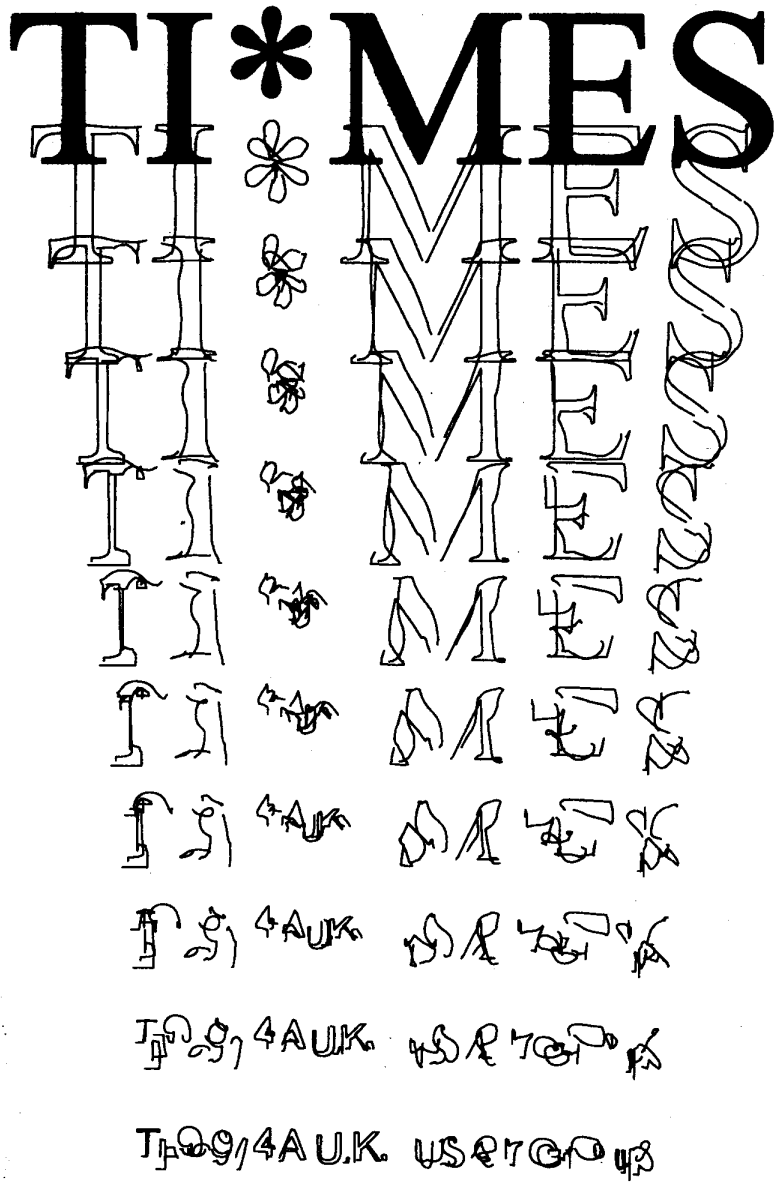
#44

SPRING 1994



TI-99/4A U.K. User Group (ISSUE 44)

TI-99/4A User's Group (U.K.)

CHAIRMAN:
Trevor Stevens. Tel: 0623 793077
249 Southwell Road East, Rainworth, Notts. NG21 0BN

VICE CHAIRMAN, CASSETTE LIBRARIAN, PROGRAMMING:
Mark Wills. Tel: 0743 350588
12 "Rosehill", Betton Street, Shrewsbury, Shropshire, SY3 7YN

GENERAL SECRETARY, AND ALSO PROGRAMMING!
Richard Twyning. Tel: 0623 27670
24 Peel Road, Mansfield, Notts. NG19 6HB
London telephone number (after 5pm) 081 5308038

MEMBERSHIP SECRETARY, BACK ISSUES:
Alasdair Bryce. Tel: 0389 65903
51 Dumbuie Ave., Silverton, Dumbarton, Scotcland. G82 2JH

TREASURER:
Alan Rutherford. Tel: 0625 524642
13 The Circuit, Wilmslow, Cheshire. SK9 6DA

TI*MES EDITOR, and the odd bit of hardware!
Gary Smith. Tel: 0636 706767
55 Boundary Road, Newark, Notts. NG24 4AJ

HARDWARE:
Mike Goddard. Tel: 0978 843547
"Sarnia", Cemetary Road, Rhos, Wrexham, Clwyd. LL14 2BY

DISK LIBRARIAN, JOURNAL EXCHANGE:
Stephen Shaw.
10 Alstone Road, Stockport, Cheshire. SK4 5AH

PUBLICATIONS:
Mike Curtis. Tel: 0209 219051
21 Treliske Road, Roseland Gardens, Redruth, Cornwall. TR15 1QE

MODULE LIBRARIAN:
Francesco Lama. Tel: 0865 721582
14 Granville Court, Cheney Lane, Oxford. OX3 0HJ

Editor's Comments

Many thanks to every one who has contributed to this issue of TI*MES!! You should be aware that the AGM is on the 14th pf MAY at Derby. It would be nice to see as many people as possable so notch this date down in your diary. There should be a demonstation of the SCSI card by Richard Twyning!

Disclaimer

All views by contributors to this magazine are strictly their own, and do not represent those of the committee. Contrary opinions are very welcome. Errors will be corrected upon request.

CONTENTS.

PAGE 1. From The Chairmans Chair. **(AGM NEWS)**
By Trevor Stevens.

PAGE 2. RND: Alternative And Epilogue.
By Walter Allum.

PAGE 8. News And Reviews From The Man With The Blues.
By Richard Twyning.

PAGE 26. Somewhere In Tl*MES.
By Richard Twyning.

PAGE 40. Consoletation Zone-Queries from Walter Allum.
By Gary Smith.

PAGE 41. Answer to Issue 43 Puzzle.
By Walter Allum.

PAGE 45. Disk Library New Additions (Nov.. 1993).
By Stephen Shaw.

PAGE 46. Games.
By Steve Burns.

PAGE 47. The Undocumented Features Of 'G'.
By Mark Schafer.

PAGE 49. Funlweb Vn 5.0 Editor.
By Stephen Shaw.

PAGE 52. Tippsy.
By Trevor Stevens.

Back Page. Map for the AGM.

Mark this date down in your diary and support your Texas user group.

This venue is very easy to find and is about five minutes walk away from the train station. There is no problem with parking and everyone is welcome.

New Members.

The 99/4A user group gladly welcomes two new members. These are : Edward Forbes and R.W.W.Gleave.

The current membership stands at 89 fully paid members with 6 outstanding from issue 43.

FROM THE CHAIRMANS CHAIR

Well I sit here writing this looking out on a scene of snow and ice. I suppose this really is the Spring issue. I hope all you TIers had a very good Christmas and a *merry* New Year. I got my copy of TI*MES a little late this last quarter. On looking inside as a result of a query I found that somewhere along the line a bug in the printing crept into the program that we were discussing. It had some how omitted the line 120. So for all you confused people out in TI land trying to figure out what Call SPeek was then here is the Real bit!!!!

```
110 CALL HCHAR(24,1,40,64)::
 CALL VCHAR(1,31,40,96):: CA
LL HCHAR(2,5,33,26):: CALL S
PRITE(£1,34,2,17,17)
120 FOR R=1 TO 2 :: FOR C=65
 TO 90 :: RANDOMIZE :: CALL
PEEK(-31808,X,Y):: CALL HCHA
R(INT(X/13)+4,INT(Y/10)+4,C)
:: NEXT C :: NEXT R :: R,C=3
```

£ = #

Thats the bugs out of the way. I must apologise for any confusion caused.

This month we are going to go further into the control of sprite and getting them to pick and drop things. This next program will pick up dots as you pass over them and also lay down a blue wall trail behind you. We will also do some minor animation which is directional. So lets do it.....

THE PROGRAM

```
100 CALL CLEAR :: CALL SCREE
N(11):: CALL CHAR(33,"000000
006060",96,"AAFFFEDFBEBBBEBF
AAFFFE8F8E9B9E9FAAFFDEFFFEF7
0EFFAAFFDEFFE60706FF")
110 CALL CHAR(100,"BFBEBBBED
FFEFFAA9F9E3B8E8FFEFFAA55FF7
BFF7FEF70FF55FF7BFF67E060FF"
):: CALL COLOR(1,15,2,2,5,5)
120 CALL HCHAR(24,1,40,64)::
 CALL VCHAR(1,31,40,96):: P=
102 :: CALL SPRITE(£1,P,16,1
7,17)
130 FOR R=1 TO 50 :: RANDOMI
ZE :: CALL PEEK(-31808,X,Y):
: CALL HCHAR(INT(X/12)+2,INT
(Y/10)+4,33):: NEXT R :: R,C
=3
140 CALL JOYST(1,X,Y):: IF X
 AND Y THEN 140 ELSE IF X TH
EN P=100+X/2 :: X=SGN(X)ELSE
```

```
 IF Y THEN P=98+Y/2 :: Y=-SG
N(Y)
150 CALL GCHAR(R+Y,X+C,CH)::
 IF CH=40 THEN CALL SOUND(-6
0,110,9):: GOTO 140 ELSE C=C
+X :: R=R+Y
160 CALL PATTERN(£1,P):: CAL
L LOCATE(£1,R*8-7,C*8-7):: I
F CH=33 THEN CALL SOUND(-100
,-7,6):: CALL HCHAR(R,C,32)
170 CALL PATTERN(£1,P+1):: I
F X OR Y THEN CALL SOUND(-90
,660,9):: CALL HCHAR(R-Y,C-X
,40) :: GOTO 140 ELSE 140
```

£ = #

That's the program. Now we will go through it to explain as usual. However I will now be expecting you to do some of your own work so try and sort out what the program is doing before you read the next bit. I found this was the best way to get into programing, do it, experiment and read other peoples programs and figure out how they work. SO GO TO IT .....

Ok so you got stuck. So here is the explaination.
Line 100 This clears the screen and starts to set up the character definitions from chars 33 then 96,97,98 with a long string. This breaks down into three banks of 16 by the computer which save space.

Line 110 This carries on with Chars 100,101,102 also in a long string format. Then we do a multi color statement. This defines set 1 and 2 to color 15,2 and 5,5 respective. This makes ASCII 32 to 39. This makes our space character (32) Black and our dots grey on a black background (33). Set Set 2 characters (40 to 47) contains our walls and the trail we will leave. All characters in this set will be dark blue blocks.

Line 120 This puts the statements Hchar and Vchar drawing our wall around the perimeter of the screen. Hchar top and Bottom, Vchar the two sides. You will note the bottom is placed first and 64 repetitions. This allows a wrap round to the top of the screen. The same goes for the Vchar which also uses wrap from right hand to left hand side. Saves on extra statements!!! We now set P equal to 102 which is the starting pattern character number for our sprite. We then set our sprite number 1 with character P which is the right facing face/mouth in the closed position. Colour for our sprite will be white and will be placed at dot col 17 which is graphics col 3.

Line 130 Now we set up a 50 times loop and with our new RND routine place dots on the screen. We start with the Randimize statement then we Peek -31808 to obtain the two numbers X and Y. Since the numbers of X and Y are 0 up to 255 we have to convert the values down for the screen places. The formula for this is in the Call Hchar Statement. This will return values of 2 to 23 for the column 4 to 29. We then increment R and make another dot on the screen until 50 is greater than 50. Having done with R we now reuse it, and set R and C equal to 3 which is the starting Row and Column positions of our sprite.

Line 140 We are now set up. We now start the main loop, which runs through lines 140,150,160,170. First we look for our joystick. We place -4,0 or 4 into X and/or Y. The next statement looks for diagonal inputs and puts you back to the Joystick line 140 for a new input.(If X<>0 and Y<>0) then if this is not true it passes to (ELSE IF X). This is the same as IF X <>0 and this statement will be true when the stick is moved to the right or the left. If the stick is moved to the right then X will equal 4 and P will equal 102 (P=100+4/2) Which equals P=100+2. For the other move left, X is equal to -4 and P will equal 98. (P=100+-4/2) which equals P=100-2. If you remember P is the variable for the sprite pattern. This is carried to lines 160 and 170 for the correct movement. In the next part we change the value of X to -1. for negative X or 1 when it is positive. S if we have moved the stick right or left at this point we would jump down to line 150 to continue the program. If the fist or second test were false then we check to see if the stick was pushed down or up. Once again the if IF Y statement is the same as if Y<>0 will only be true if the stick is pushed down or up. If the stick is pushed up the Y will equal 4 and p will equal 100 (P=98+4/2) which equals P=98+2. If the stick is pushed downwards then Y = -4 and P = 96 (P=98+-4/2) which equals P=98-2. We then Change the value of Y to -1 when Y is positive or 1 if it is negative. If the Joystick is not moved then Y and X will be 0, so all three tests above will false and no change will be made to our sprite.

LINE 150 This statement does a Get character routine to find the screen character at the location we want to move our sprite to and then place its value into CH. So if we push the stick down then the value of the character below our sprite will be placed in CH, The same is true of all the other stick positions. If CH = 40 the we are trying to move our sprite into a wall so we will sound a warning and then jump back to line 140 for a new joystick input. However if the move is valid, then we can move our sprite. We then add the value of X to C and then the value of Y to R

Line 160 This statement will set up our Sprite with its mouth closed and facing in the proper direction. This is due to value P passed from line 140. Next we place the Sprite in the right place with the changed values of R and C from the Joystick routine. Since R and C contain details of correlate to the graphic Row and Column positions, we change them into dot row and column values by using our conversion formula. All this is in Call Locate, via Call Pattern. We then test to see if we have moved our sprite on top of a dot, (Char 33). If we have then we generate a noise (type 7 white noise). We then place a space character on the screen at this location to delete the dot. If CH did not equal 33 then we would jump down to line 170.

Line 170 Having jumped here we set our sprite with its mouth open, P+1 equals the character number for the open mouth Sprite. Since P will not have been changed since the last Call Pattern our Sprite will still be facing in the same direction. The first Call Pattern sets our sprite in the proper direction, according to the joystick input with its mouth closed. In the second Call Pattern it will be an open mouth. We will test to see if the joystick has been moved out of centre position. The If X or Y statement is the same as If X<>0 OR Y<>0, so, if we have moved the joystick then X or Y will equal 1 or -1 and we

generate a beep sound. Then we place a blue block on the screen from where the sprite just moved from. So this will only happen if there was a move. We jump back then to line 140 for new inputs from the joystick regardless of state. Hence goto 140 else 140.

It looks very complicated in words and seems harder than it really is. So when you start writing your own program start first with a plan then test all you routine individually. If you have a full system with Extended Basic you have a really good tool to hand, that is the merge command. Save your little routines in Merge format and you can keep them to add into your programs at a later date. I have quite a few at hand on one disk.

To show you how to merge a file I will give you a little exercise.
Type this in:

```
100 Print "Hello this is me from another place"
120 Print "Shown to me by TI*MES"
130 For A=1 to 1000 :: Next A
140 Call Clear
```

SAVE this file as MERGE = SAVE "DSK1.MERGEIT",MERGE (enter)

Now pick from your library of extended programs a program. It does not matter what. Load it up and pick a place in the listing you want to place the new file into. Note down the numbers ie I want it in after line 300

Now quite out of your program and load the merge program with (MERGE DSK1.MERGEIT (enter)) You now have your original back. Renumber it to start at 300. IE RES 300,10 or RES 300 Now save it back in merge format as before. Load up your picked program and now type MERGE DSK1.MERGEIT (enter). The Merge file has now loaded starting at 300 and takes up the line through to 140 and pushes the rest of the program forward and redoes the line numbers. Clever isn't it!!

Last time I mentioned the game I have been working on. Well here it is. I have called it Fair Ground. It is a shoot em up. You control the gun sight and plug away at the targets. If you miss you loose points if you hit you gain points. When you get to a mystery high score you win a prize. I hope that you like it. Try looking through and see how it works. Try say, putting in more targets. Try animation like the duck. Add call peek RND to the program instead of the standard RND I have used. This program is free to all TI*MES members and Users groups world wide.

STOP PRESS: AGM 14th May 1994
At the St Johns Ambulance Centre, Trinity Street, Derby.
Anyone wishing to exhibit please contact Gen Secretary.

