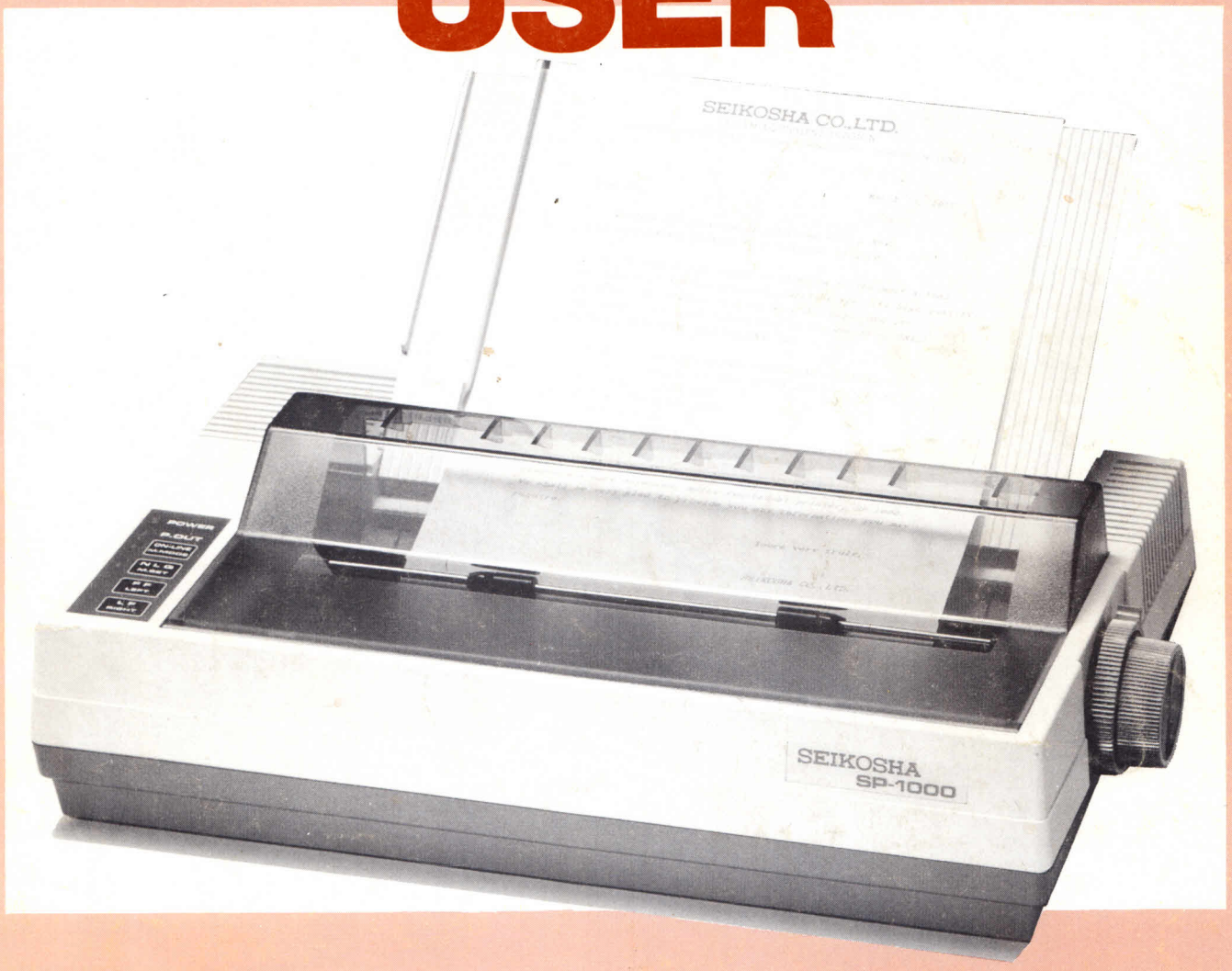
THE AMSTRAD USER

Issue No. 8 $3.00 September 1985



- *EIGHT PAGE SOFTWARE REVIEW SUPPLEMENT*
- *GRAPH PLOTTING & HOME BUDGET PROGRAMS*
- *SPEECH SYNTHESISER & PRINTER REVIEW*
- *USER GROUP INFORMATION*

FOR THE NOVICE & EXPERIENCED USER

Registered by Australia Post – Publication No VBP7017

Have you *really* discovered your Amstrad yet?

Want to get down to the nitty-gritty and start to see what you can *really* do with your Amstrad computer?

The following items are now in stock:

NEW Joysticks	JY-2
Blank Discs	CF-2
Dot Matrix Printer: 100 cps	SP-1000
Colour Printer	GP-700
Speech Synthesiser	SSA-1
Light Pen	LP-1
RS232 Interface (available soon)	
Keyboard Covers for CPC464	
Starter Kits	

Available from your Amstrad dealer

THE AMSTRAD USER

Issue No. 8 September 1985

CONTENTS

For Tape Subscribers, the programs/routines can be found at these approximate counter readings:
Side 1 - Grafload: 2, Grafplot: 14, Noteplay: 58, Budget: 69, T.Bas: 110 Side 2 - Mower: 2

All enquiries and contacts concerning this Publication should be made to The Amstrad User, Shop 2, 33 The Centreway, Blackburn Road, Mt. Waverley, Victoria 3149, Australia. [Telephone: (03) 232 7055].

The Amstrad User is published each month by Strategy Publications.

The single copy price of $3.00 is the recommended retail price only. The subscription rate (for Australia only) is $30.00 for 12 issues of the magazine only, or $70.00 for 12 issues of the magazine plus tape containing all programs appearing in that issue. Postage is included in the above prices. Overseas prices available upon application.

Please note that whilst every effort is made to ensure the accuracy of all features and listings herein, we cannot accept any liability whatsoever for any mistakes or misprints.

Contributions are welcome from readers or other interested parties. In most circumstances the following payments will apply to published material: Letters-$5.00, Cartoons-$5.00 and a rate of $10.00 per page for programs, articles etc.

Contributions will not be returned unless specifically requested coupled with suitable stamped and addressed padded bag (for tapes) or envelope.

THE AMSTRAD USER

G'day

Hands up all those people who failed to notice the duplicated lines 2530 to 3060 on Page 23 last month. Not too many I hope!

By way of amends, this month's issue has been increased to forty pages - an additional eight pages. The 25% increase in size is due to the Software Review Supplement which contains opinions on eight software products. Our software review panel had to work hard and fast to meet the deadline of this issue and deserve a vote of thanks from me. I will try to give you more notice next time. You will notice that each piece of software reviewed has been allocated a TAU Index (The Amstrad User Index) which can be taken as an overall "acceptability percentage". The idea is that you can remove the supplement if you wish, and file together with future supplements, building up a record including ratings of the software available for the Amstrads.

You may have wondered what I was doing by putting a picture of a Seikosha printer on the front of the magazine. Well, further examination of this issue will reveal a review on this printer which has been released by AWA for use with your machine. Things are now beginning to pick up on the Amstrad peripheral front with a number of new pieces of equipment becoming available. You'll also find a review on the new speech synthesiser.

The competition has now closed and the shelf above my desk is creaking under the weight of entries. Poor old Postie had to work overtime on the last two days before the closing date as many people had left it to the last minute to submit their entries. I must admit that I was surprised at the number of different applications people have chosen to run on their Amstrad - proving that it really is much more than a good games machine. The results will be published in the next issue.

See you next month.

Ed.

Letters

I am sure that there are many people who have managed to reach a score that was a personal best, yet find certain games so easy that it is a bore. Then there are others who spend countless hours early in the morning tearing their hair out in complete frustration because they can't get passed a certain stage in a game.

The point I am making is why don't you devote one page in The Amstrad User for top scores and hints on play. I am sure that this would be beneficial to many.

Just to get the ball rolling I scored 14,685 on Electro Freddy. My hints for play are:

1. On the first two levels, get as many points as possible by eating pies and bumping your uncle.
2. On the third level, only bump your uncle when he is near and try to eat all the pies.
3. On the fourth level the game starts to get difficult - only bump your uncle when he is directly above or below, **don't chase him!** Only eat the pies when you are in the top right-hand corner.
4. Good luck!

Nathan Yim,Victoria

We have received a number of requests along the same lines as above. Cognisant of these demands we will include an 'Amstrad Achievers' column next month if enough gamesters respond. A form appears in a later page of this issue to record your score. To avoid cheating, the score must be witnessed or, even better, shown on a photograph. And you will need to be quick to make the next issue - we require any scores by the end of the first week of the month prior to publication to ensure inclusion. (If Nathan would like to supply his full address, we can organise a payment of $5 for this letter).

Since my last letter I am continuing to find bugs in the Locomotive Basic. This time it is the CHAIN command which causes the computer to get hung up if an array is DIMensioned and ERASEd. Try SAVing the following two trivial programs, the first of which loads the second.

```
PROG1.BAS
10  z=n1(6)
20  CHAIN"!PROG2.BAS"

PROG2.BAS
10  DIM X(200)
20  PRINT"start"
30  ERASE X
40  PRINT"finish"
```

RUN" the first and leave the play key depressed so the second can be CHAINed. The computer will get hung up at line 30 in PROG2 and must be reset. If the following line is added to PROG1:

```
15  ERASE n1
```

then PROG2 will execute to completion. However in a more complex program PROG2 will eventually get hung up again. Does anyone have a solution?

Is there a list of bugs in Locomotive Basic available from AWA or Amsoft that we can get access to without having to spend hours finding out the hard way? Could I suggest that this magazine devote a page to such bugs and their solutions as a service to readers. This should be reprinted every issue and added to as more problems are encountered and hopefully solved.

And what about the obvious bug of being unable to merge Basic files on disc?

B.M. Chapman, Valley Hts.,NSW

To our knowledge, no list of bugs in Locomotive Basic exists in Australia but we are taking up the point with AWA. Your request for a page to be devoted to informing readers of bugs past and present would be an overkill at the moment as we are happy to publish letters as and when the problems occur. Nevertheless, we will continue to monitor the situation.

I have just purchased the Amsword Word Processing program. As there are a few commands that I do not understand and they are not covered in the instruction book, I wondered if you could help me with them. The Tasprint options are:

(V) Lectura Light
(W) Median
(X) Compacta
(Y) Data run
(Z) Palace Script

In the March edition of The Amstrad User there is a review on the DDI-1. It states that a second or backup drive can be a 5.25" and with CP/M this makes it compatible with most other micro computers as so many of them use the larger disc. I rang AWA in Sydney and was told that a larger drive could not be used unless costly modifications were carried out. Could you clarify this for me? C.W. Hall, Newcastle, NSW

Amsword will print in a type style or font which is dictated by the printer being used. To print in other styles, it is necessary to have an extra piece of software, like Tasprint. The 'commands' you mention are the different styles which can be obtained using a dot matrix printer and Tasprint. (See example on this page).

On your other point, many users have managed, quite successfully, to use a 5.25 inch disc as a second disc drive. We are aware of a third party who intends releasing this type of drive for the Amstrad, and will notify readers when this happens.

Amstrad Cordon Bleu

Somewhere between my submission of the program "Recipe" and its publication in the August issue of The Amstrad User, someone has done a RENUM with the result that the accompanying text does not match the program line numbers. The line numbers for the program modules should be:

10 to 60	**Preface**
80 to 600	**Subroutines Area**
610 to 2000	**Main Prog. Area**
700	Command 1: Add
1040	Command 2: Search
1370	Command 3: Display
1850	Command 4: Modify
2030 to 2370	**Definition of variables**
2380 to 2440	**Initialisation of all indexes**

Section 4 of the STARTING UP paragraph needs to be altered to read "GOTO 2390" and section 6 should read "REM-out lines 2390 to 2430".

Sorry about the confusion, but I take it as a lesson learned (hopefully other program submitters will take note also!) that software intended for manual user input needs to be arranged so that auto-line numbering can be used.

By the way, we recently had the fortunate experience of being asked the recipe of a certain dish during a dinner party. Boy, did we impress 'em! Good old Ami.

A.M. Urankar, Eltham, Vic

*Yes, someone 'very kindly' did a RENUM of the program before it was printed for the magazine, but **after** the narrative had been typeset! Mr. Urankar is not the only one who has learnt a lesson. He makes the valid point that programs submitted for publication should be RENUMbered to take advantage of the AUTO function.*

AMSTRAD ACHIEVERS

Get your name in our "HALL OF FAME"

In the next month or so we will publish proven high scores for games or adventures which have been achieved on an Amtrad computer. Register your name and score on the form below, and, if possible, send a photograph to put doubt out of everyone's mind!

Name.....

Address.....

.....

Telephone Number.....

Game..... Score.....

Achieved (date) .....Game lasted (mins.secs).....

Signed.....

THIS NEXT PART MUST BE COMPLETED

Witness' Name .....

Address .....

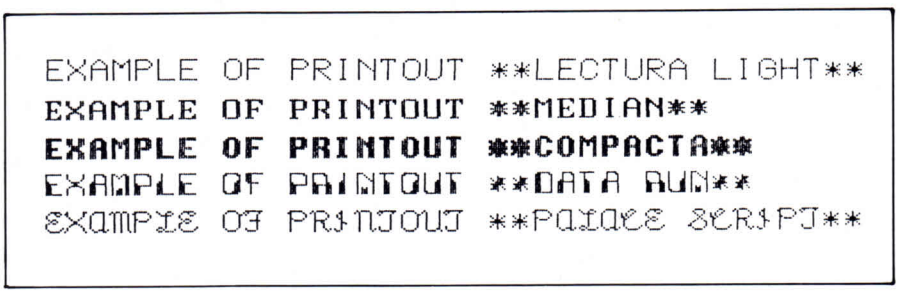
Telephone Number .....

Occupation .....

I confirm that the above claimed score is accurate and genuine

Signed .....

Post this form along with your tips for playing the game to:
Amstrad Achievers, The Amstrad User, Shop 2, 33 The Centreway,
Blackburn Road, Mt. Waverley, Victoria 3149.



Graph Plotter

by Peter Campbell

FUN MATHEMATICS

If maths was not your forte at school, you may be inclined to mutter somewhat unkindly about that heading and who is to blame you. Nonetheless, fun can be had with mathematics. That is the theme of the book, "Fun Mathematics on your Microcomputer", by Czes Kosniowski (Cambridge University Press, Cambridge, 1983, ISBN 0 521 27451 6). I bought the book a while back and while delving into its pages came upon the author's program for plotting graphs.

To quote the author: "The BASIC on computers varies from one machine to another and writing a general program is quite hard. When it comes to a graphics program the situation is almost impossible." He therefore presents three versions. One for computers which understand an implementation of Microsoft BASIC (with hints for conversion), one for the Vic 20 and one for the Sinclair Spectrum.

"Well", thought I, "I can get that up and running on my Amstrad" and embarked on the project. It did not take long to realise that the program was, of necessity, written to suit the lowest common denominator (and very common it seems to be too!) A few pages on and he was at it again, writing programs to plot polar graphs.

It soon occurred to me that one program could be written to plot both types of programs, using the CPC464's superior graphics and BASIC. From that moment on the program took on a life of its own, but I must thank Kosniowski for the inspiration.

DESCARTES VS THE POLE

No it's not the bill for World Championship Wrestling! There are two ways of plotting a graph. The first, attributed to Descartes, uses a horizontal and a vertical axis. Any point on a flat surface (plane) can be described in terms of measurements along these axes with just a pair of numbers (x,y), x is measured from left to right along the horizontal axis and y is measured vertically upwards. Negative numbers are measured in the opposite direction. The pair (x,y) are called co-ordinates, or cartesian co-ordinates.

The second method of locating a point in our plane is by means of polar co-ordinates. Here we have only one axis, with a point on it called the pole. Our point is now represented by a pair of numbers (r,z), where r is the distance to the pole and z is the angle, measured anticlockwise, between the axis and a line from the pole to this point. Confused? Good, because there is a little more to come when we get to plotting our graphs.

FUNCTIONS

I certainly hope it does! Actually the word, function, is used to describe expressions, such as y=x+1, y=sin(x), and r=sin(z), and we say that y is a function of x. In Locomotive BASIC there is a method of DEfining a FuNction for later use in a program.

A function which involves polar co-ordinates is called a polar function (and you thought it was a mid-winter celebration in Antarctica!)

GRAPHS

The interesting part of functions is not all the boring theory, which I have kept as short as possible, but the graphs that we can draw from them. A simple way to see the difference between cartesian and polar co-ordinates is to plot the graphs of the similar functions, x=1 and r=1. The first produces a straight line parallel to the x axis, but one unit (measured on the y axis) above it. The axis itself is x=0. Ah, but r=1 produces a circle!

