THE AMSTRAD USER

Issue No. 10 $3.50 November 1985

- AN EIGHT PAGE PULL-OUT BOOK SUPPLEMENT
- COMPETITION - WIN AN 'AMSTRAD LIBRARY'
- REVIEW of the CPC6128 and the GP-700
- USER GROUP INFORMATION

FOR THE NOVICE & EXPERIENCED USER

Registered by Australia Post – Publication No VBP7017

THE AMSTRAD USER

Issue No. 10 November 1985

CONTENTS

For Tape Subscribers, the programs/routines can be found at these approximate counter readings:
Side 1 - PERCENTS:5, MUSLIST2:26, MENU10:39, MENU13:61, HEADREAD:85
Side 2 - CLOCKPT1:5

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

The single copy price of $3.50 is the recommended retail price only. The subscription rate (for Australia only) is $35.00 for 12 issues of the magazine only, or $75.00 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.

As the number of new subscribers to The Amstrad User continues to increase at a healthy rate, so too do the queries from the new users. Naturally, we do our best to answer the various questions but production of the magazine always takes priority. This means that answers to specific queries may be a bit tardy, although those with stamped and return-addressed envelopes are dealt with first. So please remember, write (don't ring), and if it concerns a piece of software published in this magazine that you can't get to work, send in your tape (or disc) with a return-addressed padded bag and we will have a look at it. I can tell you that 99.9% of the time the problem can be traced to a keying-in error, so do check carefully before adding to Australia Post's revenue.

If you hadn't already noticed, this month's magazine has gone to 40 pages which includes an eight page supplement covering 24 Amstrad titles, 14 of which are new. This is by no means a definitive list as new titles pop up from time to time and take a little while to reach these shores. However, it is the result of CRS's endeavours to bring Amstrad users as up to date as possible. Of course, we will continue to keep an eye open for other titles and provide reviews when possible.

I guess you need not be reminded that Christmas is closing fast on us. It seems to come around more quickly each year! But what you will need to be reminded of is the closing dates for submissions for both the December and January editions. Normal closing dates apply for December, namely 4th November so you are probably too late. Submissions for January 1986 should reach these offices by 18th November to be sure of a chance of getting published.

Finally, if you are looking for a review of the PCW8256 - you won't find it. We didn't get hold of one until the last minute and clearly a machine of this nature warrants more than a rushed and skimpy review. We shall give it our full attention next month.

See you next month,

Ed

A Different Approach to Program Design

from Alan Harris

If you decide to really examine this program you will find hat unlike many other programs this one has been designed ther differently than is usual.

In most instances I find myself thinking in a procedural shion because of the very strict structure of PASCAL ource. I, like many others, originally thought that BASIC ould never really amount to much, but I decided to 'bite the llet' and see just how procedural or linear it could become. his clock is the result and if you decide to type (or tape) the st of it in next month you will see that it is possible to rite a program of around 26k length without a single OTO in it.

The program has been used to help novice programmers to etter understand the principles of program design. I have ed a similar program on other computers for the same rpose even though the language used was not BASIC. It is ot a game or a business program or even a demonstration. ll it is is an example of one way to write a program. The nportant point about it is that it provides a skeleton with hich you can experiment. A number of rules have been plied that enhance the discipline needed to program ccessfully. There are as many ways to program as there are ys on the keyboard, but there are only a few ways to ogram well. If you examine the program carefully you will nd that it is easier to read than most and even without EMs it is self documented. Despite this I have added some mments on the way the program works. Feel free to do th it what you will.

Line 10: sets the colours so that they are suitable for ther a green screen or colour monitor.

Line 20: is a DEFined FuNction that tests a date passed it in the form y,m,d (year, month, day) and checks that ese parameters are legal.

Lines 30-80: take a date in a numeric form of year, onth, and day, and then produce a string that contains a lly expanded date.

Lines 90-100: These two functions calculate the mporary origin for drawing or erasing a hand on the clock ce. It simply moves the origin out from the centre by an nount sufficient to miss the digital display.

Lines 110-120: This function simply draws (or un-draws) the hands of the clock. p1, p2 are the function names, s is the angle and h is the size of the hand. I designed it this way so that less words would be needed in the time sensitive sub-routine that actually runs the clock.

Lines 130-140: Line 130 is a function type that I use quite frequently when I need to provide what I call 'pretty printing' with strings. If you examine it you will find that FN make$(n) will take a number from 0 to 99 and convert it to it's STRing form. Now unfortunately ARNOLD insists on adding a 'space' character in front of the number if it's greater than 0 and a '-' sign if it's negative. When you look at the clock you will see why this FN is used as I don't want the spaces, just the numbers. Let's look at this function in detail:

The rightmost part of this function is (RIGHT$(STR$(N), LEN(STR$(N)-1),2). If STR$(n) has a length of 2 then we use the last byte only and if the length is 3 we use the last two bytes only (That's why we have the -1) and this removes the space that would normally be included. Now that we have a number without the leading space we need to add a leading '0' if it's only 1 byte long and that's all there is to it really. To use the function in a program we could use:

PRINT FN make$(3) or maybe a$=FN make$(4) to assign the function to another variable. You'll see how it is used later on.

Line 140: is used to determine the MODulous of the counter used to adjust the clock during the Update mode. When Update is selected a digital type display in the centre of the clock can be set or adjusted in a similar fashion to that used to set a digital watch. When finished the hands are drawn at a corresponding position and then both the digital and normal clocks keep time.

The modulus function in line 140 is needed so that the update routine knows how many days to expect for the current month. As the UP/DOWN arrow keys are used, the day window will increment from 1 to the last day of the month and then go back to 1 and start again.

To examine how it works we look at he innermost set of parentheses first. '(month-1)*2+1' is used to point to a pair of characters in a table that contains the maximum number of days for each month. We then get the VALue of these two

bytes and that is the number of days+1 in the month.

Lines 150-180: I have often referred to procedural program design when I help novice programmers and use as an example the language PASCAL. I have made references to using PASCAL-like procedures and now find that my sub-conscious has been thinking PASCAL while I have been writing BASIC. I have included in some of the REM lines a comment such as (* comment *). In fact the (* and *) are used in PASCAL to indicate REMarks or comments.

Here we are again thinking PASCAL, for the variable names contained in this line are in fact not variable at all. They are CONSTants, which means that the program does not alter the value of them at all. Once more this form of declaration is compulsory in PASCAL, and very useful in BASIC. The reason I include them at the start of a program is that when they are used a number of times elsewhere and I need to alter them, I only have to do it once at the start of the program where I have declared them, instead of going all through the list to find where they are used. When you write a program do the same. Each time you find the need for a new variable, go back to the earlier lines, decide whether it's a variable or constant and add it to the DECLARATION lines. Make it a habit and it'll pay rich rewards by saving boring and needless effort should you have to make major alterations at a later date.

Let's look at them in detail.

When I use the WHILE/WEND loops I often need to use flags to indicate whether a loop should be entered or not. Therefore I need to SET and RESET the flags at times. I could have used 'LET flag=0' or maybe 'LET flag=1', but I chose to use 'flag=set' and 'flag=reset' since when I read my own programs I prefer them to *sound* like instructions. So I SET a flag to enter a loop and RESET it once I'm finished to get out again. If you say this aloud it does in fact make sense.

'centre.x' and **'centre.y'** are, as their names suggest, the co-ordinates for the centre of the clock.

'max.hour' can be set to 12 or 24, depending on whether you work with 12 or 24 hour days.

'color.1' and **'color.2'** are used to switch paper/pen colours back and forth. Since I use a monochrome display, my colours are naturally black and white.

'trim' is used because contrary to popular belief the computer does not count 50 ticks to the second (well mine doesn't anyway). What I found was that approximately every 900 seconds the clock would be ahead by about 1 second. The idea of the trim value is that it can be compared with the time of the clock and used to skip 1 second every 900 or so seconds.

'second.hand', **'minute.hand'**, and **'hour.hand'** are the lengths of the hands on the clock. They are used to draw the hands when necessary.

'zero', **'one'**, **'twelve'**, and **'sixty'** are obvious and are included simply to make the listing more readable.

'colors' is used as a modulus to rotate the colours of p paper and border.

Lines 200-210: Line 200 contains the stri CHARacter CONSTants. **'reset$'** is used to reset anot string. What could be less ambiguous that using a line su as key.pressed$=reset$. If you read it without the '$' it obvious what the function is.

'setup$'** is a special list of functions I want to occur wh the system is first set up. This will become clear next mo when the SETUP routine has been added. When you exami the display once the clock is running you'll see th functions and switches are selected by the first letter of t function. 'setup$' simulates the operator pressing t sequence of keys **T C G A V P I B and **U**. These cau the program to turn on the switches for T(ick), C(hime G(ong), A(larm), V(ideo), P(aper), I(nk), and B(orde Finally the U(pdate) function is selected to allow the clock be set for the first time.

'prompt$' are exactly that - prompts.

**'finished$'** - think about that. What do you usually do the computer when you've finished typing something Answer is you press the [ENTER] key, which is precise what CHR$(13) is. You don't even have to think about it function as it's obviously to tell the program that you' 'finished' this section of the program.

'switch$()' is simply a method used to indicate on th display whether a particular function is on or off.

Lines 230-240: For those of you fortunate enough have a floppy disc a great deal of time can be saved wi graphic programs if most of, (or in this case all) of th screen is saved on disc. The reason is that in BASIC drawin takes a long time.

The **'discfile'** variable is used to inform if the clockface stored on disc and if so it bypasses the drawing stage an loads the picture into memory. Later you will see how it used but for now, remember that it's used to bypass th drawing section if a discfile called 'clckface.bin' is available.

'flag1' is a secondary flag that is used when 'flag' already in use.

The **'angle'** variable is used within the clockface drawin routine, and also when it's required that the clock hands b un-drawn and drawn. It is incremented by 6 degrees for ever second/minute mark on the clock face.

'setup' is a variable that is used when the program is firs run. It is used to select the characters that comprise setup$ for the initialisation routine, by producing phanton keystrokes as if the operator had done so.

'tick.on, chime.on, gong.on, alarm.on, video.on are switches set by the operator to turn on/off the specifi functions that are optional. The only one that may not b obvious is 'video.on'. This one is used because I believe tha all electronic products suffer the most stress when turned o and off. Therefore when I go to sleep at night I turn off th

display, by setting all inks to 0, which means the clock can continue in operation throughout the night, without shining like a beacon.

'temp' is simply a temporary storage variable.

'update' is used to signal the fact that the update routine is selected, and since the operator is adjusting the clock by means of the digital display in the centre of the clock, the timing routine is prevented from updating the digital display until the operator has finished with the update routine.

'position' is used for identifying the horizontal position of the selected digital display during the update mode. When firs selected the cursor is at the extreme left of the particular display.

'p.max' is set to 4 as a rule but is changed to 3 for updating the date, since only 3 positions are needed.This is to prevent the cursor from messing up the centre of the clock.

'modulus' is used within the update routine for each of the adjustable features, (clock, alarm, date) to ensure that when the arrow keys are used the characters only rotate through the correct sequence of digits for the particular value that is being set at the time.

'second', and **'minute'** are obvious, but we'll discuss **'tock'** in a little more detail. The 'tock' variable is used to adjust the hour hand. It moves the hand 6 degrees at a time to it's new position. Now the purist may think that a new position for the hour hand should be calculated every minute, but the graphics resolution of the 464 makes this a bit of a waste. In addition since the second and minute hands move at 6 degree intervals, by making the hour hand do the same we can use the same block of code for all three. It simply means that the 'tock' increments every 5th of one hour or every 12 minutes.

'year', **'month'** and **'day'** are also obvious, except that the year is not 4 digits long, since we can assume that for now the year is between 1900 and 1999. We only need the last two digits so 'year' will vary between 0 and 99. The modulus for year therefore is 100.

'do.trim' is used to signal that the clock is a second ahead of time and causes a return from the timer routine without any increase. 'do.trim' counts in MODulus (trim), therefore by adjusting the constant 'trim' we can miss a second at regular intervals when we need to correct the time.

'bright' is another ink setting this time it's really a constant of bright white or colour 26.

'paper.color' and **'pen.color'** are used to provide highlighting of the display during update, in particular the function is to reverse the PEN/PAPER settings so the operator can easily see the current ACTION AREA. The secondary function of these two is that the operator can adjust the pen colour by using the 'I' key which will cause the ink 1 colour to rotate from 0 to 26 and then back to 0. In addition the 'P' key causes a similar function to be carried out on paper and border colours.

Now we come to string variables.

All data entered from the keyboard by the operator is by means of a WHILE/WEND INKEY$ routine that I call WINKEY. 'key.pressed$ is assigned a value from INKEY$.

**'work$'** is used as a temporary variable during the Update routine to hold either the 'clock$', 'alarm$', or 'date$'. The contents of the string are displayed at the correct position in the centre of the clock, and once a particular feature of the clock has been completed, the contents of work$ are copied to the correct feature string.

'alarm$'** and **'clock$' are obviously used for display of the digital representation of the time and alarm settings. In addition they are compared every second so that the alarm can be sounded when they are similar.

**'date$'** holds the current date in the form DD/MM/YY. At midnight the day is incremented and the string updated. In addition a feature is available for the user to have messages appear on certain important days. This could be implemented by an array of date$(n). At midnight the array is scanned to see if there is a corresponding date$ in it that agrees with the date just set. For example when DD=01 and MM=01 the computer will display a new year greeting. There is no reason why birthdays and public holidays cannot be added by the user.