****** FULL MAP WITH THIS ISSUE ******
On back cover

```
 100 GOTO 170
110 CALL CLEAR :: CALL SCREE
N(5):: CALL DELSPRITE(ALL)::
 CALL COLOR(8,2,1)
120 DISPLAY AT(10,1):"CONGRA
TULATIONS YOU HAVE WON"
130 DISPLAY AT(13,1):"
A TEDDY BEAR"
140 CALL PEEK(-28672,X)
145 REM START
150 IF X=96 THEN CALL SAY("Y
OU WON")
160 FOR A=1 TO 1000 :: NEXT
A
170 SC=0 :: RANDOMIZE
180 CALL CLEAR :: CALL MAGNI
FY(3):: CALL SCREEN(5)
190 CALL CHAR(96,"003868385B
1F1F1F0F070300000000000000
04C4ECFCF8F0E0C00000000000")
!DUCK(1)
200 CALL CHAR(100,"003868381
B1F1F1F0F070300000000000000
000C0ECFEF8F0E0C00000000000"
)!DUCK(2)
210 CALL CHAR(104,"010618302
8444281814244283018060180601
80C142242818142221 40C186060"
)!TARGET
220 CALL CHAR(108,"000000010
7051F3F1F0F0000000000000008
0C0F050FCFEFCF8000000000000"
)!SPACE CRAFT
230 CALL CHAR(112,"030303031
F101F1F1F03030303030303E0606
060FC84FC7C7C6060606060E0"
)!TOMBSTONE
240 CALL CHAR(92,"FFFFFFFFFF
FFFFFF")
250 CALL CHAR(93,"000000E0FF
FFFFFF")
260 CALL HCHAR(13,1,92,384)
270 CALL HCHAR(14,1,93,32)
280 CALL HCHAR(17,1,93,7)::
CALL HCHAR(18,15,93,9)
290 CALL HCHAR(20,5,93,4)::
CALL HCHAR(22,14,93,5)
300 FOR A=25 TO 31 :: CALL V
CHAR(21,A,32,4):: NEXT A
310 DISPLAY AT(22,24)SIZE(5)
:"SCORE"
320 DISPLAY AT(24,23)SIZE(5)
:SC
330 CALL COLOR(1,15,15):: CA
LL COLOR(8,9,4):: CALL SCREE
N(6)
340 CALL SPRITE(£1,104,16,13
0,220)
350 SHIP=INT(RND*6)+1 :: CAL
L PATTERN(£13,100)
360 X=INT(RND*40)+30
370 IF SC=X OR SC>X THEN 110
380 ON SHIP GOSUB 400,410,42
0,430,440,450,460
390 GOTO 470
400 CALL DELSPRITE(£12):: RE
TURN
410 CALL SPRITE(£12,108,7,50
,50,0,-15):: RETURN
420 CALL MOTION(£12,0,15)::
CALL MOTION(£15,1,17):: RETU
RN
430 CALL SPRITE(£13,96,12,98
,100,0,-10):: RETURN
440 CALL DELSPRITE(£13):: RE
TURN
450 CALL SPRITE(£15,108,13,1
0,100,0,-18):: RETURN
460 CALL DELSPRITE(£15):: RE
TURN
470 GOSUB 630
480 C=INT(RND*10)+1 :: CALL
PATTERN(£22,98,£23,98)
490 REM DUCKS AND THINGS
500 ON C GOSUB 520,530,540,5
50,560,570,580,590,600,610
510 GOTO 350
520 CALL SPRITE(£25,112,2,11
8,40):: CALL DELSPRITE(£24,£
26):: RETURN
530 CALL DELSPRITE(£25):: RE
TURN
540 CALL SPRITE(£26,112,2,11
8,20):: CALL DELSPRITE(£24,£
25):: RETURN
550 CALL DELSPRITE(£26):: RE
TURN
560 CALL SPRITE(£24,112,2,15
9,120):: CALL DELSPRITE(£25,
£26):: RETURN
570 CALL DELSPRITE(£24):: RE
TURN
580 CALL SPRITE(£22,100,12,1
47,40):: RETURN
590 CALL DELSPRITE(£22)
600 CALL SPRITE(£23,100,12,1
30,150,0,-2):: RETURN
610 CALL DELSPRITE(£23)
620 REM CONTROL
630 CALL JOYST(1,X,Y):: CALL
 MOTION(£1,-Y*4,X*4):: GOSUB
 650
640 REM FIRE
650 CALL KEY(1,K,S)
660 IF S=0 THEN 680
670 IF K=18 THEN CALL SOUND(
100,110,0):: GOSUB 690
680 RETURN
690 CALL COINC(ALL,C)
700 IF C=-1 THEN CALL SOUND(
100,400,0)ELSE SC=SC-1 :: DI
SPLAY AT(24,23):SC :: GOTO 7
20
710 SC=SC+1 :: DISPLAY AT(24
,23):SC
720 RETURN
```

£=#

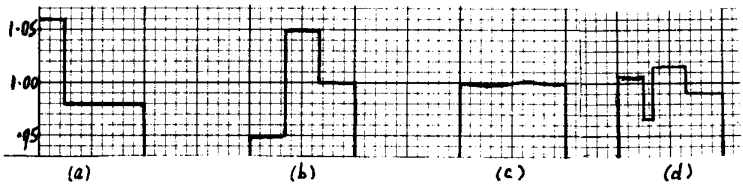
RND: ALTERNATIVE AND EPILOGUE.

Walter Allum

Most members are probably not much worried about RND being, maybe, a bit defective. However, the following alternative program which I found in my files is short enough to be nearly as interesting to them as to any concerned users. It has a good pedigree and was tested by many millions of samples, using methods mentioned in my Issue 43 article. Although in Basic, as against RND being in Assembler and GPL, it is not unduly slower (approx.0.18s per call instead of 0.073s). Its cycle length (according to me--but see later) is nearly 7*10^12 so that it would not repeat for some 40000 years if worked continuously! And, of course, there's no problem identifying its algorithm.

It is based on the fact that the fractional part of the sum of n independent rectangularly distributed random numbers is itself rectangularly distributed between 0 and 1, excluding the end points, regardless of the value of n. You might ask: if the program requirs rectangularly distributed inputs to produce a similar output, where's the benefit? As the authors say "no pseudo-random generator is perfect and adding several...(irons out the imperfections)".

I can give a simplistic illustration of this. Figs (a) & (b) show the tops of two imperfect rectangular probability distribution functions (pdf) over the range 0 - 1.



They should, of course, be level at ordinate 1. By standard statistical theory, it can be shown that the pdf for the fractional part of the sum of one sample from each "rough" distribution is as in Fig (c), much smoother than a simple average of the pdfs, Fig (d).

The program uses three multiplicative congruential generators with prime moduli and with the multipliers chosen to be primitive roots of the corresponding moduli. For those not familiar with number theory, it is enough to know that this ensures that each generator achieves its maximum possible cycle length (=prime-1). The authors also say that the overall cycle length is the product of the individual lengths but I believe this is a mistake for the LCM of those lengths. A correction may have appeared after I stopped taking the journal. If I'm the one who has got it wrong, the cycle length is four times bigger.

The pseudo-random numbers output are theoretically fractions with a denominator exceeding 2.7*10^13 but, due to the TI's 7 radix-100 representation, there is a certain degree of blurring among "adjacent" numbers in the 13th and 14th decimal places. In a system based on a single congruential generator, this might corrupt the sequence. But, in the present scheme, two pseudo-randoms that have been "confounded" in this way will not have the same three parents and it is these that control the sequence. The point is mainly academic, of course, since users will usually be rounding the output numbers.

The program now follows:

```
1000 IW=INT(IX/177)
1010 IX=171*(IX-177*IW)-2*IW
1020 IW=INT(IY/176)
1030 IY=172*(IY-176*IW)-35*IW
1040 IW=INT(IZ/178)
1050 IZ=170*(IZ-178*IW)-63*IW
1060 IF IX>=0 THEN 1080
1070 IX=IX+30269
1080 IF IY>=0 THEN 1100
1090 IY=IY+30307
1100 IF IZ>=0 THEN 1120
1110 IZ=IZ+30323
1120 R=(IX/30269)+(IY/30307)+(IZ/30323)
1130 R=R-INT(R)
```

Statements appropriate to GOSUB or SUB have to be added. NB: IX,IY and IZ should be set before any call is made to the program and then not changed except by the program itself. Unless required otherwise for a particular purpose, they should be random integers in the range 1-30000 (Use RND!). R is the pseudo-random output fraction.

Note: Readers may wonder why statements 1010,1030,1050 were not simplified by writing e.g. 1010 IX=171*IX-30269*IW. Originally, it was because I was following the authors' Fortran 66 integer arithmetic statements e.g
IX=171*MOD(IX,177)-2*(IX/177)
Then I just felt it wasn't a bad idea to keep the numbers in the products smaller.

References. (1) Wichmann & Hill Algorithm AS 183 Applied Statistics 1982
(2) do. NPL Report DITC 6/82

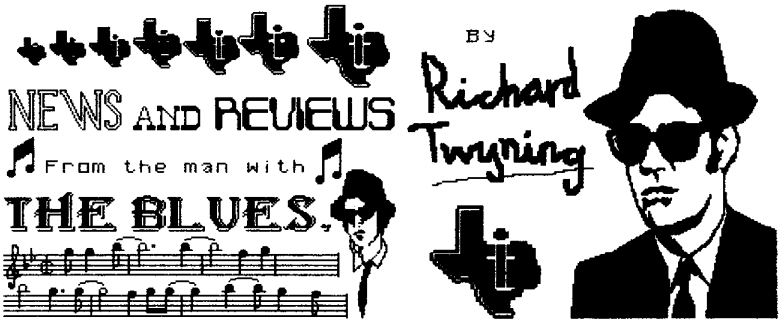
"Last Words" on RND. Although I have no firm evidence that RND is materially non-random within its proper scope, I think there's a need for that scope to be clarified. My belief is that console RND uses the integer sequence
I(N+1)≡28645*I(N)+31369 (mod 65536) Note that this modulus is 2^16 and the first parameter is a primitive λ-root of it.
My calculations and experiment indicate a cycle length 65536 instead of the 16384 that would apply if the 31369 had been absent, as in more conventional relationships. I also have evidence to suggest that there could be serious correlations between successive slabs of 16384 numbers. Before conversion to the fractional RND output, the 16-bit I() have their bytes interchanged, possibly(??) to offset this defect.

My current command of Assembler and GPL is too feeble to make much more progress and so, while continuing to poke at the problem, I do not intend to write any more about it unless seriosly provoked (Distant cheers?). It would be helpful if one of our specialists could confirm/deny my view of the console algorithm and tell us what XB uses. I don't think it can be the same. Finally, the cycle length (be it 16384 or 65536) seems pretty meagre especially when you consider the TI's capability. Surely, the whole point of having random numbers in computing lies in being able to reach out to virtual certainty through really large samples.

(Document continued from page 51)

TO USE CHARACTERS BELOW ASCII 32 enter special character mode with a toggle: CTRL U. In SCM the cursor is an under- line. When the underline cursor is flashing, ASCII 1 is CAP A up to ASCII 27 which is CAP Z. ASCII 0 is @. ASCII 27=(esc)=FCTN R. ASCII 28=FCTN Z. ASCII 29=FCTN T. ASCII 30=^ (shift 6). ASCII 31=FCTN U. Some of the above functions are carried forward from earlier issues but may have different or alternate keys! And to see these help screens, from the command line you just enter a H, the use letters Q and A to page up and down. BACK will return you to your text where you left it.

The help screens are 24 x 40 characters, and are transformed into 5 sector "program" type files by a conversion utility. * ENJOY *



Dear TI'ers,

How are things in the four corners of the T.I. World? Did I get carried away in my last article? I did didn't I!! I hope you found it interesting though, especially the comparison of the 4A to the PC, and the bit of TI history that I included.

I also hope you found my description of bitmap mode useful, if not informative. You'll have to watch out from now on. Writing the bitmap article has inspired me to write more assembler. I might write whole sections of my CAD program in 9900, and then use C99 to link it all together!

Well, I hope I'll have enough time to fit in all the subjects that I intend to cover. I suppose I might have started writing this article a bit late (at the time of writing it's 10pm 28-1-94 and I'm beginning to write this article as Blackadder II is just starting!)

At the present time I've also not started work on my Screenwriter program, but despite the late start, I'm sure I'll manage to complete everything.

I've had similar deadlines with University assignments, and have sometimes achieved nearly impossible tasks. The problem with University assignments is that they are mostly quite meaningless, and don't come anywhere near the cutting edge of technology.

The real cutting edge of technology is what you read every quarter in TI*MES. The concept of using Workspaces and having 16 registers anywhere in memory is still a remarkable one, and is far beyond anything that the Nottingham Trent University The most relevant thing we've been taught so far was about "intelligent memory." To the rest of the students this was a totally foreign subject, and it went straight over their heads. The reason for this is that it's not found in IBM PC's, Macintoshes, BBC's, Spectrum's etc. Even to the lecturers it's still a concept, and they couldn't easily find an actual example of intelligent memory (or an intelligent user for that matter!!!).

To me though, intelligent memory was something very close to home since I've been using it for over ten years! Have I? Yep! And so have you!

Intelligent Memory is memory that is intended to relieve the processor of time wasting duties. The most common thing that a processor has to do is access memory by setting up its address bus to the pattern that is required to access the desired memory location. You might not think that this would take too much time, and really it doesn't, but it takes even less if the processor doesn't have to do it at all!

A common use for intelligent memory is taking this searching process out of the hands of the processor. It makes more sense when the program has a branch or jump instruction and the processor has to go forwards or backwards to find the correct location. This is an even more lengthy process. Some of you may have already realized that the intelligent memory I'm talking about is the faithful GROM chip.

A GROM is a ROM chip that has its own program counter, and if you've got a GRAM Kracker or GENEVE, then you've got RAM with its own program counter. On the GENEVE it's handled by the Gate Array, but I'm not sure what handles it on the GRAM Kracker; probably a few PAL chips. Another legendary design to include with the design that makes the 9938 or 9958 use Static RAM, when it is only designed for Dynamic!

So what does the G in GROM or GRAM stand for?
It stands for Graphics, which as everyone on the planet should know, puts the G in GPL (Graphics Programming Language)! Unfortunately, for mankind, thanks to IBM, the uneducated world knows nothing of the benefits of GPL.

Despite having the full GPL development package on six disks, it's still a bit of a mystery. It was developed on TI mainframes, so since we're instruction set compatible it was simply downloaded to the 4A and run with virtually no conversion!

Try running mainframe software on a PC!
The mystery is, why was it developed in the first place? Was it intended as a way of making mainframe programming easier? In a way it was I suppose! It was designed to make the programming of TI cartridges easier! It's a bit of a mixture between 9900 assembly and FORTRAN, and it does relieve you of the more time consuming things, such as providing a single instruction for scanning the keyboard, or playing around with a few 64-bit numbers. Those familiar with FORTRAN will notice that this is where GPL inherits its logical operators.

It seems incredible doesn't it, that TI could spend time and money on developing a hybrid programming language like GPL, just for writing cartridge based software (In fact the entire 4A console operating system, including TI-BASIC is written in GPL).

That's TI for you though. Everything's done alot more professionally. Even the TI-Extended BASIC manual's got an International Standard Book Number (ISBN), and the LOGO II manual was published by McGraw Hill book company.

TI-LOGO was the very first licensed version of LOGO, which isn't surprising, since it was written at M.I.T. (Massachusetts Institute of Technology), in conjunction with TI, and with its creator, Seymour Papert. For this reason, TI-LOGO is the most powerful version available, even today.

There were later releases for BBC's, Commodore 64's, Spectrums etc., but I don't doubt that some of these weren't even licensed officially by M.I.T., and despite their more flashy graphics, do not possess the full list processing features of

M.I.T./TI LOGO. You could write a word processor with TI-LOGO, and that's official!

Back to the subject of GROM's!

I suppose one reason for them might have been to make the copying of cartridges more difficult, but nowadays we've got GRAM Krackers and C-SAVE to solve that problem.

Their two biggest advantages are speeding up the operation of GPL (since it's an interpreted language) and also saving on memory. Since each GROM has its own program counter built in, there's no need for it to be fully decoded into the memory map. There's just a single GROM Read and Write address that take up a total of 16-bits of the memory map. Every GROM, when it's required, is mapped into this single location. On the 4A, there's the device available that allows 16 cartridges to be loaded simultaneously, and switched with software, and it's possible to put 8 GROM's on each cartridge, since I think that this is how many are in the Triton Super Extended BASIC cartridge. This is why "REVIEW MODULE LIBRARY" sometimes appears on the TI title screen. The software to handle it has been built into the console's operating system from day one, but TI never released the hardware to handle it. Those who attended the Maidenhead show a couple of years ago will have seen Richard Blanden demonstrating his home-built version. Gary had a personal demonstration of all this, so if you want to get an article on this, then that's Gary's department! Richard Blanden is a typical, dedicated TI owner. He's a 747 pilot, and when he's flying trans-Atlantic, he puts the plane on auto-pilot and disassembles TMS9900. He's memorized every op-code of the 9900!

Back to the subject again!

The reason that GROM's speed up GPL, is because they have their own program counter and are "auto-incrementing", as soon as the 9900 has read an instruction from the GROM, the GROM automatically increments its program counter.

The memory inside the GROM is internally address-decoded. This helps when the GPL program contains jumps, or branches. The GROM sorts all this out for itself and updates its own program counter while the 9900 is busy doing something else. This must help to speed up GPL programs, and because GPL is very low level, GPL programs run much faster than compiled C, which isn't bad when you realize that GPL's really an interpreted language. When you also consider that the TI-BASIC and Extended BASIC interpreters are written in GPL, it shows just how fast the machine is.

According to Mark, despite GPL appearing to contain 9900, he says that every opcode is different, and has to be translated to true 9900 op-codes as it runs, so GPL is a true interpreted language. Mark also says that the Extended BASIC cartridge contains ROM and not GROM, but in the interests of research I've taken mine apart (don't try this at home because I can always load it from disk in emergencies!).

It does contain two 24-pin chips, which I would say fairly conclusively are ROMs, but it also contains four other chips, which are also definitely conclusively GROM's. How do I know? Well, this is the most unbelievable thing about GROM's! They are piggy backed in pairs, but every pin is touching!!!! That can't possibly work can it? It wouldn't appear so, but somehow it

does! The extra-terrestrial intelligence from which TI stole the idea in the year 2058 have apparently solved this problem, and when travelling back to the late 70's, TI managed to recreate what they saw using silicon technology!!! Gary had experience of this when putting his TI Disk Manager II cartridge GROMs onto his Zeno board. He saw two GROMs piggy backed with every wire touching, and though something was not quite right. He separated them and put them on the Zeno board individually, and it didn't work! He joined them up again with every pin touching, and it worked! It also doesn't matter which way round the GROMs go. Either one is happy underneath or on top!

Also in the XB cartridge is a 7400 and 7474, which are probably handling address decoding of the ROMs. There are also some components which appear to be decoupling capacitors as they are near the GROMs and ROMs. The numbers are 15Z, and below that 104, and altogether, there's 8 of them. Also there's a 20 Volt and 50 Volt diode, and a single 100 ohm resistor which is near the front of one of the GROMs, and must be related to it in some way.

Don't worry, I've just put my XB back together and tried it in my console, and it works! I've got my console set up at the side of my GENEVE on the desk, and it's permanently set up, and plugged into the composite input of my monitor. Whenever I want to do something quick, like test a little idea for a routine, or test a cartridge after ripping it apart!, then I can just do it quickly without having to fire up the Cray II and load MDOS!

Right, where am I? Firstly, I must say that I have been informed by Trevor that I've had a complaint about certain colourful metaphors (though censored) which appear in my articles! I'll try to tone things down in the future. I just get carried away when writing about the IBM PC! After having had to put up with them at college and work, and been forced to write articles on them, I just can't see the sense behind the world using them instead of what we've got. Our hardware, software and I/O capabilities are still incredible when comparing it to the so-called "state-of-the-art". Buying six Megs of memory so I can run a windowing program and a wordprocessor is not my idea of a very good computer. It's like buying a SKOD (SKODA) and realizing that you've got to buy NASA booster rockets if you want to beat anything away at the traffic lights, when you could have bought a high powered German car such as a ROVER!, and saved quite a bit of expense. Booster rockets are re-usable, but still not cost effective when compared to a 2 litre engine.