To add even further convolutions to our polar graphs, we can multiply the angle z by a factor, say d, and I have made provision in the program to do just that.

A couple of things that microcomputers do not like when it comes to calculating the value of functions are division by zero and very large numbers. In the program I have tried to circumvent Arnold's protests by the use of a number of statements to which the program is diverted by ON ERROR GOTO. If you have the misfortune to discover a circumstance I missed, add another suitable statement after line 2350 and let us all know what was required, please!

There is one "bug" in the program. In graphs of functions like 1/x^2 the graph disappears into the wild blue yonder and then re-emerges almost directly opposite. What is happening, of course, is that division by zero is approached, giving the function a value too large to be plotted. After the point of division by zero is passed, the value of the expression, whilst still large, becomes opposite in sign. The vertical line at the point of division by zero, which the graph approaches but never touches within a finite distance, is called an asymptote and Arnold cheerfully draws it in, if you use the D option. In any one graph, it is quite easy to write a statement to prevent this. For example:

```
175 IF abs(v)>=250 THEN PLOT
```

u,v,1:goto 200

will prevent it happening in equation 1. However, the dysfunctional consequence (I spent years studying the jargon - I've got to use it on someone!) is that other graphs will be interfered with. If anyone comes up with a general expression that appears to work in all cases, please let me know! Of course the "if" statement in line 175 could be altered so that it only applies to equation 1, but what happens then if I make equation 5: 1/(x^3-1)?

HOW IT WORKS

Listing 1 is a loader screen, which not only gives you something to look at while the main program (listing 2) is loading, but also shows off just a couple of the CPC464's graphic features. These are the ability to change the colours on the screen and the ability to mix text and graphics by printing text at the graphic cursor position.

LINES 250-310: Initialise the screen setting all four inks to black and defining the windows used.

LINES 350-390: Set the transparent mode and origin for the text "plotting".

LINES 40-170: Complete the screen and turn off the transparent mode.

LINE 210: Loads and runs the main program.

Main Program

LINES 2310-2340: Error handling.

LINE 60: Call &BB4E initialises the text VDU in the same way as a "cold start". This clears the loader screen when first encountered and the graph drawing screen when you elect to have "Another Go" (line 1590 on). Call &BB03 resets and clears the keyboard buffer.

LINES 1700-1730: Initialise the first screen, setting paper and pen to blue and assigning values to key variables.

LINES 1770-1960: The first screen displays a brief explanation of the program and invites the user to choose mode and type of graph. Note line 1960, which ensures that the keyboard buffer is clear before proceeding, could be replaced with call &BB03.

LINE 80: Branches to the information screen for the chosen type of graph.

LINES 590-770: The second screen lists the 8 graphs which can be plotted. If amending the program to include your own choice of functions, substitute your choice for one of the 8.

LINES 780-790: These subroutines enable the user to select one of the functions, assess the appropriate scale for the screen plot. plot the graph and finally invite the user to have another go.

LINES 830-970: The third screen performs the equivalent functions for polar graphs. This time there are 7 choices.

LINES 980-990: These subroutines enable the user to make a choice not only of function, but also the five constants a,b,c,d and e, which are added into and/or multiplied by the elements of the various functions and, as mentioned, used to vary the angle z. By varying these, equation 2's simple circle becomes a familiar logo. (Try a=0, b=0, c=0, d=1 and e=3). They also assess the correct scale, plot the graph and invite the user to continue.

Other lines of interest are lines 1080 and 1220 which select the appropriate definition for the function from lines 1360-1570. The latter are where you make your other sustitution if using the program to plot a graph of your own choice.

With the polar graphs you have a choice of plotting or drawing (lines 2220 on), but for the other graphs you have a third choice, that of shading in the area below the graph (lines 2090 on).

VARIABLES USED

Loader Program

i: Control variable in the for.... next loop.

title: String variable from which the title is sliced a letter at a time using p. Note that both are predefined as string variables in line 250.

Main Program

r,z,x,xx,y,yy,u,v: The first six are used in the functions from which the graphs are plotted, whilst u and v are used for the points to be plotted.

a,b: Used firstly as the lower and upper limits of the range which is to be plotted for ordinary graphs. For polar graphs a,b,c,d and e are used as explained earlier.

m: Calculated by sampling the function at various points. Used to scale the graph to the screen.

st: Step variable in for..... next loops.

dgs: Number of degrees being plotted in polar graphs.

p$: Required option from (p)lot, (d)raw or s(hade).

n,n$: Equation number.

keystr$: The choice of equation numbers. (An alternative approach to coding the choices would have been ON n GOSUB).

gsb: Used for the selection of graph type.

HINTS:

The program will plot any range of values of x in the ordinary graphs. Try -50 to 50, -10 to 10, -2 to 2 and -1 to 1. If any of these ranges give a hint of an interesting area of the graph, zero in on it, by choosing the appropriate range.

With the polar graphs an almost infinite variety of patterns can be created. Try the so-called standard plot. Then try a=1, b=1, c=1, d=1 and e=1. (Why doesn't that give a graph in the case of function 7? If you apply a little algebra, you will see that the variable, z, is eliminated, leaving nothing to graph!). A third choice is a=2, b=2, c=3, d=4 and e=1 and finally try a=2, b=2, c=3.1, d=3 and e=3. By the time you have tried all of those, you will have an idea of the effect of changing these numbers. Then do some experimenting.

Use AUTO when typing the program in. To make it more compact, type <ENTER> instead of the REM and ' lines. These lines will then be omitted, eliminating more than 50 lines. Also omit the spaces used to indent wrap-around lines.

If the graph seems to get "stuck", a touch of the ESC key will jump you forward to the "Another Go?" option.

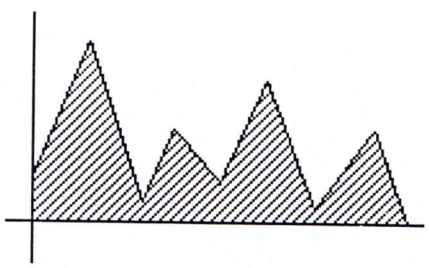
Have fun!

```
10 REM *********** GRAFLOAD ***********
20 '
30 GOSUB 250:GOSUB 350
40 FOR i=1 TO 360 STEP 5
50 MOVE 130,110
60 DRAW 180+180*SIN(i),100+100*COS(i),3
70 PLOT 180+180*SIN(i),105+105*COS(i),0
80 PRINT CHR$(143);:NEXT i
90 FOR i=420 TO 60 STEP-20
100 p=MID$(title,i/20,1)
110 PLOT 180+180*SIN(i),
        105+105*COS(i),1
120 IF NOT p="" THEN PRINT p;
130 NEXT i:TAGOFF:PEN 1:LOCATE 12,12
140 PRINT CHR$(164);CHR$(128);
150 PRINT "The Amstrad User"
160 LOCATE 16,13:PRINT"1985"
170 PRINT CHR$(22);CHR$(0);
180 '
190 REM ******** Load Grafplot ********
200 '
210 RUN"!grafplot"
220 '
230 REM  ********* Initialise *********
240 '
250 DEFINT i:DEFSTR p,t:MODE 1
260 INK 0,0:INK 1,0:INK 2,0:INK 3,0
270 PEN#0,0:PAPER#0,1:CLS#0:BORDER 5
280 WINDOW#1,2,39,2,24:PAPER#1,3:CLS#1
290 WINDOW#2,5,36,4,22:PAPER#2,0:CLS#2
300 WINDOW#3,6,35,5,21:PAPER#3,2:CLS#3
310 RETURN
320 '
330 REM  ******* Prepare Screen *******
340 '
350 INK 1,24:INK 2,14:INK 3,7
360 PRINT CHR$(22);CHR$(1);
370 DEG:ORIGIN 130,105:TAG
380 title="   * RETTOLP * GRAPH      "
390 RETURN
```

```
10 '************************************
20 '**** G R A P H   P L O T T E R ****
30 '************************************
40 '
50 ON ERROR GOTO 2310
60 CALL &BB4E:CALL &BB03
70 GOSUB 1700:GOSUB 1770
80 IF gsb=1 THEN GOSUB 830
            ELSE GOSUB 590
90 '
100 REM ********* Plot  Graph *********
110 '
120 MODE mde:y=FNa(x):x=a-st
130 PLOT 630*(x-a)/(b-a),
         195+195*FNa(x)/m
140 FOR x=a TO b STEP st
150 IF INKEY(66)=0 THEN x=b:GOTO 1630
160 y=FNa(x)
170 u=630*(x-a)/(b-a):v=195+195*y/m
180 IF p$="p" THEN PLOT u,v,1
              ELSE DRAW u,v,1
190 IF p$="s" THEN DRAW u,-v,1
200 NEXT x:RETURN
210 '
220 REM  ****** Plot Polar Graph ******
230 '
240 r=ABS(FNa(0)):MODE mde
250 st=9/(a+b+c+d+e)
260 IF st>1 THEN st=1
270 IF st<0.25 THEN st=0.25
280 dgs=180/st
290 IF dgs<390 THEN dgs=390
300 IF NOT INT(a)+INT(b)+INT(c)+INT(d)+
    INT(e)=a+b+c+d+e THEN dgs=3600:
    st=2*st:m=1.25*m
310 PLOT xx+yy*COS(0)*r/m,
         yy+yy*SIN(0)*r/m
320 FOR z=0 TO dgs STEP st
330 IF INKEY(66)=0 THEN z=dgs:GOTO 1630
340 r=ABS(FNa(z))
350 u=xx+yy*COS(d*z)*r/m
360 v=yy+yy*SIN(e*z)*r/m
370 IF p$="d" THEN DRAW u,v,1
              ELSE PLOT u,v,1
380 NEXT z:RETURN
390 '
400 REM  ***** Assess Graph Scale *****
410 '
420 y=FNa(x):st=(b-a)/500:RAD
430 IF b-a>10 THEN st=(b-a)/1000
440 IF b-a<2  THEN st=(b-a)/5000
450 FOR x=a TO b STEP st*5
460 m=MAX(m,y,ABS(FNa(x+2.5*st)))
470 IF m>30000 THEN m=75
480 NEXT x:RETURN
```

```
490 '"""
500 REM  ***** Assess Polar Scale *****
510 '
520 DEG:r=ABS(FNa(z))
530 FOR z=0 TO 360 STEP 5
540 m=MAX(m,ABS(FNa(z)),FNa(z+3))
550 NEXT z:RETURN
560 '
570 REM ******* Graph Functions *******
580 '
590 INK 1,1:CLS:LOCATE 21,2
600 PRINT"G R A P H   P L O T T I N G"
610 WINDOW#0,4,78,2,24
620 LOCATE 1,4
630 PRINT"1. y=1/(x^2-1)"
640 LOCATE 40,4
650 PRINT"2. y=x*sin(1/(x^3))"
660 LOCATE 1,6
670 PRINT"3. y=sqr(x^2+2)
680 LOCATE 40,6
690 PRINT"4. y=cos(x+exp(-x/5)"
700 LOCATE 1,8
710 PRINT"5. y=6+2*x^2-x^4"
720 LOCATE 40,8
730 PRINT"6. y=sqr(x^2+2)*sin(x)"
740 LOCATE 1,10
750 PRINT"7. y=x*sin(1/x)*sin(1/x)"
760 LOCATE 40,10
770 PRINT"8. y=sin(x/log(abs(x)+1.1))"
780 INK 1,24:GOSUB 1030:GOSUB 420
790 GOSUB 120:GOSUB 1610:RETURN
800 '
810 REM ******* Polar Functions *******
820 '
830 INK 1,1:CLS:LOCATE 21,2
840 PRINT"P O L A R   G R A P H S"
850 WINDOW#0,5,78,2,24
860 LOCATE 1,4:PRINT"1. r=atn(z)"
870 LOCATE 25,4:PRINT"2. r=1"
880 LOCATE 44,4:PRINT"3. r=a+sin(b*z)"
890 LOCATE 1,6
900 PRINT"4. r=a*cos(c*z)+b*sin(c*z)"
910 LOCATE 33,6
920 PRINT"5. r=a+b*cos(c*z)+c*sin(c*z)"
930 LOCATE 1,8
940 PRINT"6. r=a+b*sin(c*z)+a*cos(b*z)"
950 LOCATE 33,8
960 PRINT
 "7. r=sin(z-z*a)*cos(z-z*b)*sin(z-z*c)
970 INK 1,24
980 LOCATE 1,10:GOSUB 1180:GOSUB 520
990 GOSUB 240:GOSUB 1610:RETURN
1000 '"""
1010 REM  *** Graph Plot Parameters ***
1020 '
1030 Keystr$="12345678"
1040 LOCATE 1,12
1050 INPUT "ENTER EQUATION NO. ",n$
1060 n=VAL(n$)
1070 IF NOT n=INT(n)OR(n<1 OR n>8) THEN
     LOCATE 1,12:PRINT"EQUATIONS ARE ";
     "1 - 8 ONLY!";SPC(15):GOTO 1050
1080 ON INSTR(keystr$,n$)GOSUB 1360,
     1370,1380,1390,1400,1410,1430,1450
1090 LOCATE 1,14
1100 PRINT"VALUES OF x FOR PLOT"
1110 LOCATE 1,16:INPUT"LOWEST VALUE";a
1120 LOCATE 40,16:INPUT"HIGHEST";b
1130 IF NOT b>a THEN LOCATE 1,18:PRINT
     "Error - Try again please!":LOCATE
     1,14:GOTO 1090
1140 GOSUB 2090:GOSUB 2000:RETURN
1150 '
1160 REM ****** Polar Parameters ******
1170 '
1180 Keystr$="1234567"
1190 INPUT"EQUATION NO. ",n$
1200 n=VAL(n$)
1210 IF NOT n=INT(n)OR(n<1 OR n>7) THEN
     LOCATE 1,10:PRINT"EQUATIONS ARE ";
     "1 - 7 ONLY!";SPC(14):GOTO 1190
1220 ON INSTR(Keystr$,n$) GOSUB 1470,
     1480,1490,1500,1520,1540,1560
1230 LOCATE 1,12:PRINT"STANDARD PLOT";
1240 PRINT": a=0, b=1,c=5,d=1,e=1"
1250 LOCATE 1,14:INPUT;"VALUE OF a";a
1260 LOCATE 25,14:INPUT;"VALUE OF b";b
1270 LOCATE 45,14:INPUT;"VALUE OF c";c
1280 LOCATE 1,16:INPUT;"VALUE OF d";d
1290 LOCATE 45,16:INPUT"VALUE OF e";e
1300 GOSUB 2220:GOSUB 2000
1310 tim=TIME:WHILE TIME<tim+600:WEND
1320 RETURN
1330 '
1340 REM ****** Define Functions ******
1350 '
1360 DEF FNa(x)=1/(x*x-1):RETURN
1370 DEF FNa(x)=x*SIN(1/(x*x*x)):RETURN
1380 DEF FNa(x)=SQR(x*x+2):RETURN
1390 DEF FNa(x)=COS(x*EXP(-x/5)):RETURN
1400 DEF FNa(x)=6+2*x*x-x*x*x*x:RETURN
1410 DEF FNa(x)=SQR(x*x+2)*SIN(x)
1420 RETURN
1430 DEF FNa(x)=x*SIN(1/x)*SIN(1/x)
1440 RETURN
1450 DEF FNa(x)=SIN(x/LOG(ABS(x)+1.1)
1460 RETURN
```

