illegible]
*COLORS
       DATA 1,8,[illegible]
       DATA [illegible]
[illegible]
[illegible]
       DAT[illegible]
       DATA >366[illegible],>8[illegible]09,>347[illegible],>[illegible]009 *3,4
       DATA >3486,>9C04,>2490,>940C *5,6
       DATA >1793,>980[illegible].>2[illegible]69,>9001 *7,8
       DATA >D000
```

Hydro from page 8

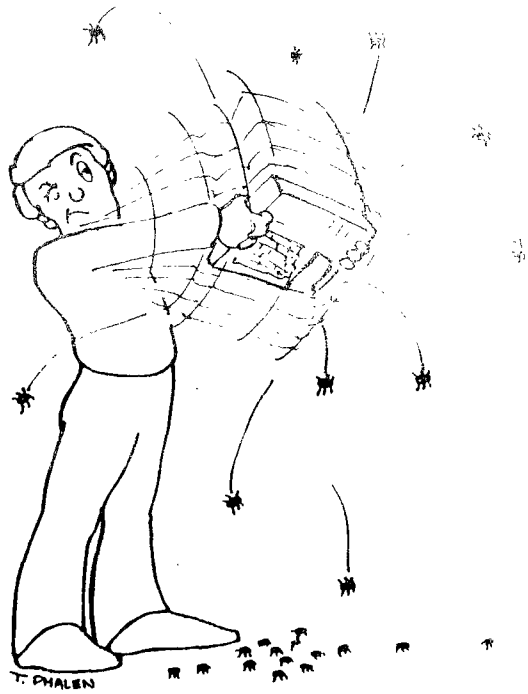
```
770 PRINT #1:CHR$(27);CHR$(7
7);CHR$(86)
780 PRINT #1:
790 PRINT #1:CHR$(14);" ~~~
ONTARIO HYDRO ~~~"
800 PRINT #1:"DATE";CHR$(9);
" #  ";CHR$(9);" Read";CHR$(
9);" Kwh";CHR$(9);" 1/4"
810 PRINT #1:"----";CHR$(9);
"Days";CHR$(9);" -ing";CHR$(
9);" Used";CHR$(9);" Cost"
820 PRINT #1:"     ";CHR$(9);
"----";CHR$(9);"----";CHR$(9
);"----";CHR$(9);"----"
830 GOTO 230
840 PRINT #1:D$;CHR$(9);E;CH
R$(9);R;CHR$(9);K;CHR$(9);C
[illegible] GOTO [illegible]
[illegible] PRINT #1:
870 PRINT #1:CHR$(14);"    --
- SUMMARY ---"
880 PRINT #1:"Kwh used(last
4 readings).......":TKWH
890 PRINT #1:"Total # DAYS u
sed (last 4 rdgs..";TE
900 PRINT #1:"TOTAL COST (la
st 4 rdgs. .......";TC
910 PRINT #1:"      "
920 PRINT #1:CHR$(14);".....
 CONCLUSION ....."
930 PRINT #1:"ANNUAL Kwh use
d.................";AKWA
940 PRINT #1:"Av.daily Kwh
used (last 4 rdgs)";KD
[illegible]
[illegible]
..................[illegible];DC
970 PRINT #1:"ANNUAL COST...
..........
980 PRINT #1
990 PRINT #1
1000 CLOSE #1
1010 DATA [illegible]
0,95,84
1020 DATA [illegible]
0,85,09
1030 DATA [illegible]
0,98,90
1040 DATA [illegible],88,8807,120
[illegible]
```

END

DEBUGGING

by

Debugger



I thought the Editor had done it to me again in the September issue. All the Extended BASIC keyed in and ran like a dream.

"Well," I said to myself. "Looks like another column I'll have to fake for the magazine."

And then I tackled Iain's ANIMAL program. First, I dug out all my listings on the animals that had been keyed in as single entries, eliminated all the programming, thus leaving all the DATA intact.

Then I removed all the END statements from each file. (This I didn't do the first time through).

Then, following Iain's instructions I assembled the whole thing and saved it in memory image format. So far so good.

Now for the program itself. It was summarily keyed in and assembled, and promptly crashed on the first attempt to run it.

I shut the console off, put in my Mini-Memory, and ran a dis-assembler I have just to get a look at the contents of my expansion RAM. (remember, shutting off the console does not erase the contents of the expansion RAM, only the pointers to it. I like the Mini-Memory dis-assembler because it loads in the 4K of RAM in the module and allows me to see all the expansion RAM, as nothing is overwritten by the program.)

A quick look at the beginning of the high memory RAM told me nothing had been loaded. (I am writing this as if I just sat down at the console and did it... in fact, the whole process took just over one week to complete)

Why wasn't the first program file loading? A thought occured to me. The program file ANT-KAN occupies 33 sectors. A bit of quick multiplication told me that 8448 bytes make up 33 sectors. Out came the HEX-DEC converter. Turns out, the buffer needed in the VDP PAB is >21CC.

Those BITty GREMLINS had struck again!

I had further problems getting the program to run, but they were of my own creation. In the throes of debugging it, I made some rather extensive changes to portions of the program, and inadvertently left out necessary coding. But that was my problem (and the bulk of that week to debug it).

If you had the same problem I did, just change line 315 of the published program to DATA >2100, and it should work fine.

One of the changes I made may interest the group, as I assembled all the files and have, currently, 13 animals appearing on screen.

Assemble the ANT-KAN file as Iain suggests, and assemble the INS-ZEB file as follows;