Our device independant operating system is the biggest reason that we are still where we are today. The 4A was compared to machines of the day such as BBC's which quite criminally were recommended for schools. They could not possibly educate anyone with an 8-bit processor that didn't even have a 16-bit index register. Yes! The 6502 has only got two 8-bit index registers named X and Y! Each of our Workspaces has got more index registers than the 6502, 6800, and Z80 put together!

A supposed selling point was that the BBC had alot of ports, including an A to D convertor! Oh, it must be very expandable if it's got all of these ports! Well, it's so expandable that you can't call a drive a drive! The BBC must

divide each floppy into two, and treat the top of one floppy as drive 0, and the bottom as drive 1. The TI controller can do much better than this. The trouble with these ports are that they are memory mapped, and the BBC is designed to use them at these locations. Memory is already set aside for them, and there are routines in the operating system to control them. They are "hard-wired" into the machine's design, and not designed to work independantly.

Our operating system is taken from TI's mainframes, and is totally device independant. This means that the 4A is not tied down by its hardware. The 4A is designed so that it can take advantage of any new hardware when it becomes available. There is nothing in the 4A's operating system that allows it to control floppy disks. There is only a very powerful protocol for talking to DSR's, which are the control ROM's that are found on each device that is plugged into the side of the console, or more commonly into the expansion box. All of the routines that control floppies are contained in the TI floppy controller's DSR. The CALL FILES command is not contained in the console. There is nothing in the 4A's operating system that can control any type of disks. It's all in the DSR.

If you've only got a console and no disk controller, then if you type CALL FILES(1) you'll get an error. SUBPROGRAM NOT FOUND when doing it from Extended BASIC, or BAD NAME from TI-BASIC. The console knows nothing about controlling disks. Disk control is not part of the 4A's operating system software.

The BBC's operating system, however, contains all of the routines needed for disk access. Acorn's "Advanced" Disk Filing System! The trouble is though, that these routines were written in the early 1980's, and are therefore only able to control the disks available in the early 1980's. The BBC's operating system has these routines embedded within it. If drive technology suddenly improved a week after the BBC was released, the entire operating system would have to be re-written.

The 4A however does not contain any software to allow it to control any type of disks. This software is only contained in the DSR of the respective device, such as the TI-Disk controller.

Therefore, if you want the 4A to control a different type of disk, then you just need to change the DSR. You don't need to change the computer. What am I getting at with this. Where does all of this lead? Some of you will know already!

Yes, the BBC is forced to use the devices that are defined in its operating system. There is nothing in its operating system about adding any new type of devices. So can a BBC control hard disks? Yes, but you can't expect a microcomputer to use independant device names when it was designed to use a limited set of ports and hardware. What they have to do is fiddle it! You could buy a 52Meg hard disk for the BBC, but since the BBC will only use floppies, it must be fooled into thinking it's a floppy. Therefore it has to run as drive 0 or drive 1, which means that you've reduced the number of floppy drives that you can connect!

Because the 4A has independant hardware control, we have the advanced feature of device names, which allow serious things to be performed such as listing a file to disk. Say this to a BBC owner, like I did once, and his neurons almost spontaneously combusted!

Because their operating system is fixed, and is totally

inflexible, they can't do something like this, and they can't comprehend it. To us, it's a simple matter of typing a device and filename. Our system is based on device names, followed by an extension. Once the device name is output to the Communications Register Unit, it is matched against the device name in the DSR of each peripheral. When the correct DSR matches up, it is selected and mapped into >4000. The DSR itself contains the routines that make sense of the extension.

For example:

DSK1.PARSEC

RS232/2.BA=9600.DA=8.PA=N

HDS1.UTILS.DISK.MG

When the 4A scans the box for DSR's, the first device filename (DSK1.PARSEC) is recognized by the DSR at 1100 (normally). The DSR at 1100 is usually the TI Disk Controller, or any other disk controller for that matter. All it recognizes is the DSK1. which causes it to say "Yep! That device name's for me!". When it is mapped in to >4000 and executed, it looks at the remaining PARSEC bit and decides what to do. In the code of the DSR is a list of the devices that the controller can recognize, which in the case of the TI controller is DSK1., DSK2., DSK3.

When the DSR recognizes the device name when the 4A scans the box, it already matches the called device name to whichever drive name is being called. This causes the controller to cross reference the code in its DSR, which then makes the controller look at the correct drive. Then it executes the code that finds the file PARSEC on the drive. It already knows what type of file it is, because the application that's asked for the device.filename, has transparently to you, passed a Peripheral Access Block (PAB) to the DSR. This PAB contains a code that defines everything that the controller needs to know about the file. These codes can be found in the Editor/Assembler manual, and I know that they are also contained in the File Handling section of Ralph Molesworth's Introduction to Assembly Language. In the case of PARSEC, it's probably an Option 5 E/A file, which means it's a memory-image PROGRAM file. There is no OPEN or CLOSE. The DSR just uses the LOAD operation that grabs an entire block of memory from the drive. There is also a SAVE operation that will take a specified area of memory and transfer it to disk in one go. There's nothing better for showing up PC's as a complete embarassment than loading memory-image files from RAM disk, or hard disk! I can go from the Editor/Assembler screen to YAPP's 256 * 424, 256 colour mode, in less than five seconds! With a very optimized RISC instruction set, we don't need to load as much data, and we don't use anywhere near the same amount of memory or disk space. PC owners talk about Megs of RAM, but to the 4A owner, 8K or 32K extra on the cartridge port opens up hundreds of new possibilities.

Where was I? Oh yes!

The other device names that might be typed. When the RS232 card receives the RS232/2. device name, it's DSR has caused it to intercept the I/O call, and it also knows from the /2 that the call is trying to access Port 2 on the RS232 card. This time

though, when the RS232's DSR starts to operate, there is no filename. There are commands that change the characteristics of the port in some way. The BA=9600 alters the speed to 9600 baud, the DA=8 forces it to work with 8 Data bits, and the PA=N causes it to ignore Parity. There is nothing in the 4A that operates on these instructions. The 4A knows nothing about serial communications. This is all written into the DSR. The same thing could be true about the IBM PC, but the trouble is, you need to load a device driver into memory! Whenever you buy a new piece of hardware, you get "driver software" with it. You've got to play about with your config.sys file to make it load the driver that has been supplied with your new card. This however takes up some of the machines memory, and the more hardware you add, the more memory is taken up, which is why there are so many problems with new application programs. The PC only really has 640K base memory, and you're extremely lucky to get over 600K of free memory for application programs. Some are so critical that, as I mentioned in my last article, you need different autoexec.bat and config.sys files for each program, just so that it's got the correct amount of memory available. Gary had this problem with GRAND PRIX. It needs about 600K, so he had to have a separate autoexec.bat file that didn't load as many device drivers, just so that there would be enough memory for GRAND PRIX to run! And, when he wanted to use it, he had to rename his existing autoexec.bat file to something else, and then rename whatever he'd called his GRAND PRIX setup file to autoexec.bat, and then reboot his machine so the new autoexec.bat file could be executed.

If we need to do this on the GENEVE, for example, to give 512K systems a smaller RAMdisk if the user wishes to run GPL, then there is no renaming to be done. Any batch file can be treated as an AUTOEXEC by simply typing an & in front of it! If you have created a batch file called SMALLRAM that contains the following:

```
TIMODE
RAMDISK 90
ASSIGN A=DSK1:
ASSIGN B=DSK2:
ASSIGN E=DSK5:
PROMPT $T$B$D$_$N$P$G
GPL TI_XB
```

Then to run this instead of the normal AUTOEXEC batch file, you just need to type &SMALLRAM which tells MDOS that the following file should be treated as an AUTOEXEC file, even though it is not named AUTOEXEC. If you just typed SMALLRAM without putting the & in front, then you would get the error "only in AUTOEXEC" because TIMODE can only appear in an AUTOEXEC file.
There is much more of a need for this feature on PC's, but they won't do it, because they haven't had the thought put into them that our system has.

So where was I again? Well, what am I showing you with all this? I suppose I'm making you realize that, however flashy the modern PC might be, it's still nowhere near as good as the 4A or GENEVE. Anything they can do, we can do better. We used to compare ourselves against the home computers that were available at the time of the 4A, but now, they have all disappeared, and

the only reason we are still here is that the 4A was designed properly. It's a bit of a let down that Don Bynum didn't get his own way with the design he really wanted, but TI thought it would be too expensive. It would, and they would probably have started out making a loss, but it would have given them a massive user base, and they would have ended up making a massive profit, bigger than anything IBM have ever achieved. As Mark says, TI are the best company in the world, and they have more foresight when it comes to design than anyone else. You only have to look at the innovations put into the 4A. Stackless RISC architecture, sprites, intelligent memory, the best speech system available, a cartridge port miles better than anyone elses, and the best thing of all, Device Independance. You can get a 4A emulator for the PC, but you need the fastest of 486's for it to handle sprites, and it could never hope to produce graphics as fast as the 9938 or 9958 with it's hardware graphics commands. With the 4A and GENEVE you can draw a line with a single instruction. On the PC it still takes a program to achieve the same thing. That's why they can't do that much more than us, even though their CPU's are apparently getting faster. Why bother executing a line drawing program on the CPU, when your graphics chip can do it! Therefore the 4A can draw lines faster than a 486! Unless the 486 has got a decent graphics card. Using the Texas Instruments TMS34020 Video Display Processor.

I've strayed from the subject of DSR's again! The point is that with our operating system, we can add anything to our boxes and the 4A will talk to it with no problems. Each card has its own DSR to tell the 4A and TMS9900 what to do with the device, and how to transfer data to and from the peripheral.

The 4A's operating system itself knows nothing about Floppy disks, RAM disks, RS232 cards, print spoolers, internal modems, hex-bus interfaces, hard disk controllers, or SCSI cards!

YES!!! That's what I'm getting at! The SCSI card! It is finally ready. Honest! It really is! Some people will find this hard to believe since there have been statements saying "The DSR will be ready in May" and then in May 93 they said the same thing, followed by "Yes, I know we said that last May, but this time we really mean it!".

It seemed to be a bit of a Lou Phillips (Let's hope he rots in hell for not sending me my GENEVE back! I had to buy another one from Texaments!) special to get people's hopes up, and then let them down again. Those interested in the subject of the SCSI card will have doubt in their minds about the possibility of the card ever being released, but after reading my article, you will not complain any longer. You will realize the cause of the delay. But, you will have to wait just a little bit longer, as I've got a couple other things to write about first.

No!!

Hold it right there! Don't skip pages to the SCSI review!! You've waited this long, and something this good is worth waiting for. BELIEVE ME. IT IS THIS GOOD.

In my Winter article I reported that I was building a computer room upstairs. It's now complete, and I moved in on

Wednesday 15th of December. The Sunday before that, Mark Wills came to stay with us, because he'd got an interview in Sheffield on the Monday. British Rail were going to charge over £70 for a return ticket from London to Sheffield on Monday morning, so Mark asked if it was alright if he came to stay with us on the Sunday as it would only cost half as much. I go swimming regularly on Sunday mornings, and I was paged while I was in the pool with the information that he was catching an earlier train. I picked him up from Nottingham station just as it started snowing. We were a bit stuck in it on the way back, and it took over an hour. I showed Mark the new room, but all I had done was laid my carpets, assembled my desk and coffee table, and put up a shelf. I hadn't had time to take the computer upstairs, so it was still set up in my bedroom. I can't exactly remember exactly what we did now. I did a demo of Workspace and the Myarc Hard Disk controller. I also think I did a demo of YAPP, and 80 Column Funnelweb etc.

We drove Mark to Sheffield for the interview, which was the longest I've ever known! In the afternoon before he went back, we visited Trevor's place and had a bit of a get together. After this I took him back to Nottingham station.

I had a bit of a rush over the next two days, moving everything upstairs and setting up my system. A few days earlier, Francesco phoned and said he had a problem he would like sorting out, and would like to come and see me before Christmas. I said it would be good if he could come on the 15th, since we were having the get together which acted as a "roof warming party!". I sent Francesco a map which guided him from junction 28 of the M1.

Unfortunately, Derek was unable to attend, which left Francesco, Gary, and Trevor in attendance. The problem that Francesco had was with the Myarc Floppy controller with the GENEVE. When I put the controller in my box with my GENEVE, it had the same problem, which proved that Francesco's GENEVE was O.K. The problem was, I think that he had got the wrong EPROM. The EPROM was telling the Myarc Floppy that the drives were 40 track, and not 80 track. When he tried using a disk, he got the message: "80 track disk in 40 track drive".

Francesco had to leave early enough to allow time to drive back to Oxford, but Trevor, Gary, and I, carried on for a bit longer, and then decided to go for chips. As it was a Wednesday, I had taped Star Trek, and we watched this while sitting round in my new T.V. room eating our chips. I can remember distinctly which episode it was. It was "The Tholien Web" which had these things that spun a web around the Enterprise. Trevor had seen it and remembered it when the BBC repeated the original series during the 70's.

In my previous article, I spoke alot about my work placement at SYNEL U.K. in Sheffield. I started this on the 19th of July, and was just beginning to get into the swing of things, and then, at the end of November, I was made redundant! Everyone seems to think that it's impossible to be made redundant on a work placement, but they did it! It wasn't a particularly good placement. It didn't pay too much, but it was close to home, which allowed me more time to get my programming out of the way, or would have done if it had carried on! While I was at SYNEL I was building my new room, and this prevented me from spending as much time on my TI programming as I would like. It was a bit of a struggle to get my Winter article finished! I suppose this

wasn't helped by the fact that it was 54 pages long!!!

SYNEL U.K. was a distributor for SYNEL Israel, and sold Time and Attendance Terminals. These were remote terminals that had either a magnetic, or bar code reader, and a dot-matrix plasma display. When an employee entered a factory or where ever he/she worked, their card would be swiped through the slot, and their details would be stored in the terminal's battery backed memory (Only 32K!). This data could then be downloaded via RS232 or RS422 (the plug connections were completely non-standard of course!).

SYNEL was owned by MHG systems, who supply EPOS Point of Sale terminals to supermarkets. Their largest customer is Sheffield CO-OP. Due to a lack of orders to SYNEL, and a list of phantom orders created by a so-called salesman, MHG decided to sell SYNEL, and so I was made redundant. I wrote quite an abrupt letter to my placement tutor, telling him that I felt very badly treated by the University. It seemed that other students had been given more help in finding a placement, and despite my knowledge, I seemed to have been given much less help than anyone else. This might have been because I didn't do quite as well in the second year. I had lost interest in the course. I know much more than the rest of the students since my TI has given me a much wider range of skills than Nottingham Trent University could ever teach anyone. I don't know what it is about the education system though. They just don't see that they're doing it all wrong. My friend Dave at University had a BBC when he was about 11. He's still got it somewhere, buried in a cupboard, but now uses a 486 PC. If the BBC was any good, he'd still be using it! What the world thinks is though, that everything needs to get alot faster, and have more memory for things to improve. They thought the BBC was a good computer, and BBC owners even argued with me that their machine was better. After showing my 4A to one of them, he soon stopped arguing. I had just bought GRAPHX, and had already got Extended BASIC, TE II and speech, and had also got my disk system running. He was speechless when he saw the Expansion Box! At school we had a Research Machines 380Z which had a box about as big as my expansion box, and everyone thought that because of the cooling fan and the size of it, it must be really powerful. It wasn't anywhere near the power of my 4A, since it only had a Z80 processor, and hadn't got a Device Independant operating system! When the BBC owner saw my box, he was almost down on his hands and knees saying "I'm not worthy, I'm not worthy!"

If the BBC had been any good, then my friend Dave would have as much knowledge of computing as me, but he hasn't. Also, if it was any good, he would still be using it, and wouldn't have a PC. I consider PC's to be faster equivalents of a BBC. The main reason for this is that both BBC's and PC's have to waste valuable cycles displaying a cursor! Why do you have to have a

cursor when you're displaying things to the screen. It makes me laugh when either the BBC or PC are updating things like numbers on the screen. You can see a cursor whizzing about under the numbers, and if you are displaying the numbers in more than one location, then you can see it jump between the two! SAD!! That's why processors get faster, but PC's don't. The world just doesn't see it.

Back to my friend Dave again, I went around to his place in Southwell once, and he tried accessing the University's VAX via MODEM. It took well over half an hour to try and set it up properly. There were just too many options, and pull down menus. Gary will vouch for this too. He has tried using the MODEM on his PC, and has ended up giving up altogether, and firing up the GENEVE, and getting straight in with TELCO.

Returning to the subject again, my phrase is still correct! "You may have been to college, or you may have been to school, but if you ain't got a Texas, you're an educated fool."

So what's happening at the moment? Well, my placement tutor has finally pulled his finger out and found me a decent placement. The only trouble is that it's in Landun (London!). So I've got to live down there. The time of writing this is 17:53, 25-2-94, and I start on Monday. I've already found accomodation in Leytonstone, which is four stations away on the Central Line. The place I'm working is Mitsubishi Finance, which is located in Broadgate, which is a new office block which has been built on the old site of the Liverpool Street tube station after the new station has been built. It's a sort of a financial dealing house, and has a dealing floor with people sat around on phones trading shares. On the top floor they've got a real network which is run by, among other things, a Sun Sparc Station 10, which contains a Texas Instruments Super SPARC. They've lowered themselves into getting two COMPAQ Tower PC systems to run a Novell network, and also an IBM AS-400. My job is technical support, which is fixing problems caused by users. If they spill coffee in their keyboard, I'll have to take them a new one, or if they've brought software in from home, and fiddled their config.sys to make it run, then I'll have to go and re-copy the old version back on to make the company software run again!

I'll also have to learn what patches go where on the patch panel! There's a massive panel of sockets with a wire from each Sun Workstation. It's a bit like an old film of a telephone exchange. If a user wants access to a different bit of the network, or another network, then their plug on the patch board has to be moved! Some days I'll start later in the morning, and work until 7pm. When doing this shift, you have to put a tape in the IBM so it can back itself up overnight. It probably takes it all night too!

I'm not taking my Cray II down with me. I will take my Amiga down I think, if I can get my mini-TV working. I like to know that my real computer is safe at home in the loft! It will be good not being able to use it. That doesn't sound right does it? Well, yes it does sound right. When I wrote Workspace, I didn't have access to a GENEVE, and was only able to test it a few times when Gary got a GENEVE. It improved my theory, and forced me to double check everything. I learnt just about everything about using XHI from that. This should force me to write my CAD program properly now. Not having the GENEVE will

force me to improve on my theory. Hopefully, by August, which is when my placement will finish, I should know C99, and Bit-map mode inside out! I bet you are all worried when I mention GENEVE aren't you, but I'm writing it for the 4A. It will run from the 4A with 32K and disk. Don't worry, I'm going to write it so if you've only got one single sided 360 sector drive, it will still work. It will allow you to create CAD-object files as big as your disk capacity, and will allow true pathnames, which will make it compatible with any floppy disk, or hard disk. The biggest advantage will be if you've got a battery-backed RAM Disk if you haven't got a SCSI card!