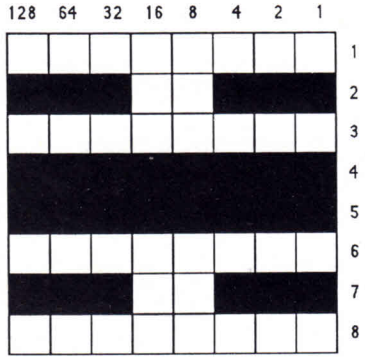
```
1470 DEF FNa(z)=ATN(z):RETURN '""
1480 DEF FNa(z)=1:RETURN
1490 DEF FNa(z)=a+SIN(b*z):RETURN
1500 DEF FNa(z)=a*COS(c*z)+b*SIN(c*z)
1510 RETURN
1520 DEF FNa(z)=a+b*COS(c*z)+c*SIN(c*z)
1530 RETURN
1540 DEF FNa(z)=a+b*SIN(c*z)+a*COS(b*z)
1550 RETURN
1560 DEF FNa(z)=SIN(z-z*a)*COS(z-z*b)*
     SIN(z-z*c)
1570 RETURN
1580 '
1590 REM ********* Another Go *********
1600 '
1610 LOCATE 1,1:PRINT"*";
1620 WHILE INKEY(47)=-1:WEND
1630 PRINT" ANOTHER GO? Y OR N"
1640 IF INKEY(43)=-1 AND INKEY(46)=-1
     THEN 1640
1650 IF INKEY(43)=0 THEN 60
1660 CALL &BB03:MODE 2:CLS:END
1670 '
1680 REM ********* Initialise *********
1690 '
1700 INK 0,1:INK 1,1:MODE 2:BORDER 1
1710 CLS:ORIGIN 0,0:INK 2,1 INK 3,1
1720 m=0.001:xx=318:yy=198:x=2:y=2
1730 RETURN
1740 '
1750 REM  ******** Set Up Menu ********
1760 '
1770 WINDOW#1,1,80,1,25:PAPER#1,1:CLS#1
1780 WINDOW#0,3,78,2,24:PAPER#0,0:CLS#0
1790 PEN 1:LOCATE 21,3
1800 PRINT"G R A P H   P L O T T E R"
1810 LOCATE 1,6
1820 PRINT" THIS PROGRAM PLOTS AND ";
1830 PRINT"DRAWS GRAPHS OF ";
1840 PRINT"ORDINARY AND POLAR ";
1850 PRINT"FUNCTIONS. ":PRINT
1860 PRINT" GRAPHS MAY BE DRAWN IN ";
1870 PRINT"MODES 0,1 OR 2"
1880 LOCATE 1,11:INK 1,24
1890 INPUT" ENTER MODE REQUIRED: ",mde
1900 IF (mde<0 OR mde>2) OR
     NOT INT(mde)=mde THEN INK 1,1:
     GOTO 1770
1910 LOCATE 1,13
1920 PRINT" DO YOU WANT POLAR GRAPHS";
1930 PRINT" - Y OR N?"
1940 WHILE INKEY(43)=-1 AND
          INKEY(46)=-1:WEND
1950 IF INKEY(43)=0 THEN gsb=1
     ELSE gsb=2
1960 WHILE INKEY$<>"":WEND:RETURN
1970 '""
1980 REM *** Scale Assessment Pause ***
1990 '
2000 PRINT"PREPARING GRAPH - ";
2010 PRINT"P L E A S E   W A I T"
2020 PRINT"A * WILL INDICATE ";
2030 PRINT"COMPLETION OF GRAPH ";
2040 PRINT"THEN PRESS <SPACE> ";
2050 PRINT"TO CONTINUE":RETURN
2060 '
2070 REM *** Draw/Plot/Shade Choice ***
2080 '
2090 PRINT
2100 PRINT"Use PLOT for ";
2110 PRINT"the most accurate result!"
2120 PRINT
2130 PRINT"PLOT, DRAW, OR SHADE - ";
2140 PRINT"P, D, OR S?"
2150 PRINT
2160 WHILE INKEY(27)=-1 AND INKEY(60)=-1
       AND INKEY(61)=-1:WEND
2170 IF INKEY(27)=0 THEN p$="p"ELSE
     IF INKEY(61)=0 THEN p$="d"ELSE
     p$="s"
2180 WHILE INKEY$<>"":WEND:RETURN
2190 '
2200 REM *** Polar Draw/Plot Choice ***
2210 '
2220 PRINT
2230 PRINT"PLOT OR DRAW - P OR D?"
2240 PRINT
2250 WHILE INKEY(27)=-1 AND
          INKEY(61)=-1:WEND
2260 IF INKEY(27)=0 THEN p$="p"ELSE
     IF INKEY(61)=0 THEN p$="d"
2270 WHILE INKEY$<>"":WEND:RETURN
2280 '
2290 REM ****** Error Processing ******
2300 '
2310 IF ABS(x)<1E-15 THEN x=x+st
2320 IF u>30000 THEN u=0.95*u
2330 IF v>30000 THEN v=0.95*v
2340 RESUME NEXT
```



The Trials of Tony Blakemore

A Column for the absolute beginner

Having learnt how to drive your friends crazy by redefining the characters on the keyboard, this month we will move on to graphic characters. The basics are exactly the same as changing the B to an A as we did last month. Take a piece of graph paper and mark off an eight by eight grid. The character we are going to design is an aerial representation of a racing car.



The symbol will be CHR$(201). Count the first row - it is blank so has a value of zero. The second row should come to 231, the third another zero, the fourth and fifth rows are the same with values of 255 and the last three are zero, 231 and zero respectively.

The coding to change the character CHR$(201) from a double slash to a car is:

```
10  SYMBOL AFTER 200
20  SYMBOL 201,0,231,0,255,255,0,231,0
30  PRINT CHR$(201)
```

Now we have a car and nowhere to go. Listed below is a simple program to display and race six cars. All sections have remarks (') to explain what is happening, and though very simple, illustrates the ability of the Amstrad to produce animation with the minimum amount of programming.

```
10  'Define variables as whole numbers
20  (integers) - this speeds up all the loops
30  and makes the cars go faster.
40  Define CHR$(201) as a car.
50  '
60  DEFINT A-Z:MODE 1
70  SYMBOL AFTER 200
80  SYMBOL 201,0,231,0,255,255,0,231,0
90  '
100 'Print numbers and the finish line
110 'Set start location of the cars
120 '
130 FOR NUMBER=1 TO 6:PEN 2
140 LOCATE 5,NUMBER+NUMBER:PRINT NUMBER
150 X(NUMBER)=7:NEXT
160 FOR NUMBER=1 TO 6
170 LOCATE 35,NUMBER+NUMBER:PRINT "|"
180 NEXT:PEN 1
190 '
200 'Print title
210 '
220 LOCATE 11,14
230 PEN 3:PRINT"AMSTRAD CAR RACES":PEN 1
240 '
250 'Print car at random location
260 'Check for first car finished
270 '
280 FOR NUMBER=1 TO 6
290 LOCATE X(NUMBER),NUMBER+NUMBER
300 PRINT "  ";CHR$(201)
310 IF X(NUMBER)=34 AND Y=0 THEN Y=NUMBER
320 CAR=INT(RND*2+1)
330 IF CAR=1 THEN X(NUMBER)=X(NUMBER)+1
340 NEXT
350 IF Y=0 THEN 280
360 '
370 'Print winner and check for another game
380 '
390 'LOCATE 14,16:PEN 2
400 PRINT"NUMBER";Y;"WINS";CHR$(7):PEN 1
410 LOCATE 12,18:PEN 3
420 PRINT"ANOTHER RACE Y/N?":PEN 1
430 I$=UPPER$(INKEY$):IF I$=" " THEN 430
440 IF I$="Y" THEN RUN
450 END
```

The remark symbol (') is the shifted 7. All remark lines can be removed if you wish and are only there to explain how the program works. Next month we will look at more complicated characters and delve a little deeper into animation.

The Learning Centre

An Introduction to Machine Code - Part Three

by Shane Kelly

What we will do now is to develop a general purpose routine for passing parameters to a M/C routine that is placed in a reserved area of memory. Cast your mind back to last month's article. In it was a BASIC program that POKED a value into a known area of memory and then a subroutine picked up that value and manipulated it, and finally displayed the result. Analysing theese concepts we find that :-

a) The parameter was put in a known place.
b) The manipulating routine picked it up from the know place.
c) The second routine stored it (in that case, to the screen).

So, our general purpose routine will follow along these lines. We will set up a byte of memory for each of our registers (including the FLAG register) in an area set aside for that purpose. We will make this area known to our machine code routine so that all registers may be loaded and all results passed back to our BASIC program.
To set aside the memory type:-

```
PRINT HIMEM: MEMORY HIMEM-1024:
AREG=HIMEM+1: PRINT AREG
```

Take note of the value of AREG. It will form the base of our 'known register area'. Use the above as the first line of any BASIC program that uses this method of passing parameters. The second line is as follows:-

```
FREG=AREG+1: CREG=AREG+2: BREG=AREG+3:
EREG=AREG+4: DREG=AREG+5: LREG=AREG+6:
HREG=AREG+7
```

You will note that the register areas are set up in reverse order. This is because the Z80 expects the first place of data in the lowest memory address and the most significant piece of data in the next highest memory address. This means that to load, say, the HL register pair, L (or Low register) is loaded first from the Low byte address and then the Hi register is loaded from the next Highest memory address. (Why it is that way we need not concern ourselves). Now follows the first part of any machine code routine that is to be used if parameters need to be passed. I will present the opcodes first, then the mnemonics followed by an explanation.

```
DD 21 00 00     LD IX,0
DD 39           ADD IX,SP
31 00 00        LD SP,0
F1              POP AF
C1              POP BC
D1              POP DE
E1              POP HL
DD F9           LD SP,IX
CD FIRMWARE     Call required firmware routine
```

Line one LOADS the IX register with zero. Line two ADDS the contents of the STACK POINTER register to the IX register. This is a normal method of saving the value of the STACK POINTER without storing it in memory. IX will now contain the value of the SP register. Line three LOADS the SP register with zero. Line four is an opcode that POPS out of memory a 16 bit value and places it into the AF register. You may be wonderirng how the sytem knows which area to take the data from in order to load into the AF register pair. Bear with me and all will be revealed! The following lines do the same for the other register pairs. The next line puts the value held in the IX register into the STACK POINTER register. You will remember that we saved in the IX register the previous value of the STACK POINTER register so it is now back as it was when we entered our M/C routine. We are now able to call our FIRMWARE ROUTINE knowing that all registers are set up as required by POKING the required value into the variable representing the register, namely: AREG,BREG etc. Our FIRMWARE ROUTINE will return to the byte following the high address byte of the firmware's address.
Now we will present the routine to pass back to BASIC the values held in the registers when we return from our CALLED FIRMWARE ROUTINE.

```
DD 21 00 00     LD IX,0
DD 39           ADD IX,SP
31 00 00        LD SP,0
E5              PUSH HL
D5              PUSH DE
```

C5	PUSH BC
F5	PUSH AF
DD F9	LD SP,IX
C9	RETURN (TO BASIC)

You can see that it is very nearly the reverse of our routine to load the registers. The PUSH opcode puts the values held in the pushed registers into an area of memory 'pointed' to by the STACK POINTER. You may still be wondering how the system knows where to put the values that are returned by our CALLED routine. The answer is that the BASIC program supplies the addresses because, although they can be in the same place for every program that will use CALLED FIRMWARE routines, this could be inconvenient when there is a need to move our M/C routine to a different area of memory. So BASIC calculates the neccessary places for us and POKES the values needed into our LOAD and SAVE register routines.

Now we must set up some variables which are of great importance for correctly interfacing of our BASIC program and FIRMWARE ROUTINE.

```
FIRMIN=HREG+3: PICKUP=FIRMIN+7:
DROP=FIRMIN+25
```

These variables are set to these values so that we may easily tell the M/Code routine from where to POP the register values and where to PUSH the returned values. The SP register works like this: when a value is POPped from the memory pointed to by the SP, then the SP is automatically incremented by two. When values are PUSHed into the memory area pointed to by the SP, the SP is automatically decremented by two. Thus you can see how our M/C routine will get the registers loaded as required. (if you don't see, re-read this section and any other information you have on the SP).

Now the following lines are to be incorporated into our basic program:

```
POKE
PICKUP,VAL("&"+RIGHT$(HEX$(AREG),2)):POKE
PICKUP+1,VAL("&"+LEFT$(HEX$(AREG),2)):
POKE
DROP,VAL("&"+RIGHT$(HEX$(HREG),2)):
POKE
DROP+1,VAL("&"+LEFT$(HEX$(HREG),2))
```

Study this section until you are sure that you know what is going on. It is not necessary to understand in intricate detail, but the more you study the method of interfacing, the easier it will be to use. It may seem that we can call only one firmware routine with each set of the above routines. Not so! We can call as many as we like merely by poking the required addreses into the BASIC variable MCROUT which is set up as above.

Let's now recap on the main points:

a) The basic routine reserves an area of high memory.

b) The general purpose PASS REGISTER and DROP REGISTER ROUTINES are poked into it.

c) The basic variables are set up to allow us to tell the M/C routine where to pick up its parameters and where to put the returned values.

d) We poke the required values into our M/C routine using BASIC.

e) We set up the address of the FIRMWARE ROUTINE to be called.

```
10 MEMORY HIMEM-1024: AREG=HIMEM+1
20 FREG=AREG+1:
   CREG=AREG+2:BREG=AREG+3:
   EREG=AREG+4: DREG=AREG+5:
   LREG=AREG+6: HREG=AREG+7
30 FIRMIN=HREG+3: PICKUP=FIRMIN+7:
   DROP=FIRMIN+25: MCROUT=FIRMIN+16
40 FOR X=0 TO 33: READ A$ : POKE
   FIRMIN+X,VAL("&"+A$): NEXT
50 DATA DD,21,00,00,DD,39,31,00,00,F1,C1,
   D1,E1, DD,F9,CD,00,00,DD,21,00,00,DD,39,31,
   00,00,E5, D5,C5,F5,DD,F9,C9
60 POKE PICKUP,
   VAL("&"+RIGHT$(HEX$(AREG),2)):
   POKE PICKUP+1,
   VAL("&"+LEFT$(HEX$(AREG),2)):
   POKE DROP,
   VAL("&"+RIGHT$(HEX$(HREG),2)):
   POKE DROP+1,
   VAL("&"+LEFT$(HEX$(AREG),2))
70 POKE MCROUT,&6E: POKE MCROUT+1,&BC:
   'TURN ON CASSETTE FOR EXAMPLE
80 REM
```

Both the Firmware Manual and 'The Ins and Outs' present the address of the firmware routines with the high byte of the address first followed by the low byte of the address.

Having used our firmware routine to do what we wanted, we can now get any values that were returned by PEEKING the appropriate register variable, e.g. PEEK HREG would return any value left there by the firmware routine. NOTE that not all firmware routines return valid data, so be sure to study any EXIT CONDITIONS associated with your firmware routine.

It was never the intention of these articles to teach you to use specific FIRMWARE ROUTINES, but to give you a method that is fairly painless to use and can be employed in the majority of circumstances. Therefore I leave it up to you to experiment with the above routines and hence to gain a greater understanding of the workings of ARNOLD in particular and computers in general.

Remember - experimentation is the first step to knowledge!

Music

A preview of next month's Learning Centre

Have you ever wanted to program music on your Amstrad, but found that you can't read music? Or found, perhaps, that you had forgotten how?

The latter was more my situation when I was showing off my newly-acquired Amstrad to my nephew, last Christmas. You see, it's about 37 years since I took a dislike to my music teacher and refused to take any further lessons from 'that man'! I might add I had not made much progress, only being able to play a few scales and the simples of tunes.

What have these personal reminiscences to do with programming music, you ask. Well, if I can do it, so can you. It isn't necessary to be able to play an instrument, or to be able to read music 'fluently'. As Arnold and the newcomer ('Fred'?) have only three voices, the music cannot contain complex chords. All you need to be able to do is to decipher musical notation and to translate the notes and their values into the appropriate BASIC commands.

Of course, there are a few tricks to getting the best out of Arnold's sound chip and so your esteemed Editor has come up with the brilliant idea of a series of short articles on music programming on the Amstrad. (The series in that English magazine had nothing to do with it, despite what the rumour mongers are suggesting!) Anyway, starting next month (if I meet the deadlines, as promised) is all you need to know to get started exploring this fascinating area of the Amstrad computers' capabilities.

If you cannot wait until then, try this short program. It has no frills and is not particularly user-friendly. What it does do, however, is respond to the keys 'cdefgabCDEFGABH-^' with two octaves of music, including FLATS (-) and SHARPS (^). Pick out a tune and it will play it back and it does not take all night to type the program in!

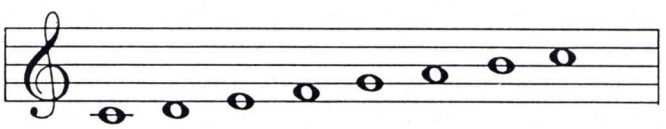
For something to play on this 'rudimenti synthesiza', try the following and see if you can recognise the melody:

```
e,f,g,a,D,C,a,g,f,d,e,f,f,f,f,f,e,f,g,a,
a,a,g,a,D,C,a,g,f,d,d,d,f,a,b,-,C,C,C,D,
C,a,f,a,g,g,g,g,g,e,f,g,a,a,a,g,a,D,C,a,
g,f,d,d,d,e,f,g,a,a,a,b,-,a,g,f,g,f,f,f,
f,f,C,D,E,F,F,F,E,E,D,C,D,C,a,f,f,f,C,D,
E,F,F,F,E,E,D,C,a,g,g,g,g,g,C,C,C,A,G,G,
F,D,F,C,a,f,f,f,e,f,g,a,D,C,a,g,f,d,e,f,
f,f,f,f
```

(That's the length of 142 notes; the '-' converts the preceding note into 'b flat).