```
        DEF SLOAD,SFIRST,SLAST
SLOAD
SFIRST B    @START
START   COPY "DSK1.INSS"
        COPY "DSK1.JACS"
        COPY "DSK1.KANS"
        COPY "DSK1.LIOS"
        COPY "DSK1.ZEBS"
SLAST   END
```

Now, you will have to make some changes to the program to read the files. Lines 50 to 67 are fine as published. Starting at line 68, you'll have to set some new pointers to read the current files. Change line 68 to point to >C4B0, line 70 to point to >C6EE, line 72 to point to >CAFE, and line 74 to >CE26. (at least that's where the files start on my system. Use TI's DEBUG program to inspect memory locations near these to find the start points on your system.) Or, if you are worried that the DEBUG program may overwrite some of the data, use a dis-assembler that will load and run in BASIC as it will load in VDP RAM, and will allow you to view expansion RAM undisturbed.

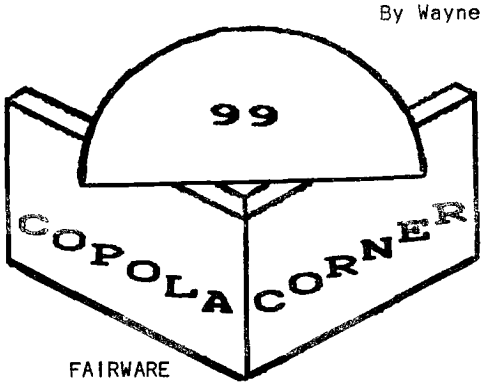
The beginning of each DATA file will look something like this;

```
0000
01E0
070X - X being a different num.each file
```

This way, each month there is a new file, include it between the LIOS and ZEBS files, the Lion file ends at >CE24, thus >CE26 is the start of the new file. Just go in to find the start of the ZEBS file, change the program to suit, and as long as Iain includes the

Continued on page 2

By Wayne Anderson



FAIRWARE

Fairware, also called Freeware and Shareware, is a fascinating concept. Software authors, most of whom are not "professional" programmers and who created their works because of a lack of software for their own use, distribute their material to all other users essentially free of charge. They rely on the honesty of the persons receiving and using their programs to compensate them for the many hours of work in providing something useful to the computing community. Some suggest a fee that they believe is appropriate while others leave us to decide what the material is worth.

How many of us take the time to thank these people for their time and effort. "Not many" seem to be the answer. The long-term cost of this attitude very likely will be that these talented people will not bother to share their talent with us. And where would we be then? I appreciate the opportunity to test software before buying. Too often I have paid the $15 to $60 for commercial software and found it inferior to the Fairware in my library.

In an effort to encourage Shareware authors to continue their support of the TI many user groups have begun collections at their meetings. Channel 99 has done the same. The monies are sent to the various programmers who very often have shown their appreciation by sending us updates or new programs.

The usefulness of some of this Fairware dictates that the small donation made at the meetings is not enough! Why not write the author and say....."Your efforts are really worthwhile. Please continue.". Include a few dollars along with the letter. I'm sure we will all benefit.

There are several Fairware items in the Channel 99 library. We are attempting to increase the number so that everyone has access to this excellent material.

These titles are available:-

c99 Compiler by Clint Pulley
A0196-SAF/1/2

Disk Manager 1000 by Bruce Caron et al
A0144-SAF

FAS-Tran by Bill Harms
E0195-SAF/1/2

Funlwriter by Tony and Will McGovern
A0143-SAF/DISK

Neatlist by Danny Michael
A0091-SAF

Screendump by Danny Michael
A0092-SAF

Super Cat by Larry Duke and Scott Beeker
A0194-SAF

TI99-OPOLY by Ross Mudie
E0193-SAF/DISK

All programs may be ordered from the Librarian from Tom Arnold at the monthly meeting [illegible] time.

PROGRAM REVIEW FUNLWRITER V 3.3

One of the best pieces of Fairware available is Funlwriter, written by a pair of Australians, Tony and Will McGovern. This is by far the most versatile program available for the T.I. It allows you to use T.I. Writer and the Editor/Assembler without the command modules. Disk Manager 1000, another very popular program, is also included as well as a disk sector reader/editor, a Forth loader and c-Compiler by our own Clint Pulley. To make use of this program you need a console, 32K memory, and one disk drive. It is also very helpful to have a second disk drive and a printer.

There are six documentation files included which should be printed out using T.I. Writer and read before you attempt to beg it. Make sure to make a back-up [illegible] place.

Examining the load program will reveal the lines necessary for changing things within Funlwriter. Line 120 controls the colour of the screen and colour. Lines 130 and 140 set defaults for the printer type, either parallel or serial output. Lines 150-190 may be used for entering the names of programs which you would like on the User's List. Lines 240-280 contain the load commands for the User's List programs. Do NOT RESequence the Load program because it will be destroyed.

The first menu which comes on the screen asks you to choose TI-Writer, Editor/Assembler or the User's List. If you select option [illegible] menu will be displayed. This shows the Editor, Formatter, DM 1000, Utility, Switch and Reset. Selecting Switch changes the menu so that option 2 reads c-Compiler. Selecting Switch again changes option 2 to Modem, then to Disk Edit and then to Assembler.

Pressing Reset stores the name of the file you are currently working with in memory so that if you use one of the other utilities on the disk then [illegible] to either the Editor or the Formatter [illegible] will be included in the name of your program.

Utility, option 4 displays five various assembly file loaders.

Option 1 will load various TI-Writer utilities such as spell check programs.

Option 2 sets up a GPL environment for loading program image files.

Continued on page 25

C BASIC

by
David Storey

As I promised last month I will be talking about the [WHILE] statement. But before we continue with the game we will have to understand more about the C-BASIC control commands. In normal BASIC we have:
[IF-THEN-ELSE/FOR-NEXT-STEP/GOTO and GOSUB].

In C-BASIC we have:
LOOPS = for - while - do
DECISION and CHOICE = if - else - switch
JUMPS = break - continue NOTE:- goto is not supported in c99c.

This month we will discuss the [while] loop.
Keyword = while.
The while statement creates a loop that repeats until the test expression becomes false or zero.

Let's start with a short program segment that goes through two loops five times each. Each time through both loop it will print "Number" and a number that represents the number of times it has been through the loop. Following that it will print either "First while loop" or "Second while loop."

The first while loop increments the column that the line is being printed on each time through the loop. When it exits the loop it will print "End of First While loop." The second loop does the same thing except it will decrement the column it prints on each time through the loop. When it exits that loop it will print "End of Second While Loop".

The program looks like this:

```
/*************************/
/* c99c c-basic compiler */
/*  The while statment   */
/*   by David Storey.    */
/*************************/
/* LOAD WITH:- CSUP      */
/*             PRINTF    */
/*************************/

#asm                          /* start of assembler code */
 REF PRINTF                   /* place printf in ref table*/
#endasm                       /* end of assmblr & cont.with c-basic */
main()                        /* start of main       */
{
int row,colm,c,tim;           /* set up [illegible]ariables */
  putchar(12);                /* clear [illegible] curser*/
row=0;                        /* make row [illegible] */
colm=3;                       /* make column =3   */
locate (row,colm);            /* place curser at row colm */
tim=0;                        /* set tim to zero */
 while(colm++ < 8)            /* start while [illegible] checks if [illegible] */
                              /* [illegible] loop, incr[illegible] */
printf("Number %d First While Loop",[illegible]); /* print [illegible] */
locate(++row,colm);
}                             /* similer to next in basic */
locate(11,9);
puts("End of First While loop ");
colm=7;
row=1[illegible];
[illegible]
[illegible]
[illegible]
{
printtf("Number %d Second While Loop",++tim);
locate(++row,colm);
}
locate(20,8);
puts("End of Second While Loop \n");
}                             /* end of main        */
```

You will have noticed some new statements that were not used last month and they are :- #asm
#endasm
REF
PRINTF

First, the #asm and #endasm go together. These two commands let you use assembler code directly within your C-BASIC program. I used them to place a REF PRINTF in the program. This allows us to use the statement PRINTF which you have to load and run with your program and CSUP. When using #asm - #endasm the spacing of the assembler code between these two statements is very important. You will notice that there is a space before REF PRINTF, this is required. If there is no space the compiler places the REF statement in the wrong column which will result in an error. So look out for this in future listings. The PRINTF file is on the same disk as your c99c compiler.

The file gives you the ability to print a string with a variable within the printed string. If you take a look at the PRINTF statement in the listing you will notice the [%d]. This is where the variable [tim] is placed in the string. At the end of the string you will see the variable to be printed. You can have more than one variable in one PRINTF statement. This is how you could do it: printf("string %d string %d string",vari1,vari2); The first variable would appear at the first [%d]; the second variable would appear at the second [%d].

Here is a list of the C-BASIC identifiers.

IDENTIFIER	OUTPUT
%d	decimal integer
%c	a single character
%s	character string
%u	unsigned decimal integer
%o	unsigned octal integer
%x	unsigned hexadecimal integer

The general form of the while loop is:-