Where was I?

Oh yes, back to my placement. You will find on the list of contacts, next to my name, my London phone number for weekdays. If you're in need of spiritual guidance, then you're welcome to call me for inspiration. It'll cheer me up too, to hear from fellow TI'ers! I'm hoping to return to Mansfield every other weekend to catch up on episodes of Star Trek that my video will have faithfully taped, and to test the next stages of my CAD program that I will have written, and possibly typed in while I'm down there. And, don't worry, I'll be back for the AGM, and Gary and I will be demonstrating the first SCSI cards to enter the country! Just so you can't miss it, here's the date of the AGM:

AGM, 14th May '94

The AGM will be held at the regular venue, in the middle of the country at the St. Johns Ambulance Center on Trinity Street, Derby.

After Francesco came and said my map was quite easy to follow, I thought it might be nice to invite anyone to visit me, since I'm quite easy to find from the M1. This has been disrupted now though, by my lodging in London. But, if you want to visit me one weekend, that'll be a definite possibility, as long as you phone me in London during the week to confirm when I will be in Mansfield. Somewhere in my article where Gary finds free space you will find a map which shows how to reach 24 Peel Road from the M1.

Well, what's next then?

O.K. Alright then. You've waited long enough. I said last issue that it would excite our members. After reading this you will possibly not believe it, but it is all true.

The SCSI card has arrived.

This card is not just a new piece of hardware. It's a tribute, and a turning point. It's a tribute to the design of the 99/4A, and to Don Bynum, who was responsible for the design of the 4A. Last year, when our own little team considered the possibility of building a new computer, we also fleetingly considered building a SCSI interface, since Gary did his

placement at National Semiconductor, and had information of a National Semiconductor SCSI interface chip. At around this time there were rumours of the SCSI card beginning to circulate, and I can't actually say when I heard about it first. I don't think many people were aware of it over here at last years AGM.

As I have said earlier in my article, I suppose it seemed to me that it might never arrive, since there were so many problems with the DSR. It might go the same way as the IDE interface. WHAT? Yes, a prototype IDE interface for the TI does exist, but will probably never go into production, especially now that the SCSI card is officially ready.

The SCSI card only seemed official to me from around August or September, when I seemed to be hearing more about it. This was when I first started pricing up SCSI hard disks!

A couple of weeks ago, I visited Trevor to show him an early version of my ScreenWriter program, and he informed me that he'd got the latest issue of MICROPENDIUM. This was the December issue, which contained an official report that the card was ready, and had been completely modified to allow it to work with both the GENEVE and 99/4A. Bud Mills says twelve cards were sold at a show that shouldn't have been, because they hadn't been modified. He said that the cards were available for sale though, and printed his voice and Bulletin Board phone numbers to contact for information about the card. Trevor let me borrow MICROPENDIUM, and I said I was going to give Bud a call.

I did call Bud, on the 11th of February, and after phoning him, I was literally in shock! I couldn't believe what he had told me. Mark phoned just before I phoned Bud, and I phoned Mark back to tell him the news. He was also amazed, and so was Gary when he heard the news. The news from Bud on the 11th was that the DSR was ready, and would use hard disks perfectly, but the emulation of floppies was still being worked on.

I will try and cover what the card will do from the information provided by Bud.

Are you sitting down? Do you have a glass-topped coffee table in your lounge? If so, then please do not stand or sit near it while reading this review. The author cannot be held responsible for damage to household contents when suffering shock after reading the following information. My apologies to console-only owners, as Gary will not be able to design a mini-expansion box until he finishes his course. Don't worry though, this box will contain a socket for optional 32K RAM expansion!

Are you sitting comfortably? Then we will begin the future here.

What is SCSI? Is it an hard disk interface? No, it isn't! The SCSI (Small Computer Systems Interface) interface was not designed to run hard disks. It isn't as good at controlling hard disks as the Myarc Hard Disk Controller. How can that be true?

The Myarc Hard Disk Controller is an hard disk controller!! It uses the now obsolete (but still excellent for our purposes) ST506/412 interface in which the ST stands for SHUGART. Named after Al Shugart who invented the interface, and who invented the interface that we all use for floppy disks.

The controller has to think of all aspects of controlling the hard disk, and the drives are less intelligent. SCSI on the

other hand was not designed for controlling hard disks. It was designed as a general purpose parallel interface for mini-computers. You can virtually buy anything that will work from the SCSI interface, including printers, plotters, scanners, and of course, hard disks. The reason the SCSI interface is better though, is because it was not intended to control hard disks, so it's not tied down by worrying about drive control information, so its data transfer rate is quicker.

This means though that the drives need to be more intelligent to make up for the fact that they are not being controlled, they are just being communicated with. With the Myarc Hard Disk controller, when you want to format an hard disk, the controller controls the formatting process. With SCSI however, the formatting software sends a command over the SCSI interface to the selected device that more or less tells the drive to format itself! There are even drives available that can repair corrupted files themselves! If it finds a faulty sector in the middle of a file, it will move the file to a new location, and attempt to re-insert the data that was lost. This relieves the computer of alot of machine cycles.

So what about our SCSI card?

The SCSI card will work at any CRU location from >1000 to >1F00, and will handle seven SCSI devices. Each device is jumpered to a different SCSI device number, but I'm not exactly sure of the device names that will be used. The most likely thing they will do must be to call them SCS1., SCS2., etc. up to SCS7. You can't plug hard disks into all of these though. Two are reserved for different devices. What are these? Well, devices 3 and 6 are reserved, and the team are still working on the code that will control the device that sits on SCSI address 6.

Device 3 on the other hand cannot be used by a SCSI hard disk, because it is reserved for an optional floppy disk controller! This part of the DSR is complete, and the floppy option is available when you purchase the SCSI card. Because the SCSI card is an industry standard Bus interface, it can allow us to interface to a new floppy controller. The SCSI card connects to the 4A or GENEVE, and the optional floppy controller connects to the SCSI card.

So, what will this floppy controller do? Are you still sitting down? At the moment there is a slight bug in the DSR, which means that it won't format double density TI disks, but if you do put a double density TI disk in, it will use them perfectly. It will format and handle 90K SS/SD disks though, and also 180K DS/SD disks. Also, 80 track TI disks which give 180K and 360K at single density. Why do I keep saying TI? Does it mean that it will do other formats? YES! Any IBM format!

It will format/read/write ANY IBM format, and some formats that the top of the range IBMs can't handle!

It will use a standard 40 track drive that most of you already have, and will format and use it at the IBM 360K formatting standard. Their format is different because they have 512 bytes per sector, and half as many sectors. We have twice the number of sectors, and only 256 bytes. Ours is the best format though, because file storage has to be done on whole

sectors. If a file which is being saved has just used up an even number of sectors, but there is another byte to be stored, then this byte will need to take up a whole new sector. Since PC's use 512 bytes per sector, 511 bytes are wasted. We only use 256 bytes per sector, so we would only waste 255 bytes. Besides being better structured, the TI filing system is much more efficient!

The card will also use 80 Track disks and will give the IBM 720K format. If you can get hold of any cheap 1.2 Megabyte, 5.25" drives, then it will use these too, along with 1.44 Megabyte 3.5" disks. This is where IBM'ers stop, but not TI'ers I'm very happy to say. The card will also do 2.88 Megabyte disks! YES, that's 11520 sectors according to my calculations! It sounds impressive doesn't it. It is, but not as impressive as 3.6 Megabytes! Yes, they also have a routine that will format these 2.88 Meg drives as far as 3.6 Megabytes, which at 14400 sectors, is the equivalent capacity of twenty standard TI DS/SD 40 Track disks.

To cover that again, the formats it will handle are:

Capacity	Format	Tracks	Sectors	Comments
90K	TI SS/SD	40	360	
180K	TI DS/SD	40	720	
360K	TI DS/DD	40	1440	<<Use but not format yet
360K	TI DS/SD	80	1440	
720K	TI DS/DD	80	2880	<<Use but not format yet
360K	IBM	40	720	<<512 bytes per sector
720K	IBM	80	1440	<<512 bytes per sector
1.44Meg	TI or IBM	?	5760	<<Unsure of exact format
2.88Meg	IBM	?	?11520?	<<Unsure of exact format
3.6Meg	SCSI team!	?	?14400?	<<Unsure of exact format

I'm unsure of the higher formats, because I don't know if they will be formatted to the IBM or TI standard sectors. The number of sectors shown for 1.44 Meg, 2.88Meg, and 3.6Meg are the equivalent number of TI sectors if the drives are formatted at 256 bytes per sector. Using the 3.6 Meg standard would mean that the cassette library would fit onto three 3.5" disks!

The advantage of the floppy controller card being used as a SCSI device, means that it benefits from the SCSI card's built-in 32K cache which buffers data transfer between the computer, and SCSI devices. This means that if you are saving or loading files to a floppy disk, you can consider it as being faster than an HORIZON RAMdisk! The reason that SCSI device 3 is reserved for the floppy controller is because the DSR intercepts calls for standard DSK devices, and directs these calls to SCSI device 3 and to the floppy controller. The DSR tells the 4A that these floppies are standard TI disk devices. I dare say that it is theoretically and practically possible to even have more than one of these floppy controllers plugged into the SCSI card!

O.K., if SCSI device 3 is reserved for floppy control, what it device 6 reserved for? CD-ROM! Yes, they are working on code for the DSR that will read IBM CD-ROM's! They are using an NEC CD ROM Drive, which according to Bud, is about the best drive for price and features. They have already succeeded in

reading Sound files, and GIF (Graphic Interchange Format) image files, and text files straight from an IBM CD-ROM. Encyclopedia Britanica here we come. I think it was the January MICROPENDIUM that contained reports of someone transferring archived TI files to an IBM so they could be put onto an IBM CD-ROM. They didn't envisage the possibility of the SCSI card, but now with these possibilities, we will not need an IBM to read the data back, and when the data is on CD-ROM, it will be permanent, and not suffer magnetic drop outs etc.

Right, that's the reserved devices out of the way. There is no restriction on what you can do with the remaining devices. If you must, you can run a hard disk off of each one! SCSI devices 1,2,4,5,7 can all connect to SCSI hard disk drives. That's five SCSI hard disks, compared with the three ST506/412 drives controllable with the Myarc HFDCC.

How is each drive divided up? Well, disk users may know that we've got a limit of 127 files per directory on floppy, so what they have done, is divide each hard disk up into a maximum of 127 folders. These are 32 Megabytes each, and since the DSR has fooled the TI into thinking that the drive contains files, and not folders, I think that the directory structure below this can be virtually unlimited, since this is all handled by the DSR.

Don't worry about rushing for the calculator. The drive addressing capacity means that we can get 4064 Megabytes per drive, and with five drives, gives us a total SCSI hard disk capacity of:

20320 Megabytes!

That's:

20.32 Gigabytes!!

That's 79375000 sectors, or 110243 DSSD 40 track disks!

Officially from Bud, the SCSI card is faster than an HORIZON RAM Disk, which has been confirmed and explained by Gary. He says that every block of memory in the HORIZON has to be addressed by CRU selects, whereas, to transfer data to/and from the SCSI card only requires one CRU select. The drives will provide or accept data continuously, and will be cached by the SCSI card's 32K cache. Even better if the hard disk has its own cache. I've had my eye on a DEC 1.05 Gig drive which is a fast SCSI 2 hard disk and has its own 512K cache built in, which must be contained on one chip, since the drive is only a 3.5" device! Did I forget to mention that the SCSI card will use SCSI 1 or SCSI 2 devices. Standard SCSI devices will work with no problem on the card, it's just that SCSI 2 devices offer higher data transfer rates, and are much faster.

You're not limited to standard SCSI hard disks on the remaining SCSI devices. You can also use removable disks. These are like floppy drives, in that they normally use big floppy disks, but they work like hard disks, and offer transfer rates as fast as the Myarc Hard Disk Controller. Bud recommends SyQuest drives, since this is what they use, but any major name in drives

will work with the card. If it's a Phillips, Seagate, Fujitsu, Connor, DEC, Panasonic, etc. then it will work. I've seen some very nice Panasonic floptical drives. These are 3.5" drives that use optically re-writable disks, and give 128Megabytes per disk. Most TI'ers can get their life's collection of data and applications into this space, since our files are so compact. Imagine that. All your disk collection backed up onto a 3.5" floppy! The entire User Group disk library in a box of 10 disks! And, it's secure from data loss, because it's optical!

I've only really mentioned data storage haven't I! SCSI will run other devices, and these can be used on devices 1,2,4,5, and 7. These include laser printers, plotters, and scanners. Bud is already considering the use of scanners, and with this in mind, they have added another unique feature to the card.

The DSR ROM can be copied into the 32K cache, and the computer can be fooled into thinking that this 32K RAM cache is actually the DSR ROM! This is called shadowing, and under software control, can allow different applications to load additional routines, into the DSR area, that control devices that relate to the application being used. They are already considering this, and the number one example is the use of scanners. SCSI scanners work out at about $300 for 256 grey scales, or $600 for 16.7 million colours. 256 Grey scales would be perfect for our purposes, as it allows us to scan things for editing, and re-inclusion in TI*MRS, and it also opens up the possibility of Optical Character Recognition.

Well, that almost covers everything. Bud reported that DSK emulation was still being worked on, but I can report that this is now finished! Todays date is the 26th of February, and if I look at the clock on MyWord, it says 20:00. On Thursday night at 11:30pm I phoned Bud. For him it was 6:30pm. I told him I would like to order two cards, both with floppy option, for me and Gary. Bud said it was O.K., but they had been slightly put back with the DSR due to demonstrating at Fest-West (A SCSI card and 20Meg hard disk was the first prize in the raffle at Fest West! The second prize was the Myarc HFDCC and 20Meg drive!)

He said he thought, if I sent my order straight away, it should take a week to get there, and for the International Money Order to clear. By this time the DSR should be complete in a 100% usable form. He said he would hang on to the cards for a couple of days longer, just in case, so that we would get the most up-to-date DSR. He says this DSR will do everything you would want for normal operation, and then there will be an upgrade DSR that includes "a few more bells and whistles" as Bud puts it!

I said to him, "Would these bells and whistles include DSK emulation?" He said, "No, that's already working!"

Users of the Myarc Hard Disk controller will know what this is already.

Most software will work from the hard disk by simply copying it on, and then altering the pathnames that are contained within the program. TI-Artist allows any pathname to be set, and so do some other programs. Alot of new programs nowadays even allow themselves to be installed to any device name, so that they will load from it, and use any hard disk pathnames.

Some software, however, will not do this. It will only work from disk drive one, because the pathname DSK1 is hardcoded into the software, and cannot be changed, without complicated sector editing. You might think that sector editing is a good solution, but the program will only contain space for the four character DSK1 device name, and a ten character filename.

Hard disks on the other hand, allow much longer device.pathnames, which in the case of the Myarc HFDCC can be up for 40 characters. Sector editing the program to make it load from hard disk it no good, because the longer device.pathname will overwrite assembly instructions in the program.

The only answer to this is to emulate the floppy drive on the hard disk. This means that an area of the hard disk acts as floppy disk one (DSK1) and the 4A and the software thinks that it's talking to floppy disk, but really it's talking to the hard disk, and is working much faster.

The Myarc controller allows three forms of floppy emulation. If you've got a subdirectory named DSK1 on hard disk 1, any calls to DSK1 get directed to HDS1.DSK1. and it searches the hard disk first. Also, if you've got a subdirectory named DSK which contains further subdirectories, then any call for a named disk, such as DSK.TIMP. (for the Microsoft Multiplan disk) will cause the system to get the data from HDS1.DSK.TIMP. first, so that you don't have to waste time and speed putting your floppy in a drive.

The most advanced form of emulation on the Myarc Hard Disk Controller, is the use of EMULATE files. These are a new file type that is unique to the Myarc HFDCC, and are created by Myarc Disk Manager five. There is no other way of creating them, unless you mess about with assembly language. An EMULATE file contains an entire sector map of an entire floppy disk and is copied into the EMULATE file sector by sector. Only one EMULATE file can be active at a time, and they are activated by setting a flag at the side of the file from the catalog option of Myarc DM five.

When an EMULATE file is active, any calls for DSK1, go to the EMULATE file on the hard disk, no matter whereabouts on the hard disk the EMULATE file is saved. It can be at the lowest possible sub-directory path, or on the root directory, it makes no difference.

I'm unsure if the SCSI card extends to "named" disk emulation, but this may well be already included, or may be one of the "bells and whistles" that Bud was talking about.

What the Myarc Hard Disk Controller will not do is emulate other disk drives such as DSK2, DSK3, DSK4, etc.
This is already all definitely possible on the SCSI card. You can "alias" any disk to any folder on an hard disk, so you can alias any disk number (DSK2, DSK3, DSK4, DSK5, DSK6, etc.) to any folder on the hard disk. This has the effect of giving you the equivalent of a 32Meg floppy, so you're not limited by the number of applications that can be placed into a folder, and "emulated" from it.

Don't let me put you off the Myarc Hard Disk Controller. I've been pleased with mine. The only off-putting point is that my 80Meg drives have not yet run correctly, and one of them contains a backup of my 40Meg drive. It's formatted to 40Megs, and works perfectly like this, but I need to back it up and then

try formatting it up to 80Megs. This is the reason that I'm buying the SCSI card. So I can sort my filing system (and therefore my life) out! I suppose I was a bit ambitious in buying 80 Meg drives. For most floppy users out there, the purchase of a Myarc HFDCC and a couple of 40Meg drives would be adequate. The Myarc HFDCC will soon be available again, since Cecure are going to manufacture it again! Let's hope that they fix a price that is competitive with the SCSI, and not overprice themselves, otherwise everyone will disregard it and go straight for the SCSI, but I think that there is room for both, and you could even buy both and use both if you want.

ST506/412 drives can be found at most radio rallys etc. for about £20 for a 40Meg, or £10 for 20Meg, but make sure they are MFM (Modified Frequency Modulation) encoding, because RLL (Run Link Limited) encoding drives pull to much current from the controller. Amstrad made this mistake when supplying drives with their PC's. The users got them home and turned on, and the controllers blew up!