The program only accepts an input of one note at a time and has no provision for correcting errors after <ENTER>. I told you it wasn't user-friendly! Still, what do you expect of 873 bytes of BASIC?

```
10 '***** RUDIMENTARY NOTE-PLAYER *****
20 MODE 1:CLS:PEN 2:number=1
30 PRINT"CAPS LOCK for upper octave."
40 PRINT"Notes are 'cdefgabCDEFGABH'"
50 PRINT"Use'^'for SHARP;'-'for FLAT,";
60 PRINT" entered as the next note."
70 PRINT:INPUT"Length of tune";length
80 DIM music(length)
90 notenumber$="c d ef g a bC D EF G A B
H-^"
100 WHILE NOT number>length
110 INPUT"Enter a note";note$
120 music(number)=INSTR(notenumber$,note
$)
130 IF note$="-"OR note$="^" THEN number
=number-1
140 IF note$="-"THEN music(number)=music
(number)-1
150 IF note$="^"THEN music(number)=music
(number)+1
160 number=number+1:WEND
170 INPUT"Tempo: 1-9";tempo
180 FOR number=1 TO length
190 IF music(number)=0 THEN 230
200 frequency=440*(2^(octave+((music(num
ber)-10)/12)))
210 period=ROUND(125000/frequency)
220 SOUND 1,period,10*tempo,7
230 NEXT number
```



USER GROUP INFORMATION

You would have noticed that since the first issue of THE AMSTRAD USER, the User Group Contact List has grown quite considerably.

You may also have noticed that User Group news varies in length from month to month depending on whether anyone from the groups remembers to let us know what is happening. (Now there's a thought - perhaps no one has been allocated the post of 'The Amstrad User Liaison Officer' or more simply, 'Publicity Officer'). Plus, it has not always been absolutely clear as to which groups exist, where they meet and so on.

Not any more! From now on, established groups will be permanently listed and separated from the User Group Contacts, the latter containing only those people who are seeking to find other users in their area with a view to setting up a group. We hope this will make things clearer.

Now we've got a new format, it is up to individual groups to keep us advised of any changes to their listings. We should also mention that there are a number of groups in existance which are not listed. This is quite simply because no one from the groups has officially contacted us. Now that The Amstrad User is very firmly established throughout the country as the only Australian Amstrad user's magazine (reaching many thousands of enthusiasts every month), it really does make sense to register your group so that newcomers know where you are. You'll never grow if you keep quiet, and we should remind you that it doesn't cost a penny (or a cent!) to be included in either list. So - get to it!

NATIONWIDE USER GROUPS

WESTERN AUSTRALIA

AMSWEST, Perth

President:	Tony Clitheroe	(09 275 1257)
Secretary:	Mrs. P.T. Ardron	(09 361 8975)
Treasurer:	Eric Stallard	(09 339 6361)

Regular meetings take place at a venue in Shenton Park on the first and third Tuesdays of each month starting at 7.30p.m.

ROCKINGHAM/KWINANA USER GROUP

Contact Bob Harwood on 095 27 1777 for further details on meeting times.

SOUTH AUSTRALIA

AMSTRAD COMPUTER CLUB INC.

President:	Chris Sowden	(08 295 5923)
Secretary:	Vince Alfonso	(08 384 2394)
Treasurer:	Les Jamieson	(08 356 9612)

The group meets each Tuesday at the Grange Primary School between 6.30 p.m. and 9.00 p.m. but may be moving to new premises shortly. You are advised to first check with Chris Sowden. Any correspondence can be addressed to PO Box 210, Parkholme, 5043.

PORT PIRIE AMSTRAD USER GROUP

President:	Rick Cable	(086 32 5967)
Secretary:	John Coleman	
Treasurer:	Dave Green	

The group meets at 7.30 p.m. on the first Monday of each month at the Princess Park Scout Hall, Solomontown. For further details contact Rick Cable.

VICTORIA

WESTERN AMSTRAD USER GROUP

President:	Mike McQueen	(03 312 5594)
Secretary:	Peter Pilbeam	(03 336 0705)
Treasurer:	Frank Melino	(03 337 2495)

The meetings are held on each alternate Tuesday and Sunday (to allow for shift workers) at the Tottenham North Primary School, South Road, Braybrook.

CENTRAL AMSTRAD USER GROUP

President:	Rimon Russo	(03 428 4281)
Secretary:	Melanie Leith	(03 383 1498)
Treasurer:	Fred Gillan	(03 598 5780)
Publicity:	Don Leith	(03 383 1498)

Meetings are held twice a month in the Hall at the corner of Church and Somerset Streets, Richmond on a Sunday afternoon commencing at 4.00 p.m. A short business session is followed by a predetermined topic and concludes with a friendly group session.

EASTERN AMSTRAD USER GROUP

President:	Tony Blakemore	(03 878 6212)
Secretary:	Andrew Martin	(03 729 8471)
Treasurer:	Ron Dunn	(03 277 7868)

Regular meetings are held on the first Sunday of every month at the Box Hill Scout Hall, Tyne St. (The Hall is located in Halligan Park between Watts and Mersey Streets). Proceedings commence at 2.00 p.m.

SOUTHERN AMSTRAD USER GROUP

President:	Mike Prezens	(03 781 2158)
Secretary:	Martin Scragg	(059 78 6949)
Treasurer:	Steve Issell	(03 786 9340)

Meetings are held on the third Tuesday of every month (except December) from 7.30 p.m. to 10.30 p.m. The venue is the Senoir Campus at John Paul College, Frankston.

ACT

ACT AMSTRAD USER GROUP

Convenor:	Arthur McGuffin	(062 31 9437)
Secretary:	Kevin Loughrey	(062 31 2991)
Treasurer:	Kevin Cryer	(062 91 9881)

The group meets at 7.30 p.m. on the first Wednesday of each month in the Seminar Room of the Oliphant Building at the Research School of Physical Science, Australian National University.

QUEENSLAND

BRISBANE AMSTRAD COMPUTER CLUB

President:	Paul Witsen	(07 371 9259)
Secretary:	Mal Harper	(07 288 3578)
Treasurer:	Ian Cartwright	(07 369 9364)

Meetings are held on the first Tuesday of each month at Junction Park State School, Annerley starting at 7.30 p.m. in Room 15a.

User Group Contact List

Please note that the following names are listed as contact points for new user groups and should NOT be viewed as a problem solving service. See other list for established groups.

NSW

Mark Kelloway	Barrack Point	(042) 95 1581
Hans Hill	Blacktown	(02) 671 2929
Chris Craven	Canowindra	(063) 44 1150
Bruce Jones	Coffs Harbour	(066) 52 8334
T.J. Webb	Glossodia	(045) 76 5291
David Higgins	Inverell	(067) 22 1867
John Patterson	Lismore	(066) 21 3345
Paul Wilson	Moruya	(044) 74 3160
Frank Humphreys	Mummulgum	(066) 64 7290
Martin Clift	Narrabri	(067) 92 3077
Bob Hall	Newcastle	(049) 52 6915
R. Vijayenthiran	Newtown	(02) 519 4106
Ken Needs	St. Ives	(02) 449 5416
Chas Fletcher	Toongabbie	(02) 631 5037
Nick Bruin Snr.	Tweed Valley	(066) 79 3280
Jim Owen	Uranga	(066) 55 6190
John Harwood	Windale	(049) 48 5337

ACT

Chris Rogers	Fraser	(062) 58 5749

Vic

David Carbone	Burwood	(03) 29 4135
Rod Anderson	Camperdown	(055) 93 2262
Paul Walker	Heathmont	(03) 729 8657
Andrew Portbury	Leongatha	(056) 62 3694
Ron Butterfield	Leopold	(052) 50 2251
Sue Kelly	Manangatang	(050) 35 1402
Alan Harris	Sale	(051) 44 1454
Mrs. G. Chapman	South Clayton	(03) 551 4897

QLD

Steven Doyle	Caloundra	(071) 91 3147
Mick O'Regan	Gladstone	(079) 79 2548
Kylie Telford	Goondiwindi	Calingunee246 (weekendsonly)
D.F. Read	Ingham	(077) 77 8576
Tim Takken	Ipswich	(07) 202 4039
Michael Toussaint	Loganlea	(07) 200 5414
Alan Laird	Maryborough	(071) 22 1982
R.C. Watterton	Toowoomba	(076) 35 4305

SA

Lindsay Allen	Murray Bridge	(085) 32 2340

WA

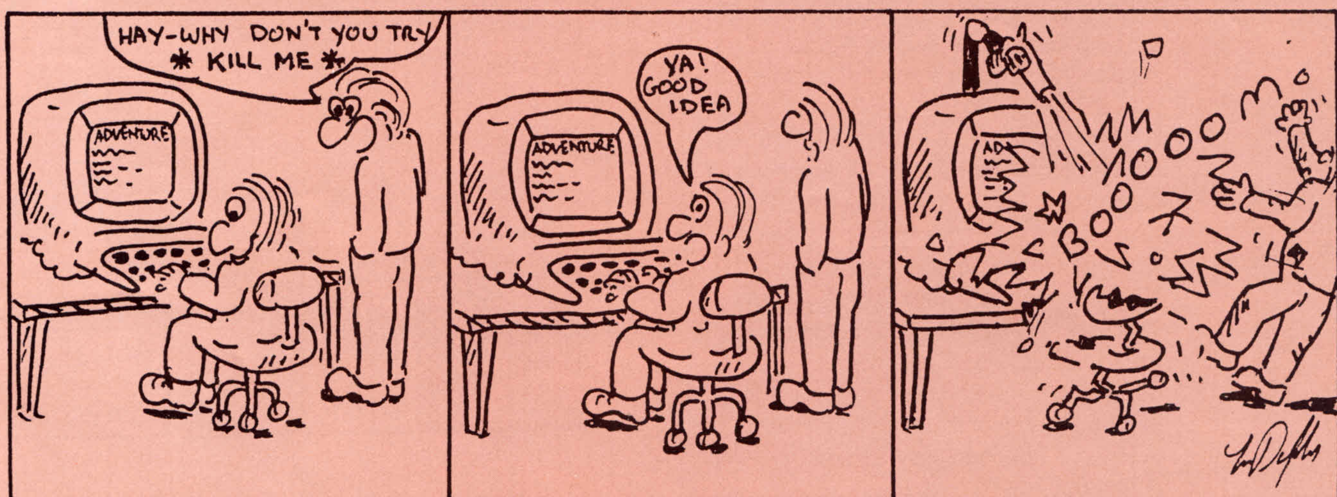
Dave Andersen	6 Kitchener Rd Merredin, 6415	
Graeme Worth	Scarborough	(09) 341 5211
P.M. Nuyens	Waroona	(095) 33 1179

TAS

Conal McClure	Scottsdale	(003) 52 2514

NT

G.P. Heron	Tiwi	(089) 27 8814



Home Budget

A Program from R. Page

I'll wager that most domestic arguments are about money. Where is all the money going? What have you/we spent it on this month? How much did *that* cost?

If it sounds familiar, this Home Budget program could well provide the answers. It allows you to keep track of up to 100 items, calculate totals and store them for future use. It will produce a list sorted by item name with a total at the end of each entry. A prompt is given at the end of each update or deletion stage until END is entered.

Since the program was originally written for a Commodore 64 not all the Amstrad facilities have been used to their fullest extent, but you will still find this a handy utility which will assist in keeping your accounts in order and prove to your family (wife) that the computer is not just another fancy video game.

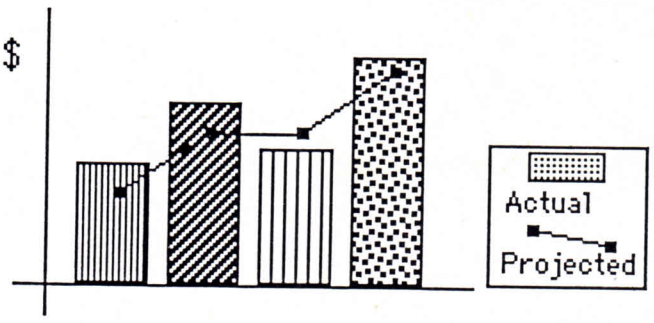
The following list provides an explanation of the options displayed in the menu:

DISPLAY	(1)	Displays a list of 20 items each with an index number. This number is used in the UPDATE/DELETE sections of the program (Lines 1000-1999).
NEW	(2)	Allows you to initialise the list and add new expenses. This section ends if you try to enter more than 100 items or when you type END and Enter. (Lines 2000-2999).
UPDATE	(3)	Allows you to correct any item name or amount by using the index number from the display screen when you receive the ITEM £ prompt. On entering the number, the item and amount will be displayed. Again, type END to finish this section (Lines 3000-3999).
SAVE	(4)	Allows you to save the data for future reference into a nominated file name which is entered at the prompt FILE NAME. Don't forget the name you use - I find that by using the month and year, I can keep a permanent record of each month's expenses. Type END and Enter to finish the section (Lines 4000-4999)
DELETE	(5)	START AT and END AT uses the index numbers to delete the items in between. To delete one item line put a zero in END AT (Lines 5000-5999).
OPTION	(6)	This displays the main menu (Lines 6000 - 6999).
LOAD	(7)	Allows you to load data for studying and modification (Lines 7000 - 7999).
PRINT	(8)	Allows printer owners to get hard copy of the screen display (Lines 9000-9999).
END	(9)	The end of program routine, which also has a SAVE option in case you have omitted to do so during the program (Lines 8000 - 8999).
CTRL U		This is used only when there are more than 20 line items. It provides the facility to step back 20 items.
CTRL D		For viewing the next 20 line items on the list. Remember that the CTRL and D keys should be pressed at the same time.

The program is written around various subroutines called by the main loop (Lines 200-300). There are two utilities included for use by any module - the sorting subroutine (Lines 500-599) and an accumulator subroutine (Lines 300-399).

In line 275 there should be a symbol for CTRL U inserted between the inverted commas. Similarly in line 280, there should be a symbol inserted for CTRL D.