```
while(expression)
statement
```

```
/**************************/
/* c99c c-basic compiler */
/*  The while statment    */
/*    by David Storey.    */
/**************************/
/* LOAD WITH:- CSUP       */
/*             PRINTF     */
/**************************/
```

The expression can be a mulititude of things. As you can see in the program we are using a variable incrementing it, then checking to see if it is less than 8. The expression can be more complex or it can be very simple as in the second program. In the second program I have used an endless while loop to keep the program running until a certain key is depressed.

When you construct a while loop, it must include something that changes the value of the test expression so that the expression eventually becomes false. Otherwise, like in the second program, the loop only ends when FCTN 4 is hit.

Let's look at a fragment of a program:-

```
times=1;
 while(times < 8)
   puts("hello \n");
```

This fragment will print "hello" indefinitely because nothing in the loop changes. The value of times is always less than 8. So we have to change the value of times like this:-

```
times=1;
 while(--times < 8)
   puts("hello \n");
```

This fragment changes the value of times by decrementing it, but this will not work as the value of times will never be a other than 1. This will have the same result as the first fragment. To get the first fragment to work it should look like this:-

```
times=1;
 while(times++ < 8)
   puts("hello \n");
```

Now we are incrementing the value of times so that it's value will at sometime be greater than 8. The second listing is only slightly different. This time instead of going to the c99c end screen, it waits to scan the keyboard. If the space bar is pressed the program returns to the beginning. If FCTN 4 is pressed the program returns to the c99c end screen.

Enter these shorties, play around with the values of the variables and see what happens. Although this is not a program as such of the what it is doing it will give you a start and next month we will look at the while statement and continue the loops. Until next time

```
#asm
 REF PRINTF                  /* place printf in ref table*/
#endasm
main()
{ while(1)
{ int row,colm,c,tim;
  putchar(12);               /* clear screen home curser*/
locate (2,7);                /* place curser at row2 column 7 */
  puts("Press Space Bar to Continue."); /* just like print */
locate(3,7);
  puts("FCTN 4 TO END.");
```

```
getchar();                    /* get input from keyboard */
poll();                       /* scan keyboard */
                              /* make row =5      */
colm=3;                       /* make column =3   */
locate (row,colm);            /* place curser at row colm */
tim=0;                        /* set tim to zero */
 while(colm++ <8)             /* start while lp chcks if colm<8 */
                               /* if yes go through loop,increment colm */
printf("Number %d First While loop",++tim);   /* prnt While lp # */
locate(++row,colm);
}                             /* similar to next in basic */
locate(11,9);
puts("End of First While loop ");
colm=7;
row=5;
tim=0;
locate(row,colm);
while(colm-->2)
{
printf("Number %d Second While Loop",++tim);
locate(++row,colm);

locate(20,8);
puts("End of Second While Loop \n");
getchar();                    /* get input from keyboard */
poll();                       /* scan keyboard */