Well, I think that's it for now on the SCSI. Go and get a strong coffee, or something even stronger, and sit down, or preferably lay down, and contemplate the future. I said in my last article, "The future starts here", and I wasn't wrong was I.

Since it's quite late, and I've still got to draw my map, I'd better make it quick with my next items. As I promised in my last article, I said I would have my ScreenWriter program completed for this issue. You can rely on the Bluesman. Unfortunately, I've wasted alot of time this afternoon trying to sort out one last problem. The program works completely, and when you LINK the routines it creates, it draws the screen, and redefines all of the character (CALL CHAR) codes in a split second, but unfortunately I've still not found the correct address for the starting address of the Colour Descriptor Table. This table controls the character set colours which are normally defined with CALL COLOR subprograms. This will only require changing the address that's contained in my program, to the correct address. Maybe Mark can help with this. I'm also sorry Mark that I didn't have time to re-write the code that generates the screen image table, so there are 768 individual BYTE directives in the generated source code!

I've got to also write ***Somewhere In TI*MES***, so I think I'll multi-task my text. ScreenWriter will be listed on the left of the page in 28 column format (generated with Triton SUPER XB) and ***Somewhere In TI*MES*** will be on the right of the page!

Here goes!

```
1 DIM SCREEN(24,32),COLORS(1
4,2),H$(15):: GOSUB 11000 ::
 CALL CLEAR :: CH=65 :: SC=8
 :: CALL SPRITE(#1,143,7,96,
128,#2,65,16,96,128)

2 DISPLAY AT(15,1):"L-LOAD P
REVIOUS SCREEN       S-SAVE C
URRENT SCREEN        9-GENERA
```

Somewhere In TI*MES
Issue 4
Spring 1984

Having this listing on our left hand side doesn't leave us much room does it!

On the front cover is a

```
TE 9900 ROUTINE"

3 DISPLAY AT(19,1):"B-CALL S
CREEN(A)              C-CALL C
OLOR(A,B,C)           D-CALL C
HAR(A,A$)             N-NEW CH
ARACTER               E-EXIT"

4 CALL JOYST(1,X,Y):: CALL M
OTION(#1,-Y,X):: GOSUB 8 ::
CALL KEY(1,K,S):: IF K=18 TH
EN GOSUB 10 ELSE CALL KEY(3,
K,S):: IF S=0 THEN 4

5 IF K=76 THEN GOSUB 1000 EL
SE IF K=83 THEN GOSUB 2000 E
LSE IF K=57 THEN GOSUB 3000
ELSE IF K=66 THEN GOSUB 4000

6 IF K=67 THEN GOSUB 5000 EL
SE IF K=68 THEN GOSUB 6000 E
LSE IF K=78 THEN GOSUB 7000
ELSE IF K=69 THEN GOTO 8000

7 GOTO 4

8 CALL POSITION(#1,R,C):: CA
LL LOCATE(#2,MIN(MAX(1,8*INT
(R/8)+1),192),MIN(MAX(1,8*IN
T(C/8)+1),256)):: RETURN

10 CALL POSITION(#1,R,C):: R
=INT(R/8)+1 :: C=INT(C/8)+1
:: IF R<1 OR R>24 OR C<1 OR
C>32 THEN CALL SAY("OFF+OF+S
CREEN")):: RETURN

11 CALL HCHAR(R,C,CH):: SCRE
EN(R,C)=CH :: RETURN

1000 ! L-LOAD SCREEN

1001 INPUT "ENTER DEVICE.FIL
ENAME:":F$ :: OPEN #1:F$,DIS
PLAY ,VARIABLE 20 :: FOR D=0
 TO 14 :: INPUT #1:COLORS(D,
1):: INPUT #1:COLORS(D,2)

1002 NEXT D :: INPUT #1:SC :
: FOR R=1 TO 24 :: FOR C=1 T
O 32 :: INPUT #1:A :: IF A=0
 THEN SCREEN(R,C)=32 ELSE SC
REEN(R,C)=A

1003 NEXT C :: NEXT R :: FOR
 D=32 TO 143 :: LINPUT #1:A$
 :: CALL CHAR(D,A$):: PRINT
D,CHR$(D),A$ :: NEXT D :: CL
```

picture of two children playing with a console, running an educational program that I've never seen. On the inside cover was a full page advert for PARCO, and they were still at New Street, Honiton.

They had cut the price of Atari cartridges to members, and offered them at £18 each, instead of £19.50. I got DEFENER, SHAMUS and PROTECTOR II for £5 for the lot the year before.

On the right of the contents page were screen shots of PacMan, Defender, DigDug, and Donkey Kong cartridges.

At the bottom of the contents page in big lettering it said, **"Keeping you in touch with all the latest developments".** After reading my SCSI review, it seems we still are!

The first item in this issue of TI*MES was a greeting to new members who had just joined the group. This was followed by the message that the TI Users Library was not being supported! Sounds familiar, doesn't it!!

Page 3 was "Your Letters" which started with a thank you to Peter Brooks from V. Comley, YMCA, Welwyn Garden City. He was thanking him for an article on single pixel drawing. He was worried that he wouldn't be able to draw a decent graph with the 4A. He bought a graphics package from Stainless Software (Norton Graphics) but said the program did not give one the ability to use it in conjunction with any other program, and since it was protected, he couldn't modify it.

He'd have no need now, and no trouble producing a very decent graph now that we've got things like the Missing Link for XB.

The second letter was from D.A.Jones, Haverfordwest, Dyfed. He had got Terminal Emulator 2, and Speech Synthesizer, and had programmed the 4A to speak Welsh!

```
OSE #1 :: GOSUB 9000 :: RETU
RN

2000 ! S-SAVE SCREEN

2001 INPUT "ENTER DEVICE.FIL
ENAME:":F$ :: OPEN #1:F$,DIS
PLAY ,VARIABLE 20 :: FOR D=0
 TO 14

2002 IF COLORS(D,1)=0 THEN P
RINT #1:2 :: PRINT #1:1 ELSE
 PRINT #1:COLORS(D,1):: PRIN
T #1:COLORS(D,2)

2003 NEXT D :: PRINT #1:SC :
: FOR R=1 TO 24 :: FOR C=1 T
O 32 :: IF SCREEN(R,C)=0 THE
N PRINT #1:32 ELSE PRINT #1:
SCREEN(R,C)

2004 NEXT C :: NEXT R :: FOR
 D=32 TO 143 :: CALL CHARPAT
(D,A$)

2005 PRINT A$ :: PRINT #1:A$
 :: NEXT D :: CLOSE #1 :: GO
SUB 9000 :: RETURN

3000 ! 9-GENERATE 9900 ROUTI
NE

3001 PRINT "GENERATE SOURCE
CODE TO      PRINTER, DISK, O
R BOTH?":"":"ENTER:" :: INPU
T "(P)RINTER D(ISK) B(OTH)":
C$ :: IF C$="P" THEN S=2 ::
E=2

3002 IF C$="D" THEN S=1 :: E
=1 ELSE IF C$="B" THEN S=1 :
: E=2

3003 IF C$="D" OR C$="B" THE
N PRINT "":"" :: INPUT "PLEA
SE ENTER DEVICE.FILENAMEFOR
SOURCE CODE:                ":D$

3004 IF C$="P" THEN OPEN #2:
"PIO" ELSE IF C$="D" THEN OP
EN #1:D$ ELSE IF C$="B" THEN
 OPEN #1:D$ :: OPEN #2:"PIO"

3005 F$="" :: FOR D=LEN(D$)T
O 1 STEP -1 :: K$=SEG$(D$,D,
1):: IF K$="." THEN F$=F$&RP
T$(" ",7-LEN(F$)):: GOTO 300
7 ELSE F$=K$&F$
```

His main use for TE-II was communication, and he wanted to hear from other users with the same interest in linking up via MODEM.

Next was from S.Merret, from Richmond in Surrey. He provided a list of 4A compatible cassette recorders that a TI representitive handed out at the Barbican Computer show in 1983.

The next letter sounds very interesting. It's from F.W. Seaman, of Derby. He writes that he saw a book in his local library that had a picture on the front, and made a lot of references to the 4A. He says it was pleasing to find so many relevant details of the TI-99/4 in a general guide book.

I wonder if it's still available. "COMPUTERS AND YOUR CHILD" by Ray Hammond, published by Century Publishing Co. Ltd. ISBN: 07126 0092 2, it sold for £5.95. When I'm in London, I'm going to try and buy that.

Page 4 was **SCENE U.S.A** which was a report on Walt Disney Sierra On-Line and Imagic signing software agreements with TI.

Disney was to become the first third party software publisher to manufacture and market software for the 99/4A.

Below this was something very distressing! An announcement of TI launching a "32-bit" system called the NU MACHINE. It used a 68010 processor! The 9995 and 99000 would have left it standing. It had 512K RAM, and 84 Megs of hard disk.

The model designed for "computer room" applications would have a 474 Meg hard disk.

Next was a warning that your computer might be a V2.2 console. If it says 1983 on the colour bar screen, then it will not run third party software because TI made changes to the console.

```
3006 NEXT D

3007 FOR COPY=E TO S STEP -1
:: PRINT #COPY:"       DEF
"&F$:"VMBW    EQU  >2024":"VW
TR    EQU  >2030"

3008 PRINT #COPY:F$&"LI    R0
,>070"&H$(SC-1):"       BLWP
 @VWTR":"        LI    R0,2048
":"        LI    R1,CDAT":"
    LI    R2,15"

3009 PRINT #COPY:"        BLW
P @VMBW":"        CLR   R0":"
      LI    R1,SDAT":"
LI    R2,770":"        BLWP @V
MBW":"        LI    R0,1024"

3010 PRINT #COPY:"        LI
  R0,1024":"        LI    R1,P
DAT":"        LI    R2,872":"
      BLWP @VMBW":"        RT
"

3011 PRINT #COPY:"SDAT    BYT
E ";SCREEN(1,1)+96 :: FOR C=
2 TO 32 :: PRINT #COPY:"
   BYTE ";SCREEN(1,C)+96 ::
NEXT C

3012 FOR R=2 TO 24 :: FOR C=
1 TO 32 :: PRINT #COPY:"
   BYTE ";SCREEN(R,C)+96 ::
NEXT C :: NEXT R

3013 PRINT #COPY:"CDAT    BYT
E >"&H$(COLORS(0,1)-1)&H$(CO
LORS(0,2)-1):: FOR D=1 TO 14
 :: PRINT #COPY:"       BYTE
 >"&H$(COLORS(D,1)-1)&H$(COL
ORS(D,2)-1):: NEXT D

3014 CALL CHARPAT(32,A$):: P
RINT #COPY:" ":"PDAT    DATA
>"&SEG$(A$,1,4)&",>"&SEG$(A$
,5,4)&",>"&SEG$(A$,9,4)&",>"
&SEG$(A$,13,4)

3015 FOR D=33 TO 143 :: CALL
 CHARPAT(D,A$):: PRINT 143-D

3016 PRINT #COPY:"        DAT
A >"&SEG$(A$,1,4)&",>"&SEG$(
A$,5,4)&",>"&SEG$(A$,9,4)&",
>"&SEG$(A$,13,4):: NEXT D ::
PRINT #COPY:"        END"
```

On page 6 was something very interesting. **Charlie's Page**, written by Charles La Fara of the international 99/4 Home Computer Users Group. I've read some not very nice things in "The Orphans Chronicles" about the way he did business. The International User Group was a profit making group, and took someone to court for distributing public domain software that they claimed a copyright over. He writes about many things, including the GREAT EXTENDED BASIC shortage of 1980, the rebate wars of 1981-1982, and the third-party lockout of 1983.

He had an official statement saying that 4A hardware would no longer be available, including UCSD PASAL, Video Controller, HEX-BUS Interface, MBX Expansion and additional 99/4A consoles.

Page 7 was the start of a six page RAMBLES from Stephen Shaw. This included everything from making back up copies of commercial software, to little example programs.

The most interesting though was a report on PERSONAL RECORD KEEPING, and STATISTICS modules. This was what I mentioned in my last article, and is about them adding new commands to TI BASIC.

If you've got one of these cartridges, try:

```
CALL D(6,2,26,"SURPRISE")
```

These modules add CALLs which simulate DISPLAY AT and ACCEPT AT and also, uniquely allow you to partition the VDP RAM, store data there, and save it in PROGRAM format: verifiable, and FAST.

He offered copies of a TI booklet on the subject for £1 each. I wouldn't mind reading about this. Any chance of an article on it Stephen?

He also reviewed Atarisoft modules, and covered the subject of V2.2 consoles that the modules caused problems with.

Atarisoft said all their

```
3017 NEXT COPY :: GOSUB 9000
 :: IF C$="B" THEN CLOSE #1
:: CLOSE #2 ELSE IF C$="P" T
HEN CLOSE #2 ELSE IF C$="D"
THEN CLOSE #1

3018 RETURN

4000 ! B-CALL SCREEN

4001 DISPLAY AT(1,1):"SCREEN
 COLOR:" :: ACCEPT AT(1,14):
SC :: CALL SCREEN(SC):: GOSU
B 10000 :: RETURN

5000 ! C-CALL COLOR(A,B,C)

5001 DISPLAY AT(1,1):"CALL C
OLOR(" :: ACCEPT AT(1,12):A
:: DISPLAY AT(1,14):"," :: A
CCEPT AT(1,15):B :: DISPLAY
AT(1,17):"," :: ACCEPT AT(1,
18):C

5002 DISPLAY AT(1,20):")" ::
 CALL COLOR(A,B,C):: COLORS(
A,1)=B :: COLORS(A,2)=C :: G
OSUB 10000 :: RETURN

6000 ! D-CALL CHAR(A,A$)

6001 DISPLAY AT(1,1):"CHAR("
 :: ACCEPT AT(1,6):A :: DISP
LAY AT(1,9):","&CHR$(34):: A
CCEPT AT(1,11):A$ :: DISPLAY
 AT(1,20):CHR$(34)&")"

6002 CALL CHAR(A,A$):: GOSUB
 10000 :: RETURN

7000 ! N-NEW CHARACTER

7001 DISPLAY AT(1,1):"ENTER
ASCII CODE:" :: ACCEPT AT(1,
18):CH :: CALL PATTERN(#2,CH
):: GOSUB 10000 :: RETURN

8000 ! E-EXIT

8001 CALL SCREEN(8):: PRINT
"":"* READY *":"" :: CALL HC
HAR(24,2,62):: FOR D=1 TO 10
 :: CALL HCHAR(24,3,32):: FO
R C=1 TO 90 :: NEXT C

8002 CALL HCHAR(24,3,30):: F
OR C=1 TO 90 :: NEXT C :: NE
XT D
```

modules worked on their ONE TI console!

Stephen also reviewed the Smart Programming Guide for Sprites, from which Trevor has obtained some excellent routines for his recent articles on improving the response of sprite motion with joystick control.

Pages 13 and 14 where a two page advert for Stainless Software.

Page 15 was ARCADE which included two short programs from Kerry Martin of LA 99'ers User Group. This was a PACMAN game, and Russian Roulette! Did it make the console explode if you lost!

Page 16 was a page from the Sidney, Australia User Group. It included a program to play and print the frequencies of two additional base octaves on the TI 99/4A, and also the world's shortest Tic-Tac-Toe program.

There were also two other very interesting programs that I might just re-print. Both for console-only, and TI BASIC.

Pages 17 to 19 was a Mini-Memory special about disassembling TMS9900. This was from Geoff Nunn of the Perth, Australia, Users Group, and it included a complete disassembled program in TMS9900.

Page 20 was something very pleasing to hear from Paul K. Dunderdale of Preston.

He had owned 5 ZX Spectrums, 2 ZX81's, and a TI-99/4A.

Four of the Spectrums, and a ZX81 all died, but his TI-99/4A lives on!

He printed a very good routine which tells you what day it will be on a given date.

Why not! Here it is!

```
10 CALL CLEAR

20 INPUT "DAY NO? ":D
```

```
8005 PRINT "":"":"JUST KIDDI
NG!":"":"":"IF YOU'VE QUIT A
CCIDENTALLY,YOU CAN STILL SA
VE YOUR WORK":"":"DO YOU WAN
T TO SAVE IT (Y/N)"

8006 CALL KEY(0,K,S):: IF S=
0 THEN 8006 ELSE IF K=78 OR
K=110 THEN PRINT "O.K. BYE!"
 :: END ELSE GOSUB 2000 :: E
ND

9000 ! RE-DRAW SCREEN

9001 FOR R=1 TO 24 :: FOR C=
1 TO 32 :: IF SCREEN(R,C)<32
 THEN 9002 ELSE CALL HCHAR(R
,C,SCREEN(R,C))

9002 NEXT C :: NEXT R :: FOR
 C=0 TO 14 :: IF COLORS(C,1)
=0 OR COLORS(C,2)=0 THEN 900
3 ELSE CALL COLOR(C,COLORS(C
,1),COLORS(C,2))

9003 NEXT C :: CALL SCREEN(S
C):: RETURN

10000 ! RE-DRAW FIRST LINE

10001 FOR C=1 TO 32 :: IF SC
REEN(1,C)=0 THEN CALL HCHAR(
1,C,32)ELSE CALL HCHAR(1,C,S
CREEN(1,C))

10002 NEXT C :: RETURN

11000 FOR D=0 TO 9 :: H$(D)=
STR$(D):: PRINT D :: NEXT D
:: FOR D=10 TO 15 :: H$(D)=C
HR$(D+55):: PRINT D :: NEXT
D

11001 FOR R=1 TO 24 :: FOR C
=1 TO 32 :: SCREEN(R,C)=32 :
: NEXT C :: PRINT R :: NEXT
R :: FOR C=0 TO 14 :: COLORS
(C,1)=2 :: COLORS(C,2)=8

11002 NEXT C :: CALL MAGNIFY
(1):: CALL CHAR(143,"8142241
8244281"):: RETURN
```

```
30 INPUT "MONTH NO? ":M

40 INPUT "YEAR NO? ":Y

50 C=INT(Y/4)+1-((Y/4=INT(Y/
4))*(M<3))+Y* 365+VAL(SEG$("0
000310590901201511812122432 7
3304334",M* 3-2,3))+D

60 PRINT ::SEG$("THUFRISATSU
NMONTUEWED",(C-INT(C/7)* 7+1)
* 3-2,3);D;".";M;".";Y:::

70 GOTO 20
```

The day no. should be between 1 and 31.