'temp$'** and **'am.pm$' are used to ensure the day/night condition is unambiguous.

Line 170: simply defines the attack/decay envelope for the chimes, gong and alarm sounds.

Well there we are that's all the variables defined and explained briefly. We shall now provide some general facts about the program. There are five principal blocks of code:

1. INIT	(Declarations etc.)	10-999
2. Draw	(The clock face).	1000-1400
3. Main	(Key scan and control)	2000-3700
4. Update	(Set and adjust)	4000-4690
5. Clock	(Timing)	5000-6240

(Note. the block of code starting at 4000 has been intentionally removed because of space constraints and will be available at a later date.)

The draw routine uses a single WHILE/WEND routine which increments a counter 'x' from 0 to 360. While x is less than 360 degrees, the sections of the clockface that are to be filled in are drawn. the variable 'angle' is x degrees-90 degrees, because using x on its own would start the clock at the right hand side of the screen instead of the top. The minute marks on the outer edge of the clock occur every 6 degrees so within the main loop another WHILE/WEND loop is entered if x is evenly divisible by 6. There is a condition however where the operator may decide that a 24 hr clock is needed and this is allowed for by using function to calculate the angle depending on the setting of 'max.hour'. The marks for the hours are drawn in the loop which has the command TAG added, so that as the mark is drawn the number of the hour is also drawn beside it.

Summary of operation

Unfortunately we must now leave the descriptive text for this month, so I'll briefly describe the operation of the clock.

By pressing the key that is associated with each function, that particular function can be turned 'On' or 'Off'. If the 'T' key is pressed, it will be found that the 'Tick' is turned 'on' or 'off' depending upon its condition when the key is pressed. The chime, gong and alarm will function similarly. I suggest that to examine all of the features that you set the clock up as follows:

Set the time to 11:59:30:am and the date to 31/12/85
Set the A(larm) for 12:01:00:am and make sure that A(larm), C(hime) and G(ong) are all ON.
Turn the volume control up!

As the time approaches midnight, all features will be used. The chime will sound 16 times and a New Year message will be displayed. Just after midnight, the gong will sound 12 times followed shortly after by the alarm.

Once you have checked all is working, you may decide to set the clock to REAL TIME and leave it on overnight, (remember that Arnold uses only a few watts of power, about one light globe's worth). Press the 'v' for video until the display is blanked out and then turn off the chime and gong functions. If you wish, you can press the 'v' again during the night to check the time. *(I think Alan is an insomniac - Ed)*

Next month we shall provide a function to UPDATE the clock/date while the clock is running and will describe the CLOCK and UPDATE routines in detail. Let me know if you have any problems, either when typing in and running the clock or in understanding how it works. For now, while I am in between jobs, I'll be glad to hear from you. If you wish, you can write to me:

Alan Harris, 28 Leslie Street, Sale, Vic 3850

It will be easier for me if you send a letter using your own word processor and a cassette tape rather than the old fashioned letter system, but make sure you label the cassette with your name and address and stick a couple of $$$ in for my postage back to you.

```
10 EI:INK 0,13:INK 1,0:BORDER 13:
   PAPER 0:PEN 1:WIDTH 80
20 DEF FN date.good(y,m,d)=((y)18
   50 AND y<2000) AND (m>0 AND m<
   13) AND (d>0 AND d<(VAL(MID$("
   32293231323132323132323132",(m-1
   )*2+1,2)))) OR (y/4=INT(y/4) A
   ND d=29 AND m=2))
21 DEF FN time.good(h,m,s,t$)=(h>
   -1 AND h<60) AND (m>-1 AND m<6
   0) AND (s>-1 AND s<60) AND (t$
   ="AM" OR t$="PM")
30 DEF FN dx(y,m,d)=y*365+INT((y-
   1)/4)+(m-1)*28+VAL(MID$("00030
   30608111316192124 26",(m-1)*2+1
   ,2))-((m>2) AND((y AND NOT-4)
   0))+d
40 d$="Friday "+CHR$(0)+CHR$(0)+
   HR$(0)+"Saturday "+CHR$(0)+"S
   nday "+CHR$(0)+CHR$(0)+CHR$(0
   +"Monday "+CHR$(0)+CHR$(0)+CH
   $(0)+"Tuesday "+CHR$(0)+CHR$(
   )+"Wednesday Thursday "+CHR$(
   )
50 DEF FN dx$(y,d$)=MID$(d$,(y-1
   T(y/7)*7)*10+1,10)
60 DEF FN day$(d,y)=STR$(d)+MID$
   "  thstndrdthththththth",(VAL
   RIGHT$(STR$(d),1))+1)*2+1,2)+
   TR$(y)
70 m$="January"+STRING$(2,0)+"Feb
   ruary"+CHR$(0)+"March"+STRING$
   (4,0)+"April"+STRING$(4,0)+"Ma
   y"+STRING$(6,0)+"June"+STRING$
   (5,0)+"July"+STRING$(5,0)+"Aug
   ust"+STRING$(3,0)+"SeptemberOc
   tober"+STRING$(2,0)+"November"
   +CHR$(0)+"December"+CHR$(0)
80 DEF FN date$(y,m,d,d$,m$)=FN d
   x$(FN dx(y,m,d),d$)+MID$(m$,(m
   -1)*9+1,9)+FN day$(d,y)
90 DEF FN x(s)=308+54*COS(90-s*6)
100 DEF FN y(s)=208+54*SIN(90-s*6)
110 DEF FN p1(s,h)=h*COS(90-s*6)
120 DEF FN p2(s,h)=h*SIN(90-s*6)
130 DEF FN make$(n)=RIGHT$("00"+RI
   GHT$(STR$(n),LEN(STR$(n))-1),2
   )
140 DEF FN days.in.month(month,yea
   r)=VAL(MID$("3128313131303131313
   0313031",(month-1)*2+1,2))
150 MODE 2
160 CLS:DEG
170 REM                          (* CONST
   ant ( INTEGER ) *)

180 reset=0:set=1:centre.x=308:cen
   tre.y=208:max.hour=12:color.1=
   1:color.2=0:black=0:white=1:tr
   im=860:second.hand=60:minute.h
   and=60:hour.hand=55:zero=0:one
   =1:twelve=12:sixty=60:colors=2
   7
190 REM                          (* CONST
   ant ( string CHARacter *)

200 reset$="":setup$="TCGAVPIBU":p
   rompt$(1)="Select by first let
   ter A(larm),T(ime),D(ate) or [
   ENTER] to exit":prompt$(2)="Us
   e "+CHR$(240)+CHR$(32)+CHR$(24
   1)+" to adjust and "+CHR$(242)
   +CHR$(32)+CHR$(243)+" to move
```

```
    window, or [ENTER] to exit"
210 finished$=CHR$(13):blank$="Pre
    ss the first letter of the fun
    ction you wish to alter-rememb
    er 'V' for video":switch$(0)="
    OFF":switch$(1)=" ON"
220 REM                    (*VARiab
    les ( INTEGER ) *)

230 discfile=reset:flag1=reset:ang
    le=reset:max.hour=12:setup=set
    :tick.on=reset:chime.on=reset:
    gong.on=reset:alarm.on=reset:v
    ideo.on=reset:update=reset:col
    or.1=white:color.2=black:posit
    ion=reset
231 'discfile=set
240 p.max=4:modulus=reset:second=r
    eset:minute=reset:tock=reset:p
    osition=reset:modulus=reset:ye
    ar=reset:month=reset:day=reset
    :temp=reset:do.trim=set:trim=9
    00:bright=26:paper.color=black
    :pen.color=25:border.color=0
250 REM                    (* VARia
    ble string CHAR *)

260 key.pressed$="":work$="
          ":alarm$="00:00:00:..":cl
    ock$=alarm$:date$="30/06/85":t
    emp$="":temp$(0)="AM":temp$(1)
    ="PM":am.pm$(0)="PM":am.pm$(1)
    ="AM"
270 ENV 1,127,-1,7
280 INPUT "First please type the a
    larm time (HH:MM:SS).
                                  Lik
    e this => 01:34:45:AM or 11:23
    :45:PM";alarm$
290 hour=VAL(LEFT$(alarm$,2)):minu
    te=VAL(MID$(alarm$,4,2)):secon
    d=VAL(MID$(alarm$,7,2)):am.pm$
    =UPPER$(RIGHT$(alarm$,2)):IF F
    N time.good(hour,minute,second
    ,am.pm$) THEN GOTO 300:ELSE PR
    INT "Invalid time ":GOTO 280
300 INPUT"Now please type the date
     in the form DD/MM/YY. For exa
    mple 01/11/85 indicates   1st
    December 1985 (Remember to use
     the '0' if needed)";date$
310 day=VAL(LEFT$(date$,2)):month=
    VAL(MID$(date$,4,2)):year=VAL(
    RIGHT$(date$,2))
320 IF FN date.good(year+1900,mont
    h,day) THEN 330:ELSE PRINT "In
    valid Date ":GOTO 300
330 PRINT "Please remember before
    typing the time that you will
    be required to set it about2 m
    inutes ahead of time to allow
    for the clock face to be drawn
    .."
340 INPUT "Type the time in simila
    r form to that for the alarm (
    HH:MM:SS)";clock$
350 hour=VAL(LEFT$(clock$,2)):minu
    te=VAL(MID$(clock$,4,2)):secon
    d=VAL(MID$(clock$,7,2)):am.pm$
    =UPPER$(RIGHT$(clock$,2)):IF F
    N time.good(hour,minute,second
    ,am.pm$) THEN 360:ELSE PRINT "
    Invalid Time":GOTO 340
360 CLS:PRINT "       When clock is
     finished press <ENTER> when t
    ime catches up to clock
    "
370 tock=hour*5+INT(minute/12):hou
    r=hour-1
380 IF t$="AM" THEN am.pm=1:ELSE a
    m.pm=0
390 MODE 2
1000 REM ********************** Dr
    aw the clock face ***********
    ***************
1010 IF discfile=reset THEN flag1=s
    et
1020 WHILE flag1=set
1030     angle=90-x
1040     ORIGIN centre.x+50*SIN(ang
    le),centre.y+50*COS(angle)
1050     DRAW -50*SIN(angle),0
1060     ORIGIN centre.x+160*SIN(an
    gle),centre.y+160*COS(angle)
1070     DRAW -60*SIN(-angle),0
1080     IF x MOD 6=0 THEN flag=set
1090     WHILE flag=set
1100       ORIGIN centre.x,centre.y
1110       PLOT 120*COS(angle),120*
    SIN(angle),1
1120       flag=reset
1130     WEND
1140     IF x MOD 15*24/max.hour=0
    THEN flag=set
1150     WHILE flag=set
1160       TAG
1170       ORIGIN centre.x+120*COS(
    angle),centre.y+120*SIN(angle)
1180       DRAW  10*COS(angle),10*S
    IN(angle),1
1190       ORIGIN centre.x-13+145*C
    OS(angle),centre.y+5+145*SIN(a
    ngle)
1200       PRINT x/(15*24/max.hour)
    ;
1210       flag=reset
```

```
1220        TAGOFF
1230      WEND
1240      x=x+1
1250      IF x>360 THEN flag1=reset
1260 WEND
1270 ON ERROR GOTO 1280:GOTO 1290
1280 RESUME 1360
1290 :DISC
1300 WHILE discfile=reset
1310      SAVE"clckface",b,&C000,&4
     000,&C000
1320      CLS
1330 PRINT"Remove the (') from line
      231 then type 'SAVE ''CLOCK1'
     '. Then RUN ''CLOCK1 to ensure
      the clock has been saved OK"
1340      STOP
1350 WEND
1360 ON ERROR GOTO 0
1370 IF discfile=set THEN MODE 2:LO
     AD"a:clckface.bin"
1380 ORIGIN FN x(second),FN y(secon
     d):DRAW FN p1(second,second.ha
     nd),FN p2(second,second.hand),
     white
1390 ORIGIN FN x(minute),FN y(minut
     e):DRAW FN p1(minute,minute.ha
     nd),FN p2(minute,minute.hand),
     white
1400 ORIGIN FN x(tock),FN y(tock):D
     RAW FN p1(tock,hour.hand),FN p
     2(tock,hour.hand),white
1410 LOCATE 27,1:PRINT FN date$(yea
     r+1900,month,day,d$,m$)
1420 LOCATE 34,12:PRINT alarm$;
1430 LOCATE 35,11:PRINT date$;
1440 LOCATE 34,13:PRINT clock$
1450 day=day-1:month=month-1
2000 REM ********************* key sc
     an routine for feature control
      ***************
2010 REM ++++++++++++++++++++++ Sub
      routine 2000 start ++++++++++
     ++++++++++++++++
2020 key.pressed$=reset$
2030 WHILE key.pressed$<>CHR$(255)
2040 key.pressed$=UPPER$(INKEY$)
2050 IF setup<10 THEN flag=set
2060 WHILE flag=set
2070   key.pressed$=MID$(setup$,set
     up,1)
2080   flag=reset
2090   setup=setup+1
2100 WEND
2110 WHILE key.pressed$="T"
2120      IF tick.on=reset THEN ti
     ck.on=set:
     LSE tick.on=reset
2130      DI:LOCATE 1,1
2140      PRINT "(T)ick is  ";s
     ch$(tick.on):EI
2150      key.pressed$=reset$
2160 WEND
2170 WHILE key.pressed$="C"
2180      IF chime.on=reset THEN
     himе.on=set:

     LSE chime.on=reset
2190      DI:LOCATE 1,2
2200      PRINT"(C)hime is "swit
     $(chime.on):EI
2210      key.pressed$=reset$
2220 WEND
2230 WHILE key.pressed$="G"
2240      IF gong.on=reset THEN
     ng.on=set:

     LSE gong.on=reset
2250      DI:LOCATE 67,1
2260      PRINT "(G)ong is  "swi
     h$(gong.on):EI
2270      key.pressed$=reset$
2280 WEND
2290 WHILE key.pressed$="A"
2300      IF alarm.on=reset THEN
     larm.on=set:

     SE alarm.on=reset:alarm.acti
     =reset:chime=reset:gong=rese
2310      DI:LOCATE 67,2
2320      PRINT "(A)larm is "swi
     h$(alarm.on):EI
2330      key.pressed$=reset$
2340 WEND
2350 WHILE key.pressed$="S"
2360      restart=set
2370      key.pressed$=reset$
2380 WEND
2390 WHILE key.pressed$="P"
2400   paper.color=(paper.color+1)
     MOD colors
2410   key.pressed$=reset$
2420   INK 0,paper.color
2430   LOCATE 20,25
2440   PRINT "P(aper) ";USING"##";
     aper.color
2450 WEND
2460 WHILE key.pressed$="I"
2470   pen.color=(pen.color+1) MOD
     colors
2480   key.pressed$=reset$
2490   INK 1,pen.color
2500   LOCATE 33,25
2510   PRINT "I(nk)   ";USING"##";
     en.color
```

```
2520 WEND
2530 WHILE key.pressed$="B"
2540   border.color=(border.color+1
     ) MOD colors
2550   BORDER border.color
2560   key.pressed$=reset$
2570   LOCATE 48,25
2580   PRINT"B(order) ";USING"##";b
     order.color
2590 WEND
2600 IF key.pressed$="V" THEN flag=
     set
2610 WHILE flag=set
2620   WHILE video.on=set
2630     INK 0,black
2640     INK 1,black
2650     BORDER black
2660     video.on=reset
2670     flag=reset
2680   WEND
2690   WHILE flag=set
2700     INK 0,paper.color
2710     INK 1,pen.color
2720     BORDER border.color
2730    flag=reset
2731   video.on=set
2740   WEND
2750  key.pressed$=reset$
2780 WEND
3540   IF key.pressed$=finished$ TH
     EN flag=set
3550   WHILE flag=set:REM (* End of
      update procedure *)
3560     DI:LOCATE 1,24
3570     PAPER black
3580     PEN white
3590     PRINT blank$:EI
3600     flag=reset
3610   WEND
3620   IF key.pressed$=finished$ TH
     EN key.pressed$=reset$:EVERY 5
     0 GOSUB 5000
3630   key.pressed$=reset$
3640 WEND:REM                  (* End
      of procedure Keyscan *)
3650 REM
3660 REM
3670 REM (* End of main procedure *
     )
3680 REM
3690 REM
3700 END
3990 REM
                (* This is the adju
     st procedure *)
4700 REM ************************
     Clock routine ***************
     ***************
```

```
5000 IF do.trim=reset THEN do.trim=
     set:RETURN:REM (* Adjust for f
     requency *)
5010 do.trim=(do.trim+1) MOD trim
5020 ORIGIN FN x(second),FN y(secon
     d)
5030 DRAW FN p1(second,second.hand)
     ,FN p2(second,second.hand),bla
     ck
5040 second=(second+one) MOD sixty
5050 IF tick.on=set AND chime=reset
      AND gong=reset THEN SOUND 1,2
     00,1
5060 ORIGIN FN x(second),FN y(secon
     d)
5070 DRAW FN p1(second,second.hand)
     ,FN p2(second,second.hand),whi
     te
5080 MID$(clock$,7,2)=FN make$(seco
     nd)
5090 LOCATE 34,13
5100 PRINT clock$
5110 IF second=reset THEN flag=set
5120 WHILE flag=set:REM (* Incremen
     t minute *)
5130   ORIGIN FN x(minute),FN y(min
     ute)
5140   DRAW FN p1(minute,minute.han
     d),FN p2(minute,minute.hand),b
     lack
5150   minute=(minute+1) MOD sixty
5160   ORIGIN FN x(minute),FN y(min
     ute)
5170   DRAW FN p1(minute,minute.han
     d),FN p2(minute,minute.hand),w
     hite
5180   MID$(clock$,4,2)=FN make$(mi
     nute)
5190   LOCATE 34,13
5200   PRINT clock$
5210   IF minute MOD 12>zero THEN f
     lag=reset
5220   WHILE flag=set:REM (* Increm
     ent tock 6 degrees [ 1/5 hour
     ] *)
5230     ORIGIN FN x(tock),FN y(toc
     k)
5240     DRAW FN p1(tock,hour.hand)
     ,FN p2(tock,hour.hand),black
5250     tock=(tock+1) MOD sixty
5260     ORIGIN FN x(tock),FN y(toc
     k)
5270     DRAW FN p1(tock,hour.hand)
     ,FN p2(tock,hour.hand),white
5280     flag=reset
5290   WEND
5300   IF minute=reset THEN flag=se
     t
```

```
5310    WHILE flag=set:REM
                (* Increment hour a
    nd set GONG *)
5320      hour=(hour+1) MOD twelve
5330      MID$(clock$,1,2)=FN make$(
    hour+1)
5340      LOCATE 34,13
5350      PRINT clock$
5360      IF gong.on=set THEN gong=h
    our+1
5370      IF hour<>11 THEN flag=rese
    t
5380      WHILE flag=set:REM
    (* Set AM/PM *)
5390        am.pm=(am.pm+1) MOD 2
5400        MID$(clock$,10,2)=am.pm$
    (am.pm)
5410        LOCATE 34,13
5420        PRINT clock$
5430        IF am.pm=reset THEN flag
    =reset
5440        WHILE flag=set:REM
                        (* Dawn
    of a new day *)
5450          day=(day+1) MOD (FN da
    ys.in.month(month+1,year))
5460          MID$(date$,1,2)=FN mak
    e$(day+1)
5470          LOCATE 35,11
5480          PRINT date$
5490          LOCATE 27,1
5500          PRINT "
                    "
5510          LOCATE 27,1
5520          PRINT FN date$(year+19
    00,month+1,day+1,d$,m$)
5530          IF day <>0 THEN flag=r
    eset
5540          WHILE flag=set:REM
                        (* Dawn of
     a new Month *)
5550            month=(month+1) MOD
    12
5560            MID$(date$,4,2)= FN
    make$(month+1)
5570            LOCATE 35,11
5580            PRINT date$
5590            LOCATE 27,1
5600            PRINT "
                      "
5610            LOCATE 27,1
5620            PRINT FN date$(year+
    1900,month,day,d$,m$)
5630            IF month<>0 THEN fla
    g=reset
5640            WHILE flag=set:REM
                        (* Dawn of Ne
    w Year's Day *)
5650              year=(year+1) MOD
    100
5660              MID$(date$,7,2)=FN
     make$(year)
5670              LOCATE 35,11
5680              PRINT date$
5690              LOCATE 27,1
5700              PRINT "
                        "
5710              LOCATE 27,1
5720              PRINT FN date$(yea
    r+1900,month+1,day+1,d$,m$)
5730              LOCATE 1,25
5740              PRINT "
                   Happy New Year --
    Everyone !!
                "
5750              flag=reset
5760            WEND
5770          WEND
5780        WEND
5790      WEND
5800    WEND
5810  WEND
5820  IF second-tock<3 THEN flag=set
5830  WHILE flag=set:REM (* Redraw h
      our hand if close to second ha
      nd *)
5840    ORIGIN FN x(tock),FN y(tock)
5850    DRAW FN p1(tock,hour.hand),F
      N p2(tock,hour.hand),white
5860    flag=reset
5870  WEND
5880  IF second-minute<3 THEN flag=s
      et
5890  WHILE flag=set
5900    ORIGIN FN x(minute),FN y(min
      ute)
5910    DRAW FN p1(minute,minute.han
      d),FN p2(minute,minute.hand),w
      hite
5920    flag=reset
5930  WEND
5940  IF alarm$=clock$ AND alarm.on=
      set THEN alarm.active=set
5950  IF alarm.active=set THEN flag=
      set
5960  WHILE flag=set
5970      SOUND 1,16,100,15,1,15:SOUN
      D 2,32,100,15,1,15
5980      chime=reset
5990      dong=reset
6000      flag=reset
6010  WEND
6020  IF chime.on=set AND chime>zero
       THEN flag=set:
                                ELSE IF
       chime.on=set AND chime=zero T
      HEN RESTORE 6260
6030  WHILE flag=set
6040    READ shape1,note1,note2:shap
      e2=shape1
6050    IF note1>0 THEN SOUND 1,note
      1,100,shape1,1,shape2:
                SOUND 2,note2,100,shap
      e1,1,shape2
6060    chime=chime-1
6070    flag=reset
6080  WEND
6090  IF gong>zero AND alarm.active=
      reset AND ding=set THEN flag=s
      et:                       ELSE IF go
      ng>zero AND alarm.active=reset
       THEN ding=set
6100  WHILE flag=set
6110    SOUND 1,358,100,15,1,15:SOUN
      D 2,179,100,15,1,15
6120    gong=gong-1
6130    ding=reset
6140    flag=reset
6150  WEND
6160  IF chime.on=set THEN flag=set
6170  WHILE flag=set
6180    IF minute=14 AND second=55 T
      HEN chime=5
6190    IF minute=29 AND second=50 T
      HEN chime=10
6200    IF minute=44 AND second=45 T
      HEN chime=15
6210    IF minute=59 AND second=40 T
      HEN chime=20
6220    flag=reset
6230  WEND
6231  IF month=0 AND day=1 THEN LOCA
      TE 1,25:PRINT SPACE$(80);
6240  RETURN
6250  REM ********************** Ch
      ime data storage **********
      **********
6260  DATA 15,358,179,15,284,142,15,
      319,159,15,478,239,0,0,0,15,47
      8,239,15,319,159,15,284,142,15
      ,358,179,0,0,0,15,284,142,15,3
      19,159,15,358,179,15,478,239,0
      ,0,0,15,478,239,15,319,159,15,
      284,142,15,358,179,0,0,0
6270  END
6280  RETURN
```