}
```

Check it out from page 6

```
260 IF D$="" THEN DISPLAY AT
(20,1):"Date?" :: ACCEPT AT(
20,8):D$ :: IF D$="" THEN D$
="n/a"
270 DISPLAY AT(5,1):"PRESS
TO": : :"    1  Update accou
nt":"    2  Reconcile statem
ent":"    3  QUIT (save data
)":"    4  ""HELP""" :
280 GOSUB 720 :: IF K<49 OR
K>52 THEN 280 ELSE ON K-48 G
OTO 460,370,300,290
290 CALL HCHAR(2,1,32,23*32)
: GOSUB 720 :
: GOTO 250
300 IF F$<>"1" THEN 450 ELSE
 CALL HCHAR(2,1,32,23*32)::
DISPLAY AT(4,1):"Save data?
   y/n  Y"
310 DISPLAY AT(6,1):"Data wi
ll be lost otherwise."
320 ACCEPT AT(4,20)SIZE(-1)B
EEP VALIDATE("YN"):Y$
330 IF Y$="Y" THEN 350 ELSE
420
340 DISPLAY AT(4,1):"Insert
data disk or tape    and pre
ss any key..."
350 CALL KEY(0,K,S):: IF S=0
 THEN 350
360 IF K=15 THEN 350 ELSE DI
SPLAY AT(8,1):"Enter filenam
e":"DSK1.MYFILE" :: DISPLAY
AT(12,1):"Enter a blank file
name to   return to menu."
370 ACCEPT AT(8,1)SIZE(-15)B
EEP:FN$
380 IF FN$="" THEN 32767
390 OPEN #1:FN$,INTERNAL,OUT
P ,FIXED :: PRINT #1:
DATA"
400 PRINT #1:NR :: FOR I=1 T
O NR :: PRINT #1:ACN$(I),CBA
L$(I),DATE$(I):: NEXT I
410 CLOSE #1
420 F$="1"
430 DISPLAY AT(12,1)ERASE AL
L BEEP:"QUIT...Are you sure?
 Y/N  N" :: ACCEPT AT(12,27)
SIZE(-1):Q$
440 IF Q$="Y" THEN CALL CLEA
R :: STOP ELSE GOTO 250
450 DISPLAY AT(12,1)ERASE AL
L BEEP:"Nothing to save..."
:: CALL D(250):: GOTO 430
460 ON ERROR 1240 :: GOSUB 7
30 :: DISPLAY AT(4,1):"PRESS
  TO"
470 DISPLAY AT(7,5):"1  Load
 data from":TAB(8);"tape or
disk";TAB(5);TAB(5);"2Add or
 delete account"
480 DISPLAY AT(10,5):"3  Upd
ate account":"    4  Return
to main menu":"       (QUIT)
"
490 GOSUB 720 :: IF K<49 OR
K>52 THEN 490 ELSE ON K-48 G
OTO 500,790,550,250
500 IF F$="1" THEN CALL MESS
AGE :: GOTO 460
510 CALL HCHAR(2,1,32,736)::
 DISPLAY AT(5,1):"Filename?
DSK1.MYFILE":"" :: DISPLAY A
T(12,1):"Enter a blank file
ame to   return to menu."
520 ACCEPT AT(8,1)BEEP SIZE
(-15):FN$ :: IF FN$="" THEN
460
530 OPEN #1:FN$,INTERNAL,INP
UT ,FIXED :: INPUT #1:A$ ::
IF A$="CHECKDATA" THEN 540 E
LSE CALL MESSAGES :: CLOSE #
1 :: GOTO 250
540 INPUT #1:NR :: FOR I=1 T
O NR :: INPUT #1:ACN$(I),CBA
L$(I),DATE$(I):: NEXT I:: CL
OSE #1 :: F$="1"
550 IF F$="0" THEN CALL MESS
AGE2 :: GOTO 460 ELSE GOSUB
730
```


To be continued next mont

CLUBPAGE

By Tor Hansen

Tom Arnold opened the September session of the User's Group with a quick summary of the financial situation the group is in. Included was a rather terse message to the membership to purchase more from the Club library, as December looks like a rather lean month. To keep the group out of the red, members must purchase more from the library. Come on people, there must be something in there that you like!

There were almost forty people at this session, and they witnessed 4 full systems and one more-than-full system. More on this one later.

Tom didn't have any new "Fairware" to offer this month. Unfortunately.

What Tom has been doing at previous sessions has been to ask for a donation, contributed later, as no money may change hands at the Spectator, for the "Fairware" of the month. This money has been forwarded to the people who create the software that we all enjoy.

A more-than-worthy cause! Question. How many of you who have gained from this practice have sent some financial consideration to the source of this practice? (i.e. YOUR CLUB)

Tom has been going out of his way to make this all available to you, THE MEMBER. Please, remember the source, and include it in your considerations. The Club has to survive, too.

CLUB NEWS

By Tor Hansen

Clint Pulley has been invited to a Tech Fest in Chicago this November (as you read this column).

There, he will deliver a speech dealing with the details surrounding the creation of his "c" Compiler. The membership gets to hear this speech in October, and, I for one, am looking forward to hearing it.

Clint had a prototype of the new Myarc Computer, Geneve, running at this session. (not the one I alluded to earlier).

The LINES program indeed danced across the screen in a highly accelerated version of the TI program we are all used to. This appears to be a function of the VDP processor more than the CPU. It seems with the new processor, all you have to do is define the start and stop positions for the line, and the chip will do the rest. The original LINES program draws the whole line.

Mr. Gil Tennant from New Horizons was present with the new hardware for our machines. (This is what I was alluding to earlier).

He has a new set of hardware for our machines that seem to do almost the same as the new Myarc equipment.

I was too rushed to get all the details. Talk to Malcolm, as I think he got more out of this than I did.

Don't forget that December is election month. Make sure you show up to nominate and elect your executive committee for 1987.