The month no. should be between 1 and 12 and,

The year should be four digits. eg. 1984, or should that be 1994!

Page 21 was an advert for the UNEX Graphics Creator and Screen Editor. I've never heard of this either!

It allowed the creation of up to 52 `on-board' graphics.

Design large screen layouts for your own programs. SAVE ALL of your work onto cassette tape. Commands included ROTATE, MIRROR, INVERT, IN-HEX, OUT-HEX, ACCIDENT REPAIR, and many more.

It only cost £5.95 from PikaDee Software, 35 Parker St. Preston, Lancashire. PR22AH

Page 22 was about playing adventure games on the TI-99/4A.

This was a tutorial on the techniques of playing.

Page 24 was BABBLING BROOKS (Peter Brooks) writing about control and function keys on the 99/4A. It continued up to page 27 and even included a comparison of key funtions and ASCII codes between the 99/4 and 99/4A.

Wow! My ScreenWriter listing has finished! Shall we spread out a bit?

Page 28 was MODULATOR MODIFICATION which should be very interesting for Trevor Stevens, since he's having trouble with his modulator. Let's print it big so he can find it easily!

MODULATOR MODIFICATION

This was from Dave Hewitt of Hoddesdon and concerns the conversion to allow the 99/4A to be connected directly to the modulators on video recorders and on the newer TV's.

Yes, this means it can go straight into Composite Video on your monitor. The TI console actually outputs its signal in YUV, which is TV studio standard output.

TV cameras that are used by ITV or BBC all output their signal in YUV! Do that on a BBC!

What you need to do is disconnect the modulator from the console and TV and remove the three screws which hold the top cover on (Trevor's done this already!). The part of the circuitry that you need to get at is the input to the UHF modulator from the processing circuits that preceed it. This signal is at just the right level and impedance to feed into the already mentioned video input circuit on a TV set or into the camera or auxiliary socket which is provided on most video recorders, and top quality monitors.

The audio circuit can be connected directly to the audio phono plug of the video or monitor, but the video circuit needs coupling through a 47uF capacitor (in practice any capacitor between 10uF and 1000uF worked fine). With the modulator unit turned to that the computer is coming out of the bottom, observe this lead as it enters the box, and you will find that its inner cables divide six ways (including the outer screen which has been made tidy by wrapping in a bit of thick yellow sleeving). The audio signal is carried by the thin yellow wire which goes to the bottom right hand side of the printed circuit board.

Cut this wire and fit a suitable piece of sleeving over it (to insulate the connection again when you have finished). Join a fresh length of suitable audio single screened wire to the junction of these wires, the inner cable being joined now to the yellow audio lead. Fold back the outer screened wire of the new cable, fit some sleeving over this and solder the end on to a convenient point on the metal case of the box (the silver coloured Screening Can of the modulator is ideal for this purpose). Fit the correct type of plug on the other end of the new cable to fit the audio

input of your TV or video (normally a phono plug). This lead may be used on its own for taking the output of the computer's sound circuits to your stereo system, to give a better response than the speaker on your TV might offer.

Next, locate the small 1K ohm resistor which sits above and to the right of the modulator. Solder another inner cable from the left hand side of this resistor, again soldering the outer screening cable onto the modulator's Screening Can. Fix a suitable (phono) plug on the other end of this lead and fit into the video input socket of your video or monitor. Switch your video to the Camera, or AUX. position and you can observe the output from your computer via the channel you would normally use for playing video recordings.

This now allows you to make video recordings of your favourite games, and have action replays.
If you connect the inputs directly into the sockets on the back of a domestic TV the gain in picture definition and stability is quite marked, and frees you from that annoying tuner drift which often occurs as modulators and tuners warm up.

All that remains is to drill holes in the modulator unit's casing, large enough to accept the two new cables with protecting gromets. If preferred, one could fit suitable chassis sockets on the lid or case of the TI modulator so that the unit can be unplugged from the new leads, or maybe a line socket could be used on a short lead, but both of course should use proper co-ax screened cable.
Dave Hewitt attached 12 feet of lead on both outputs and have not encountered any undue `pick-up'. The UHF output from the TI modulator is not disturbed and may be used at the same time without affecting either cable!

I can personally guarantee composite video over forty feet since I've got a 40ft cable that goes from my video in the loft, to the video in the lounge, and I use it for tape-to-tape recordings!

At the bottom of page 30 was a note from Peter Brooks saying that he had set up his own user group (Oxon TI Users), and would be distributing a newsletter called `TI-LINES'.

Pages 31 to 40 were a massive (for 1984 anyway) article and advert for Arcade Hardware, by Howard Greenberg.
This was a review which included Mini-memory, Arcade Machine Joystick, Introduction to Assembly Language by Ralph Molesworth, a 40-column thermal

printer, TI-Writer, and a few modules. Some of these modules were the Atarisoft modules Donkey Kong, Pac-Man, and Defender.

Pages 41 and 42 were a few programming hints and tips from Cin-Day User's Group (Cincinnati).

How come we seem to have lost contact with alot of the American user groups?

Pages 43 to 44 was a tutorial on beginning to program and the writing of flowcharts, by Ian Godman.

I don't like flowcharts! I prefer Structure Charts and Data Flow Diagrams, so I'll skip this.

Structure Charts and Data Flow Diagrams are the most (only) educational thing that the Nottingham Trent University has taught me (not forgetting Finite State Machines of course, which is an efficient method for writing Lexical Analysers to be used in compilers!).

Page 45 was the continuation of the flowchart tutorial at the top,. and the rest was about the TI EXCHANGES with second hand cartridges to exchange. The beginnings of the current Module Library, of which Francesco submitted a current list of modules in the Winter issue.

Page 46 was devoted to classified adverts, for B.JACKSON SOFTWARE, and PARCO ELECTRICS. B.JACKSON SOFTWARE was based at 21 Rowan Way, New Balderton, Newark, Notts. NG24 3AU. That's not far from Gary!

The back page was devoted to the Computer Home Service, and offered for sale, a Dust Cover, Cassette Demagnitizer, Getting Started with the 99/4A by Stephen Shaw, and an Aerial Splitter!

More reviews from the 4th Dimension next issue.

Somewhere In TI*MES

Will return!

In the words of my friend Paul, quoting from "Back To The Future" when he got out of the car after we went swimming. He wished me good luck in London and said "Watch that re-entry, it can be a little bumpy!"

Oh, I almost forgot, I didn't mention any prices for the SCSI card.

The standard SCSI interface costs $170.

The optional floppy controller card costs $100.
Air-Mail costs $10.

That's $280.

As I said, I sent my order off, for two on the 25th of February, so you can look forward to a proper review of its actual operation in my next article.

There's just one more thing to write about. Last weekend I returned Mark's offer of going to stay with him for the weekend in Shrewsbury, to get a bit of programming done, along with alot of chatting on all TI subjects!

He has been given another console by John Murphy, so he's got a useable system, and we SHOULD expect more Drivel from him! He showed me his Games Writers Toolkit for Mini-Memory, and the new version he's working on for Extended BASIC. It's an amazing set of routines, and includes windowing routines, scrolling routines, screen-grabbing routines, sprite routines, and much more.

It includes colour cycling routines, that BELIEVE ME, puts DELUXE PAINT on the AMIGA to shame. And that is official. Considering that the AMIGA is multi-tasking, with the speed of Mark's colour cycling demo, if his 4A had another couple of tasks to worry about at the same time, it would still be beating the AMIGA, so don't give up on the machine yet.

I was amazed when he brought out his box of modules and I saw Miner 2049'er. O.K. the game is crap, but the amazing thing is that the cartridge plugs in the side of the console, and works like a DSR power up routine. When the 4A switches on or resets to the Title Screen, it checks CRU for DSR's and power up routines. When it hits Miner 2049'er, the power up routine causes a header to be written into the standard GROM space, and the game appears on the menu selection!

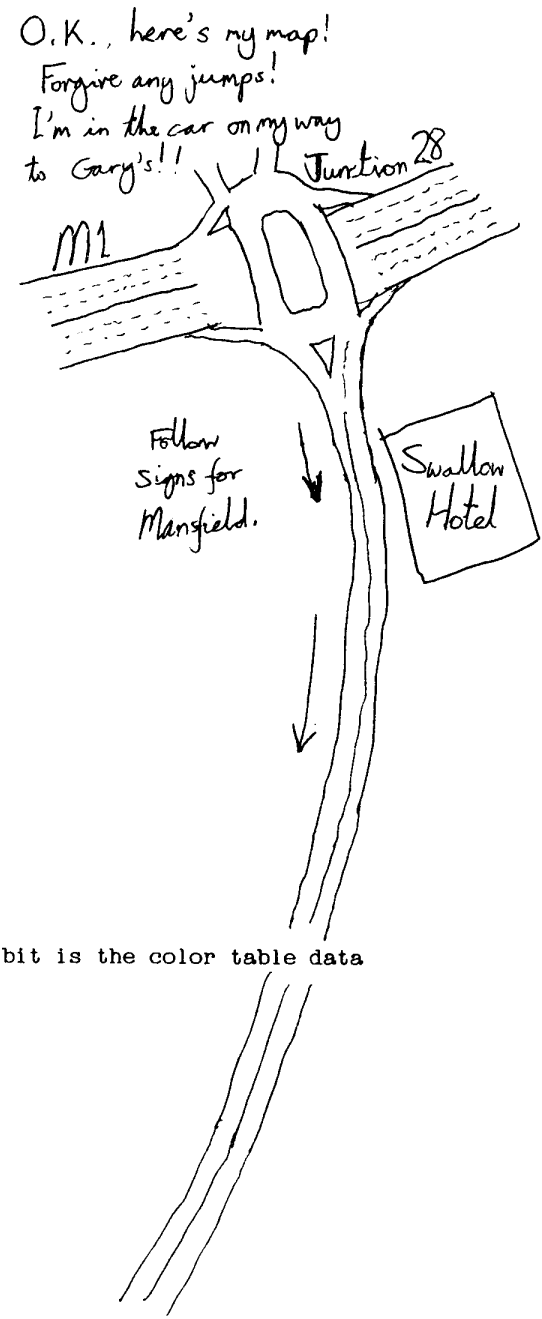
I also noticed that he'd got TI-Writer. Could you bring the TI-Writer module to the AGM Mark? I want to rip it off with C-SAVE.

Also, could anyone who owns a DataBiotics 32K Super-Space cartridge, bring it to the AGM so Gary and I can have a look how it's bank-switching works. If we produce the 256K Mega-Cart, then we want it to be compatible!

I almost forgot to include an example of the source code that is produced with ScreenWriter. I've cut out some of the screen data to save space.

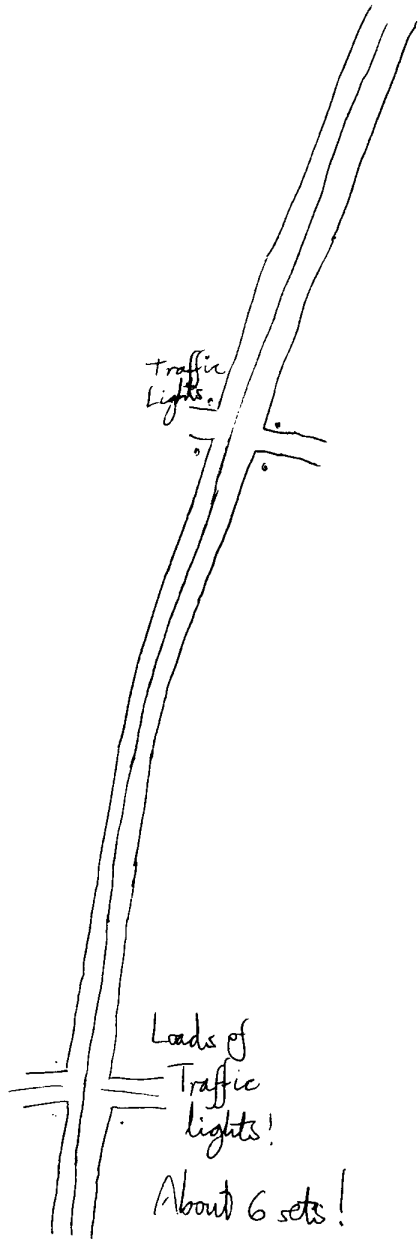
```
        DEF SCREEN
VMBW    EQU  >2024
VWTR    EQU  >2030
SCREEN  LI   R0,>0707
        BLWP @VWTR
        LI   R0,2048
        LI   R1,CDAT
        LI   R2,15
        BLWP @VMBW
        CLR  R0
```

```
        LI    R1,SDAT
        LI    R2,770
        BLWP  @VMBW
        LI    R0,1024
        LI    R0,1024
        LI    R1,PDAT
        LI    R2,872
        BLWP  @VMBW
        RT
SDAT    BYTE  142
        BYTE  142
        BYTE  202
        BYTE  236
        BYTE  236
        BYTE  202    /
        BYTE  202   /
        BYTE  202/
        BYTE  2/
        BYTE /
        BYT/
        B/
       /   These BYTE
     / directives     /
   / describe all    /
 /  of the screen   /
   display, so    /
there's 768    /2
lines of     /222
them!       /  222
         /TE   222
       /BYTE   222
     /  BYTE   222
   /    BYTE   222
 /      BYTE   222
        BYTE   222
        BYTE   222
        BYTE   222
        BYTE   222
        BYTE   222
        BYTE   222
        BYTE   222
        BYTE   222
        BYTE   222
        BYTE   202
CDAT    BYTE >10        This bit is the color table data
        BYTE >10
        BYTE >12
        BYTE >10
        BYTE >10
        BYTE >F5
        BYTE >10
        BYTE >10
        BYTE >10
        BYTE >10
        BYTE >10
        BYTE >1F
        BYTE >10
        BYTE >C1
        BYTE >1C
        BYTE >61
```

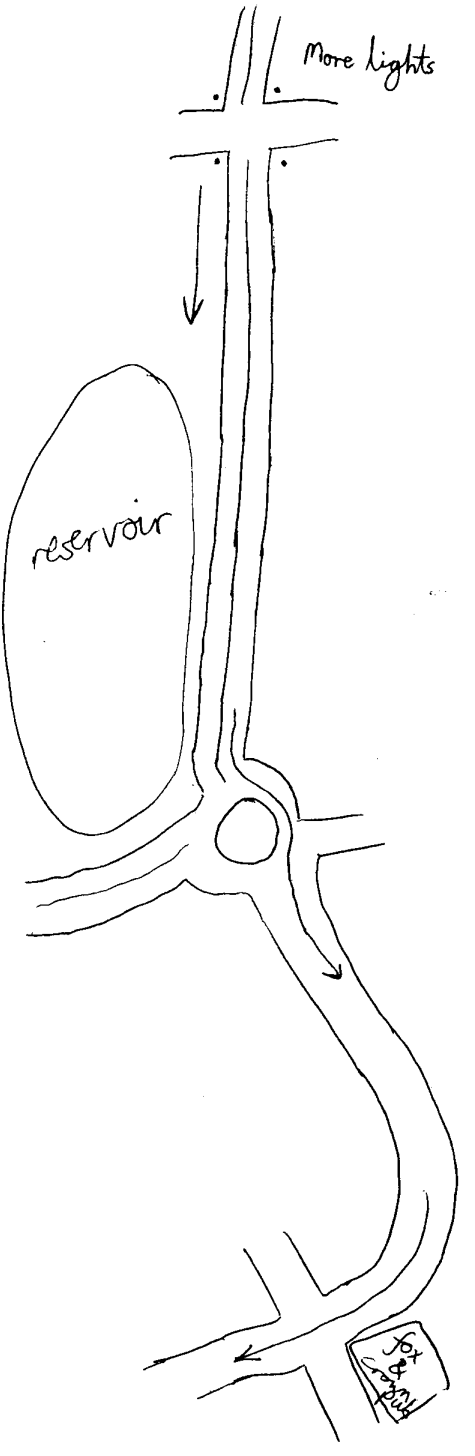


```
PDAT    DATA >0000,>0000,>0000,>0000
        DATA >0010,>1010,>1010,>0010
        DATA >0028,>2828,>0000,>0000
        DATA >0028,>287C,>287C,>2828
        DATA >0038,>5450,>3814,>5438
        DATA >0060,>6408,>1020,>4C0C
        DATA >0020,>5050,>2054,>4834
        DATA >0008,>0810,>0000,>0000
        DATA >0008,>1020,>2020,>1008
        DATA >0020,>1008,>0808,>1020
        DATA >0000,>2810,>7C10,>2800
        DATA >0000,>1010,>7C10,>1000
        DATA >0000,>0000,>0030,>1020
        DATA >0000,>0000,>7C00,>0000
        DATA >0000,>0000,>0000,>3030
        DATA >0000,>0408,>1020,>4000
        DATA >0038,>4444,>4444,>4438
        DATA >0010,>3010,>1010,>1038
        DATA >0038,>4404,>0810,>207C
        DATA >0038,>4404,>1804,>4438
        DATA >0008,>1828,>487C,>0808
        DATA >007C,>4078,>0404,>4438
        DATA >0018,>2040,>7844,>4438
        DATA >007C,>0408,>1020,>2020
        DATA >0038,>4444,>3844,>4438
        DATA >0038,>4444,>3C04,>0830
        DATA >0000,>3030,>0030,>3000
        DATA >0000,>3030,>0030,>1020
        DATA >0008,>1020,>4020,>1008
        DATA >0000,>007C,>007C,>0000
        DATA >0020,>1008,>0408,>1020
        DATA >0038,>4404,>0810,>0010
        DATA >0038,>445C,>545C,>4038
        DATA >0038,>4444,>7C44,>4444
        DATA >0078,>2424,>3824,>2478
        DATA >0038,>4440,>4040,>4438
        DATA >0078,>2424,>2424,>2478
        DATA >007C,>4040,>7840,>407C
        DATA >007C,>4040,>7840,>4040
        DATA >003C,>4040,>5C44,>4438
        DATA >0044,>4444,>7C44,>4444
        DATA >0038,>1010,>1010,>1038
        DATA >0004,>0404,>0404,>4438
        DATA >0044,>4850,>6050,>4844
        DATA >0040,>4040,>4040,>407C
        DATA >0044,>6C54,>5444,>4444
        DATA >0044,>6464,>544C,>4C44
        DATA >007C,>4444,>4444,>447C
        DATA >0078,>4444,>7840,>4040
        DATA >0038,>4444,>4454,>4834
        DATA >0078,>4444,>7850,>4844
        DATA >0038,>4440,>3804,>4438
        DATA >007C,>1010,>1010,>1010
        DATA >0044,>4444,>4444,>4438
        DATA >0044,>4444,>2828,>1010
        DATA >0044,>4444,>5454,>5428
        DATA >0044,>4428,>1028,>4444
        DATA >0044,>4428,>1010,>1010
```

This bit is the pattern table. All CALL CHAR definitions from 32 to 143!



```
DATA >007C,>0408,>1020,>407C
DATA >0038,>2020,>2020,>2038
DATA >0000,>4020,>1008,>0400
DATA >0038,>0808,>0808,>0838
DATA >0000,>1028,>4400,>0000
DATA >0000,>0000,>0000,>007C
DATA >0000,>2010,>0800,>0000
DATA >0000,>0038,>447C,>4444
DATA >0000,>0078,>2438,>2478
DATA >0000,>003C,>4040,>403C
DATA >0000,>0078,>2424,>2478
DATA >0000,>007C,>4078,>407C
DATA >0000,>007C,>4078,>4040
DATA >0000,>003C,>405C,>4438
DATA >0000,>0044,>447C,>4444
DATA >0000,>0038,>1010,>1038
DATA >0000,>0008,>0808,>4830
DATA >0000,>0024,>2830,>2824
DATA >0000,>0040,>4040,>407C
DATA >0000,>0044,>6C54,>4444
DATA >0000,>0044,>6454,>4C44
DATA >0000,>007C,>4444,>447C
DATA >0000,>0078,>4478,>4040
DATA >0000,>0038,>4454,>4834
DATA >0000,>0078,>4478,>4844
DATA >0000,>003C,>4038,>0478
DATA >0000,>007C,>1010,>1010
DATA >0000,>0044,>4444,>4438
DATA >0000,>0044,>4428,>2810
DATA >0000,>0044,>4454,>5428
DATA >0000,>0044,>2810,>2844
DATA >0000,>0044,>2810,>1010
DATA >0000,>007C,>0810,>207C
DATA >0018,>2020,>4020,>2018
DATA >0010,>1010,>0010,>1010
DATA >0030,>0808,>0408,>0830
DATA >3C7E,>DBFF,>BDDB,>663C
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >243C,>7EFF,>BD99,>99DB
DATA >0A14,>0EFF,>0E14,>0A00
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
DATA >0000,>0000,>0000,>0000
END
```



And that's more or less it from me! I'll see you at the AGM on the 14th of May, and I will return with my Summer article, including:

SCSI strikes again!!