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		
Chris Craven	Canowindra	(063) 44 1150
Bruce Jones	Coffs Harbour	(066) 52 8334
Trevor Farrell	Coolah/Mudgee area	(063) 77 1374
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
Reuben Carlsen	North Sydney	(02) 957 2505
Stephen Gribben	Singleton	(065) 72 2732
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
Vic		
David Carbone	Burwood	(03) 29 4135
Rod Anderson	Camperdown	(055) 93 2262
Paul Walker	Heathmont	(03) 729 8657
Terry Dovey	Horsham	(053) 82 3353
Andrew Portbury	Leongatha	(056) 62 3694
Ron Butterfield	Leopold	(052) 50 2251
Sue Kelly	Manangatang	(050) 35 1402
Keith McFadden	Numurkah	(058) 62 2069
Mrs. G. Chapman	South Clayton	(03) 551 4897
Lindsay Parker	Wandin North	(059) 64 4837
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		
Andrew Banfield	Launceston	(003) 44 3181
Conal McClure	Scottsdale	(003) 52 2514
NT		
G.P. Heron	Tiwi	(089) 27 8814

News from around the States

Lack of space last month prohibited inclusion of correspondence received from a number of groups around Australia. This month the situation is redressed although a little belatedly.

AMSWEST - the user group north of the river in Perth - was invited to man a stall at the recent Electronics Exhibition (August to be exact).

They assisted AWA in running a competition to win an Amstrad, and took the opportunity to sell copies of their locally produced newsletter and distribute leaflets advertising The Amstrad User *(Thanks - Ed)*

The show itself was attended by over 85000 visitors, an increase of 20% on the previous year. Mrs. Thelma Ardron (Secretary of Amswest) reports that there was a bank of ten Amstrads available for visitors to try the many pieces of software on show, with a further three 464's to one side demonstrating printers and a modem.

"As you can imagine, we were very busy talking to people... but we had to do more talking at the next meeting night to deal with all the new enquiries".

Obviously there is a lot to be said to linking up with AWA or dealers at shows or exhibitions in terms of spreading the news about Amstrad User Groups.

Chris Sowden, President of Amstrad Computer Club Inc. in Adelaide, reports that the membership continues to grow and now exceeds 70 with some as far away as 500 kms.

The club has moved and now meets at the Unley High school. This makes it a little more central to the city and, and as a point for other groups, they obtained these premises by offering free membership to the School's students.

In addition to normal activities, the club runs a successful BASIC programming course for beginners, hardware and software reviews and a tape to disc conversion service. Future activities may include a basic electronics course, a machine code programming course and information transfer by phone and radio.

The club is incorporated and runs under a professionally prepared legal constitution, copies of which are available to other groups at a nominal charge.

JUNIOR JOTTERS

A Column for Young Amstrad Users

LETTERS

I am a Junior Jotter and I have a fairly short Lotto program, not just for JJ's but for adults too.

