

Personal SOFTWARE

Summer 1983

An Argus Specialist Publication

£1.95

**Programs
galore for
your...**

**VIC-20,
PET,
8000
machines,
and
Commodore 64.**

- **Ingenious utilities
to run**
- **Educational
software
to test
the family**
- **Data base
for better
business**
- **Great
games
to
play**



**BRITAIN'S BEST BUY
FOR THE
COMMODORE
USER**

VIC20 GAMES FROM QUICKSILVA

TORNADO VIC20 GAMES FROM QUICKSILVA



VIC20
+ JOYSTICK

TORNADO: In this model of the Colony Wars ships are attacking you from the air you have three levels of ground base to bomb. Features Full Colour Hi-Res Graphics, Sound, Increasing Speed, High Score, Progressive Difficulty, Random Landscapes, Engagements. By Cheated for the unexpanded VIC20 + Joystick.

PIXEL GAMES

HARVESTER & BRAINSTORM

A cut-throat strategy game to help valuable bookkeepers around the planet. Color, Hi-Res Graphics and lots of fun for two to four players. For the unexpanded VIC20.

STARQUEST/ ENCOUNTER

A voyage of discovery and adventure in the cosmos. With the help of your friend's computer you reach a remote planet amidst the perils of deep space. For the VIC20 + 128K RAM.

QUICKSILVA are proud to introduce Skyhawk written by Charted for the VIC20 personal computer, many more amazing games to come!

SKYHAWK: Features Multi-colour Hi-Res 3-D Effect Graphics, Realistic Sounding Landscapes, Airplane Landing and Refueling, Radar and Aircraft Status Displays, High Score, Save, Full Ground Effects, Varying Dimensions of Moving Aircraft, Score, Up to 30000 pts, Full Low Warning, Realistic Explosions. For the VIC20 + 128K or 192 expansion + Joystick.

PIXEL POWER



VIC20 128K
For two to four players

PIXEL

HARVESTER



VIC20 UNEXPANDED
A cut-throat strategy game for two + joystick

PIXEL

STARQUEST



VIC20 128K GAME

A voyage of discovery and adventure in the cosmos

PIXEL



For the VIC20 + 128K RAM

PIXEL POWER

A graphics workshop packed with useful features such as Create, Delete, Store and View. For the VIC20 with 128K or more address RAM.

SUBSPACE STRIKER & ZOR

It comes from out of nowhere and then vanishes back into the ether. With your deadly armored torpedoes, you unleash havoc in the Subspace Spectacular.

For the
VIC20 +
128K
RAM

TRADER

A trilogy of 128K programs that combine to give an epic RPG graphic adventure. As a pioneer trader you deal with some very human customers instead. Will you live to tell the tale? Supplied in a box with extensive instruction booklet.



- | | | |
|------------------------|--------|--------------------------|
| Tornado | £5.95 | ☐ |
| Skyhawk | £7.95 | ☐ |
| Trader | £14.95 | ☐ |
| Subspace Striker & Zor | £7.95 | ☐ |
| Starquest + Encounter | £7.95 | ☐ |
| Pixel Power | £7.95 | ☐ |
| Harvester + Brainstorm | £7.95 | ☐ |

PLEASE SEND ME THE GAMES

£6.00 each

Total cheque/P.O. enclosed
Cheque payable to Quicksilver Limited

Please send your order to:
QUICKSILVA LIMITED
PALMERSTON PARK HOUSE
13 PALMERSTON PARK ROAD
DUNDEE DD1 1JN
TEL: 01307 200710

ADDRESS

NAME

QUICKSILVA: The only computer games publisher in the world to offer a free and complete set of sales figures at which all



Volume 2 No 1 Summer 1983

Editor: Henry Budgett
Assistant Editor:
Wendy Palmer
Advertisement Manager:
Adrian Roberts
Advertisement Copy:
Christina Susan Best
Managing Editor:
Sam Harris BSA
Managing Director: T J Connell

Original art and design by
BM Design & Print

Personal Software is a quarterly
magazine covering the world
of PCs, Macs, Apples, Novell and
Pentiums.