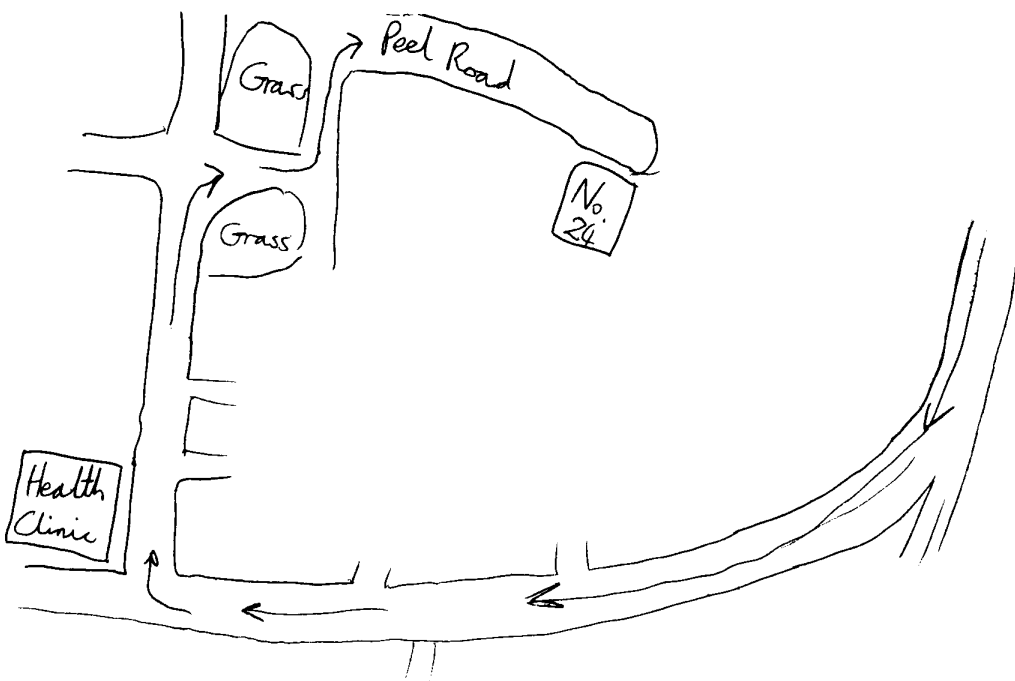
All for now from the Bluesman.

Richard Twyning signing off.

The time now is 09:52 on the 27th of February 1994, and I've been typing since 06:15, just so I could get my article finished, and delivered to Gary. Apologies for any errors, and print quality. I'm printing in Super Letter Quality Mode, but I'm using a ribbon which has been livened up with WD40!

That's it. Over and out.

Richard Twyning



I want to be in a position to write uncomplicated routines in Assembler, mostly to be called from XB. I don't see the language itself as a problem; it's the getting it implemented. My position is possibly not helped by a decision I've taken that I'm not going to keep shifting my XB module in and out. That, of course, eliminates using the official TI E/A module or Minimemory and leaves me with the E/A Emulation in Funnelweb (my version is 4.31). The trouble then is getting clear what I'm supposed to do. I have the TI Manual which is effectively a large catalogue of resources in a system I haven't got. The Funnelweb documentation seems to imply that things will mostly run according to the Manual except when they don't. I make progress in spite of my information sources rather than with their aid.

I have now been able to take a small published source listing to the no-errors object file stage. When I try to load/run, the immediate result is a DEF table heading with no contents other than a cursor that won't budge. I gather that I am supposed to make a selection and then off we'll go. As there is nothing to select from, I'm not surprised the cursor stays put. But, how have I brought this impasse about? So far as I can tell, under the official E/A, the table wouldn't even be there.

I send separately details of the various listings and would be glad if you can point to my error. Also, can you recommend any book that illuminates the principles of assembly rather than the grammar of the statements?

Unwisely, I have got more deeply involved with RND than I intended. By courtesy of Stephen Shaw, I now have a listing of Basic RND from GROM (written, I am told, in GPL) and an Assembler subroutine from ROM called by this. Mindful, no doubt, of the benefits of a challenge, he has left the decipherment for me. I think I've done pretty well with the S/R and I attach my analysis for you to confirm (hopefully). As you see, there is one obscurity. In my mind, I picture GROM as an odd kind of factory "Stores" where you hand in your chitty at one special counter and pick up your items a bit later at another. The S/R seems to visit the hand-in counter without a new chitty but simply to copy what was left there by the last applicant. This, I believe, would have been the address in GROM of the item he wanted. To this, the S/R adds 1 and uses it as a divisor in a bit of arithmetic!! Too weird to be true. Where have I gone wrong?

In passing, this raises the question how one is supposed to know what is where in GROM so that one can ask for it properly.

I am less concerned over the GPL routine but would be glad of any enlightenment particularly as to why there are two calls to the S/R and what is happening to the floating-point accumulator.

Program For Assembly And Execution..

Source

```
            IDT 'MORQ-5'          ALSO TRIED: DEF OTEST
M1          DATA >9125
M2          DATA >102C
M3          BSS 2
            LWPI >70B8
            MOV @M1,R0
            MOV @M2,R1
            C R0,R1
            JH J1
            MOV R1,R0
J1          MOV R0,@M3
            B *R11
            END
```

Object

```
0001        0001EMORQ9-5 A0000B9125B102CA0004A
0002        A0010C0002B8040B1B01BC001BC800C000
0003        7FFC9F
0004        :          99/4 AS
```

Object Code For Alternative Source

```
0001        0001E          A0000B9125B102CA0004A
0002        A0010C0002B8040B1B01BC001BC800C000
0003        7FFC9F
0004        :          99/4 AS
```

The listing showed OTEST as an undefined symbol.

Objects were assembled with conditions RLS.

Charity Race numbers. Answer to the puzzle on p.63 Issue 43: 288 people ran. Bill's number was 204. Infinitely many field sizes are arithmetically possible but the next smallest (49) and the next largest (1681) are inconsistent with the statement that "some hundreds" turned up.

Consoltation Zone....

In reply to the questions posed from Walter Allum we have a few suggestions. We are not experts in machine code or GPL and so are open to correction from anyone with a wider knowledge in this area.

By the word 'we' I mean me the editor, general secretary Richard Twyning and the chairman Trevor Stevens. After recieving these questions from Walter we decided to have a meeting at Trevors on Wednesday the 23rd. I left my work station simulating a large ASIC design that I am currently working on at university. This simulation time is now up to a couple of hours so needless to say this puts a serious dent into my free time.

I caught a bus from Nottingham at 3.00pm and eventually got to Mansfield after 7.00pm. Usually this trip will only take half an hour! What is it about snow that brings this country to a halt, imagine this problem in Sweden! Anyway, enough of a grumble about naff drivers and back on to the point.

In reverse order, your last question about what machine code books are worth while, I strongly advise '**Introduction to Assembly Language for the TI Home Computer**' by Ralph Molesworth, ISBN 0-911061-01-0. This book starts at the start which is where many other books fail. Number systems are covered first and then simple addressing. In the addressing section of the book also shows the memory layout for the 99/4A. He moves on from there to registers, assemblinmg and running programs, screen, keyboard, file access and a very useful section on linking machine code into basic!

The book has many examples to try and is a must if you are planning the jump to machine code. Another good feature of this book is that it makes reference to the Texas Instuments Editor Assembler manual which ties the whole story together.

You were correct that each program needs an identifier, this is so that when you try to execute the program it knows where to find it. The way I understand it is that when you give the program a DEF OTEST it uses OTEST as a pointer to the first executable statement. In the case here, the lable OTEST should be placed before LWPI command. Now when you load the program into memory the computer will assign the lable OTEST to the location of the first machine code command. You can now CALL LINK("OTEST") from basic or type OTEST at the prompt in E/A and the program counter will set to that point and execute your code.

That was all a bit of a mouth full but I hope it was clear. The reason OTEST is placed at LWPI and not with DATA >9125 is that data is not an executable line, it only assignes M1 and M2. The BSS line only reserves memory and is not an executable command.

The next point of note is where the LWPI command loads the workspace pointers to.

0000 - 1FFF	Console Rom
2000 - 3FFF	Memory Expansion Low Ram
4000 - 5FFF	Device Service Routines
6000 - 7FFF	Command Module Roms
8000 - 9FFF	Speech, Sound, VDP, GROMS, etc
A000 - FFFF	Expansion High Ram

70B8 appears to be in the middle of the command module roms in which case any use of the registers after this point will have no effect and the returning routine will fail and cause the system to lock up. It is possible that a module like Minimemory may have ram in this area which would be O.K. but you should make reference to the user manual for which ever module you are using.

You should be careful with the use of workspace registers. I have found it better to resume the initial workspace before exiting so that any unseen alterations in the existing workspace doesn't cause any problems.

The spaces that you saw in the first listing cause no problem. The assembler sets asside eight charcter spaces for program identification. As your first edition was called 'morq-5' which is six characters then there will be two spaces after this. The second listing I assume you deleted the IDT command which would lead to eight spaces. This command serves no useful purpose for the assembler but is useful in other respects.

My recommeded source listing would be something like:-

```
            DEF OTEST
M1          DATA >9125          assign M1 to 9125 in hex
M2          DATA >102C          assign M2 to 102C in hex
M3          BSS 2               put asside two bytes for M3
WSPACE      BSS >20             get 32 bytes for workspace
OTEST       LWPI WSPACE
            MOV @M1,R0          move 16 bits of M1 to R0
            MOV @M2,R1          move 16 bits of M2 to R1
            C R0,R1             compare values of M1 and M2
            JH J1               if R0 is logically greater
                                than R1 ( which it is ) then
                                goto J1 otherwise
            MOV R1,R0           copy R1 to R0
J1          MOV R0,@M3          put largest value in M3
          ( LWPI >83E0 )        replace GPL workspace ?
            B *R11              return to master program.
            END
```

Consoltation Zone....

Your first enquiry causes many interesting points to be brought to light. I think some of this problem will have to be solved by vice chairman and programming boff Mark Wills (grovel grovel). He has more information on GPL than the average 4A owner that you are likely to come across.

GROM is a very cleaver idea. It is a memory mapped device that appears to take up only four addresses in the memory map. I believe this was done so that GROM0 to GROM2 which contain the monitor program, parts of the operating system and the TI Basic interpreter could be in the memory map but still leave pleanty of room for 48K of user RAM.

Just out of interest, the 4A can access up to eight GROMS at one time with each GROM containing 6K of information.

GRMWA is the GROM write address register located at 9C02. This lets you select where in the GROM you are going to write. One of your points was that you cannot achieve anything from writing to GROM or ROM, this is perfectly correct. The ED/AS manual is very unclear on this subject but from what I understand, this command can be used for GRAM on hardware like the GRAM Cracker. To give an address of where you wish to write to takes two byte transfers. The most significant byte of the address is sent first followed by the least significant. When this has been done the GRMWD or GRAM write data register at 9C00 can be written to. There is no way as you suggested that this command can be used to write data to the GROMs.

A similar principle applies to reading the GROM/GRAM. The most significant byte of the address has to be sent to GRMRA at 9802 followed by the least significant byte. Then the GRMRD register at 9800 can be read. To summerise this bit, GROM needs two bytes written so that it knows where to go. Then with GROM only the read statement should be done whereas with GRAM both the read and write commands can be operated.

Just out of interest I wrote a machine code program to place R13 at address E000. A basic program then read E000 and printed this on the screen. It printed zero which I thought was a mistake so I checked the code once more. I even set the values in E000 from basic at the start of the program and these were cleared by the code which suggests that R13 contains zero. This was tested on the Geneve so this may be why the manual does not tie up with my test results. This is worth testing yourself just to check. Anyway, if R13 contained 9C02 as suggested in the manual, the data found there will be the least significant byte of the last address used in writing to the GROM. This is almost random in itself as we have no way of monitoring what writes there and when. I do believe that in section 16.5.3 of the ED/AS manual that GRMWA should be replaced with GRMRA. I think its over to Mark on the other GPL issues. Thanks for the enquiry and I hope what is written here is of help to you.

DISK LIBRARY NEW ADDITIONS...
NOVEMBER 1993....

Sorry its a bit late (Ed..)

>BODENMILLER DEMO DISK. DOUBLE SIDED DISK ONLY. A very unusual demo disk of two programs from Bodenmiller Computers. (dated 1993).

>GOBLINS+ STAR MISSION. DOUBLE SIDED DISK ONLY. Both games are simple shoot outs, but Goblins has some very interesting use of sampled sound-speech without a synthesiser (the speech is Italian, as this came from Rome!). Games instructions are very brief and in English! No apparent scoring for either game, just keep going as long as you can!

>ODDMOD3 now has both the UK and USA versions of Household Money Management/Household Budget Management. Take your pick!

>XBMOD4 now has an ancient unreleased module added, DISK DUPLICATOR, a cut down version of disk manager BUT this one has the facility to copy USCD Pascal Disks, disk manager cannot do that!

>FUNLWEB STANDALONE, by special request, the simplified version, only TWO DISKS please, for a working copy of the enhanced VERSION 5 EDITOR plus the original formatter plus Disk Review and the loaders for machine code programs, together with dos for ONLY the editor and disk review. Nothing else to confuse you. But DONT complain about missing utilities or docs! You get only what is described here! 40 column version. Probably all that most owners will use! TWO DISKS.

>MINESWEEPER. An old program with varieties from TI Basic to PC+Windows, this one is in c99, and will run from XB. Several options, and quite a challenge!

>UTIL C is a new disk and at present JUST has a program to print labels for those rather solid 3.5" disks!

>MAZE MANIA is a nice disk with 13 mazes for you to solve on screen (some of them are bigger than the screen and scroll up and down!) or you can make it easy and print them out to do!

A complete listing of the disk library is available in the form of DV80 (text) files on FOUR SSSD disks, and a copy is yours if you send four disks plus return post and packing.
Library copying charges are one pound per side copied plus a flat one pound handling fee. You should send blank (or full!!) disks for the programs to be recorded on, or the library will supply for an additional 50p per disk.

STOP PRESS.

The TI user group U.K. passes on its sympathy to family and friends of Jim Peterson of Tigercub Software who passed away at home (Whitehall, Ohio) on the 12th of January. The funeral was held on the 17th of January. He will be greatly missed in the 4A community.

GAMES : by Steve Burns

Walk into any store that handles computer software and notice what actually fills the shelves, not rows and rows of data bases, word processors and spreadsheets, but endless shelves of GAMES! This is not necessarily an indication that computer users do not have a need for serious software, they do, but games are something that everyone can "use" and are also not "threatening" to the computer neophyte or non-user. They are an introduction to computing to many people, both young and old, and continue to be a source of entertainment and a challenge to anyone who enjoys them.

What bearing does this have on the TI "world"? In the past, games were an integral part of the TI software library, since it was the ORIGINAL home computer. Since then, the TI "world" has sought to prove its worth as a productive machine and to many, games have been all but forgotten. Some users are content to play an occasional game of Munchman or one of the other early games. This is not true for all though, and the TI is lacking in the area of new graphic-intensive games. Exceptions to the rule are Tetris, Rock Runner, and a few others. To interest new users we need more of these types of games. Many parents buy computers for their kids and we ALL know what the kids are interested in.

The NINTENDO Connection Can the TI compete with the Nintendo and other new game systems? I feel it can. We need to keep in mind what type of games are popular now. Baker Software has been selling a SUPER MARIO BROS. "clone" for the TI and although the reviews I have read are far less than stunning, the game still seems to be selling well. Tetris made it through the conversion a little better. My family actually prefers the TI version to the Nintendo game. Other Nintendo games that would convert well include LOOPZ and LOLO (if you do have a NES, I recommend you check these out). Both of these could possibly even be programmed in XB without a serious loss in play value although most conversions would require the talents of an assembly language programmer.