Also don't forget that for most of you December is the last month you are paid up for receiving this publication. In the new year you will be on your own to arrange for this publication.

Things are far from dead with our systems. Malcolm has new software available in his store, and there is the promise of more to come.

Stay tuned for more developments.

1986 CLUB MEETING DATES

Friday 10 January	Friday 23 May	Friday 19 September
Friday 28 February	Friday 27 June	Friday 24 October
Friday 28 March	Friday 25 July	Friday 21 November
Friday 25 April	Friday 22 August	Friday 19 December

All meetings are held from 7 P.M. to 10 P.M.

"WHO'S-WHO"

CHANNEL 99 USERS GROUP HAMILTON.

PRESIDENT
Tom Arnold.....phone (416) 385-5576
VICE PRESIDENT
David Storey.....phone (416) 387-1295
SECRETARY
Malcolm Johnson.....phone (416) 528-5756
TREASURER AND CHIEF LIBRARIAN
Tom Bacolini.....phone (416) 734-9958

CHANNEL 99 LIBRARY

COPOLA CHAIRMAN
Wayne Anderson.....phone (519) 632-7329

CLUB PAGE
Tor Hansen.....phone (416) 279-0437

Any and all written communications should be addressed to:
CHANNEL 99 HAMILTON
P.O. BOX 1005 Station 'A'
HAMILTON ONTARIO
CANADA L8N 3R1

THE INCREDIBLE SHRINKING PROGRAM

by Tom Arnold

How would you like to write a program that shrinks as you write it? I have found a way to do this! However before you get too excited I must warn you that it won't do you any good as this method is caused by a flaw in the language. Maybe I should begin at the beginning. I had written a program which I though might be a good program to show the merits of Myarc XBII. I loaded the program into the computer and started to edit it to conform to the Myarc XBII version 2.1. After several hours of revisions my computer spoke to me, "OUT OF MEMORY" I tried to re-edit it but it wouldn't let me. I saved the program to disk and tried to run it. No go! Size indicated that there was only about 600 bytes left (all SIZE commands are estimates as I didn't keep track). In desperation I deleted all the REM's. Still no luck, in fact the program size did not even shrink. Now I was really desperate so I deleted a whole large subroutine. Size was almost the same. I then saved to disk and noticed that the number of sectors used was 97!! I was saving into Ram Disk so you can CALL FDDIR to see the catalog. I now realized that the system to keep track of program size is defective in version 2.1 of the Myarc XBII.

I now loaded TI XB and loaded the program into it. SIZE indicated that the program had now shrunk to an expected level. I loaded this into the Myarc XBII and did some more editing. Each time I looked at the SIZE. The program grows as you make changes, any change at all! Thinking there must be a simple way of solving this problem I saved the program in MERGE format. After typing NEW I loaded the MERGED program. SIZE was now what it would be in TI XB. A solution!! Your final program version should be saved in MERGE format , loaded using MERGE and then saved normally. This will give you a normal sized and runable program.

Now I haven't mentioned the shrinking program yet you ask? Well I inadvertently MERGED the program on itself instead of typing NEW first. When I sized the program it was several thousand sectors lower than normal! I realized this and decided to MERGE it into itself again and again. After three or four merges, I can't remember the exact number, the program shrunk to 880 bytes! I didn't try to go lower. However this 23K program normally would take up about 45 sectors on the disk. When I saved the 880 byte version it took up 97 sectors!

What have I learned from all this? Mainly that the Myarc XBII is flawed but there is a way to overcome it. Either save all programs in MERGE format and then reload and save using SAVE or save your final version and reload into TI XB and resave in that language. Both methods will reset the program to it's proper size.

Play with Sparky from page 7

jump must be made before the end of the pad or even from the far end. To achieve high scores on this game takes a lot of practice. I have found several patterns that will complete the levels fairly quickly but no one pattern on any level that makes a great deal of difference.

I have reached level 8 but never completed it as there is usually only one man left when I arrive. I consider any score over 30,000 as being okay.

There is only one thing in MINER 2049er that bores me and that as usual is the theme song as Bob sets up the Title screen and goes through the short Demo Mode. Everything else is top rate. The graphics are good and there are no surprises popping out of the hat at unexpected times to upset your carefully laid plans. MINER 2049er sucks you in and forces you to continue striving for a new high score.

So there, now you have a target to shoot at. Remember to send in your high scores. Support them with a photo or negative because every so often the high scores will be given in the Hall of Fame.