As you read through the listing, you will find that all the functions on the menu go to various subroutines. This makes it easier to trace through the steps of the program.



```
1 REM   BUDGET IS A PROGRAM ORIGINALLY PUBLISHED BY COMPUTE FO
R THE
2 REM  COMMODORE C64  AND CONVERTED BY R PAGE TO RUN ON THE CPC 464
3 REM     TO END ANY SUBROUTINE TYPE IN END
5 SPEED WRITE 0
10 MODE 2
20 sz=100:i=-19
30 r$=CHR$(18) : ta=0
40 DIM a$(sz),ae(sz)
50 DEF FN rn (x)= INT (x*100+0.5)/100
210 GOSUB 6000
220 z$=INKEY$:IF z$="" THEN GOTO 220
230 IF z$="1" THEN GOSUB 1000
235 IF z$="2" THEN GOSUB 2000
240 IF z$="3" THEN GOSUB 3000
245 IF z$="4" THEN GOSUB 4000
250 IF z$="5" THEN GOSUB 5000
255 IF z$="6" THEN GOSUB 6000
260 IF z$="7" THEN GOSUB 7000
265 IF z$="8" THEN GOSUB 9000
270 IF z$="9" THEN GOSUB 8000
275 IF z$="" THEN i=i-20:GOSUB 1000
280 IF z$="" THEN i=i+20:GOSUB 1000
299 GOTO 220
300 REM ACCUM TOTALS
310 ta=0
320 FOR j=1 TO mx
330 ta=ta+ae(j)
340 NEXT j
399 RETURN
400 REM load files
410 INPUT "File Name";f$
420 IF f$="*end" THEN GOSUB 6000 :RETURN
450 OPENIN f$
460 INPUT #9 ,mx
470 FOR j=1 TO mx
481 INPUT #9,y
482 INPUT #9,r$
483 INPUT #9,a$(j)
484 INPUT #9,ae(j)
485 INPUT #9,r$
490 NEXT j
495 CLOSEIN
499 RETURN
500 REM sorting by name
505 IF mx=1 THEN GOTO 599
510 PRINT "Sorting"
520 FOR j=1 TO mx-1
530 FOR k=j+1 TO mx
540 IF a$(k)>a$(j) THEN GOTO 590
550 sm$=a$(k) :sm=ae(k)
560 a$(k)=a$(j):ae(k)=ae(j)
570 a$(j)=sm$:ae(j)=sm
590 NEXT k
595 NEXT j
599 RETURN
1000 REM DISPLAY
1010 IF (i<1) OR (i>mx) THEN i=1
1020 CLS:PRINT TAB(5)"Expenses "TAB(61)"Amt"
1030 FOR j=i TO i+19
1040 IF j>mx THEN PRINT " ":GOTO 1080
1050 pr$=STR$(ae(j)+0.001):pr$=MID$(pr$,2,(LEN(pr$)-2))
1060 IF ae(j)=0 THEN pr$="0.00"
1065 j$=MID$(STR$(j),2)
1070 PRINT TAB(3-LEN(j$));j$;TAB(4);a$(j);TAB(66-LEN(pr$));pr$
1080 NEXT j
1090 ta$=STR$(ta+0.001)
1100 ta$=LEFT$(ta$,LEN(ta$)-1)
1110 IF ta=0 THEN ta$="0.00"
```

} SEE NOTES

```
1120 PRINT:PRINT TAB(20)"TOTAL     ";TAB(66-LEN(ta$));ta$
1999 RETURN
2000 REM ADD NEW EXPENSES
2010 r=mx+1:n$="":e1$=""
2015 IF r=101 THEN GOTO 2999
2020 CLS:PRINT"  Add New Expenses"
2030 PRINT:PRINT SPC(12) "Item #";r
2040 PRINT:INPUT "Item Name ";n$
2050 IF n$="end" OR  n$="END" THEN GOTO 2999
2055 IF LEN(n$)>30 THEN n$=LEFT$(n$,30)
2060 a$(r)=n$
2070 PRINT:INPUT "Item Amt  ";e1$
2080 IF e1$="end" OR e1$="END" THEN GOTO 2999
2085 IF VAL(e1$)=0 THEN ae(r)=0:GOTO 2100
2090 ae(r)=FN rn (VAL(e1$))
2100 mx=mx+1
2105 IF mx=101 THEN GOTO 2999
2110 GOTO 2010
2115 IF mx=101 THEN GOTO 2999
2200 mx=mx+1
2999 GOSUB 500:GOSUB 300:GOSUB 6000:RETURN
3000 REM ************ UPDATE EXPENSES ********************
3010 CLS :  PRINT"Expenses";" Update"
3020 PRINT:INPUT "Item # ";p1$
3025 IF p1$="end" OR p1$="END" THEN GOTO 3999
3026 IF (VAL(p1$)=0) OR (VAL(p1$)<1) THEN PRINT:PRINT" Input Error!":GOTO 3
020
3027 p=INT (VAL(p1$))
3030 n$="":e1$=""
3040 IF p>sz THEN PRINT"Max Exceeded":p=sz:mx=p
3050 IF p>mx THEN mx=p
3060 pr$=STR$(ae(p)+0.001):pr$=MID$(pr$,2,(LEN(pr$)-2))
3065 IF ae(p)=0 THEN pr$="0.00"
3070 PRINT p;TAB(4)a$(p)TAB(42-LEN(pr$))pr$
3080 PRINT:INPUT "Item Name";n$
3090 IF n$="end" OR n$="END" THEN GOTO 3999
3100 IF n$<>"" THEN a$(p)=n$
3105 IF LEN(a$(p))>30 THEN a$(p)=LEFT$(a$(p),30)
3110 INPUT "Amt ";e1$
3120 IF e1$="end" OR e1$="END" THEN GOTO 3999
3125 IF e1$="" GOTO 3010
3130 IF(VAL(e1$)=0) AND (e1$<>"0") THEN PRINT:PRINT"   Input Error":GOTO 31
10
3135 IF VAL(e1$)=0 THEN ae(p)=0:GOTO 3800
3140 ae(p)=FN rn(VAL(e1$))
3800 GOTO 3010
3999 GOSUB 500:GOSUB 300:GOSUB 6000:RETURN
4000 REM SAVE FILE
4010 CLS :PRINT"   Save Expense List"
4020 PRINT:PRINT:INPUT"File Name";f$
4030 IF f$="end" OR f$="END" THEN GOSUB 6000:RETURN
4050 OPENOUT f$
4060 PRINT #9,mx
4070 FOR j=1 TO mx
4080 WRITE #9,j
4081 WRITE #9,r$
4082 WRITE #9,a$(j)
4083 WRITE #9,ae(j)
4084 WRITE #9,r$
4090 NEXT j
4100 CLOSEOUT
4999 GOSUB 6000:RETURN
5000 REM DELETE
5005 dt=0:tm=0
5010 CLS :PRINT"          Delete"
5020 s1$=""
5030 PRINT:PRINT:INPUT"Start at";s1$
5040 IF s1$="end" OR  s1$="END" THEN GOTO 5900
5050 ds=INT(VAL(s1$))
```

```
5060 s1$=""
5070 IF ds=0 THEN PRINT:PRINT"          Input Error":GOTO 5020
5080 s1$=""
5090 PRINT:PRINT:INPUT"End At";s1$
5100 IF s1$="end" OR s1$="END" THEN GOTO 5900
5110 IF s1$="" OR s1$="0" THEN de=0:GOTO 5200
5120 de=INT(VAL(s1$))
5125 IF de>mx THEN de=mx
5130 IF de>ds THEN GOTO 5200
5135 PRINT:PRINT:PRINT"  0 or Number Greater"
5140 PRINT:PRINT:PRINT"  Than";de;"Required"
5150 GOTO 5080
5200 IF de=0 THEN de=ds
5205 tm=de-ds+1
5207 dt=dt+tm
5210 FOR j=ds TO de
5220 a$(j)=CHR$(143)+CHR$(137):ae(j)=0
5230 NEXT j
5240 GOTO 5010
5900 GOSUB 500
5910 mx=mx-dt
5999 GOSUB 300:GOSUB 6000:RETURN
6000 REM OPTIONS MENU
6010 CLS:PRINT"   Options"
6020 PRINT "        "
6030 PRINT:PRINT"     1 - DISPLAY EXPENSES    CTRL U = LOWER INDEX"
6035 PRINT"                                   CTRL D = HIGHER INDEX"
6040 PRINT:PRINT"     2 - NEW EXPENSES"
6050 PRINT:PRINT"     3 - UPDATE EXPENSES"
6060 PRINT:PRINT"     4 - SAVE EXPENSES"
6070 PRINT:PRINT"     5 - DELETE FROM LIST"
6080 PRINT:PRINT"     6 - OPTIONS SCREEN"
6090 PRINT:PRINT"     7 - LOAD FILES"
6100 PRINT:PRINT"     8 - PRINTER"
6110 PRINT:PRINT"     9 - END"
6999 RETURN
7000 REM LOAD
7010 CLS:PRINT"       Load"
7020 PRINT:PRINT"          Expense Files"
7030 INPUT"Load    (Y/N)";an$
7040 IF an$="y" OR  an$="Y" THEN mx=0:GOSUB 400:GOTO 7999
7050 IF an$="end" OR  an$="END" THEN GOSUB 6000:RETURN
7060 IF an$="n" OR  an$="N" THEN GOSUB 6000:RETURN
7065 GOTO 7030
7200 NEXT
7999 GOSUB 500:GOSUB 300:GOSUB 6000:RETURN
8000 REM.END OF JOB
8010 CLS:PRINT"    End of Program"
8020 PRINT:PRINT:PRINT"Would you like to save (Y/N)":INPUT an$
8030 IF an$="end" OR AN$="END" THEN GOSUB 6000:RETURN
8040 IF an$="n" OR an$="N" THEN GOTO 8060
8050 GOSUB 4000
8060 CLS:PRINT" Thank you good-bye"
8070 END
9000 REM DISPLAY ON PRINTER
9010 IF (i<1) OR (i>mx) THEN i=1
9015 PRINT #8,CHR$(14)+"Budget"+CHR$(15)
9016 PRINT #8,
9017 PRINT #8,
9025 PRINT #8,TAB(5)"Expenses "TAB(61)"Amt."
9030 FOR j=1 TO mx
9040 IF j>mx THEN PRINT " ":GOTO 9080
9050 pr$=STR$(ae(j)+0.001):pr$=MID$(pr$,2,(LEN(pr$)-2))
9060 IF ae(j)=0 THEN pr$="0.00"
9065 j$=MID$(STR$(j),2)
9075 PRINT #8,TAB(3-LEN(j$));j$;TAB(4);a$(j);TAB(66-LEN(pr$));pr$
9080 NEXT j
9090 ta$=STR$(ta+0.001)
9100 ta$=LEFT$(ta$,LEN(ta$)-1)
9110 IF ta=0 THEN ta$="0.00"
9125 PRINT #8,:PRINT #8,TAB(20)"TOTAL     ";TAB(66-LEN(ta$));ta$
9999 RETURN
```

Review of the SP-1000 Printer

by Simon Anthony

Any printer, providing it has a Centronics parallel interface, is capable of being used on a 464/664. A reasonably fair statement but some printer owners will justifiably argue that it often requires a deal of 'tinkering' to make the statement true.

DMP-1 owners avoid this irritation having invested in a piece of equipment which is already set up for use on the Amstrad, and it works well. Many letters and other submissions to The Amstrad User are printed on this machine, and the results are quite adequate. Nevertheless, one of the major criticisms directed at the DMP-1 is the lack of lower-case decenders. (This makes the word 'program' look like 'Prog̲ram' although a little exagerated in this example).

A new printer, jointly marketed by AWA and Seikosha, appears to answer the requirements of those users who are looking for a more professional finish to their printed text. The SP-1000 weighs in at just under 5 kilos.

It takes either single sheet paper or continuous stationery, and with the latter, the tractor unit must be removed. This is easily done. I searched vainly for a friction/tractor switch and discovered that the paper bail (the bar that applies pressure to the paper against the platen) was linked to a paper loading knob situated next to the paper-feed knob. When in the forward position paper is free to be moved around for loading or when on tractor feed, and when pushed back, applies friction for feeding single sheets.

Loading the ribbon cassette was simple as was adjusting the head position for multi-part paper. The printer can certainly handle two part paper but I didn't get the chance to try anything thicker, although the specs indicate that it is capable of handling three part.

It has an automatic paper loading function where the paper automatically advances to the top of form position by using just the paper loading knob and will take paper from 4" to 10" wide.

It is a possibility that a cut sheet feeder will be available by the end of the year.

Margin designation and 1" Skip perforation (for continuous paper) can be set by either DIP switch or manual commands.

The DIP switches are located at the back of the printer and are protected by a plastic cover (possibly to keep out 'small fingers'). These provide a number of functions, for example: specifying the paper length (1-6), selecting italic characters (2-1), language fonts (1-1 to 1-3) and many more. On the subject of language fonts, there are eleven from which to choose, but not being a linguist I can't tell you why Denmark is allocated two of them!

Naturally, commands can be entered through the Amstrad as the following examples show:

```
1 OPEN "LPT1:" AS #1
2 WIDTH #1,255
5 'HIGH QUALITY (CORRESPONDENCE)
10 PRINT #1,"NORMAL QUALITY 1";CHR$(10);
20 PRINT #1,CHR$(27);"x1";'HIGH QUALITY
30 PRINT #1,"HIGH QUALITY";CHR$(10);
40 PRINT #1,CHR$(27);"x0";
41 'CLEARS HIGH QUALITY
50 PRINT #1,"NORMAL QUALITY 2"
```

```
NORMAL QUALITY 1
HIGH QUALITY
NORMAL QUALITY 2
```

```
1 OPEN "LPT1:" AS #1
2 WIDTH #1,255
5 'SMALL CHARACTER
10 PRINT #1,"PICA CHARACTER 1";CHR$(10);
20 PRINT #1,CHR$(27);CHR$(15);'SMALL
30 PRINT #1,"SMALL CHARACTER";CHR$(10);
40 PRINT #1,CHR$(18);'CLEARS SMALL
50 PRINT #1,"PICA CHARACTER 2"
```

```
PICA CHARACTER 1
SMALL CHARACTER
PICA CHARACTER 2
```

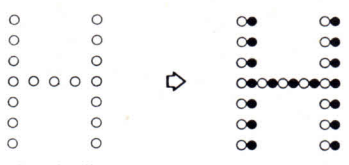
It is worth noting here that the printer has a 1.5k bytes RAM area. When the DIP switch 2-4 is turned off, the printer uses this area as a communications buffer, and when on, it becomes a down-load area.

Another useful function is the Hexadecimal Dump List function. Input data is printed in 2-digit hexadecimal numbers with a space between each. This allows 16 bytes of data to be printed on each line. This function is simply set by pressing the FF switch when initially turning on the printer.

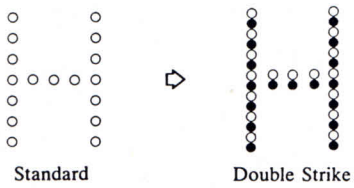
The SP-1000 offers Standard or Italic Cursive characters in PICA, ELITE or Condensed. Near letter quality can be obtained with PICA or ELITE. Other features include:

Proportional
Super and Subscripts
Underline
Double Width
Double Strike
Bold

If you've ever wondered how these are achieved with a dot matrix printer, the diagrams below give the answer in relation to producing Bold and Double Strike Characters.



Standard Bold



The printing speeds tend to vary depending upon the style chosen. For example, standard PICA, which consists of a 12x9 matrix (including half dots), will print at 100 characters per second. On the other hand, near letter quality ELITE, a 24x18 matrix including half dots, will print at just 24 characters per second.

The size of each printed character determines the number of characters that can be printed in a line. PICA will produce 80 in a line, ELITE produces 96 and condensed gives an impressive 137.

There is also a Graphic Print mode - in either standard or Double density. The former has a maximum of 480 horizontal dots, whilst the latter has 960. However, the print speed is halved, which is natural enough with double density printing. Just to complicate matters, it is possible to print in Quadruple Density Graphic mode where the maximum number of horizontal dots is 1920, but I won't go into detail now.

Printing is bi-directional.. To the novice, this means that it will print from left to right and again from right to left as it performs a 'carriage return'.

Printout Sample

789:;<=>?@ABCDEF
<=>?@ABCDEFGHIJKLM
789:;<=>?@ABCDEFGHI
/0123456789:;<=>?@ABCDEFGHI
789:;<=>?@ABCDEF
<=>?@ABCDEFGHIJKLM
789:;<=>?@ABCDEFGHI
EFGHIJKL
789:;<=>?@ABCDEFGHIJKLMNOPQ

I found the printing operation to be smooth and quiet and the final result - the all important printed page - to be of good quality, and certainly very acceptable for my needs, and indeed for those in a business environment. In fact a number of the program listings appearing in this magazine have been printed using the machine.

The documentation is very comprehensive and consists of a manual of nearly ninety pages in length. It's the best I've seen for a printer for a long time.

The price of the SP-1000 is likely to be under $500 and is a most welcome addition to the peripherals available for the Amstrad.

Character Specification Table

Character Font			Character Structure (H×V +Space)	Maximum Column Number	Character Spacing (Character/inch)	Print Speed (Character/Second)
Standard Character	Standard	Pica	*12×9	80	10	100
		Elite	*12×9	96	12	50
		Condensed	*14×9	137	17	70
	Near Letter Quality	Pica	*24×18	80	10	20
		Elite	*24×18	96	12	24
Italic Cursive Character	Standard	Pica	*16×9	80	10	100
		Elite	*16×9	96	12	50
		Condensed	*18×9	137	17	70
	Near Letter Quality	Pica	*32×18	80	10	20
		Elite	*32×18	96	12	24
Graphic	Single Density Graphic		n×8	480 dot column	–	600 dot column
	Double Density Graphic		n×8	960 dot column	–	490 dot column
	Double Speed Double Density Graphic		n×8	*960 dot column	–	600 dot column
	Quadruple Density Graphic		n×8	*1920 dot column	–	*980 dot column
	640 Dot Graphic		n×8	640 dot column	–	326 dot column
	576 Dot Graphic		n×8	576 dot column	–	294 dot column
	720 Dot Graphic		n×8	720 dot column	–	367 dot column

• Indicates that the specified figure includes half dots.

Disc/Joystick Utility

by Andrew Martin

As my young children use the Amstrad for playing games (and because I want them to continue to use the system), I have developed a program which allows programs to be loaded or run from disc using just a joystick or the cursor keys. In addition, it provides for CAT, ERASE and RENAME functions. I have called the program simply T.BAS.