Distribution by: Argus Press Sales &
Distribution, Ltd., 27 Bedford Street,
London WC2E 8RF. Tel: 01-461 5000.
Printed by: Aldenham Presses &
Sons Ltd., York, Yorkshire, Great
Britain.

The material in this publication
including all articles, designs, plans,
drawings and programs and all
copyright and other material
properly remains the property of
Argus Presses & Distribution
Limited. All rights reserved in the
use of copyright and other
intellectual property rights and by
virtue of international copyright
conventions are specifically reserved
to Argus Presses & Distribution
Limited and its representatives
throughout the world except insofar
as the Company, its staff Argus
Presses & Distribution Limited.

Subscription Rates: UK £7.50 per
year (includes postage).
Overseas and other rates upon
application to Personal Software
Subscriptions, 40 Lonsdale Road,
Barnsley South, South Yorkshire
S10 2BB.

CONTENTS

Editorial & Advertisement Office
145 Charing Cross Road, London WC2H 0EE
Telephone: 01-437 1002 Telex: 8811896

Preface	4	Commodore Communications 58
Getting Converted	6	Get Commodores talking to each other using this application.
Hints on how to convert programs from one Commodore machine to another.		
Leapfrog	13	Address Book
Our micro version of that old pub-game played with coins.		63
Power Boat	16	Compile an address book, or any other similar list, and throw away those bits of paper.
Have all the fun of power boating without feeling seasick!		
The Valley	19	Multicolumn Records
Save the kingdom of the Valley by combatting dragons, Balrogs and Wraiths. Choose your character with magic or physical strengths and do battle in our epic game, The Valley.		66
Towers of Braham	38	A multipurpose data base program for use at home or in the office.
Moving rings from one pillar to another may sound easy but just try it!		
Micro Examination	42	Subroutine Library
Test your friends and children with this multiple choice program.		74
Quiz Time	45	A library of BASIC subroutines.
Assess your performance in terms of speed and accuracy with a multiple choice program.		
Multipurpose Records	50	Toolkit Program
Set up your own filing system which enables you to store, search, edit and retrieve data.		77
VIC Editor	54	A simple toolkit program for the Commodore 64.
Take one VIC-20 and add this program and what do you have? A VIC-20 with an 'enlarged' screen.		
		Tailoring VIC's Characters ... 79
		Create your own characters on your VIC-20.
		Max-Mander
		84
		Bomb-proofing your software against unskilled fingers.
		VIC Blow Up
		92
		Find out how characters are made up and generate giant versions on the VIC-20.
		Program Protection
		95
		Simple tips on how to protect your program from being easily copied.
		Bibliography
		97
		A brief perusal of some of the multitude of books on Commodore computers.

Credits: Our grateful thanks to the following individuals for their assistance in the production of this issue: Enterprise Features Limited, Karen, Adrian, Javier, Tom, Euan, Wesley, Chris, Walter, John and Bill.

PREFACE



Welcome to this issue of *Personal Software* which is dedicated to the Commodore series of microcomputers. The series of machines includes the very popular VIC-20, the fairly new Commodore 64 and the 2000, 300 and 4000 range so we have tried to put together a variety of software that should appeal to Commodore users regardless of their particular machine. We say regardless of machine because we know how frustrating it can be when you see an interesting program for a VIC-20, say, and you sit gazing longingly at your Commodore 64 wishing you knew how to adapt the program for your micro. Here we have endeavored to be of help: on page 6 of this magazine the Editor has put together some hints and techniques to show you how to convert the programs from one machine to another. So with only a little alteration you should be able to implement almost all of the programs in this issue on your Commodore microcomputer.

Armed with the knowledge that you really can use most of the programs in this magazine you will probably want to know exactly what there is, and we have tried to pull together some of the best material available for Commodore machines that have been published in *Computing Today* and have added some new specially commissioned features and programs to produce a varied package.

The material included in this issue ranges from the fairly elementary level to the more advanced stuff. A lot of VIC-20

users will probably be in the younger age groups and will no doubt head for the games programs first. Here they will find such games as *Leapfrog* and *Towers of Babel* to test their abilities. To test them even further (and also very much for adult users) is our tamed game *The Valley*. This was originally published in *Computing Today* and has developed an enthusiastic following, but it is a very large program and so you shouldn't be put off by its size. And if you really can't face keying in the whole thing you can buy the game ready for loading into your micro directly from ASP Software.