Send in your own review - tell us what you like or don't like about any of the arcade style games - send in tips on strategy - suggest games that you would like to see reviewed.

The computer industry has a saying---GIGO (garbage in, garbage out.) Computer Club magazines have another---NINO (no input, no output.) Dust off your word processor or heaven forbid, use a pencil and paper and let's hear from those games players out there.

Copola from page 20

Option 3 is RUN PROGRAM FILE from the Editor/Assembler.

Option 4 is LOAD AND RUN also from the Editor/Assembler.

Option 5 allows immediate re-entry to a program without reloading it

One of the most impressive parts of the program is the User's List. It contains Dpatch, a disk sector reader and TI-Forth, a Forth loader which still requires Forth. It also comes ready to load the Myarc DM, if you have it connected to your system. Another five utilities can be added to the list, and programs on the list can be changed at any time by changing the LOAD program.

This is truly a fine piece of software so let's see that its writers are compensated for their great work on this program.

Review

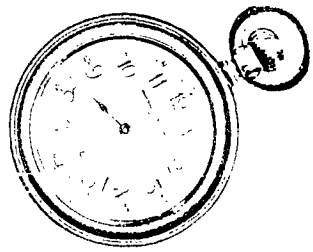
	Mark	Max.
Ease of Use	9	10
Performance	10	10
Documentation	8	10
Over-all Value	10	10
General Impression	9	10
Total	46	50

END

OPTION 3

by

V.C. MacArthur



I have recently undergone a change of jobs. Some might call it a promotion, while others would just stand aside and snicker.

But this change has engendered a completely new problem with my programming! And maybe I am beginning to understand why a lot of you are not responding to this, and other columns in this magazine.

It's the problem of time.

If you have a tough and demanding job (as I now have), you have hardly enough time to sit and enjoy your own diversions with this accursed machine. And once a month, you get your issue, and dutifully type in the programs, run them, and, hopefully, enjoy them.

But you don't have the TIME to send us your reactions.

How do you spend your valuable time with your computer? How do you FIND time to spend with this domineering machine?

I try to split my time between playing, and programming. After all, no play and all work etc.!

But most importantly, I try to ENJOY my machine. And the simple fact is that I enjoy writing this column!

Even if nobody reads me, I will still enjoy contributing to this project. Because, for me, it's part of the fun of owning a toy that can perform so many marvelous and magical functions.

If you don't have the time to send us a note, I promise to stop yelling at you. However, if you can, even if you give Ted or myself a disk file at the meetings,(when, of course, I have the time to attend them!), then maybe you, as well, can feel a new enjoyment.

The enjoyment of pulling your weight. The enjoyment of contributing. The enjoyment of putting your own mark on this club, and indeed, the world wide club of TI/99 owners. Don't be intimidated. Don't think your stuff has to be perfect the first time around.

That's what editors are for!

Even if you're working on a program that isn't running properly, give us a copy! Even if you don't want it published!!!

We are supposed to be the experts!

We want to help you learn how to program better.

We want to help you to enjoy your machine to the fullest.

No matter how much time it takes!

DON'T TAKE IT APART

by Dick Tilley

I had a problem with my ss/sd disk drive system recently that was a little strange to say the least.It would read well,it would write fine,but it would'nt initialize my disks.Perhaps one time in ten,it would work. Usually I would get a "NO DISK IN DRIVE" error reading.

After much fiddling about and frustration I decided to give everything a good cleaning. As soon as I got the metal cover off I could see the problem. One of the tiny leads on the light emitting diode that beams through the hole in the floppy disks was broken. The connector a pinch type of thing, was holding the broken lead in a manner that allowed it to contact, sometimes.

That should have been all there is to this story, however I didn't know that you could easily pop out the diode and pop in another one. MR. FIXIT had to disassemble the door mechanism.In reassembling it, something got FOULED UP. Now my drive is in for repairs.

What should have been a five minute and $1.80 repair is now going to be something quite different.

I hope I have helped someone out there who may have had the same problem with their drive,and I hope also they learn from my mistake and fix their's the easy way.

By the way, the diode was obtained from MAYBE ELECTRONICS, 4190 Fairview Ave. Burlington, Ph.632-4345(Good people). Cost, about $2.00.

Big program from page 5