INITIAL LOADING

Once the program has been checked carefully and saved to tape, you will need to take the following action to get it on to disc:

1. Put the tape in the recorder and a disc in Drive A
2. Type |TAPE then CTLR and ENTER together. This will load the program and do a CAT on the disc.
3. Using the Joystick or the cursor keys, move the cursor across the top line and position on the word 'Break'.
4. Press the FIRE button or the Copy key.
5. Press number 1 on the numeric keypad. This will save T.BAS to disc.
6. Press number 0 on the numeric keypad to re-load and run T.BAS. This will check that the save was successful.

Alternatively, you may type RUN and Enter.

LOADING OR RUNNING A PROGRAM

1. Move the cursor to Load or Run.
2. Press the Fire or Copy button.
3. The cursor will move to the top left corner of the screen to indicate acceptance of the command.
4. Again, with the Joystick or cursor keys, move the cursor to the first character of the program name.
5. Press the Fire or Copy key. The cursor will move to the bottom left of the screen while the program loads or runs.

BREAK

This function simply clears the screen and returns the user to direct command mode.

CAT

Move the cursor to CAT and press the Fire or Copy button.

RENAME

1. Move the cursor to Rename and press the Fire or Copy button.
2. Move the cursor to the first character of the program to be renamed and press the Fire or Copy button.
3. Respond at the bottom centre of the screen with the new program name in full. For example, COMBAT.BAK or T.BAS etc.
4. Press Enter.

The program will change the name and perform a CAT with the cursor positioned at the end of the line containing the new name.

NON-STANDARD FILETYPE (NSF)

The CAT function lists programs/files in the following format: PPPPPPPP.TTT where PPPPPPPP is the program/file name of up to 8 characters and TTT is the file type of up to 3 characters. Both are separated by a full stop.

T.BAS checks the file type and will only accept BAS, BAK or ED. The latter, ED, is a non-standard file type name which I use, but you can change it within T.BAS to whatever name you prefer to use most often.

If a program has a non-standard file type, it can still be loaded/run by specifying the NSF as follows:

1. Move the cursor to NSF and press the Fire or Copy button.
2. Input the non-standard file type name (maximum 3 characters) and press Enter.
3. Move the cursor to Load/Run as required and press the Fire/Copy button.
4. Move the cursor to the first character of the program name to be loaded and press the Fire or Copy button.

Note that file types which consist of three blank characters will not be accepted. If necessary they should be renamed to an acceptable file type.

ERASE

To erase an unwanted file:

1. Move the cursor to Erase and press the Fire or Copy button.
2. Move the cursor to the first character of the program/file name to be erased and press the Fire or Copy button.

T.BAS will erase the file and provide a CAT with the cursor positioned where the erased file had been shown.

CHANGING USER AREA

1. Move the cursor to the location before the user number if a single digit or on to the most significant digit if larger than nine.
2. Press the Fire or Copy button.
3. Use the left or right cursor keys or move the joystick left or right to change the numbers.
4. Press the Fire or Copy button when the correct number is showing.

CHANGING DISC DRIVES

This feature is for owners of two disc drives only, and it is necessary to REMout line 880. It provides a CAT of either drive.

1. Move the cursor to the A or B after the word 'DRIVE'.
2. Press the Fire or Copy button. If the original letter is A, then this will be changed to B and a CAT of that disc provided, and vice-versa.

```
10 '*****************************************************************
*********
20 '              SIMULATED IKON CONTROL-JOYSTICK OR CURSOR KEYS
30 '                          ANDREW MARTIN  6/8/85
40 '*****************************************************************
**********
50 DEFINT a-z:c=12
60 |DISC
70 x=1:y=1
80 KEY 1,"save"+CHR$(34)+"t.bas"+CHR$(13)
90 KEY 0,"run"+CHR$(34)+"t.bas"+CHR$(13):SYMBOL AFTER 32:INK 0,1:INK 1
,26:GOTO 540
100 '************* READ CHARACTER FROM SCREEN ************************
*********
110 GOSUB 250
120 FOR a=1 TO c
130 IF usr=1 THEN LOCATE 7,2 ELSE LOCATE y,x
140 CALL mc
150 xx=PEEK(&97FF)
160 A$=CHR$(xx):TIT$=TIT$+A$
170 IF usr=1 THEN a=12:RETURN
180 Y=Y+1
190 NEXT
200 IF ER=1 THEN RETURN
210 '************* CHECK FOR VALID FILETYPE  ***************************
*********
220 IF  RIGHT$(tit$,3)<>"BAK" AND RIGHT$(tit$,3)<>"BAS" AND RIGHT$(tit
$,3)<>"ED" AND RIGHT$(tit$,3)<>ALTER$ THEN tit$="":can=1:tit1=0:RETURN

230 IF tit$=""THEN tit1=0:can=1:RETURN
240 can=0:RETURN
250 '************* PUT MACHINE CODE INTO RAM ***************************
*********
260 MC=&9700
270 POKE &9700,&CD
280 POKE &9701,&60
290 POKE &9702,&BB
300 POKE &9703,&32
310 POKE &9704,&FF
320 POKE &9705,&97
330 POKE &9706,&C9
340 RETURN
350 '************ PRINT CURSOR AND CHECK FOR LOAD/RUN/PROG SELECTION **
********
360 LOCATE Y,X:CALL &BB81:IF LOD>0 THEN ON LOD GOSUB 600,660,720
370 '************ CHECK FOR POSITION/INPUT ****************************
********
380 IF (JOY(0)=1 OR INKEY(0)=0) AND x=1 THEN GOTO 500 ELSE IF (JOY(0)=
1 OR INKEY(0)=0) THEN X=X-1
390 IF (JOY(0)=2 OR INKEY(2)=0) AND x=24 THEN GOTO 500 ELSE IF (JOY(0)
=2 OR INKEY(2)=0) THEN x=x+1
400 IF (JOY(0)=4 OR INKEY(8)=0) AND y=1 THEN y=79:GOTO 500 ELSE IF (JO
Y(0)=4 OR INKEY(8)=0) THEN y=y-2
410 IF (JOY(0)=8 OR INKEY(1)=0) AND y=80 THEN y=1:GOTO 500 ELSE IF (JO
Y(0)=8 OR INKEY(1)=0) THEN y=y+2
420 IF (JOY(0)=16 OR INKEY(9)=0) AND X=1 AND Y>63 AND Y<70 THEN GOTO 7
90 ELSE IF (JOY(0)=16 OR INKEY(9)=0) AND X=1 AND Y>71 AND Y<80 THEN GO
TO 830
430 IF (JOY(0)=16 OR INKEY(9)=0) AND X=1 AND Y>2 AND Y<7 THEN LOD=1:Y=
1:GOTO 510 ELSE IF (JOY(0)=16 OR INKEY(9)=0) AND X=1 AND Y>9 AND Y<13
THEN LOD=2:Y=1:GOTO 510
```

```
440 IF (JOY(0)=16 OR INKEY(9)=0) AND X=1 AND Y>15 AND Y<21 THEN GOTO 1
010
450 IF (JOY(0)=16 OR INKEY(9)=0) AND tit=1 AND X<>1 AND (y=1 OR y=21 O
R y=61 OR y=41) THEN tit1=1
460 IF (JOY(0)=16 OR INKEY(9)=0) AND y=15 AND x=2 THEN GOTO 920
470 IF (JOY(0)=16 OR INKEY(9)=0) AND X=1 AND Y>22 AND Y<26 THEN LOD=3:
Y=27:GOTO 510
480 IF (JOY(0)=16 OR INKEY(9)=0) AND X=2 AND Y=7 THEN GOTO 880
490 IF (JOY(0)=16 OR INKEY(9)=0) AND X=1 AND Y>58 AND Y<62 THEN LOCATE
 1,2:PRINT CHR$(20):LOCATE 1,1:CAT
500 IF tit=1 AND tit1=1 THEN RETURN
510 IF Y<1 OR y>80 THEN Y=1 ELSE IF X<1 THEN X=1
520 IF y MOD 2=0 THEN y=y+1
530 GOTO 360
540 '************ START OF PROGRAM ************************************
*********
550 MODE 2
560 CAT
570 tit=0:tit1=0:lod=0
580 LOCATE 3,1:PRINT "LOAD":LOCATE 10,1:PRINT "RUN":LOCATE 16,1:PRINT
"BREAK":LOCATE 24,1:PRINT "NSF";SPACE$(33);"CAT  RENAME  ERASE"
590 GOTO 360
600 '************ LOAD A PROGRAM ************************************
*********
610 lod=0:tit=1:GOSUB 510
620 GOSUB 110
630 IF can=1 THEN can=0:GOTO 610
640 LOCATE 1,22
650 CLS:WHILE INKEY$<>"":WEND:LOAD tit$
660 '************ RUN A PROGRAM ************************************
*********
670 lod=0:tit=1:GOSUB 510
680 GOSUB 110
690 IF can=1 THEN can=0:GOTO 670
700 LOCATE 1,22
710 WHILE INKEY$<>"":WEND:RUN tit$
720 '************ CHANGE LAST 3 CHARACTERS ****************************
*********
730 WHILE INKEY$<>"":WEND:LOCATE 29,1:INPUT "NON-STANDARD FILETYPE:",A
LTER$:ALTER$=UPPER$(LEFT$(ALTER$,3)):GOTO 570
740 '************ RENAME PROGRAM ************************************
*********
750 LOD=0:TIT=1:GOSUB 510
760 GOSUB 110
770 IF can=1 THEN can=0:GOTO 750
780 GOTO 710
790 WHILE INKEY$<>"":WEND:LOCATE 1,22:PRINT"FILE NAME TO BE RENAMED ?"
:lod=0:tit=1:GOSUB 510
800 ER=1:GOSUB 110:ER=0
810 WHILE INKEY$<>"":WEND:LOCATE 1,22:PRINT" NEW NAME FOR ";TIT$;:INPU
T N$:|REN,@N$,@TIT$:TIT$="":GOTO 540
820 '************ ERASE A PROGRAM ************************************
*********
830 WHILE INKEY$<>"":WEND:LOCATE 1,22:PRINT"FILE NAME TO BE ERASED ?":
lod=0:tit=1:GOSUB 510
840 ER=1:GOSUB 110:ER=0:IF tit$="" THEN GOTO 530
850 |ERA,@TIT$
860 TIT$="":GOTO 540
870 '********* CHANGE DISC DRIVES ************************************
*********
880 GOTO 550:'(delete this line if using twin disc drives)
890 usr=1:GOSUB 110:usr=0:IF tit$="A" THEN |B:tit$=""  ELSE IF tit$="B
" THEN |A:tit$=""
900 GOTO 540
910 '********* CHANGE USER AREA ************************************
*********
920 er=1:c=2:LOCATE 2,15:GOSUB 110:er=0:c=0:user=VAL(tit$)
930 IF (JOY(0)=4 OR INKEY(8)=0) THEN user=user-1
940 IF (JOY(0)=8 OR INKEY(1)=0) THEN user=user+1
950 IF user=16 THEN user=0
960 IF user=-1 THEN user=15
970 LOCATE 15,2:PRINT user;
980 WHILE INKEY$<>"":WEND:FOR t=1 TO 1000:NEXT:IF (JOY(0)=16 OR INKEY(
9)=0) THEN |USER,user:RUN
990 FOR t=1 TO 200:NEXT:GOTO 930
1000 '******** break ************************************************
****
1010 WHILE INKEY$<>"":WEND:CLS:END
```

JUNIOR JOTTERS

A Column for Young Amstrad Users

A reminder from last month

As promised, I have now found the space to include a special section for our younger readers.

Of course, to continue this page will mean that you will have to keep me well supplied with information about your Amstrad. This could cover small programs you have developed to help you in a particular way (like the one below) or a routine which you have found useful in your programs, and you think would help other people.

Don't forget that I would prefer to receive a tape containing your program as my typing is not very good, and a listing if possible, together with an explanation of what the program is supposed to do and any other information you think may help.

Alternatively, you could tell me for what you use your Amstrad - is it for playing games, for learning about computers or helping you with your homework? Do you use an Amstrad at school?

Whatever you send in, it would be helpful if you could mention your age so that I can give a fair coverage to as many age groups as possible.

I look forward to hearing from you.

The Editor, Junior Jotters.

REVIEWS

QUACK a JACK - Reviewed by Stephen Kerr

This game is one of the best I have seen. After loading it you are shown a demonstration, a list of the characters (which also shows how they look), the keys to use and the hi-scores to beat. At the bottom of each of these screens, it gives you the chance to start at any time. This means you can either play the game, set the level or check if the keys work properly.

The keys you use are simply the cursor keys. On some other games I have played, to use the keyboard instead of joystick, you have to use certain letters, numbers or symbols which becomes very confusing.

Now, back to the actual game itself. It is full of weird and wonderful creatures like Sue's Nose, Vampire Rabbits, Burgers, Computers and Kevin the Prawn. There are nine levels and a number of monsters on each level.

The main character in the game is Red Jack, a pirate. When he discovered that there were no diamonds in the Palace Castle, he went to the Palace Dungeons. A mistake! That was when he discovered the Vampire Rabbits, Spacehoppers, etc. The only thing I thought was strange about this game was that the pirate is supposed to be a duck!

This game works fairly fast, especially on the hardest level. It is very entertaining and it held my interest for a long time, as it has many different nasties and screens.

By the way it is loosely based on the role-playing game Fandonia. I would recommend this original game to almost everyone and even my 5 year-old brother enjoyed it.

ROLAND IN TIME - Reviewed by R. Herbert

This game is quite complicated. For a start, you are in Doctor Who's Tardus, flying through space to the tune of the theme song from the T.V. show. You then get to choose a time zone (there are 10 to choose from, but there are 53 scenes in all). The object of your mission is to collect as many crystals as you can, with 10 lives. But don't be fooled. This game is very difficult, and takes a long time to master, because of all the different screens. There are also flying birds, aliens, rolling wheels, polar bears, penguins, cotton reels, and many other objects out to get you. Very entertaining.

LETTERS

For those other J.J's who have pesky brothers or sisters who get into your most secret programs, here is something to put at the start, which makes sure that only the person who knows the password can run the program. But beware! If someone enters the wrong password, the whole program (and listing) erases from the memory, so you can't list to find out the program. I'm 13 years old, and I think the Amstrad is great for playing games, but I haven't used if for homework yet. (I don't know how to).

```
10 PRINT "ENTER PASSWORD"
20 INPUT P$
30 IF P$ = " your own password " THEN GOTO 60
40 PRINT "WRONG PASSWORD - ACCESS
   DENIED"
50 NEW
60 The rest of your program
```

R. Herbert, Warrnambool, Vic

(Make sure you have got a copy of your program hidden away, so that if you do lose it from memory you can load it again - Ed)

Speech Synthesiser Review

by Kevin Poynton

Speech output is one of the facilities of which most computers are not capable , yet, judging by the demand, is one of the features a great number of end users would like.

The major problems with building speech synthesis into a production model computer would appear to be expense and usage of available memory. Unfortunately the same is true when speech synthesis is available as a peripheral. AWA-Thorn, the Australian distributors of both the 464 and 664 Amstrads, have now released the SSA-1 as their answer to providing speech at a reasonable cost.

How does it stack up? - Read on.

Hardware

The unit is constructed from a gun metal grey plastic similar to that used in construction of the computer itself. For 464 owners, provision is made to connect at the same time as the disc drive. The drive draws a sufficiently low amount of power to allow its use at the same time as the SSA-1.

Two speakers of average quality are provided for sound output.

Sound Quality/ Speech Clarity

During operation, the unit finds difficulty in reproducing some sounds (more of that later) and produces utterances a little like a Cylan warrior from Battlestar Galactica.

However, accepting the limitations mentioned, speech clarity is better than adequate and should be easily understood by most people.

Software

Software for the unit is provided on tape only and is protected. This means most 664 owners will have difficulty in transferring the software to disc. It seems to me to be a rather silly attitude on the part of Amstrad to protect software which will only be of use to anyone with a SSA-1.

The software is relocatable (for experienced programmers) or is located in high memory by default and uses approximately 5k of RAM.

Commands are activated by way of RSX's, eg. |SAY, a$ will speak the text currently in a$.

There are nine commands in all and these should cover most situations. Problems can arise using 'text to speech' conversion. For example, the software can have trouble with words such as 'wall' which sounds fine in the singular but has to be spelt 'worls' to work in the plural. Some words don't seem to work at all, in spite of spending some time trying different spellings. This can usually be overcome by using allophones but this will take some practice by the end user.