I am 12 years old and I am the eldest of 5 children. We all enjoy playing games on the Amstrad, and the ones we enjoy most are Moon Buggy, Defender and Harrier attack. I also like playing a spelling program because I am not a very good speller.

I write a lot of programs and at the moment I am writing a Haunted House program. What you do is go through a four storey house and pick up gold coins, fall through a trap door and a lot more.

Michael Spozetto, Medina, WA

When you have finished your Haunted House program, spirit me a copy and, if it is good enough, we may be able to publish it.

LOTTO

```
1   CLS
10  LOCATE 1,8: PRINT"What is your name?"
20  INPUT N$: CLS
30  LOCATE 1,6: PRINT"Well ";N$;" You and I"
40  LOCATE 1,10: PRINT"are about to get lucky."
50  FOR T=1 TO 2500: NEXT
60  CLS
70  A=INT(RND*45)
80  B=INT(RND*45)
90  IF A=B THEN 80
100 C=INT(RND*45)
110 IF C=A OR C=B THEN 100
120 D=INT(RND*45)
130 IF D=A OR D=B OR D=C THEN 120
140 E=INT(RND*45)
150 IF E=A OR E=B OR E=C OR E=D THEN 140
160 F=INT(RND*45)
170 IF F=A OR F=B OR F=C OR F=D OR F=E
    THEN 160
180 LOCATE 1,5:PRINT"Well ";N$;"Here are "
190 LOCATE 4,8:PRINT"your Lotto Numbers"
200 LOCATE 1,10:PRINT A; B; C; D; E; F
210 LOCATE 1,14:PRINT"Feeling lucky ";N$
220 LOCATE 1,14:PRINT"How about some more
    numbers ?"
230 LOCATE 1,18:PRINT"Y/N"
240 INPUT A$
250 IF A$="Y" OR A$="y" THEN 60 ELSE 260
260 CLS:LOCATE 1,6:PRINT"Bye ";N$;" Good l
270 LOCATE 1,20:PRINT"Don't forget my sha
    END
```

In issue No. 8, September of The Amstrad User, there an article in the Junior Jotters section concerning Passv for programs by R. Herbert of Warrnambool, Victoria.

The program is very useful but had to overcome problem of the program being erased if the wrong pass was entered.

My version is much shorter and easier.

```
10  CLS
20  INPUT "ENTER PASSWORD";P$
30  IF P$="Your own password" THEN GOTO 40
    ELSE END
40  The start of your program
```

As you can see, the good thing about this small progra that it will not be erased when the incorrect password been entered. I place it at the beginning of my progr (mainly games) to stop my family and friends playing th

I use my Amstrad very much and do everything on it. soon as I get home from school, I quickly do my homew so I can use my magnificent Amstrad.

I am also another who would like to see a section of magazine atributed to the achievements of people at t games on the Amstrad.

I am a complete Amstrad Addict, your publication is a g help with my understanding of the uses of my Amstrad.

David Fennessy, Modbury North,

Unless you have already noticed, the first of our Amstr Achievers to respond appear in the 'Hall of Fame' on anot page in this month's magazine.

PERCENTAGES

Last month Brendan Piner gave us his version of a progra to draw a Histogram. This month we include his progra which calculates and displays a percentage from a fracti that you have entered then draws a pie chart showing t percentage.

```
10 REM ****************************
   **************
20 REM ** Percentages ** By ** Br
   endan Piner **
30 REM ******* Amstrad CPC464 ***
   * (1985)******
40 REM ****************************
50 MODE 1:BORDER 0:INK 0,0:PAPER
```

```
    0:INK 1,24:INK 2,0:INK 3,0:CLS
 60 PEN 3:LOCATE 1,25:PRINT" Perce
    ntages";
 70 FOR x%=0 TO 312
 80 FOR y%=0 TO 16 STEP 2
 90 IF TEST(x%,y%) THEN PLOT 11+x%
    *3,348+y%*3,3:PLOT 11+x%*3,350
    +y%*3,1:PLOT 11+x%*3,352+y%*3,
    1:PLOT 11+x%*3,354+y%*3,1
100 NEXT:NEXT
110 LOCATE 1,25:PRINT"Amstrad  CPC
    464 ";
120 LOCATE 17,6:PEN 1:PRINT" For t
    he"
130 FOR x%=0 TO 304
140 FOR y%=0 TO 16 STEP 2
150 IF TEST(x%,y%) THEN PLOT 205+x
    %,242+y%*2,2:PLOT 205+x%,244+y
    %*2,2:PLOT 205+x%,246+y%*2,1
160 NEXT:NEXT
170 a=0:PEN 3
180 LOCATE 1,25:PRINT" Written  By
        ":PEN 2
190 FOR x%=0 TO 220 STEP 2
200 FOR y%=0 TO 16 STEP 2
210 a=a+2
220 IF TEST(x%,y%) THEN PLOT (x%+a
    )+105,172+y%*2,3:PLOT (x%+a)+1
    05,174+y%*2,3
230 NEXT:a=a-16:NEXT
240 LOCATE 1,25:PEN 3:PRINT " Bren
    dan Piner!":PEN 3
250 FOR x%=0 TO 312
260 FOR y%=0 TO 16 STEP 2
270 IF TEST (x%,y%) THEN PLOT x%*2
    .5+15,52+y%*3,3:PLOT x%*2.5+15
    ,54+y%*3,3:PLOT x%*2.5+15,56+y
    %*3,3:PLOT x%*2.5+15,58+y%*3,1
280 NEXT:NEXT
290 LOCATE 1,25:PRINT SPC(30)
300 INK 3,5:PEN 1:INK 2,6:LOCATE 8
    ,24:PRINT "Press <SPACE> to co
    ntinue"
310 j$=INKEY$:IF (j$<>" ") THEN 31
    0
320 SOUND 1,500,20,7:GOTO 670
330 REM ** main part of program **
340 MODE 0:BORDER 0:INK 0,0:PAPER
    0:CLS:INK 1,22:INK 2,15:INK 3,
    14:PEN 1:PRINT "Percentages"
350 INK 11,12:PEN 11:PRINT:PRINT"G
    ive Fraction "
360 PEN 2:PRINT:PRINT:INPUT "Nomin
    ator";n
370 SOUND 1,500,20,7:PEN 3:PRINT:I
    NPUT "Denominator";d
380 SOUND 1,500,20,7:INK 6,6:PEN 6
    :PRINT:PRINT "That's";INT(n/d*
    100);"%"
390 FOR t=1 TO 5000:NEXT
400 MODE 0:INK 15,19:PEN 15:PRINT"
        Pie Chart"
410 PRINT:INK 1,15:PEN 1:PRINT"
                "
420  FOR a=1 TO 360
430 DEG
440 INK 2,5:PLOT 400,150,2
450 PLOT 400+148*COS(a),150+147*SI
    N(a)
460 NEXT
470 FOR a=1 TO 360
480 DEG
490 INK 12,6:PLOT 400,150,12
500 DRAW 400+144*COS(a),150+144*SI
    N(a)
510 NEXT
520 f=INT(n/d*100)
530 d=INT(f/100*360)
540 LOCATE 2,12:PEN 4:PRINT f;"%"
550 FOR a=1 TO d
560 DEG
570 INK 9,9:PLOT 400,150,9
580 DRAW 400+144*COS(a),150+144*SI
    N(a)
590 NEXT
600 INK 8,21:PEN 8:LOCATE 2,21:PRI
    NT"Press"
610 PRINT:INPUT"[ENTER]",h
620 SOUND 1,500,20,7:CLS:MODE 1:IN
    K 1,26:PEN 1
630 PRINT "Would you like the comp
    uter to work out another perce
    ntage.<y or n>"
640 k$=UPPER$(INKEY$)
650 IF k$="" OR (k$<>"Y" AND k$<>"
    N") THEN 640
660 IF k$="Y" THEN GOTO 340 ELSE P
    RINT:PRINT:PRINT"Bye bye !!":E
    ND
670 MODE 1:INK 0,0:PAPER 0:BORDER
    12:INK 1,26:INK 2,23,17:INK 3,
    6:PEN 2:LOCATE 14,1:PRINT "Per
    centages"
680 PRINT:PRINT
690 PEN 3:PRINT:PRINT"          By Br
    endan Piner  (1985)"
700 PRINT:PRINT
710 PEN 1:PRINT:PRINT:PRINT"Percen
    tages is a program which works
     out a percentage from a fract
    ion. It asks  for a fraction.Y
    ou enter the  fraction  by typ
    ing the nominator, then  press
        'ENTER'. Then you enter th
    e denomintor   Then press 'ENT
    ER' again."
```

The Learning Centre

An Introduction to Music - Part Two

from Peter Campbell

Last month I gave a little of the background to electronic sound. Now let's tackle the problem of deciphering musical notation.

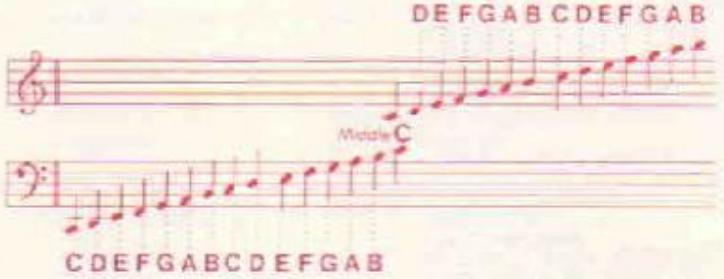
Staves

The sight of musical notation can be rather daunting for those unfamiliar with it. Don't let this put you off. It's not so hard as it might seem. There are parallels to ordinary writing which can help us to decipher it.

Just as handwriting is often written on ruled paper, music is written on sets of five lines. Often these staves, as they are called, are grouped in pairs. If you are totally unfamiliar with the notation, let your computer show you what staves look like. RUN last month's listing. (While I think of it - don't discard any of the listings as we shall be reusing much of each listing in a later program.)

The range of the stave is shown by adding a sign at the beginning. There are two which are commonly used. To denote the higher pitch, a treble clef like that on screen is used. This is placed on the top stave. To denote the lower pitch, a bass clef is used.

There is, however, another way in which musicians extend the range of notes which can be shown on a stave. These are ledger lines, indicating the position of notes which lie above or below the stave. These can be seen in Figure 3.



Notes

Returning to our handwriting analogy, let us now look at how notes are written. Just as differently shaped letters convey different meanings, differently shaped notes convey different durations. The pitch of the note is shown by placing it higher, or lower, on its stave. The notes are named with the letters A to G, which are then repeated, as shown in Figure 3.

If you look at a piano or organ keyboard, or merge the lines in Listing 2 to the earlier program, you will see that there are both white and black notes. What I have told you about the writing of notes applies only to the white ones. The black ones, you will notice, are 'squeezed' in between the white and the notation follows the same pattern. If the composer wishes to describe the 'black note' to the right of, say, D, a sign (#) is added to the stave. If every time the note D is encountered, it is to be played as D#, then the sign is placed at the beginning of the stave. However, if it is to be just that particular note that is effected, then the sign will appear just to the left of the note.

Similarly, the 'black note' to the left can be denoted by '♭', called a 'flat'. Sharps and flats can be cancelled by another sign, '♮', called a 'natural'.

Figure 4 shows the different value notes commonly used. A whole note is called a 'semibreve'. The 'breve' from which the term derives is now rarely used. If we add a vertical tail, the value is halved and the note is called a 'minim'. Shading in the note converts it to a 'crochet' and again the duration is halved. Adding a tail to the tail converts a crotchet into a 'quaver'. More tails are added to give 'semiquaver' and 'demisemiquaver'. Guess how many tails it has! If a .ote has a dot after it, its duration is increased by half. The commonly used notes are also shown on screen when you RUN listing 1, or the combined listing.

Stop and Have a Rest!

When a composer requires the player to pause, he uses a sign called a 'rest'. Rests are given the same name as a note of the same duration. Figure 5 shows the signs used. If a rest is of very long duration, the composer places a number above it and the period is that rest multiplied by the number.

Tones, Semitones and Scales

There are twelve semitones in an octave, or scale. Looking

FIGURE 5

our keyboard, you will notice that twice in the octave, two ite notes lie side by side. The interval between adjacent tes is still a semitone. A full tone consists of two mitones. All major scales follow the pattern: tone, tone, mitone, tone, tone, tone, semitone. This is most easily en by playing each of the white notes from one 'C' on the yboard to the next. A tune written with this scale as its sis is said to be 'in the key of C'. As it contains no sharps flats, the musical notation will contain none either.

f you now play from one 'D' to the next, following the me pattern, you will play F# and C#. A piece of music t in this key will have two sharp signs, indicating that the tes, F & C, should be raised by a semitone each time they e encountered. The signs are placed on the top line and the cond space from the top of the treble clef and on the second e and the third space in the bass clef. They form what is own as the 'key signature' of the piece of music. Some amples are shown in Figure 6.