Data base programs are becoming something of an 'in-thing' at the moment so we have included such features as *Multipurpose Records*, *Address Book* and *Multicolumn Records* to help you set up your own data base, invaluable for all sorts of information storage and retrieval.

For the more educationally minded, *Quiz Time* and *Micro Examination* are included to help you set up your own system for testing knowledge, be it for fun or for more serious purposes (although I hear my teacher saying that education was fun!).

Those of you who like to delve deeper into their micros, rather than just keying in a program and accepting what it does, will be more interested in the utilities and applications programs herein. You can enlarge the VIC-20's characters using *VIC Blowup*, or you can alter the screen size by

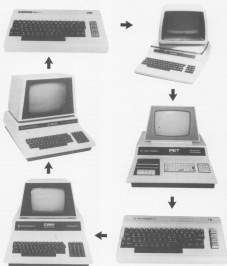
implementing our *VIC Editor* program. And for those of you with more than one Commodore machine at your disposal you can actually get them to pass programs from and to each other after reading *Commodore Communications*. Of course we should never forget the ever present problem of software security and we show a simple but effective way of protecting your own programs in our *Program Protection* feature.

Considering that there must be getting on for 250,000 Commodore micros out there in 'Moss Land' the number of books written on the machines, and their software are more than a few. We have not attempted to present a comprehensive list of the books that are available since that would probably take a whole issue in itself, but we have taken a brief look at the books that we had to hand and that should at least give you a taste of what's around.

We know from bitter experience (and your experience) that there is nothing worse than spending hours typing in a program only to find that it doesn't work. So let us assure you here that we have checked these listings as carefully as possible before putting them into the magazine. And those that have previously appeared in *Computing Today* have been updated with any changes that were brought to our attention. Thus when you find your program doesn't work you should at least initially assume that it is correct in the magazine and you should check it **very** carefully before you contact us. It is often helpful to have someone check it who did not type it in, since it is often difficult to see your own mistakes. If after checking the listing very carefully you still cannot find a typing error (and you have ensured that you are using the correct machine!) then it is better if you write to us rather than phoning in with your query since it gives us some time to try and help and also gives us more time to try to get the next issue of *Personal Software* together!

CONVERSIONS

Getting Converted — Hints on how to convert programs from one Commodore machine to another.



GETTING CONVERTED

Commodore must be one of the very few microcomputer manufacturers who have tried to maintain at least some degree of compatibility between the various computers that they make. Before you can make any kind of start at adapting a program from the magazine for, let's say, a VIC-20 to run on a standard 3000 series PET there are a number of important points to be observed. As usual with a range of machines that has developed across a long period of time minor, subtle variations occur within each particular model — in this case we are concerned with the changes to the BASIC language ROMs. You can establish the particular variety of BASIC fitted to your machine just by looking at the screen of the machine when you turn it on.

TRIED AND TESTED

If you've got an original PET 2001-8 system, those were the ones with a calculator style keyboard, you've got BASIC 1.0 and, unless you've had the new ROMs fitted, certain POKEs to the operating system memory (addresses less than 1024) will definitely need to be changed. As we could fill the entire magazine with the list of these I suggest that you consult one of the books referenced at the end of this article! If your PET has a 'clever' keyboard or is labelled as being a CBM 3008, 3016 or 3032 then you've got another version of BASIC called BASIC 2.0. This is also the version of the language supplied on the VIC-20 and Commodore 64 systems. Owners of 4008, 4035, 4032 or 8000 series machines have yet another variant known as BASIC 4.0 which is equipped with built-in disc operating commands.

As if these changes weren't sufficient to make you doubt just

earlier statement that Commodore machines were generally compatible with one another there is another, more substantial difference between certain models. This area of change is the size of the display screen and, as a direct consequence, the locations in memory left for the display to occupy. Original 2001-8 machines, the 3000 series, the 4000 series and the 'Fat Forty' 4000 models with a 12" display instead of the usual 9" one have a display format of 25 lines of 40 characters. The new Commodore 64 system also has a 25 by 40 screen format. When Commodore introduced the VIC-20 they decided, for reasons unknown, to provide a screen format of 23 lines of 22 characters. Not satisfied with this unusual format they also provided a second area of memory arranged in the same way to act as a colour memory. Table 1 shows the necessary details for all the current Commodore machines.