The manufacturers have thoughtfully provided some pre-programmed keywords. As an example, |SAY, "1234" will be spoken as "one thousand two hundred and thirty four". Also provided are 'Mr.' - speaks 'Mister', 'Dr.' - speaks 'Doctor' and others. All in all the software provides sufficient tools to produce intelligible speech of reasonable quality.

Documentation

A manual, comprising of 20xA5 pages, gives plenty of indications of how to create good speech. Included is a demonstration program of a talking clock which is worth keying in. The manual is well presented and easy to follow, even for a first-time user.

Summary

Apart from the hobbyist aspect there is good potential for commercial software in games, education and adventure programs. Indeed I saw (and heard) a partially finished 'talking' adventure featuring vampires. The extra dimension given to programs using the speech facility certainly adds something to a program and I feel we can look forward to a great many 'talking' programs. The SSA-1 will provide hours of fun and challenge to Amstrad owners and represents good value for money at around $75.

Seven Colour Graphic Printing for the Amstrads

Closely following the recent release of the SP-1000 dot matrix printer for the Amstrads comes the news that AWA will also release the GP-700. This dot matrix printer features a special printing head that incorporates four logically controllable 'Uni-hammers' and a four colour cassette ribbon.

By combining Black, magenta, cyan and yellow in one ribbon, and having the facility to specify colours in units of just one dot, it is possible to produce 'smudge-free' additional colours.

The SEIKOSHA GP-700 prints from left to right only (uni-directional) at speeds from 38 characters per second to 50 characters per second. The graphics printing has an arbitary combination of 8-dot graphic data with a maximum of 640 dot columns in a line.

The printer is fully compatible with the Amstrads, and should retail at around $850.

A New Book Review

by Robin Nicholas

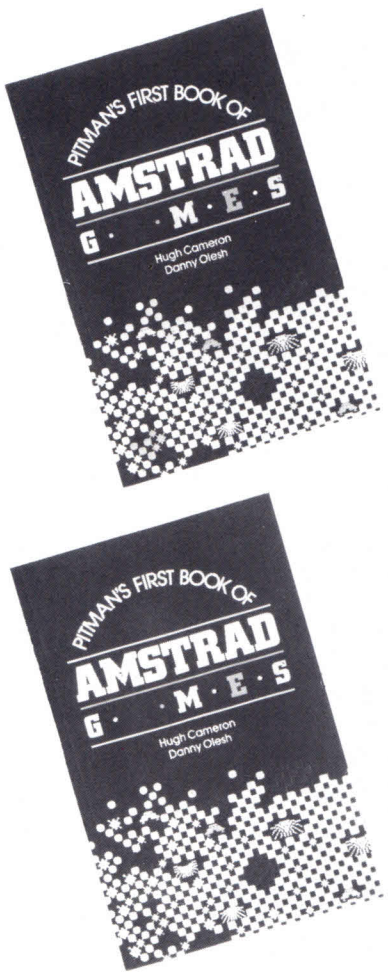
Pitman Publishing in Australia don't publish very many computer books, but when they do, they do it properly. This book is no exception. The cover of the book states "Pitman's First Book of Amstrad Games provides an introduction to writing programs on the Amstrad series of microcomputers". And that is certainly what the book does.

Many authors and publishers have lost track of what BASIC programming is all about. There is an abundance of books consisting of pages and pages of program listings and maybe an explanation of the BASIC keywords, which claim to teach you how to write programs. They appear to have overlooked the fact that there is a lot more to BASIC programming than stringing keywords together on consecutive line numbers.

This book takes a new approach. While still placing a heavy emphasis on typing in program listings, each program listing introduces one or more new programming concepts. Each program is completely different but expands upon the uses of the concepts detailed in the previous programs. I was very impressed. The book teaches BASIC programming rather than BASIC keyword familiarisation.

Enough of the ideology of the book, you probably want to know what's inside.

Well, it consists of twelve chapters. The first chapter, an introductory chapter, goes into some detail on the pitfalls of programming and how to overcome them. It describes some of the most common error codes and what they really mean in terms of your program, and also covers the main procedures for debugging programs that won't run.



The second chapter gives a series of listings for computer graphics. Graphics programs are a favourite of mine because they are short and easy to type in and yet they give you so much satisfaction. Those in the book are good examples of what I mean. I especially enjoyed the last one, "Wave", which produces an effect similar to the arcade game "QIX". The main feature of these graphics listings is that each one is based upon a single mathematical formula. By expanding on these programs it is easy to develop your own graphics programs.

From here on each chapter consists of a program listing complete with notes on what it does, how it does it and how the program has been written. Each chapter introduces BASIC programming concepts in such a way that makes the book of great benefit to newcomers and experienced programmers alike. As if this feature alone is not enough to warrant my recommendation for all Amstrad programmers, the programs themselves are excellent, both in quality and content.

I won't go into detail on all the programs in the book, but just to whet your appetite, here are a few.

Touch Typing Tutor: If you can't touch-type, this program will teach you how, and if you can, this is a great way to improve your speed and accuracy.

Maze Plays: Features include randomly generated maze, level of difficulty, timer, option to play same maze or new one and computer replay.

Triathlon: If you have seen the arcade game "Hyper-Olympics" then you'll know what this one is about. This program shows you how it's done.

Teledex: A telephone directory which works like a phone book not a card file. This program can easily be adapted for many different applications.

That's just a sample. There are also Ski Run, Emergency Landing, Draw Straws, Stop the Invasion, Sound Envelope Generator and Australiana Smith and the Forbidden Temple.

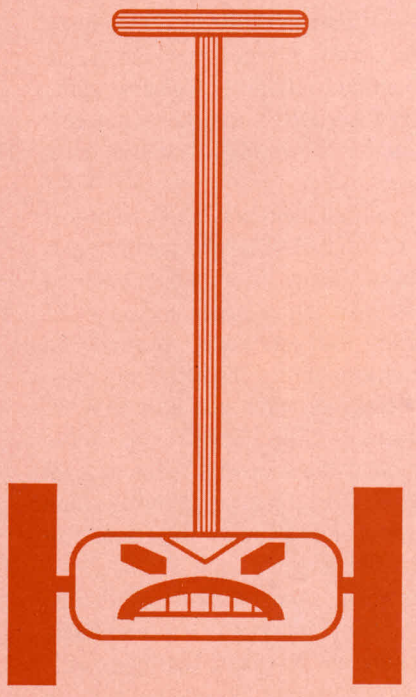
The only criticism I could make would be that there should be more of it. This book may herald the coming of a new style of computer-books. I will eagerly await the release of "Pitman's Second Book of Amstrad Games".

Maniac Mower

A Game from James Brown

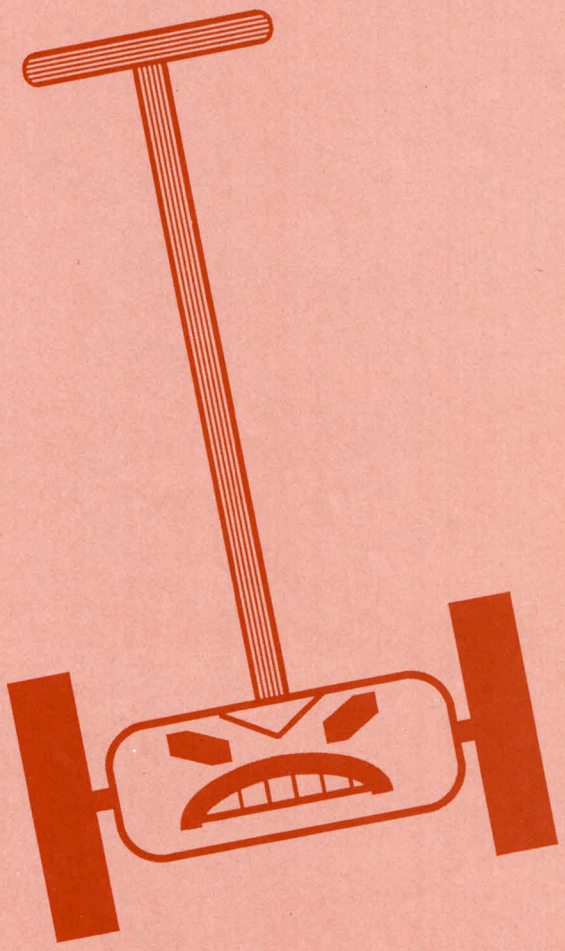
Here's a frustrating game from James Brown in Moranbah, Queensland. The object is to mow as many lawns as possible without hitting any obstacles or running out of fuel. It sounds easy, but once the "mower" is pointed in the right direction it keeps on going!

It can be played using a joystick or the cursor keys where A = UP, Z = DOWN, O = LEFT and P = RIGHT.

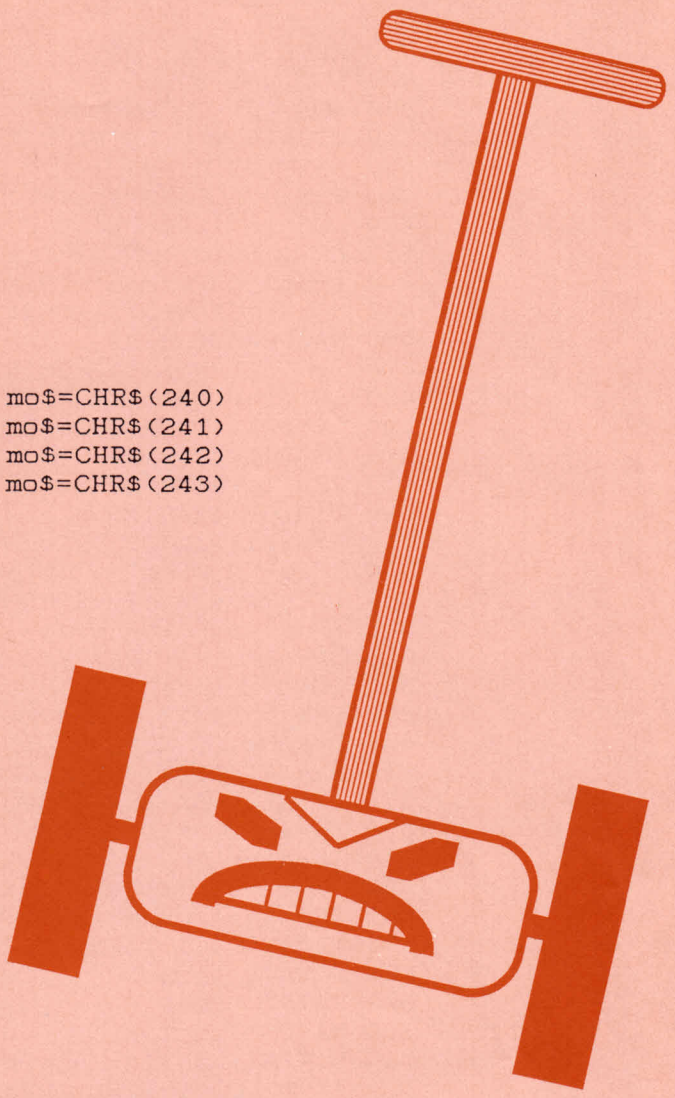


```
10 REM **************
20 REM *              *
30 REM *  MANIAC      *
40 REM *              *
50 REM *  MOWER       *
60 REM * by James.B  *
70 REM *              *
80 REM **************
90 SYMBOL 240,6,137,137,249,249,137,137,6
100 SYMBOL 241,96,145,145,159,159,145,145,96
110 SYMBOL 242,126,129,129,126,24,24,24,126
120 SYMBOL 243,126,24,24,24,126,129,129,126
130 SYMBOL 244,253,253,253,0,223,223,223,0
140 SYMBOL 245,15,31,31,15,3,3,3,7
150 SYMBOL 246,224,240,240,224,128,128,128,192
160 SYMBOL 247,0,15,31,63,63,31,15,0
170 SYMBOL 250,0,240,248,252,252,248,240,0
180 MODE 0:PAPER 1:BORDER 1:CLS:INK 0,1
190 PEN 0:LOCATE 1,2:PRINT "    MANIAC
                                MOWER  "
200 SOUND 4,450,-32767,4,2,0,2
210 FOR x=1 TO 20
220 LOCATE x,10
230 PEN 3:PRINT CHR$(240)
240 FOR p=1 TO 150:NEXT
250 LOCATE x,10
260 PRINT " "
270 NEXT
280 SOUND 132,1
290 BORDER 9:PAPER 1:PEN 0:MODE 1
300 CLS:LOCATE 14,1:PRINT"INSTRUCTIONS"
310 LOCATE 2,2:PRINT"On the holidays you decided to earn someextra money by mowi
ng lawns.You must    avoid the walls,trees,gnomes an
d garden  ponds.If you hit any of these you will  lose a life and get a kick in
the rear."
320 PRINT "You will also lose a life if your fuel  runs out.A bell will sound wh
en your fuel is low.When you have finished mowing  t
he lawn you may proceed to  the next by leaving through the hole in the wall.":L
OCATE 8,15:PRINT"PRESS ANY KEY TO START"
330 a$=INKEY$:IF a$="" THEN GOTO 330
340 sc=0:l=3
350 f=999
```

```
360 MODE 0:PEN 12
370 FOR x=1 TO 20
380 FOR y=1 TO 25
390 LOCATE x,y:PRINT CHR$(207);
400 NEXT:NEXT
410 PEN 3:LOCATE 1,1:PRINT STRING$(60,CHR$(244))
420 LOCATE 1,25:PRINT STRING$(19,CHR$(244))
430 LOCATE 1,1:PRINT"FUEL=";f:LOCATE 12,1:PRINT"LIVES=";l
440 tree$=CHR$(245)+CHR$(246)
450 pond$=CHR$(247)+CHR$(250)
460 RANDOMIZE TIME
470 r1=INT(RND*20):r2=INT(RND*23)
480 IF r1=0 OR r2=0 THEN GOTO 470
490 IF r2<4 THEN GOTO 470
500 LOCATE r1,r2
510 PEN 6:PRINT pond$
520 RANDOMIZE TIME
530 s1=INT(RND*20):s2=INT(RND*23)
540 IF s1=0 OR s2=0 THEN GOTO 530
550 IF s2<4 OR s2=r2 THEN 530
560 IF s1=r1 OR s1=r1+1 THEN 530
570 IF s1=r1-1 THEN 530
580 LOCATE s1,s2
590 PEN 9:PRINT tree$
600 RANDOMIZE TIME
610 a1=INT(RND*20):a2=INT(RND*23)
620 IF a1=0 OR a2=0 THEN 610
630 IF a2<4 THEN 610
640 IF a1=r1 OR a1=r1+1 THEN 610
650 IF a1=s1-1 THEN 610
660 IF a1=s1 OR a1=s1+1 THEN 610
670 IF a1=r1-1 THEN 610
680 LOCATE a1,a2
690 PRINT tree$
700 RANDOMIZE TIME
710 b1=INT(RND*20):b2=INT(RND*23)
720 IF b1=0 OR b2=0 THEN 710
730 IF b2<4 THEN 710
740 IF b1=r1 OR b1=r1+1 THEN 710
750 IF b1=s1 OR b1=s1+1 THEN 710
760 IF b1=a1 OR b1=a1+1 THEN 710
770 IF b1=s1-1 THEN 710
780 IF b1=a1-1 THEN 710
790 LOCATE b1,b2
800 PRINT tree$
810 t1=INT(RND*20):t2=INT(RND*23)
820 IF t2<4 OR t2=r2 OR t2=s2 THEN GOTO 810
830 IF t1=0 OR t2=0 THEN GOTO 810
840 IF t2<4 THEN 830
850 IF t1=r1 OR t1=r1+1 THEN 810
860 IF t1=s1 OR t1=s1+1 THEN 810
870 IF t1=a1 OR t1=a1+1 THEN 810
880 IF t1=b1 OR t1=b1+1 THEN 810
890 IF t1=r1-1 OR t1=s1-1 THEN 810
900 IF t1=a1-1 OR t1=b1-1 THEN 810
910 LOCATE t1,t2
920 PEN 7:PRINT CHR$(248);
930 RANDOMIZE TIME
940 c1=INT(RND*20):c2=INT(RND*23)
950 IF c1=0 OR c2=0 THEN 940
960 IF c2<4 THEN 940
970 IF c1=r1 OR c1=r1+1 THEN 940
```



```
980 IF c1=s1 OR c1=s1+1 THEN 940
990 IF c1=a1 OR c1=a1+1 THEN 940
1000 IF c1=b1 OR c1=b1+1 THEN 940
1010 IF c1=t1 OR c1=t1+1 THEN 940
1020 IF c1=r1-1 OR c1=s1-1 THEN 940
1030 IF c1=a1-1 OR c1=b1-1 THEN 940
1040 IF c1=t1-1 THEN 940
1050 LOCATE c1,c2
1060 PRINT CHR$(248)
1070 mo$=CHR$(240)
1080 x=1:y=24
1090 SOUND 4,450,-32767,4,2,0,2
1100 IF INKEY(27)=0 OR INKEY(75)=0 THEN mo$=CHR$(240)
1110 IF INKEY(34)=0 OR INKEY(74)=0 THEN mo$=CHR$(241)
1120 IF INKEY(69)=0 OR INKEY(72)=0 THEN mo$=CHR$(242)
1130 IF INKEY(71)=0 OR INKEY(73)=0 THEN mo$=CHR$(243)
1140 x2=x:y2=y
1150 IF mo$=CHR$(240) THEN x=x+1
1160 IF x>20 THEN x=20
1170 IF mo$=CHR$(241) THEN x=x-1
1180 IF x<1 THEN x=1
1190 IF mo$=CHR$(242) THEN y=y-1
1200 IF mo$=CHR$(243) THEN y=y+1
1210 IF y>25 THEN y=25
1220 LOCATE x,y
1230 PEN 0:PRINT mo$
1240 LOCATE x2,y2:PRINT" "
1250 IF x=r1 AND y=r2 THEN GOTO 1430
1260 IF x=r1+1 AND y=r2 THEN GOTO 1430
1270 IF y=s2 AND x=s1 THEN GOTO 1430
1280 IF x=a1 AND y=a2 THEN 1430
1290 IF x=a1+1 AND y=a2 THEN 1430
1300 IF x=b1 AND y=b2 THEN 1430
1310 IF x=b1+1 AND y=b2 THEN 1430
1320 IF x=s1+1 AND y=s2 THEN GOTO 1430
1330 IF x=t1 AND y=t2 THEN GOSUB 1430
1340 IF x=c1 AND y=c2 THEN 1430
1350 IF x=20 AND y=25 THEN SOUND 132,1:sc=sc+f:GOTO 350
1360 IF y<4 THEN GOTO 1430 ELSE f=f-1
1370 IF y>24 THEN GOTO 1430
1380 PEN 3:FOR p=1 TO 15:NEXT:LOCATE 6,1:PRINT f
1390 IF f=0 THEN PEN 0:SOUND 132,1:GOTO 350
1400 IF f<100 THEN SOUND 132,1:PRINT CHR$(7):PEN 3:LOCATE 6,25:PRINT " FUEL LOW
";
1410 LOCATE 18,1:PRINT 1
1420 GOTO 1100
1430 SOUND 132,1:FOR j=7 TO 1 STEP -2:SOUND 2,0,50,j,0,0,31:NEXT
1440 CLS:LOCATE 5,10:PRINT "YOU ARE DEAD":l=l-1:IF l<0 THEN GOTO 1460
1450 FOR p=1 TO 1500:NEXT:GOTO 350
1460 RESTORE 1460
1470 SOUND 132,1:FOR n=1 TO 13
1480 READ a
1490 SOUND 1,a,60,15,7
1500 NEXT n
1510 MODE 1:CLS:PRINT"TOO BAD.YOU RAN OUT OF LIVES"
1520 LOCATE 3,3:PRINT "YOU SCORED:-";sc
1530 LOCATE 10,10:PRINT "ANOTHER GAME(Y/N) ?"
1540 a$=INKEY$:a$=UPPER$(a$)
1550 IF a$="Y" THEN GOTO 340
1560 IF a$="N" THEN END
1570 GOTO 1540
1580 DATA 568,0,568,568,568,0,478,506,506,568,568,602,568,999
```