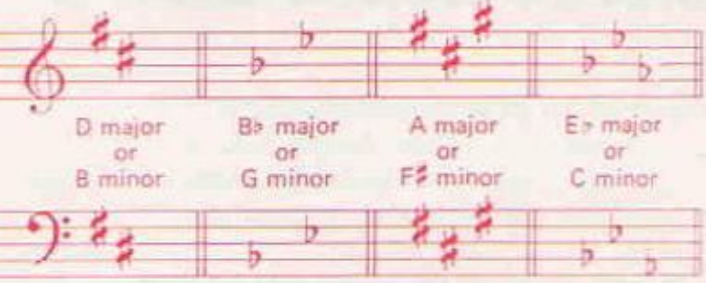


FIGURE 6

mentioned earlier that musical scales were created by erweaving tetrachords. When this is done, a problem arises that each note can have two values, depending on what te forms the base of the tetrachord. String quartets can ust to this, the result being known as 'perfect pitch'.

eyboard instruments, however, are a more difficult oblem. An attempt was made to build an instrument with y-six notes to the octave. This proved quite impractical. other solution adopted was to tune for key signatures ving only a small number of sharps or flats. This gave ch a poor sound as sharps or flats were added that the term olf keys' was used to describe the howling which resulted.

he logical solution was to make the intervals of 'equal nperament'. C was taken as a base and assigned a value of . with the interval ratio being the twelfth root of two '(1/12)]; a nice mathematical solution which gives an ceptable compromise. Both perfect pitch and equal temperament can be created on a computer, the values given in Appendix VII of the User Instructions for the Amstrad CPC464 being for the latter. I will leave the problem of programming perfect pitch to your ingenuity.

Slurs and Bars

There are a number of other terms which we have yet to meet, such as slurs and bars (and they are not as seedy as they sound!). These will be introduced as we discuss the programming techniques. We will also bring Listing 1 and 2 to life by adding a third module to make the computer play the notes for you when you touch the appropriate key.

Two programming techniques can be used. One I call 'Look it up in the Back of the Book'. As this name suggests, the tone period for a given note is obtained from the tables in Appendix VII of the User Instructions for the CPC464.

The other technique is to make use of the formulae for calculating frequency and then tone period from that result. Both of these will be considered and a comparison made later on along with a few other things that I will need to include.

The following listing (Listing 2) will provide a screen of information concerning this month's Learning Centre, however, it will not run by itself. It must be MERGEd with Listing one from last month.

Type in Listing 2 carefully and SAVE it as a backup, then SAVE it again as an ASCII file, namely:

```
SAVE "MLIST2",A
```

Load last month's file (we'll assume you called it MLIST1) and MERGE it with MLIST2, namely:

```
LOAD "MLIST1"
MERGE "MLIST2"
SAVE "MLIST1"
```

MLIST1 will now contain both listings with three screen displays.

(Tape subscribers please note that Listing 2 has already been saved as an ASCII file called MUSLIST2).

Listing Two

```
  10 GOSUB 1630:GOSUB 1000:GOSUB 1
     760:GOSUB 2480:GOSUB 1890:GOS
     UB 1410:GOSUB 2350
  30 PAPER 0:PEN 1:WINDOW#1,1,40,2
     4,25:CLS#1:WINDOW SWAP 1,0:EN
     D
1380 '
1390 REM **** Keyboard ****
1400 '
1410 ORIGIN 0,0,8,631,39,174:CLG 1
1420 MOVE 8,39:DRAW 631,39,0:MOVER
      0,1:DRAWR -631,0,0:MOVE 8,17
     4:DRAW 631,174
```

```
1430 FOR i=8 TO 632 STEP 48
1440 MOVE i,40:DRAW i,173,0:PLOT 6
     41,401,0:MOVE p*48-72,64:TAG
1450 PRINT MID$(b$,p+1,1);:TAGOFF:
     p=p+1
1460 NEXT
1470 MOVE 630,20:DRAW 630,204:ORIG
     IN 0,0,14,22,98,174:CLG 0
1480 FOR i=0 TO 11
1490 IF i=1 OR i=4 OR i=8 OR i=11
     THEN 1510
1500 GOSUB 1560
1510 NEXT
1520 ORIGIN 0,0,616,624,98,174:CLG
     0:RETURN
1530 '
1540 REM **** Black Keys ****
1550 '
1560 ORIGIN 0,0,i*48+40,i*48+70,98
     ,174:CLG 0:PLOT 641,401,1:TAG
1570 MOVE i*48+42,158:PRINT MID$(c
     $,i+1,1);:MOVER -1,0:PRINT CH
     R$(254);
1580 MOVE i*48+42,130:PRINT MID$(d
     $,i+1,1);:MOVER -1,0:PRINT CH
     R$(255);:TAGOFF
1590 RETURN
2320 '
2330 REM **** Keyboard Layout ****
2340 '
2350 WINDOW 1,40,1,13:PEN 0:PAPER
     2:CLS:LOCATE 12,3:PRINT"KEYBO
     ARD LAYOUT"
2360 LOCATE 2,6:PRINT"The keyboard
      has white and black keys,"
2370 LOCATE 2,7:PRINT"adjacent ke
     ys being separated by a"
2380 LOCATE 2,8:PRINT"semitone.
     Notice that the black keys"
2390 LOCATE 2,9:PRINT"have two n
     ames.  They are either the"
2400 LOCATE 2,10:PRINT"sharp (";CH
     R$(254);") to the right of a
     white note"
2410 LOCATE 2,11:PRINT"or the flat
      (";CHR$(255);") to the left
     and derive"
2420 LOCATE 2,12:PRINT"their names
      from that white note."
2430 t=TIME:WHILE TIME<t+4000:WEND
2440 RETURN
2450 '
```

Competition Results - Final Lis

Well, we made it in the end, but not without a few problems.
As mentioned last month, we had a number of obstacles put in our way to determine the winners of the four classes.
Now that the dust has settled, here are some observations about the entries.

1. In most cases, originality was sadly lacking.
2. A number of novice programmers entered the Competition and their efforts are to be applauded.
3. Many entries were downright user unfriendly and lost many points.
4. Program documentation was average.
5. Not all the programs had been fully tested before submitting.
6. In a number of cases much thought and work appeared to have gone into screen design.

With those major observations stated, we are pleased to announce the final result of the first Amstrad User Competition.

CLASS 1 - Best overall program
Name: J.F. Driver, Aspley, Qld
Entry: SALACC (Sales Accounting Package)
Wins: An AWA Video Recorder

CLASS 2 - Best Amusement/Adventure
Name: Geoff Camp, Burra, South Australia
Entry: Girder Guider
Wins: A DDI-1 Disc Drive

CLASS 3 - Best Educational Software
Name: Mark Snoxell, Upper Ferntree Gully, Vic
Entry: Science Tutor
Wins: A DDI-1 Disc Drive

CLASS 4 - Best Business Software
Name: Bernard Chapman, Valley Hts., NSW
Entry: Plotting
Wins: A DDI-1 Disc Drive

OUR CONGRATULATIONS TO THE WINNERS AND ENCOURAGEMENT TO THE LOSERS

THE TRIALS OF TONY BLAKEMORE

A column for the absolute beginner

o finalise the graphic part of our series we will this month k at producing multi coloured characters and larger racters. To enable construction of multi coloured racters we will step into an area seldom explained.
ontrol characters, the series of CHR$ commands between 2, were always a mystery to me and it was many months r I got my Amstrad before I could fully understand them. s beyond the scope of this series of articles to go into a y detailed explanation, perhaps at a later date. Suffice to that to produce any combination of colours or characters have to use control characters. The main ones that rest us at this stage are:-

CURSOR MOVEMENT - Four CHR$ commands ve the cursor one place at a time.
CHR$(8) Moves the cursor back one character.
CHR$(9) Moves the cursor forwards one character.
CHR$(10) Moves the cursor down one line.
CHR$ (11) Moves the cursor up one line.
using any combination of the above we can create any e of string. Vertical, horizontal and diagonal. In fact any we like.

SET PEN INK - CHR$(15) + CHR$(n) where n is number of pens available in a particular mode. Look at e 9,2 of the 464 manual and you will find a list of mands. To set the pen use, the alphabetical equivalent of numeral. i.e. for pen 1 which in mode 1 will produce ow, the command is CHR$(15)+CHR$(a), where a=1, c=3, etc.

SET TRANSPARENT MODE - To enable us to t one character on top of the other we need to make the of the character grid not containing any pixels sparent. We also need to switch if off when we have shed transposing the character. The command for sparent mode is CHR$(22)+CHR$(1). To switch off, the mand is CHR$(22)+CHR$(0).

PRINT USING "&" - This is not a control character important to enable us to print a string combination at far right of the screen without being forced down onto the line.

using a combination of the above we can create a string haracter independent of normal basic commands. Using beetle program from last month, we will now alter a few e lines to produce a multi coloured character. Load the ram and alter the following lines:

```
80 SYMBOL 201,0,24,126,255,255,126,24,0
81 SYMBOL 202,195,66,0,0,0,0,66,195
83 SYMBOL 203,0,24,126,251,251,126,24,0
84 SYMBOL 204,102,66,0,4,4,0,66,102
```

The legs are now separate from the body and we will combine them all in a large string and change their colour.

```
85 b=1
86 b$(1)=CHR$(15)+CHR$(1)+CHR$(201)+CHR$(8)
+CHR$(22)+CHR$(1)+CHR$(15)+CHR$(3)+CHR$(202)+
CHR$(22)+CHR$(0)
87 B$(2)=CHR$(15)+CHR$(1)+CHR$(203)+CHR$(8)+
CHR$(22)+CHR$(1)+CHR$(15)+CHR$(3)+CHR$(204)
+CHR$(22)+CHR$(0)
340 PRINT USING "&";" "+b$(b);
390 IF b=1 THEN b=2 :GOTO 410
```

Now save the new program as "colrbtle". The combination of the four characters in the two strings gives the beetle (FROG!!) coloured legs and flashing eyes.

To illustrate a combination of characters in a string and not overlayed we will use the series from CHR$(212) to CHR$(215). By joining the four triangles we will produce a diamond shape. Using a random number for the pen selection we will then print a screen full of coloured diamonds.

```
10 DEFINT A-Z:CLS
20 a$=CHR$(214)+CHR$(215)+CHR$(8)+CHR$(8)+
CHR$(10)+CHR$(213)+CHR$(212)
30 FOR B=2 TO 24 STEP 2
40 FOR A=2 TO 38 STEP 2:PEN INT(RND*3)+1:
LOCATE A,B:PRINT USING "&";A$:NEXT:NEXT
50 CALL &BB18:GOTO 10
```

Line 10 defines the variables as intergers and makes the loops run faster.

Line 20 combines CHR$(214) with CHR$(215) to produce the top of the diamond. CHR$(8) moves the cursor back one character. CHR$(10) moves the cursor down a line and adds CHR$(213)+CHR$(214) to the bottom of the other characters and forms the bottom of the diamond.

Line 30 Sets a loop to move the cursor down two lines.

Line 40 Sets a loop to move the cursor over two spaces at a time. The pen colour is randomly set to 1,2 or 3. The print command then prints a screen full of diamonds.

Line 50 is a machine code call to wait for any key to be pressed.

By experimenting with combinations of the above many unusual effects can be produced.

Next month Tony Blakemore Goes Somewhere Else and we will start the first part of a more serious look at Basic programming. A mailing list will be the subject and over a period we will look at designing and setting up a data base. Each month a separate module will be designed and tested. The modules will then be combined to form a complete program called AMSFILE.

Review of the GP-700

by Simon Anthony

THE GP-700A is one of a number of dot matrix printers in the GP series from Seikosha. It features a four colour printing head using a special cassette ribbon to give "clear beautiful printing of characters and 8-bit graphic data in up to 7 colours" - a claim which I wanted to put to the test. But you can't test anything without setting it up so let me give you my first impressions from taking it out of the box.

The machine weighs about 6 kgs and measures 113mm (high) by 450mm (wide) by 320mm (deep) without the paper separator in position. There are four operation switches and three indicators on a panel on the right side of the upper case. Line feed and Form feed are self explanatory and only operate when the printer is in STOP mode. The STOP switch is essentially a pause button. The fourth switch is COPY which when pressed activates a screen copy to the printer. But before you all rush out to get the answer to your screen dump problems, the copy switch won't work without a screen copy interface board - an optional extra to the GP-700A.

I don't know whether I am naturally ham-fisted or just lack co-ordination in my fingers when it comes to loading paper in a printer, but the GP-700A didn't help my problem. Unlike the SP-1000 printer that I reviewed a month or so ago, the GP-700A does not appear to allow free movement of the paper to line-up the spocket holes with the paper-feed pins. Even if the paper is accurately fed from the rear, there is no guarantee of alignment to both the pins. However, after a little fiddling I managed to load the paper successfully. Whilst on the subject of paper loading, I tried to load a single sheet just to see how near to the top I could print, and discovered that the 'Paper Empty' switch was located about 7cms away from the left ground plate and naturally enough the paper should cover this switch.

The next step was to load the ink ribbon cassette, which was simply achieved by moving the printhead away from the platen and clicking the cassette into place. The cassette itself has an inked ribbon in four coloured strips - black, magenta, cyan and yellow. There is a useful facility in that if a particular colour is not going to be used, the transfer of ink from that inker to the ribbon can be turned off. This avoids a build-up of of unused ink which could eventually streak on the paper. Individual inkers can be replaced.

The first test was a simple program listing in Black. Printing was left to write only (uni-directional) at about 50 characters per second. The result was a reasonably clear listing - there are no letter-quality facilities with this printer - but you have to accept that it is a high-quality graphics printer and not really built to churn out long listings. With the printer cover closed, the noise level was quite low. There were no problems with extra line-feeds which sometimes occur with new printers. The GP-700A has the relevant DIP switch (No.3) in the off position when shipped. In fact all the switches were off when I examined them.