One point that should be noted here is the fact that many Nintendo games are very similar. Most have a character, usually with a weapon of some type that goes through an "obstacle course" of some type. Many of the same assembly routines could be used in a clone of Nintendos' Mutant Ninja Turtles and the game Contra. These are not necessarily the types of games I would like to see converted, just the ones that seem to be the most popular (I made my suggestions earlier in the column). Don't get me wrong. Copying software from other machines is not vital to the survival of the TI, but it is a practice that has gone on since the beginning of computers. Most people who see something on another machine that they like are willing to purchase a version for their own system.

Remember all the Frogger clones from years past? Original ideas are still the most welcome addition to any computers "arsenal" of software. Unique games like DIABLO (there are clones of this for both Nintendo and Amiga) and Tunnels of Doom (the basis for most more than a few of todays popular games), will always find a market. The main things to keep in mind are bold, modern looking graphics, popular character types, and good play value. Coupled with our advantages such as speech, a full keyboard, and the new Asgard Mouse, the TI could easily keep up with the new offerings.

The Undocumented Features of 'G'

by Mark Schafer

G is a graphic programming language written in machine language for the TI. It comes on disk with on disk documentation. However, I noticed that there is more to G than the documentation shows. What I did was I looked at the program in a sector editor, and on the third and fourth sectors is a list of all the keywords in G along with the syntactical symbols. Some of these were not described in the documentation. I set out to find out what they do.

First, a list is in order; these are the commands that I did not find in the docs: GET, PUT, TEST, VPEEK, VPOKE, PEEK, POKE, VWTR, SARRAY, and LARRAY. Starting with the most obvious, PEEK and POKE do just what CALL PEEK and CALL LOAD (with memory) do in BASIC. They read and write values in memory respectively. Their syntaxes are as follows:

```
PEEK a var [var, var...]
POKE a b [c d e...]
```

Lower case letters can be any numeric expression; var is any G variable. In other words, both commands can read/write in consecutive locations.

If you want to peek or poke into an address higher than 32767, you must subtract 65536 from it, just as you do in BASIC. And as far as I can tell, all values are in decimal.

VPEEK and VPOKE do the same thing to VDP RAM. This enables you to read and write to the screen. Sure, you can write to it now using the documented commands, but only VPEEK can read it.

VWTR is the next most obvious for you assembly programmers since it happens to match the assembly mnemonic for VDP Write To Register. However, this VWTR takes only one argument and writes it to VDP register 0, so it's not very useful.

And for anybody familiar with IBM BASIC, there's GET and PUT. No, they don't read and write files; they read and write graphics, which is their other use in IBMland. Their syntaxes are as follows:

```
GET a b c d
PUT a b
```

GET's syntax matches that of CLEAR and INVERT. That is a,b is the upper left corner of a box, and c,d is the lower right corner. This area of the screen is buffered. When the PUT command is used, the last graphic that was buffered with the GET command is put on the screen with its upper left corner at the location given (a,b). Limitations apply, however. These commands will work perfectly if a, b, c, and d are all divisible by 8. If they are not, you may not get the exact area you specified.

SARRAY and LARRAY may be unique in the world of programming. The S and L stand for save and load. So their argument is a file name. SARRAY allows you to save the array (there's only one, remember) to disk. How much does it save? All 500 values. Except the zero element (@(0)). The file takes up 15 sectors on the disk. LARRAY loads in a previously saved array into the array in the program. These can be useful for programs that have a large number of array constants. There are also symbols left out of the docs. They are the percent sign, and tilde (~)

The percent sign is highly useful. It represents the CHR$ function in BASIC. It will return whatever character corresponds to the ASCII value of the numeric expression that immediately follows it. An example is in order I think: PRINT 0 0 %80 will put a P in the upper left corner of the screen since P is ASCII 80. But of course, % is more useful when a variable follows it. This allows you to echo user input to the screen. The tilde (~) is just a variable. You can set it equal to something, print out its value, use it in a formula, whatever you do with a variable. However, it shares a feature in common with the array: its value does not change from one run to the next. Therefore, it contains garbage if not set. The other variables are initialized to zero.

There's still another interesting point to cover. Notice that G prints text. This is unexpected because G uses bit-mapped mode which doesn't have text. So the G program must be converting the graphic definition of the characters to bit-mapped mode so they can be displayed. So where is it storing the character pattern table? In low memory expansion starting at >2000.

This is hex 2000, so in decimal that's 8192. This is where the space is located. The exclamation point follows that starting at 8200, and so on. They're in the same format as they are in BASIC. But you have to remember, if you play with this area of memory, you must use decimal byte values. So the range is 0-255. If you POKE into this area of memory, you can change the look of the characters. And when you do, not only will it effect your program, but the editor as well. Any font change will be permanent until you change it again. The editor is in text mode, so it will only show the first six columns of each character.

Something you can do for fun is subtract the values in this area of memory from 255, and you will invert the character set. Or you can just do parts of it. This will allow you to put inverted text on the screen easily. Then you can change it back to regular text because unlike BASIC, changing a character's definition will only effect future displays; it will not change the looks of the ones already on the screen.

To change the definition of character X, use the following POKE statement:

```
POKE X*8+7936 a b c d e f g h
```

If you already know what X is, change the formula to the appropriate constan X is in the range 32-127.

While I'm on the subject of memory, let me just mention that the zero element of the array is stored at 9184. The 1 element is after that, and so on. Each element takes up two bytes. The value of the tilde is stored at 9180. So another way to refer to it is @(-2). G does not check for negative values of the array, so if you go low enough, you can change the character set by setting negative values of the array. When you've got PEEK and POKE, you've got power. You can use these to access the joysticks, scan other keyboards, and even pronounce vocabulary words in the speech synthesizer! Watch out, though, if you write a program that accesses the joysticks. G is constantly checking to see if the user hits FCTN 9. When it does that, it does not change the keyboard to be scanned, so it won't work if you've changed it. You could immediately change the keyboard back to zero when you're through scanning it, or type Z to escape while scanning joystick 1 or N while scanning joystick 2.

The graphics programming language G is available from the disk library. Programs in G /and any other language/ are welcome!.

FUNLWEB VN 5.0 EDITOR

The pages which follow comprise the six help screens I have created for myself for use with Version 5 Baseline Editor, and include ALL baseline functions and commands. As well as acting as a good quick reference, they also summarise what you are missing if you have not yet updated! JUST the new Editor files occupy two SSSD disks, but you can of course select the bits you want for your working disk.

The bug which prevented the usual simple exit, mentioned last issue, has now been removed.

Also a bug in the support file XB4TH which was introduced with Vn 4.40 has now been removed if anyone uses that small file. Should be easy to add it onto any other disk you ask for.

An especially nice feature of the new Funlweb is the ability to pick up line numbers for command line transactions- for example, supposing a wanted to copy from line 120 to line 130 to commence after line 170...
Go to the command line, and choose COPY, then -still in command mode! - scroll the text so that line 120 is at the top of the screen and type CTRL M then scroll so that line 130 is at screen top and type CTRL M then scroll on to line 170 is at the top and press CTRL M then ENTER. Then a simple CTRL 2 places you back where you were before you started all this. Effectively we now have the ability to pick up text and move it elsewhere, very handy indeed. I also enjoy using the FREEZE control, which despite not windowing with the rest of the screen (it is frozen!) has its uses.

Here is the summary, in 40 column mode 'cos that is how the HELP screens come up....

FUNLWEB VERSION 5 HELP DATA
Relates to "baseline" editor. Extended Editor has even more functions!

FROM THE COMMAND LINE:
CR/carriage return- if rqd use CTRL 8
DISK DIRECTORY- SD then number
DISK DIR PRINT NAME if different to PF name- set with DP then name
E for EOF: use with Show only
-not available with PF or LF
-use a high number eg 999
EDIT mode...CTRL 2 or ENTER blank line returns to line left from
...CTRL 0 (for origin) returns to place FS or RS started from AFTER you use CTRL 2 to return to Editor if approp

FIND STRING- FS:
surround word with 2 chars same eg /at/ or =at= or ~at~ etc
Wildcard available-see W
N /word/..find N+1 time word occurs
M N /word/..find between col M and N
K M N /word/..find n+1 time word occurs between col M and N
afterwards-return to start location with CTRL 0 (for Origin) from Editor.

```
GOTO LINE...  just enter line number
GOTO TOP OF PAGE..CTRL 1
HELP..H..page through with Q/A
        ..exit with CTRL C

LINE NUMBER: enter top of screen line
    number with CTRL M
MARK start of line N: nnn MK
PAGE (Pa) if required-use CTRL 9

PRINT FILE: E(end of file) not available
   Precede printer name with switches:
    A...Append (add to disk file)
    F...DF80 records (eg to disk)
    C...remove control chars
    L...col 1 to 73 + line numbers
    P...start with printer set up codes
    Q...end with printer reset codes
    m n..from line m to line n
      eg PF:   P C Q 4 30 PIO

QUIT Q or QUICK QUIT with QQ

REPLACE STRING-RS:
   use /w/r/ or =w=r= etc etc
   Wildcard available- see W
   N /word/..find N+1 time word occurs
   M N /word/..find between col M and N
   K M N /word/..find n+1 time word
        occurs between col M and N
    afterwards-return to start location
    with CTRL 0 (from Editor)
   For PA use CTRL 9, for CR use CTRL 8

RETURN TO EDITOR: where left: CTRL 2
            or ENTER with line blank.

SCROLL SCREEN UP/DOWN...FCTN or CTRL E/X

SHOW DIRECTORY- Bytes is room in memory
 P and U protect / unprotect
 T passes file name to LT postbox
 SPACE passes file name to LF postbox
 CTRL P to PRINT directory (see DIR)
 D to delete file.
 V to view DV80 file on screen
 Q and A page up and down
 ENTER to exit.  FCTN 8(Redo)=next disk
 0 returns postboxes to the names they
  had on entry to SD.

SHOW LINE..just enter line number nnn or
use S then number

TAB...3 sets available.
   Left Insert Tab and Right markers.
WILDCARD CHAR-used in RS and FS eg FS
/b*t/ finds bat or bit.
WILDCARD CHAR CHANGE: WC then char
   to use as wildcard eg * or ?
```

WHEN IN EDIT MODE:

```
Break Line.........CTRL B or FCTN 2
Change to LCase...CTRL . (CTRL >)
Change to CAPS....CTRL ;
COMMAND LINE......FCTN 9 or CTRL C
Copy line above...CTRL 5
CURSOR...move with:
    CTRL or FCTN ESDX
    to EOL...........CTRL Z
    to start of line CTRL V
    to MARK.........  FCTN =
    to START OF FILE-CTRL H
DELETE CHAR...........FCTN 1
DELETE LINE...........FCTN 3
Duplicate Line above..CTRL 5
END OF FILE,top of page:CTRL J
EOL..move cursor to...CTRL Z
   ..delete to end of.CTRL K
FREEZE screen cursor to bottom of
  screen (toggle).... CTRL F
GOTO MARK............FCTN =
   -return to position FCTN = was
    used with CTRL 0 (for Origin)
GOTO TOP OF DISPLAYED PAGE..CTRL L
HOME..go to top of file..CTRL H
      go to top of PAGE..CTRL L
INSERT CHAR....FCTN 2 or CTRL G
INSERT LINE....CTRL N or FCTN 8
LINE -BREAK....CTRL B or FCTN 2
  -Rejoin (wrap off) CTRL R
  -copies line below upwards.
LINE DELETE.........FCTN 3
LINE NUMBERS at edge-toggle FCTN 0
Margin Release.....CTRL Y
MARK line.... FCTN ; (goto is FCTN =)
OOPS..........CTRL 1
PAGE DOWN.....FCTN 4 or CTRL A
PAGE UP.......FCTN 6 or CTRL Q
REFORMAT......CTRL 2
SCROLL DOWN...FCTN 4 or CTRL A
SCROLL UP.....FCTN 6 or CTRL Q
TAB...........FCTN 7 or CTRL I
TAB TO NEXT WORD.........CTRL W
TAB BACKWARDS............CTRL T
TOP OF FILE goto.........  CTRL H
TOP OF PAGE DISPLAYED.....CTRL L
TOP OF PAGE: TOP of file: CTRL H
TOP OF PAGE: END of file: CTRL J
WORD WRAP TOGGLE.........CTRL 0 (zero)
```

(Document continued on page 7)

HARDWARE RELEASE.

Bud Mills Services of 166 Dartmouth Drive, Toledo Ohio 43614-2911 now offers the following for sale:

1: True AT keyboard for the 99/4A which will interface directly with the 4A and allows the use of both keyboards $65.

2: Upgrade 64K system ROM for the 80 col cards which fixes all knows bugs.

3: RXB which is a vasy improvement on XB. This is available as a cartridge for $60.

TIPPSY

Hi there again to another page of TIPs and bits. Last quarter I made a request for **your** Tips. I have received some communication so here goes.

From group member **Eamon DORAN of Colchester**

Here is my tip for your Tippsy column. I often play a game published in the Parco Magazine Vol 2 Issue 1, 1985 called TURN OF THE CARD.
To speed up initial card display add a zero to the end of data in line 690.
To stop entering more than you have to bet change line 1200 to E2=E2/E2*T.
Omit lines 1210, 1220, 1230, 1240, 1250. This will stop display from corrupting and will leave you with the same hand being dealt.

LOGO undocumented stick routines

If you want to use your TI LOGO 1 or 2 with joysticks here is how to do it. The values for the sticks are as follows:-

```
2     6     10          0010   0110   1010
1     5     9           0001   0101   1001
0     4     8           0000   0100   1000
      FIG 1                    FIG 2
```

Moving from left to right in FIG 1, in each row you will notice that each digit is four more than one on the left. If converted to binary FIG 2, You will note that all columns are starting with the first two digits are the same. Ie 00 first col, 01 second col, 10 third col. Also if you look at the last digit in the rows, they all correlate. So what we have is a distinctive coordinate system with real meaning.

So let X and Y be your operators as in basic, we can then use the following

```
MAKE "Y (JOY1)/4
MAKE "X (JOY1) - 4 * :Y
```

Now all you do is is use the following in a procedure say STICK :S

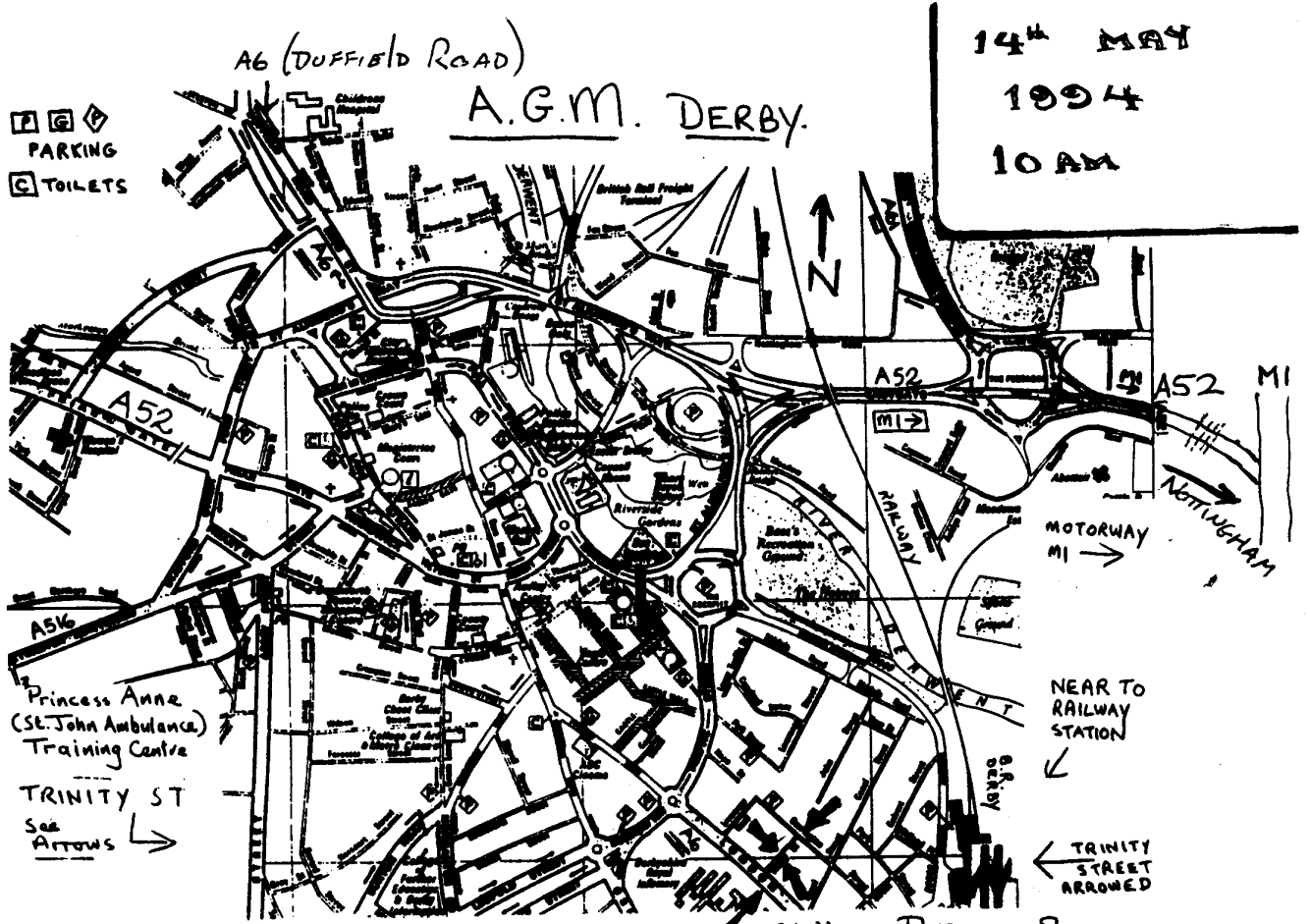
```
IF :X = 6 THEN FASTER
IF :X = 4 THEN SLOWER
```

FASTER would be a procedure such as IF SPEED < 100 THEN SS SPEED + 5

MEMORY FULL!!!! Fctn Quit.....

A.G.M. DERBY.

14th MAY
1994
10 AM



MAIN ROADS : M1
A52
A6
A516

TRINITY STREET, JUST OFF LONDON ROAD (A6)

EW LINES (BIRMINGHAM-LINCOLN VIA (NOTTINGHAM & NEWARK)
NS LINE (UP PAST MANCHESTER)