Having astounded programmers by coming up with the VIC-20 screen format the company attempted to reduce the balance somewhat by giving the business oriented 8000 series machines a decent 25 lines of 80 characters. However, the launch of the Commodore 64 saw a return to the tried and tested 25 by 40 format. The upshot of this is that programs which require direct access to the screen memory by means of the PEEK and POKE commands will need substantial changes in this area. Once again Table 1 will be of assistance as this gives the various memory addresses for the screen and colour maps.

GETTING IT IN

Actually keying the programs contained within these pages

Here we attempt to show you how to go about converting programs for one Commodore machine so that it will run on your particular one.

into your Commodore machine is a simple task, it is little laborious. You can generally make any necessary changes as you go although a screen planning grid will be an invaluable aid for re-formatting the display. If you're loading a VIC-20 program into anything else other than the Commodore 64 you must omit all references to the second display memory otherwise weird and wonderful things will happen as your program only POKEs numbers into its own memory! The VIC chip itself should also be marked down as a candidate for removal, once again the presence of Table 1 should prove helpful.

Programs which use specific facilities provided by the Operating System are generally well commented and in several cases suggestions will be made regarding changes between the various machines. Very specific commands to do with the input or output of information to cassette or disc are included in some of the programs, notably the data base systems and the two-pass assembler. The type of disc unit required is either specified in the text or the code is written in such a way that any of the various Commodore discs will work. If you don't have discs and want to make use of cassette storage instead simply look at one of the programs that uses cassette storage techniques and substitute these for the disc code. All the Commodore machines handle the cassette in the same way.

For those converting programs onto the VIC-20 a special word of caution. The amount of memory, and its location within the memory map, for the storage of programs can be a tricky point. Because there are a number of different sized RAM packs available: 3K, 8K, 16K etc you

do need to know which variant the program has been written for! The problem is not that the program might not fit into the available RAM, although that obviously does come into it, but that the screen memory map moves around the RAM

depending on how much extra memory you have plugged in. This can cause severe problems with PEEKs and POKEs to screen locations so you should be aware that if you are keying the program into a VIC-20 which has a different amount of RAM to that specified in the article you may well have to make changes.

LOADING FROM TAPE

The tape format used by Commodore is consistent right across the range of machines so if you have a tape of a VIC-20 program there should be nothing to stop you **LOADing** it into, say, a 3000 series CBM machine. The only problem is that both the VIC-20 and the Commodore 64 store their programs in a different place in memory to the various PET type machines. However, all is not lost as you can make use of the built-in monitor program on the PETs to resolve the problem. If you are working with a VIC-20 that has an extra 3K of RAM attached your programs will load correctly as though they were created on a PET, you'll still have to make the other changes though!

Programs that **LOAD** in but aren't there when you type **LIST** haven't been lost for good, they are merely hiding somewhere in memory. There are a number of ways to get the programs back to the right place but the following method is probably the easiest of all.

- 1) Ensure that the PET's memory is clear before you **LOAD** — turning it off is often the easiest method.
- 2) **LOAD** the VIC-20 or Commodore 64 program from tape.
- 3) Try to **LIST** it — you never know!
- 4) Assuming that you get no joy from **LIST** perform the following

sequence of inputs, everything you need to type is **underlined>**. The **<CR>** symbol simply means that you should press Carriage Return at this point.

```
READY
PULLDOWN <CR>
PA 00 00 00 00 00 00 00 00
-> 0000 0000 00 00 00 00 00
```

If the figures given under the letters pa, irq etc don't match yours it doesn't matter, they depend on what's been going on inside the machine and will not affect what we are about to do. Now, type in the following:

```
> 0,0000,0000 <CR>
-> 0000 00 00 00 00 00 00 00
```