The DIP switches are located on the PCB in the lower case of the unit and are accessed by removing six screws holding the upper and lower cases together. This is not what you could call convenient but it is unlikely that you would want to change them unle you need to select different languag characters. The printer is already set f USA, with a choice of UK, GERMA and SWEDISH characters.

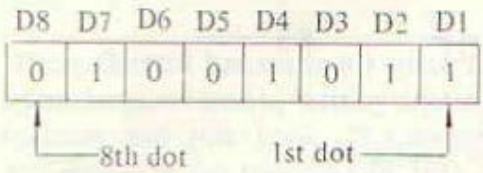
Function codes provide the following
Double width printing
Buffer clear
Setting linefeed spacing
Specifying the print position
Setting the page length
Setting the character spacing
Repeated printing of character data
Graphics data input
Colour mode specification

The last two functions are of particular interest. Graphics data input is established with ESC K n2 n1 n0 GD1 GD2 to GDn. ESC K specifies the input of graphics data in the form of 8-dot vertical columns. n2, n1 and n0 are the ASCII codes for a decimal number representing the number of vertical columns that are to be input. Then follows exactly the same number of bytes of graphics data (GD) as was specified by n2, n1 and n0. The correspondence between graphics data and the dots printed is that the uppermost dot, dot1, is represented by the lowermost bit, D1, and the lowermost dot, dot8, is represented by the uppermost bit, D8.

For example:

1 · · · · · ● · · · · · 1st dot
2 · · · · · ●
4 · · · · · ○
8 · · · · · ●
16 · · · · · ○
32 · · · · · ○
64 · · · · · ●
128 · · · · · ○ · · · · · 8th dot

Translating the above dot pattern into a byte of data gives the following:



So the graphics data for the above example is (4B)H=75

Colour mode specification is a little complex in view of the number of alternatives available, especially when you get down to specifying a colour for a particular dot, so I wont bore you with all the details. In simple terms, when the printer is first turned on, the selected colour is black. To change this the DC4 n function is input, where n is a hexadecimal number between 0 and 6 representing the seven possible colours.

For example, to print AB in red and CD in purple, the code would be DC4 (02)H A B DC4 (03)H C D LF.

All the colours are clear and quite easy to read with the exception of the yellow (I guess it would be clearer on black paper!), and may be used in any order. To specify a colour for each dot, a raster scan method (like the RGB method) is used, except that the colour code is a correspondence to each of the four hammers (and the colours they produce) that comprise the printhead.

It is worth noting here that the printer is designed so that lighter colours are printed first in a single pass. The order of colours on the ribbon also runs from light to dark i.e. yellow, magenta, cyan and black. So I found that printing lighter colours on top of darker ones tended to stain the lighter coloured part of the ribbon with the darker colour. This eventually led to the lighter colour becoming 'dirty'. However, it can be 'cleaned' by merely printing in the correct sequence for a while. If I had read the manual properly in the first place, I would not have made the mistake.

Talking about the manual, it is well presented, although I would like to have seen more examples, and I sometimes found snippets of information which would have been better placed elsewhere.

So there it is - my first attempt at using a colour graphics printer. Unfortunately I didn't have the screen copy board to really put it through its paces with one of my multi-coloured screen creations, but the areas that I did try left me with the feeling of owning a colour television while all around still had a black and white set.

Soft Top

The Top 25 Software Titles Marketed by AWA

Title	This month	Last month	Soft Number
Flight Simulator	1	1	168
Advanced Amsword	2	11	164
Word Hang	3	3	101
Harrier Attack	4	2	112
Easi-Amsword	5	13	154
The Hobbit	6	15	018
Easi-Amscalc	7	NEW	153
Codename Mat	8	8	129
Pitmans Typing Tutor	9	9	924
Amsword (Advanced)	10	4	1164
Amsgolf	11	12	185
Hunter Killer	12	NEW	135
House of Usher	13	NEW	021
Secret of Bastow Manor	14	NEW	010
Exploding Fist	15	NEW	026
Sir Lancelot	16	NEW	023
Manic Miner	17	NEW	173
Concise Firmware Gd.	18	25	158
Centre Court	19	19	921
Grand Prix Rally 2	20	NEW	06012
Fantastic Voyage	21	NEW	984
The Scout Steps Out	22	NEW	988
Dragons	23	NEW	977
Guide to Basic Part 1	24	17	111
Classic Racing	25	NEW	928

London Amstrad User Show

A report from Andre Urankar

The old phrase about "being in the right place, at the right time" was in effect during my recent European business trip. Co-incidental with this trip was an "Amstrad User Show" to be held in London on the 5th and 6th October. As a dedicated user, I made it my business to attend and find out the latest in hardware and software.

The location of the show was first advertised as being at the Novotel Exhibition Centre. A later bulletin in another magazine advised of a change of venue; this time to the Tech West Centre. On arrival at the Tech West Centre, I was greeted by a sign advising that the location was changed back to the Novotel Exhibition Centre. Not a good start!! Taxies are not cheap and the T.W.C. is in an "out of the way" area not frequented by taxies. (Apparently the original organizers of the show pulled out, and it was later taken over by another group; thus the location confusion).

I eventually arrived at the show, to discover a queue of about 200 metres long and steadily lengthening. Due to the orderliness of the British queue-ee, a quick calculation suggested that at least 1000 people were still waiting to enter. Fortunately I was able to "acquire" a trade pass, and thus go straight in.

First impression: WOW!!

The area was packed with rubber-neckers, standing 4-5 deep at just about every display. I had the discouraging feeling that I would not get the opportunity to talk to the various exhibitors to any great depth. How wrong! At each of the stands that I stopped, they were only too willing to go into detail about their product.

Most of the larger software houses were there, displaying their current games (and making a fortune selling past games at special reduced "show" prices). The newer games stretch the Amstrad to greater limits, and produce graphics and sound that they (the software houses) describe with adjectives such as superb, brilliant, exciting, etc. Seriously though, they were of a high standard. Games in general followed the tried and tested paths of adventures, strategies, ladders and ramps, and shoot-em-ups formats. Very little in the way of anything new.

Ultimate have set a graphics standard in games such as Knightlore and Alien-8, and it was encouraging to see other software houses bringing up the level of their graphics to this standard. I was very impressed by "Highway Encounter" from Vortex and "The Covenant" from PSS. These should make the top 10 very quickly and should be added to your "must get" list.

Hardware enhancements for the Amstrad were also well represented. From RAM expansions for the 464 (up to a massive 256k) through modems, RS232 interfaces, light pens, printers, 5.25" disc drives, sideways ROMS, and even a "mouse".

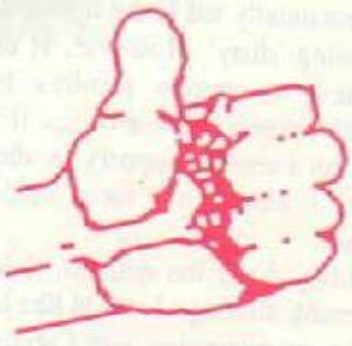
Modems are still a very tricky subject. Some exhibitors were offering simply the modem, while others the necessary interfaces and software. Care was needed to establish what really was being offered.

"Let there be light (pens)"; and so it came to pass. From a very simple-to-use offering from Datapen Technology, through the dk'tronics pen, up to what in my opinion was the best; the fibre-optic pen from Dart Electronics. This particular pen, whilst not the most easiest to use since required keyboard interactio nonetheless provided the mo impressive capability; fast "clean" routine, clear pull-down menu, but ev more impressive was its pixel accura and the ability to follow freeha drawings and writing at norm handwriting speed. All of this in mod 0, 1 and 2.

For the more serious user, "toolki packages were being demonstrated both tape and sideways ROM version The range of ROMs appears at last t be increasing, and other functions (apa from assemblers and disassemblers) a now also available as ROMs.

In a lonely corner was the solitar PCW8256 of the show. It is sad to not that this big brother to Ami was being ignored by the masses. Equally lacking user interest were the few commercia exhibitors; offering their business packages.

All in all this was a bright, informative show, aimed at the Amstrad user, and more specifically at the games player. I came away with the confirmation that the Amstrad is being more and more exploited to its amazing capacity. Long live Ami!!



BOOK REVIEW

by Shane Kelly

Peter Gerrard has made his mark in icrocomputer land by writing articles, eatures and this book, "Exploring dventures on the Amstrad CPC464". I say 'this book' (singular) because though Mr. Gerrard has many titles to s credit, they are all the same book - very time a new computer comes out at becomes popular Mr. Gerrards' ord processor springs into life and lobally replaces references to other omputers with the currently "in ogue" computer.

This is not to say that the book has o value - it does as a guide to dventure playing and also a potted story of the evolution - but it enerally means that the computer in e title is superficially served at best. las, this is the case with this carnation of 'Exploring Adventures on e Amstrad'.

The book aims to teach adventure riting by presenting a number of fully ritten adventure listings, one of which is dissected in great detail. This particular adventure is presented piecemeal as a series of listings that are followed by explanations. These explanations can be quite terse at times but usually manage to convey the 'meat' of the routine. The code is restricted to a subset of Locomotive Basic using mainly PRINT, GOSUB, GOTO and the occasional colour change with a pen command. What happened to all of our beloved screen formatting commands such as WINDOW#X, LOCATE#X, PRINT#X etc? Why, if the book is for the Amstrad, is there no concession to real time game playing so easily implemented with every and after? Why is there no use of the Amstrad's sound capabilities?

Enough griping. There is a positive side to this book. Unfortunately it is contained (in my opinion) in the first few chapters of the book which give the reader an abbreviated history of the development of adventure games from Crowther and Woods Collosal Caves (the classic adventure) to The Hobbit, a successful graphic real-time adventure from Melbourne House. Then follows a chapter on how to solve adventures using examples from the Collosal Caves adventure and others.

Next follows a chapter on basic commands used in the writing of adventure games and then a chapter on writing your own adventures. We then get into the listings mentioned above. And that ends the positive side of the book.

I am ambivalent about books that are one of a series. They generally do not use all the features of the microcomputer for which they they are titled and as such are not as applicable as they could be. But on the other hand, as in this example, they may contain a gem or two that would make the book worthwhile to someone who has a deep interest in the subject matter.

Summing up, I can say that this book will probably not teach you to write best selling adventures, but it may provide some insight into the workings of these intriguing and often overlooked games. Buy it if you have a deep interest in adventure-writing for fun(?) and if you have spare cash, otherwise give it a miss.

FOR THE LOVE OF ARNOLD

A soliloquy from Don Leith

I'm thinking of him ever
As I peform my daily grind
From clocking on to clocking off
He's forever on my mind.

I race on home to him
Dodging traffic in my stride
I fumble for the house key
And kick the dog aside.

Standing by the door
My wife gets just a peck
I have no time for romance
Or arms around the neck.

A door flung hastly open
My wife's prepared the way
The glowing screen does greet me
As Arnold's sign-on says G'day.

THE AMSTRAD USER HALL OF FAME

Game	Score	Time	Achiever
Roland in Time	*72*	*18 mins*	*Paul Azzopardi* *Hint: Practice well on cards 1,6 and 4 to give a sound base for a good score. Don't try to do too much at the beginning.*
Codename Mat	*601*	*90 mins*	*Rowland Hayes* *Hint: Leave the cruisers alone until you have destroyed all the other ships.*
Roland in the Caves	*79884*	*4 mins*	*Emma Poynton* *Hint: Always jump right as your first move.*
Moonbuggy	*66500*	*23 mins*	*Bob Brown* *Hint: Wear a sweat band!*

File Header Reader Utility

by Ian Jardine

This useful program allows the user to display information contained in the File Header of Tape or Disc files. The information includes the File Name, File Type, number of blocks in the File, Load Address, File Length, End Address, and the Execute Address of a Binary File if included. The program is particularly useful where the parameters of a binary file have been forgotten or mislaid. Also provided is a hard copy listing of the Basic program and an Assembler listing of the M/Code Subroutine used by the Basic program.

The machine code routine is poked into protected high memory. It loads the header block of the file to be analysed and returns the location of the file header in memory to Basic, along with a flag to tell Basic if the read was successful or not. The routine is relocatable and will load below the existing HIMEM value and protect itself by lowering HIMEM by 2103 Bytes, which includes a 2K buffer for use by the operating system call CAS IN OPEN.

The Basic program is heavily REMarked and should be self explanatory. When typing in the listing, you should double check the lines 50 to 160, which contain the machine code loader routine. The call to the machine code routine is at line 420 and uses the standard method of passing parameters to and fro. The location of the string descriptor for filename$ is passed to the routine, and the location of the file header is returned in the variable header%. If the read was successful the value of the variable flag% will remain 0, but if the read failed or ESC was pressed then flag% will be returned as -1.

Variables used are:-

oldmem	- original HIMEM value.
headread	- address of M/Code entry point.
addr	- pointer for loading M/Code
byte$	- holds string versions of M/Code bytes.
byte	- hold value of M/Code bytes.
disc	- contains -1 if disc system, 0 otherwise.
tape	- contains -1 if tape selected, 0 otherwise.
rev$	- chr$(24) = inverse video.
match$	- characters to be checked by INSTR routine.
select$	- user input to INSTR routine.
select	- points to selected character in match$.
x	- horizontal screen co-ordinate.
delay	- loop control for delay routine.
filename$	- filename of file to be read.
header%	- holds location of file header after read.
flag%	- holds 0 if read OK, -1 if read failed.
file$	- holds file name as read from header.
loop	- loop control for reading file$.
type	- holds file type code value.
loadaddr	- load address of file.
filelen	- length of file.
execaddr	- autostart address of M/Code file.
endaddr	- last memory location occupied by file.
blocks	- number of blocks in file.