Sorting Methods

by Arthur Harris

This article derives from research into sorting methods over a number of years and implementation on several computers. I still own and use (for various purposes) a TRS-80 Model I, a Sharp PC-1500 and a Sharp PC-1211, as well as "Arnold". The timings quoted have been taken from these machines and, in general, have been kept separate because of the different machine clock speeds and the form of the basic program used (for instance, I ran a compiled version on the TRS-80). The computers listed above form my home "stable".

From my reading over the years there appear to be four methods of sorting lists of items. The selection of the most efficient sort routine for a particular application is a complicated business which basically requires a prior knowledge of the condition of the list before sorting. The condition of a list is a measure of the disorder of that list. The selection of the method used depends on the time taken for the sort, the amount of coding necessary and the overhead required in terms of the extra memory required for additional variables.

The first method mentioned in almost all texts is called the Bubble Sort. This method compares the first and second items in the list and exchanges them if necessary. Then it compares the second and third items and continues comparing adjacent pairs of items until a complete pass of the list has occurred. At each comparison, a swap of the two items occurs, if necessary. Each time a swap is made, either a "flag" is set or a counter is incremented. At the end of a pass, the "flag" or the counter is tested. If the "flag" is set or the counter is greater than zero, another pass is made. Thus one complete pass of the list is made after the list is sorted. This is a waste of time.

The reason for calling this routine the Bubble Sort is obvious, after a little thought. The "heavy" items sink to the bottom, while the "light" ones "bubble" to the top when sorting into ascending order (which is the usual final order). For short lists, the Bubble Sort is reasonably efficient and this is sometimes utilised in some of the many variations in sort routines.

The next most popular method is called the Insertion Sort. There are a couple of variations of this method. The first takes an item from the list (sensibly the smallest) and moves the whole list down to accommodate it. Having moved the smallest item into the first position, this position can then be ignored and the rest of the list dealt with in the same manner. As each position in the list is filled with the correct item, the number of items to be handled reduces. This greatly speeds up the time taken to complete the sort. There is no need to check whether a swap has occurred.

The version that I prefer locates the smallest item in the list and instead of moving the list down to accommodate it, simply exchanges it with the first item. As each position in the list is assigned the correct item, the length of the list still to be dealt with decreases, as above.

The third method is called the Shell Sort (or sometimes the Shell-Metzner Sort). This method is similar to the Bubble Sort except that the distance between the items being compared is increased. Initially the two items are separated by half the length of the list. This sorts the list into two halves. The lower half contains all items smaller than the mid-point of the sorted list and the upper half contains items greater than the mid-point. The spacing of the items to be compared is then reduced to half of its previous value and each of the halves is dealt with in the same manner.

After each complete pass of the list, the interval between items being compared is halved until adjacent pairs of items are being compared. There is no need to check whether any swaps have occurred. Once the complete algorithm has been traversed, the sort will be complete.

The final method is called the Quick Sort. In this method, an item is chosen from the list, by some method, and the list is sorted into two partitions, one containing all items smaller than the chosen one and the other with all items larger than the chose one. Each partition is then dealt with in a similar manner until the whole list has been dealt with.

Variations in this method are based on the algorithm used to choose the item used for comparison. One algorithm locates the median value in the list. Another, which I prefer, simply takes the first item in the list. To enable the program to keep track of the start and finish of the partitions, the method uses an artificial stack. At each stage of the sort, the limits of the partition are initially pushed onto the stack and when the program is ready to deal with it, these limits are popped off again. I have not seen any discussion on how large this stack needs to be to cope with a list of any given size.

Once again, there is no need to check for swaps at any stage, as a complete traverse of the algorithm ensures correct ordering of the list.

The Bubble Sort is the only method that admits to any variations that will speed up the process. A little thought will show that (for sorting into ascending order) the largest item will be carried to the end of the list on the first pass. Thus this position can be ignored on the next and all susequent passes. Similarly on the second pass, the next largest item will be carried to its correct position, and so on. Thus one item of the list can be removed from consideration at each pass. This additional condition can be superimposed on the basic algorithm to improve performance, while still allowing an exit once the list is sorted, if this occurs before the final pair of items are compared.

Another modification, which appears to be the ultimate, is the Modified Bubble Sort. The basis of this method is to keep track of the location of the last swap during a pass through the list. It is assumed, correctly, that the list beyond this point is properly ordered and need take no further part in the process.

As an example of how involved the procedures used for sorting may become, I will quote one that I know of. The list is first divided into short segments of, say, 15 to 20 items. At this size, a Bubble Sort is fairly efficient, so it is used to sort each of the segments. Following this a merge routine is used to merge the sorted segments into the final ordered list.

An example of choosing the sort method to suit the initial condition of the list occurs when maintaining, say, address lists. In order to locate a name and address quickly, the list is usually kept in sorted order. When it is necessary to add a name to the list, these are usually added to the end and the list resorted. In this case, it is known that the list is initially in close to the final ordered condition. The most efficient method of re-ordering the list is considered to be the Insertion Sort.

The final selection of the method used to sort a given list depends on four criteria - the inital condition of the list, the amount of coding required to implement the chosen method, any overheads occasioned by the use of additional variables used exclusively as part of the sorting procedure and finally the time taken to complete the sort.

Timings

(Bub	M/Bub	Insert	Shell	Qsort	Items)	Bub	M/Bub	Insert	Shell	Qsort	Items
122.27	52.45	21.06	4.00	3.41	255	21.06	12.54	5.08	1.26	1.31	125
123.12	51.27	20.13	3.47	3.24	250	18.28	11.36	4.45	1.26	1.35	120
118.18	46.37	19.26	3.37	3.20	245	18.02	10.47	4.21	1.33	1.18	115
115.05	47.07	18.37	3.38	3.20	240	15.16	9.39	4.00	1.12	1.13	110
117.49	45.59	17.54	3.30	3.10	235	14.53	8.58	3.30	1.24	1.15	105
113.25	44.25	17.00	3.33	3.11	230	12.27	7.15	3.10	1.15	1.10	100
107.10	41.22	16.22	4.01	3.06	225	8.53	6.03	2.59	0.57	1.01	95
102.11	38.02	15.41	3.05	3.01	220	10.40	6.21	2.40	0.59	1.00	90
104.50	37.40	15.01	3.05	2.53	215	9.33	5.51	2.24	0.51	0.57	85
102.27	36.00	14.10	3.10	2.47	210	7.54	4.55	2.06	1.01	0.52	80
101.01	34.59	13.40	2.57	2.45	205	6.59	4.03	1.51	0.45	0.43	75
54.10	32.30	12.57	3.19	2.27	200	6.02	3.46	1.37	0.50	0.42	70
45.50	29.40	12.21	3.10	2.37	195	5.52	3.25	1.20	0.41	0.45	65
47.27	28.10	11.44	2.31	2.39	190	4.34	2.50	1.12	0.32	0.37	60
47.30	28.05	11.09	2.31	2.33	185	2.53	2.09	1.01	0.29	0.30	55
44.00	26.20	10.33	2.29	2.10	180	3.02	1.49	0.51	0.30	0.27	50
42.20	25.05	10.01	2.35	2.11	175	2.45	1.42	0.43	0.21	0.20	45
30.21	23.20	9.25	2.25	2.11	170	2.04	1.15	0.33	0.10	0.20	40
35.00	22.01	0.54	2.27	2.03	165	1.20	0.57	0.25	0.10	0.14	35
30.50	21.03	0.20	2.31	2.00	160	1.12	0.43	0.19	0.12	0.15	30
31.45	19.10	7.51	1.57	1.50	155	0.40	0.20	0.14	0.10	0.10	25
27.41	10.11	7.22	1.57	2.02	150	0.19	0.15	0.10	0.00	0.07	20
29.30	17.21	0.51	1.55	1.51	145	0.10	0.10	0.00	0.03	0.05	15
27.45	15.50	0.25	1.49	1.40	140	0.05	0.05	0.03	0.01	0.02	10
24.33	14.51	0.00	1.51	1.40	135	0.01	0.01	0.00	0.01	0.01	5
22.00	13.10	5.31	1.59	1.30	130						

NOTE :- Times are in the format HMM.SS as timed by the PC-1500.

Note:- The following timings are taken from the Amstrad CPC-464 and are in seconds. The only explanation I can offer for the sudden very large jump in Bubble and Modified Bubble Sorts for the 80, 90 and 100 item lists is that the number of swaps carried out must have caused the dreaded "garbage collection" routine to be called, despite having forced its invocation before starting the sort each time.

Bubble	M/Bubble	Insertion	Shell	Quicksort	Items
1.20	1.49	0.64	0.50	0.81	10.00
4.40	2.89	1.86	1.10	1.21	20.00
9.80	6.67	3.84	1.78	1.94	30.00
15.52	11.04	6.72	3.44	2.71	40.00
25.87	16.96	10.15	4.05	4.02	50.00
39.95	24.89	14.62	5.54	4.69	60.00
50.98	34.00	19.44	6.40	5.77	70.00
62.88	197.95	25.20	8.34	7.14	80.00
254.10	226.48	32.07	8.58	8.00	90.00
304.87	264.31	39.81	11.68	9.89	100.00

DISCOUNTED BOOKS FOR SUBSCRIBERS ONLY

Title	Subscriber Price	Normal Price
Pitman's First Book of Amstrad Games	$ 11.65	$ 12.95
The Ins and Outs of the Amstrad	$ 17.95	$ 19.95
Filing Sytems And Databases for the Amstrad	$ 22.45	$ 24.95
Exploring Adventures on the Amstrad	$ 17.10	$ 19.95
A Childs' Guide to the Amstrad Micro	$ 8.95	$ 9.95
Basic BASIC	$ 11.45	$ 12.75
Bells and Whistles on the Amstrad - ***SOLD OUT***	$ 16.15	$ 17.95
Dynamic Games for the Amstrad	$ 17.05	$ 18.95
On the road to Artificial Intelligence	$ 17.95	$ 19.95
Your first Amstrad Program	$ 18.85	$ 20.95
Adventure Games for the Amstrad	$ 17.95	$ 19.95
Brainteasers for the Amstrad	$ 17.95	$ 19.95
The Amstrad Games Book	$ 14.35	$ 15.95
Amstrad Computing	$ 14.35	$ 15.95
Basic Programming on the Amstrad	$ 17.95	$ 19.95
The Working Amstrad	$ 17.95	$ 19.95
40 Educational Games for the Amstrad - ***SOLD OUT***	$ 14.35	$ 15.95
Machine Code for Beginners on the Amstrad	$ 16.15	$ 17.95
The Amstrad CPC464 Advanced User Guide	$ 15.65	$ 19.50
60 Programs for your Amstrad	$ 17.95	$ 19.95
An Amstrad Compendium	$ 20.65	$ 22.95
We also offer a subscription to the English Amstrad User:		
12 Issues from the JAN/FEB 1985 edition	$ 40.00	
12 Issues from the JULY 1985 edition	$ 45.00	
Back issues also available (subject to stock):		
JAN/FEB, MAR, APR, MAY and JUN	$ 4.00 (incl. postage)	
JULY and onwards	$ 4.50 (incl. postage)	
Back Issues of the Australian Amstrad User	$ 3.50 (incl. postage)	

HOW TO ORDER

Send a list of the titles and quantities you require along with a cheque for the total **plus** $5.00 postage and packing (regardless of the quantity you order) to:

STRATEGY PUBLICATIONS

Shop 2, 33 The Centreway, Blackburn Road, Mount Waverley, Victoria, 3149

Bankcard or Mastercard orders accepted by phone on (03) 232 7055

SAVE $450

Q. How many computer magazines would I have to buy to get at least 30 pages of information and program listings for the new Amstrad CPC464?

A. On average, most popular magazines will devote 2½ pages to the AMSTRAD CPC464. This means you will need to buy 12 magazines at a cost of around $40 per month, or **$480 per year!**

Q. Surely there must be a more sensible and cheaper way of getting the information I need?

A. There is.
THE AMSTRAD USER is a brand new monthly publication packed with articles, reviews, listings, hints etc. for Amstrad users only, and costs just **$30 a year**.

Q. How can I get a copy delivered to my home each month with an optional cassette containing all the program listings?

A. By completing and returning this order.

Please send me THE AMSTRAD USER for 12 months
Magazine only: $30 ☐ Magazine and cassette: $70 ☐
Payment by: Cheque ☐ Bankcard or Mastercard ☐
Card number _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ Expiry date .....
Name ..... Phone .....
Address .....
..... Postcode .....

Signed ..... **Please start with Issue No** .....
Return to THE AMSTRAD USER, Suite 4a, 33-45 The Centreway
Blackburn Road, Mt. Waverley, Vic 3149 Tel 03-233 9227

(OVERSEAS PRICES ON APPLICATION TO ABOVE ADDRESS)