```
710 CALL DELSPRITE(#2,#3,#4
:: CALL SOUND(-700,-8,0):: S
CORE=SCORE+20 :: HIT=HIT+1
720 IF HIT=5 THEN 800
730 RETURN
740 DISPLAY AT(10,1):"  HE
OT YOU...."
750 CALL SOUND(-1000,110,0)
760 CALL DELSPRITE(ALL)
770 FOR LOOP=1 TO 2000 :: NE
XT LOOP
780 MEN=MEN-1 :: IF MEN<>0 T
HEN 250
790 CALL CLEAR :: PRINT "I'M
 SORRY BUT ALL YOUR MEN  ARE
 DEAD."
800 DISPLAY AT(10,1):"  STAN
DBY FOR YOUR SCORE" :: CALL
DELSPRITE(ALL)
810 OPEN #1:"DSK1.SCORE/REC"
820 PRINT #1:HSE:SCORE:MEN:N
AME$
830 CLOSE #1
840 RUN "DSK1.SCORCAPE"
```


ENE

UH—OH

by
Ron Marissen

Well, I feel just awful now. At the September meeting I found something that is incompatible with my 16 bit memory...sort of. It seems that Mechatronics' Extended Basic graphics routines don't work correctly. I can only guess that the routines, which work fine with my console and a 32K card, don't allow time enough for the VDP to respond correctly. I assume that the routines use Bit Map mode and maybe the VDP responds slower when in this mode. I've tried things such as TI-Artist, Graphx, and numerous games (my kids play them, not me), and have had no problem. Maybe someone with more savvy could enlighten me - remember, I'm no expert.

There are a few other things I have tried since my last submission that I will mention now.

1-I plugged a speech synthesizer into my console and powered up. I guess that it truly is too much of a load on the 5 volt supply as all I got was a blank screen. I have a fix for that in the back of my mind though.

2-Speaking of speech (cute eh?), I had the fellow from Toronto, who's name escapes me, hook up to his P-E box which had a Triple-Tech card in it. The speech seemed to work OK with the 16 bit memory. This is something I was worried about after reading the E/A manual.

3-Aside from the aforementioned incompatibility, there is nothing I've tried that doesn't work. Incidentally, this article is being written using my 32K and Funlwriter.

Here's a partial list of programs I've tried:

-TI-Artist	-CM-1000
-Advanced Diagnostics	-XB Detective
-SXB (very fast routines)	-Super Bug
-Scrabble	-TI Disassembler
-Jaln's Animal	-RMMS
-Tombstone City	-Many games

Everything seems to work flawlessly and much quicker too. I don't understand what may be happening with Mechatronics' Basic. Oh well!, I haven't seen any programs using the routines anyway.

Thanks to Malcolm's graciousness, and the end of a busy summer, I am getting to work on the second version of the 16 bit 32K. This memory, by the way, was not meant to step on any toes as far as replacing existing 32K cards goes. As I have found out, it may have it's problem(s). I developed this for two reasons. First of all, I wanted 32K but could not even afford a boxcar memory at the time, a boat I'm sure I'm not alone in. Secondly, I just wanted to see if I could do it. Having been successful, I wanted to share it with the TI community. I'm sure that if anyone has the nerve to try it, they will be very happy.

The memory runs tape based M/L programs fine, or at least the ones I've tried (all the ones in the Smart Programmer and Micropendium). You just have to be patient while loading them.

I have a number of other ideas in the back of my mind that I am going to investigate over the winter (I hope!), and if successful, you'll be the second to know. I just wish I was a little more knowledgeable in this field...it's fascinating.

Look for my article next month and I'll show you how to install a memory in the black computer.

---

from page 19

files in alphabetical order, you're all set.

I would also suggest that you change the buffer size in the INS-ZEB file to >2100 to avoid a problem later. Also, keep an eye on the size of the INS-ZEB file as it grows. When the memory image file reaches 33 sectors, its time to start a new one.

Now that I have messed up Iain's approach to his column, please read and follow Iain's suggestions, not mine. I am only showing you what one person is doing with a program. If you feel your expertise in assembly language is up to it, experiment all you want. Otherwise, let Iain show you the way.

See you next month...

DEBUG - ADDENDUM

By Tor Hansen

The Tutor program by Jim Peterson and Brady Gill for the TE11 command module appears to have travelled a rocky road with this group. Every one who has obtained a copy from the library has had a problem getting it to run with a disk drive attached to their system.

Rest easy. The problem has been identified and solved. At the last Executive meeting, a working copy of the program was left with both Jim Peterson and Wayne Anderson. Contact either one at a Club meeting if you have a non-working version, and arrangements will be made to correct your copy.

For those interested, it appears that five sectors of nonsense were written along with the program file when the transfer was made from cassette to disk. How this happened can only be guessed at. But those five sectors were enough to cause the MEMORY FULL IN XXX error that is displayed when the program is RUN.

Just wanted to keep you updated. 'Till next month...

THE BOOK AUCTION

Tom Arnold

These are the results of our book auction which closed on November 1, 1986. The final bids have been reduced to $2.00 over the next highest bid. In the case of a tie (and there were two) we drew lots to decide who got the book. The few unsold books will have been sol[illegible] occur before you read this. [illegible] contacted you already please [illegible] arrange for your books. Congratulations on the winners, they received a super bargain.

LOT #	BID	BIDDER	LOT #	BID	BIDDER
01	$5.00	Richard Liller	[illegible]	$4.00	[illegible]
02	$4.50	Malcolm Johnson	17	$4.00	Harry Sparks
03	$4.75	Bob Dudley	18	$4.00	Andy Janosik
04	$3.00	Andy Janosik	[illegible]	$4.00	Andy [illegible]
05	$0.00	no bid	[illegible]	$5.00	Bob [illegible]
06	$0.00	no bid	[illegible]	$5.00	Bob [illegible]
07	$0.00	no bid	22	$1.25	below minimum bid
08	$5.00	Dave Wells	23	$0.00	no bid
09	$5.00	Bob Dudley	24	$2.50	Glen Watson
10	$4.00	Andy Janosik	25	$3.00	Harry Sparks
11	$4.50	Malcolm Johnson	26	$5.00	Robert [illegible]
12	$5.00	Glenn Watson	27	$5.00	Richard [illegible]
13	$5.00	Robert Dudley	28	$0.00	no bid
14	$0.00	no bid	29	$0.00	no bid
15	$1.25	below minimum bid	30	$0.00	no bid