To use the program, you should load and run it and t insert the required tape or disc. Disc users will be asked select either Tape or Disc input, and will then be promp for the filename. Tape users will go direct to the filena prompt and may press [ENTER] alone to read the next f from the tape. After the header block has been read, t program will display the information about the file on t screen. You will then be asked if you wish to analy another program or not. If you press [N] the program ends

```
10 REM ** HEADREAD.BAS ** CASSETT
   E OR DISC PROGRAM HEADER READE
   R **
20 REM     Version 1.3,  by Ian Ja
   rdine.  September  1985
30 IF PEEK(HIMEM+1)=&DD AND PEEK(
   HIMEM+2)=&E5 THEN 190' Ignore
   if Loaded
40 CALL &BB48' Disable *Break*
50 SYMBOL AFTER 256' Delete Symbo
   l Table
60 oldmem=HIMEM' Get Old Memsize
70 MEMORY HIMEM-&837' Set New Mem
   size
80 headread=HIMEM+1:addr=headread
   :READ byte$' Set Pointers & Re
   ad 1st Byte
90 WHILE byte$<>"END"' Read M/Cod
   e into Protected Memory
100  byte=VAL("&"+byte$):POKE addr
   ,byte:addr=addr+1:READ byte$
110 WEND' Loop Here Until Done
120 REM ** Hex Bytes for M/Code He
   adread Routine **
130 DATA DD,E5,DD,6E,04,DD,66,05,4
```

```
    6,23,5E,23,56,21,00,00,EB,CD,7
    7,BC,DD,E1
140 DATA 30,0F,28,0D,E5,DD,6E,02,D
    D,66,03,D1,73,23,72,18,0C,DD,6
    E,00,DD,66
150 DATA 01,11,FF,FF,73,23,72,CD,7
    D,BC,C9,END
160 POKE headread+&E,UNT(headread+
    &37)AND &FF' Adjust Non-Absolu
    te Address
170 POKE headread+&F,INT((headread
    +&37)/256) ' for Relocation
180 REM ** Check if a Disc System
    **
190 disc=(1=1):ON ERROR GOTO 200:
    DISC:GOTO 210
200 IF ERR=28 THEN disc=(1=0):RESU
    ME NEXT
210 ON ERROR GOTO 0
220 IF NOT disc THEN tape=(1=1)' I
    t's a Tape System
230 MODE 1:INK 0,13:INK 1,0:INK 2,
    0,13:PEN 1:BORDER 13:rev$=CHR$
    (24)
240 IF tape THEN 320 ELSE LOCATE 1
    2,1:PRINT"Read File from ?"
250 PRINT:PRINT TAB(5)"[T] --> TAP
    E  or  [D] --> DISC"
260 match$="TD":GOSUB 810' Select
    Disc or Tape
270 IF select=1 THEN tape=(1=1):
    APE.IN' Set up for Read from T
    ape
280 LOCATE 5-(select=2)*18,3:PEN 2
    :PRINT rev$"["select$"]"rev$
290 PEN 1:PRINT:PRINT:PRINT TAB(2)
    "[ENTER] to Confirm - [DEL] to
     Change"
300 match$=CHR$(13)+CHR$(127):GOSU
    B 810' Make Sure Customer is S
    atisfied
310 CLS:IF select=2 THEN tape=(1=0
    ):GOTO 240' Changed Mind!
320 LOCATE 5,2:PRINT rev$"[ENTER]"
    rev$"  Filename of ";
330 IF tape THEN PRINT"TAPE";ELSE
    PRINT"DISC";
340 PRINT" Program":PRINT:PRINT TA
    B(5)"to be Analysed?";:x=POS(#
    0)
350 KEY DEF 66,0,0,0,0' Disable [E
    SC] Key During Input
360 INPUT" ",filename$:filename$=U
    PPER$(filename$)' Get Filename
370 IF NOT tape AND(filename$=""OR
     LEN(filename$)>12)THEN LOCATE
     x+1,4:PEN 3:PRINT CHR$(7)"*ER
    ROR*"CHR$(18):FOR delay=0 TO 1
    500:NEXT:LOCATE x,4:PEN 1:PRIN
    T CHR$(18)CHR$(7);:GOTO 360
380 KEY DEF 66,0,252,252,252' Re-e
    nable [ESC] after Input
390 CALL &BB48' Disable *Break*
400 WINDOW#1,5,40,13,13-(NOT tape)
    *2:WINDOW SWAP 0,1:PEN 3
410 IF NOT tape THEN PRINT TAB(7)"
    Loading "filename$
420 header%=0:flag%=0' Declare Var
    iables for M/Code Routine
430 CALL headread,@filename$,@head
    er%,@flag%' Call Headread Rout
    ine
440 REM ** Header% = Header Addres
    s **
450 REM ** Flag% = 0 if Load ok, -
    1 if Read Fail or [ESC] Presse
    d **
460 IF flag% THEN PRINT rev$" READ
     ERROR "rev$"  OPERATION ABORT
    ED"
470 WINDOW SWAP 0,1:PEN 1:IF flag%
     THEN 750
480 CLS:PRINT:PRINT TAB(4)rev$" ";
    :IF tape THEN PRINT"TAPE";ELSE
     PRINT"DISC";
490 PRINT" FILE HEADER  --  ANALYS
    IS "rev$
500 file$=""
510 FOR lp=0 TO 11-(tape)*4' Read
    the Filename
520  file$=file$+CHR$(PEEK(lp+head
    er%))
530  IF NOT tape AND lp=8 THEN fil
    e$=file$+"."' Put in a . if Di
    sc Used
540 NEXT
550 type=PEEK(header%+18)' Get Fil
    e Type
560 loadaddr=PEEK(header%+22)*256+
    PEEK(header%+21)' Get Load Add
    ress
570 filelen=PEEK(header%+25)*256+P
    EEK(header%+24)' Get Flle Leng
    th
580 execaddr=PEEK(header%+27)*256+
    PEEK(header%+26)' Get Execute
    Address
590 endaddr=loadaddr+filelen-1' Ca
    lculate End Address
600 blocks=INT(filelen/2048)-(file
    len/2048<>INT(filelen/2048))'
    Calc. No. Of Blocks In File
```

```
610 REM * DISPLAY INFORMATION *
620 LOCATE 1,6:PRINT rev$" PROGRAM
    NAME"SPC(9)rev$"  "file$
630 PRINT:PRINT rev$" PROGRAM TYPE
    "SPC(9)rev$"  ";
640 IF type=0 THEN PRINT"STANDARD
    BASIC"
650 IF type=1 THEN PRINT"PROTECTED
     BASIC"
660 IF type>1 AND type<6 THEN PRIN
    T"MACHINE CODE"
670 IF type=22 THEN PRINT"ASCII TE
    XT":GOTO 730' ASCII so Ignore
    the Rest
680 PRINT:PRINT rev$" NUMBER OF BL
    OCKS      "rev$" "blocks
690 PRINT:PRINT rev$" PROGRAM LOAD
     ADDRESS "rev$" "loadaddr;TAB(
    32)"(&"HEX$(loadaddr,4)")"
700 PRINT:PRINT rev$" PROGRAM END
    ADDRESS  "rev$" "endaddr;TAB(3
    2)"(&"HEX$(endaddr,4)")"
710 PRINT:PRINT rev$" PROGRAM LENG
    TH        "rev$" "filelen;TAB(3
    2)"(&"HEX$(filelen,4)")"
720 IF type>1 AND type<6 THEN PRIN
    T:PRINT rev$" EXECUTE ADDRESS
          "rev$" ";:IF execaddr=(
    HEN PRINT" NOT SPECIFIED"ELS
    PRINT execaddr;TAB(32)"(&"HE
    (execaddr,4)")"
730 PRINT:PRINT:PRINT STRING$(39
    5)
740 REM * Do it Again? *
750 tape=(1=0):IF disc THEN :DIS
    IN' Reset Tape Flag & Indire
    ion
760 LOCATE 1,23:PRINT" Do You Wi
     to Analyse Another Program"
770 LOCATE 15,25:PRINT"(Y) or (N)
    "rev$"-"rev$
780 match$="YN":GOSUB 810
790 ON select GOTO 220,840
800 REM ** Key Selection Routine
    *
810 select=0:WHILE select=0:selec
    $=UPPER$(INKEY$):IF select$>"
    THEN select=INSTR(match$,sele
    t$)
820 WEND:RETURN
830 REM ** Exit Routine **
840 MEMORY oldmem:CALL &BBFF:MODE
    1:PEN 1:SYMBOL AFTER 240
850 END
```

AMSTRAD ACHIEVERS

Get your name in our "HALL OF FAME"

Register your name and score on the form below, and if possible, send a photo of the screen. (See Page 25 for 'Achievers' to date)

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.

Review of the CPC6128

by Robin Nicholas

As I sit and look at the new PC6128, I cannot help feeling sorry or those people who have just urchased a CPC664. Why Amstrad othered to release the 664, knowing ll well that the 6128 was due to be eleased within such an indecently short pace of time, is beyond me. I suppose, be fair, the full 64k of RAM in the 64 (and the 464 for that matter) could ot support the standard versions of me of the better CP/M software, hereas under CP/M Plus (or CP/M ersion 3) on the 6128, the full 128k ecomes available - but more of this ter.

Like the previous machines in the mstrad range, the standard CPC6128 mes as a package including a green reen with the option of a colour reen for about another $200 dollars. he floppy disc drive is housed to the ght of the keyboard on top of which is chart showing the colour and key finition numbers.

The power switch and volume control n be found at the back of the unit ong with the second disc drive port, e monitor and power supply sockets, d finally an expansion port and a ntronics compatible printer port. On e left side of the unit are the sockets r the joystick, sound output and tape corder.

The trend with computers is 'the gger they get, the smaller they come' and the 6128 is no exception. ith the greater capacity of 128k, the ysical size of the unit has reduced by er two inches on its predecessors. e colour scheme is dark grey with ht grey keys.

he keyboard has 74 dished keys and s been redesigned into one block with the cursor keys positioned below the numeric keypad. Compared with the 464, the small ENTER key has disappeared and the COPY and CONTROL keys are now situated to the left of the space bar. One, much larger ENTER key is on the right of the space bar, and its original position (on the 464) is taken with a RETURN key.

The system still provides the standard 20, 40 or 80 column screen display, and although the screen is slightly different in terms of connection to the processor, the quality of performance appears to be identical.

The 'guts' of the 6128 is very similar to the 464 being the Z80 processor with Basic ROM but with the major addition of the 64k of RAM. For the technical there are seven LSI chips: the Z80A processor running at 4MHz; 128k of RAM arranged in two 64k banks (over 41k available to the user in Basic, 61k available to CP/M Plus as the Transient Program Area [TPA]); 48k bytes of ROM containing BASIC, the operating system and disc extensions; a 6845 CRT controller device; an AY-3-8912 sound generator chip; an 8255 parallel I/O device; and a 7653 floppy disc controller.

Extra Basic commands are loaded from disc, which means that there should be full software compatibility with the 664. Unfortunately this cannot be

guaranteed to apply to the 464 despite the fact that Amstrad reckon on about 99.5%.

The extra commands allow the use of the additional 64k RAM either as a RAM data file or to save up to four screen displays. The latter are held in 16k blocks each, providing the facility to switch with the main screen display block very quickly. The two commands to achieve this are |SCREENCOPY and |SCREENSWAP. To use the RAM as a data file, the commands |BANKOPEN, |BANKWRITE, |BANKREAD and |BANKFIND are used.

Two versions of CP/M are supplied with the 6128 - 2.2 and 3.1 (CP/M Plus) - and you will know that the 2.2 version is used on the 464 and 664. In addition, there are two terminal emulators - the VT52 and the Zenith Z19/Z29. The 3.1 version takes advantage of the 6128's larger memory and should be capable of running WordStar, Multiplan and other popular pieces of software. The clever implementation of this version is such that the various statements and routines are loaded just once on the initial boot. This saves the bother of keeping a note of which system files are available when changing discs. Error messages are more 'user friendly' and are scrolled along the bottom of the screen.

The manual is very thorough as usual, covering the now standard introduction and tutorial with examples, and then the Basic keywords, AMSDOS and CP/M, an introduction to Logo, Bank switching and a lot more. However, it may still be necessary for a newcomer to purchase additional reading material to get to grips with the machine.

So, how much does it all cost?

The basic green screen package, including CP/M 2.2 , CP/M Plus and Dr. Logo will retail at around $800. The colour screen package will cost around $1000 and an additional disc drive will be about $400. At these prices, Amstrad still maintain the initiative in providing low cost entry points - in this case, possibly, small business.

Menu Utility

from Reuben Carlsen

The following two modules can be used whenever you need menu for any program. They evolved because I am urrently writing business application packages. I wanted all ny menus to have only alpha-key selection, as I dislike aving menus with 1, 2, 3 etc.

I wanted to be able to use the command "ON A GOSUB 000, 2000, 3000 " etc. but this only works with numeric nputs. So the problem was that I needed to convert alpha nputs to numeric codes. Hence the line A=(ASC(A$)-64, his converts alpha inputs to numeric inputs.

The ASCII code for Capital A is 65, so when you hit "A" he program takes it as 65, subtracts 64 and leaves it as 1, nput B as 2 etc.

I had to ensure that all inputs were uppercase - this was asy with A$=UPPER$(A$). With that problem solved, I anted the cursor flashing on-screen when waiting for an nput. This was solved with a simple little sub-routine hich locates where you want the cursor to appear, prints the ursor, drops to the delay sub-routine, returns, prints HR$(8) which brings the print position back one space en prints CHR$(16) which blanks out the cursor, the rogram then drops to the delay sub-routine again, and turns to the line that starts the process over again.

The line IF A$<>"" THEN RETURN, ensures that the b-routine keeps looping until an input is detected. When input is detected it returns to the routine that handles the put.

have written three routines (two are presented in this ticle), "MENU/10" for MODE 1 with a menu for 10 otions. The next "MENU/13" for MODE 2 with a menu for options. I've included quite a few REM statements to pefully explain what is happening throughout.

I have set myself up with a complete bank of these routines, when I need a menu with anything from 2 to 20 options I just MERGE them from tape, (soon to be disk).

I have used CHR$(224) for my cursor, I plan on using this cursor as a feature of all my packages so I would appreciate it if other writers choose another character.

I hope someone can use these little routines, I enjoyed writing them a great deal. I am still refining them as I am trying to get the response delay a little shorter, maybe someone else has got some ideas.

By the way, I didn't invent the flashing cursor sub-routine, part of it came from "AMSTRAD COMPUTING" by Ian Sinclair. This is a top book, I cannot recommend it highly enough.

NOTES

1) When modifying these routines for your own programs you can either add or delete the option lines, depending on the number of options you need. Where I have "OPTION 1" etc. you can just type in what your options are.
2) In the line IF A<1 OR A>12, just change the last number to the number of options your menu has. This ensures that the program only recognises input within the menu limits.
3) The "Quit" option always returns to MODE 2, I did this so during development I could list more on the screens.

I haven't included a commentary, I think the REM statements in each listing will suffice. If you have more than one menu in a program you could use the same flashing cursor sub-routine, as long as you want the cursor at the same co-ordinates each time.

```
' *****        THIS   IS   A
MENU   WITH   10   OPTIONS
    *****
' *****            WRITTEN    B
Y    REUBEN    CARLSEN
    *****
' ***** THIS VERS' IS FOR MODE
 1 WITH OPTIONS VERTICAL ON SC
REEN *****
'
MODE 1
LOCATE 10,1:PRINT"MAIN OPTION
BLOCK  [vers 1.4]"
LOCATE 14,4:PRINT"[A] option 1
":LOCATE 14,6:PRINT"[B] option
 2"
LOCATE 14,8:PRINT"[C] option 3
":LOCATE 14,10:PRINT"[D] optio
n 4"
LOCATE 14,12:PRINT"[E] option
5":LOCATE 14,14:PRINT"[F] opti
on 6"
LOCATE 14,16:PRINT"[G] option
7":LOCATE 14,18:PRINT"[H] opti
on 8"
LOCATE 14,20:PRINT"[I] option
9":LOCATE 14,22:PRINT"[Q] QUIT
LOCATE 8,25:PRINT"SELECT OPTIO
N REQUIRED  [ ]":LOCATE 33,24
'
GOSUB 500:'*** THIS LINE SENDS
 TO 1ST SUBROUTINE FOR FLASHIN
G CURSOR ****
'
A=(ASC(A$)-64):'**** THIS LINE
 CONVERTS ALPHA INPUTS TO NUME
RIC ****
'                  ****
 A=1,B=2,C=3,D=4 etc.
    ****
'                  ****
SO LINE 240 IS POSSIBLE
    ****
'
IF A=17 THEN GOSUB 370 ELSE 22
0:'* LINE 160 CONVERTS "Q" INP
UT TO = 17 *
'
IF A<1 OR A>9 THEN 140:'* THIS
 LINE STOPS ANY INPUT OUTSIDE
THE MENU RANGE *
```

```
230 '
240 ON A GOSUB 280,290,300,310,320
    ,330,340,350,360
250 '
260 '**** LINES 280-360 ARE THE SU
    BROUTINES FROM MENU SELECTIONS
     ****
270 '
280 CLS:LOCATE 7,12:PRINT"YOU SELE
    CTED SUBROUTINE 1":GOTO 380
290 CLS:LOCATE 7,12:PRINT"YOU SELE
    CTED SUBROUTINE 2":GOTO 380
300 CLS:LOCATE 7,12:PRINT"YOU SELE
    CTED SUBROUTINE 3":GOTO 380
310 CLS:LOCATE 7,12:PRINT"YOU SELE
    CTED SUBROUTINE 4":GOTO 380
320 CLS:LOCATE 7,12:PRINT"YOU SELE
    CTED SUBROUTINE 5":GOTO 380
330 CLS:LOCATE 7,12:PRINT"YOU SELE
    CTED SUBROUTINE 6":GOTO 380
340 CLS:LOCATE 7,12:PRINT"YOU SELE
    CTED SUBROUTINE 7":GOTO 380
350 CLS:LOCATE 7,12:PRINT"YOU SELE
    CTED SUBROUTINE 8":GOTO 380
360 CLS:LOCATE 7,12:PRINT"YOU SELE
    CTED SUBROUTINE 9":GOTO 380
370 CLS:LOCATE 6,12:PRINT"YOU SELE
    CTED THE 'QUIT' OPTION":FOR Z=
    1 TO 2000:NEXT:MODE 2:END
380 LOCATE 5,22:PRINT"START AGAIN
    (Y/N)    [ ]"
390 GOSUB 570:' **** THIS LINE SEN
    DS TO 2ND SUBROUTINE FOR FLASH
    ING CURSOR ****
400 IF A$="Y" THEN 10
410 IF A$="N" THEN GOTO 640 ELSE 3
    90
420 END
430 '
440 '**** LINES 500-560 IS 1st SUB
    ROUTINE FOR THE FLASHING CURSO
    R ****
450 '**** LINES 570-630 IS 2nd SUB
    ROUTINE FOR THE FLASHING CURSO
    R ****
460 '**** THE FLASHING CURSOR IS A
     SMILING FACE, I INTEND TO MAK
    E ****
470 '**** THIS A FEATURE OF ALL MY
     SOFTWARE SO I WOULD APPRECIAT
    E ****
480 '**** IT IF OTHER PEOPLE USED
    SOME OTHER CHARACTER,...,THANKS
    . ****
490 '
500 A$=INKEY$
510 A$=UPPER$(A$)
520 IF A$<>"" THEN RETURN
530 LOCATE 33,25:PRINT CHR$(224);:
    GOSUB 560
540 PRINT CHR$(8);CHR$(16);:GOSUB
    560
550 GOTO 500
560 FOR J=1 TO 400:NEXT:RETURN
570 A$=INKEY$
580 A$=UPPER$(A$)
590 IF A$<>""THEN RETURN
600 LOCATE 26,22:PRINT CHR$(224);:
    GOSUB 630
610 PRINT CHR$(8);CHR$(16);:GOSUB
    630
620 GOTO 570
630 FOR J=1 TO 400:NEXT:RETURN
640 CLS:LOCATE 6,12:PRINT"YOU CHOS
    E TO EXIT TO 'SYSTEM'":FOR Z=1
     TO 2000:NEXT:MODE 2:END
```

```
10 ' *****          THIS   IS   A
    MENU   WITH   13   OPTIONS
          *****
20 ' *****                WRITTEN
   BY    REUBEN    CARLSEN
          *****
30 ' ***** THIS VERS' IS FOR MODE
    2 WITH OPTIONS LISTED ACROSS
   SCREEN *****
40 '
50 MODE 2
60 LOCATE 23,1:PRINT"*** MAIN OPT
   ION BLOCK ***              [vers
    2.0]"
70 LOCATE 23,2:PRINT"_____
   _____"
80 LOCATE 12,6:PRINT"[A] option 1
   ":LOCATE 45,6:PRINT"[B] option
   2"
90 LOCATE 12,8:PRINT"[C] option 3
   ":LOCATE 45,8:PRINT"[D] option
   4"
100 LOCATE 12,10:PRINT"[E] option
    5":LOCATE 45,10:PRINT"[F] opti
    on 6"
110 LOCATE 12,12:PRINT"[G] option
    7":LOCATE 45,12:PRINT"[H] opti
    on 8"
120 LOCATE 12,14:PRINT"[I] option
    9":LOCATE 45,14:PRINT"[J] OPTI
    ON 10"
130 LOCATE 12,16:PRINT"[K] option
    11":LOCATE 45,16:PRINT"[L] OPT
    ION 12"
140 LOCATE 12,19:PRINT"[Q] QUIT"
150 LOCATE 23,22:PRINT"SELECT OPTI
    ON REQUIRED  [ ]"
160 '
170 GOSUB 560:'*** THIS LINE SENDS
     TO 1ST SUBROUTINE FOR FLASHIN
    G CURSOR ****
180 '
190 A=(ASC(A$)-64):'**** THIS LINE
     CONVERTS ALPHA INPUTS TO NUME
    RIC ****
200 '                    ****
     A=1,B=2,C=3,D=4 etc.
       ****
210 '                    ****
    SO LINE 270 IS POSSIBLE
       ****
220 '
230 IF A=17 THEN GOSUB 430 ELSE 25
    0:'* LINE 190 CONVERTS "Q" INP
    UT TO = 17 *
240 '
250 IF A<1 OR A>12 THEN 170:'* THI
    S LINE STOPS ANY INPUT OUTSIDE
     THE MENU RANGE *
260 '
270 ON A GOSUB 310,320,330,340,350
    ,360,370,380,390,400,410,420
280 '
290 '**** LINES 310-420 ARE THE SU
    BROUTINES FROM MENU SELECTIONS
     ****
300 '
310 CLS:LOCATE 27,12:PRINT"YOU SEL
    ECTED SUBROUTINE No' 1":GOTO 4
    40
320 CLS:LOCATE 27,12:PRINT"YOU SEL
    ECTED SUBROUTINE No' 2":GOTO 4
    40
330 CLS:LOCATE 27,12:PRINT"YOU SEL
    ECTED SUBROUTINE No' 3":GOTO 4
    40
340 CLS:LOCATE 27,12:PRINT"YOU SEL
    ECTED SUBROUTINE No' 4":GOTO 4
    40
350 CLS:LOCATE 27,12:PRINT"YOU SEL
    ECTED SUBROUTINE No' 5":GOTO 4
    40
360 CLS:LOCATE 27,12:PRINT"YOU SEL
    ECTED SUBROUTINE No' 6":GOTO 4
    40
370 CLS:LOCATE 27,12:PRINT"YOU SE
    ECTED SUBROUTINE No' 7":GOTO
    40
380 CLS:LOCATE 27,12:PRINT"YOU SE
    ECTED SUBROUTINE No' 8":GOTO
    40
390 CLS:LOCATE 27,12:PRINT"YOU SE
    ECTED SUBROUTINE No' 9":GOTO
    40
400 CLS:LOCATE 27,12:PRINT"YOU SE
    ECTED SUBROUTINE No' 10":GOTO
    440
410 CLS:LOCATE 27,12:PRINT"YOU SE
    ECTED SUBROUTINE No' 11":GOTO
    440
420 CLS:LOCATE 27,12:PRINT"YOU SE
    ECTED SUBROUTINE No' 12":GOTO
    440
430 CLS:LOCATE 25,12:PRINT"YOU SE
    ECTED THE 'QUIT' OPTION":FOR
    =1 TO 2000:NEXT:MODE 2:END
440 LOCATE 27,22:PRINT"START AGAI
     (Y/N)    [ ]"
450 GOSUB 630:' **** THIS LINE SE
    DS TO 2ND SUBROUTINE FOR FLASH
    ING CURSOR ****
460 IF A$="Y" THEN 10
470 IF A$="N" THEN GOTO 700 ELSE
    50
480 END
490 '
500 '**** LINES 560-620 IS 1st SUB
    ROUTINE FOR THE FLASHING CURSO
    R ****
510 '**** LINES 630-690 IS 2nd SUB
    ROUTINE FOR THE FLASHING CURSO
    R ****
520 '**** THE FLASHING CURSOR IS A
     SMILING FACE, I INTEND TO MAK
    E ****
530 '**** THIS A FEATURE OF ALL MY
     SOFTWARE SO I WOULD APPRECIAT
    E ****
540 '**** IT IF OTHER PEOPLE USED
    SOME OTHER CHARACTER,...,THANKS
     ****
550 '
560 A$=INKEY$
570 A$=UPPER$(A$)
580 IF A$<>"" THEN RETURN
590 LOCATE 48,22:PRINT CHR$(224);:
    GOSUB 620
600 PRINT CHR$(8);CHR$(16);:GOSUB
    620
610 GOTO 560
620 FOR J=1 TO 400:NEXT:RETURN
630 A$=INKEY$
640 A$=UPPER$(A$)
650 IF A$<>""THEN RETURN
660 LOCATE 48,22:PRINT CHR$(224);:
    GOSUB 690
670 PRINT CHR$(8);CHR$(16);:GOSUB
    690
680 GOTO 630
690 FOR J=1 TO 400:NEXT:RETURN
700 CLS:LOCATE 27,12:PRINT"YOU CHO
    SE TO EXIT TO 'SYSTEM'":FOR Z=
    1 TO 2000:NEXT:MODE 2:END
```