Using the cursor control keys move the cursor up the screen so it rests over the first 0 after 0400 and then edit the line so that it looks like this:

```
>> 0000 00 01 00 01 00 00 00
```

Press **<CR>** and the cursor will move down to the next line. This example will link a brand new line of program at the bottom of memory to a Commodore 64 program which starts at 0800 Hex. The link byte is the second and third pair of digits after the 0400. If we wanted to link to a VIC-20 program that was generated on an unexpanded machine the line would have read:

```
>> 0000 00 01 10 01 00 00 00
```

Similarly, for an expanded VIC-20 whose programs start at 1200 Hex we would have edited the line to read:

```
>> 0000 00 01 12 01 00 00 00
```

Now, all that remains is to go back to BASIC so type the following:

```
>> 0 <CR>
READY
```

3) Now, **LIST** the program. Surprised? Well, you created a new line of program within the computer's memory which you

linked to the beginning of the program that you loaded from tape, that's why there's a new line 1.

6) As you don't want the new line 1 delete it by typing **1<CR>** and then **LIST** the program again. As the computer had to delete a line of a program from its memory it needed to tidy up all the others and has kindly moved them all down into the correct part of memory for you. You can now modify, **SAVE** or **RUN** the program quite normally.

7) The purists among you will realize that we aren't quite finished as we should really alter the variable pointers. To do this is not that difficult, the following sequence shows how.

```
READY
PULLDOWN <CR>
PA 00 00 00 00 00 00 00 00
-> 0000 0000 00 00 00 00 00
```

The system will now proceed to dump the contents of its memory onto the screen at high speed! Use the 'slow-down' key to reduce the speed to a manageable rate and watch for the apparently random selection of numbers to change to nothing but as as as as etc. At this point we have reached the end of the program. Stop the listing by pressing the **RUN/STOP** key and you should have something like this (once again the actual numbers don't matter):

```
>> 1000 00 00 01 00 00 00 00
>> 1000 00 00 01 00 00 00 00
>> 1000 00 00 01 00 00 00 00
>> 1000 00 00 01 00 00 00 00
>> 1000 00 00 01 00 00 00 00
```

Now enter the following:

```
>> 0,0000,0000 <CR>
```

The system will display something like:

```
>> 0000 01 00 00 17 00 17 00 17
```

Using the cursor keys again edit the line to read:

```
>> 0000 01 00 00 00 00 00 00
```

and press **<CR>**. You have

now told the interpreter that the program ends at the location 1403, the first occurrence of aa after the three pairs of 00 which indicate the end of a program. You have also informed the interpreter that it can store its variables in memory immediately after the end of the BASIC program. A quick tip for those with corrupted programs is to restore these three pairs of locations to the address the first byte after the end of the program; you lose your variables but you get your program back! All that remains is to return to BASIC by typing g <CR> and listing the program. The advice will have realized that I've made life much easier for the person attempting this for the first time by filling the computer's memory with nothing, indicated by the appearance of aa all through the memory. This makes it significantly easier to search for the end of a program, under normal circumstances you would have to scan for the 00 00 00 end of program marker which is a chore.

A MATTER OF STANDARDS

If you are unfamiliar with the system of standard symbols developed in *Computing Today* over the last four years to simplify the printing of graphics and cursor control characters then you may well be somewhat confused by the appearance of square brackets with such expressions as CLS inside them. We use this convention to overcome the problems of reproducing the special symbols that a number of computers generate to indicate such as the PET's reversed Heart character which indicates a Clear Screen command. Figure 1 gives the complete list of cursor control codes that you will find in the programs.

The Commodore range also sports a built-in set of pre-programmed characters which can be accessed directly from the keyboard or POKED into place. Because these characters cannot be successfully

reproduced we use a similar convention to that outlined above to include these in the listings. Figure 2 shows how it is arranged for the Commodore machines but take note that on the VIC-20 you have two graphics characters per key so the DG-@ and DG-@ conventions must be used here. The VIC-20 and Commodore 64 also allow colour commands to be embedded in PRINT statements. As these are generated by a control code sequence we show these as (CTL BLU) which indicates that at this point both the Control key and the Blue colour key should be pressed together. Both the VIC-20 and the Commodore 64 have four programmable function keys on the right of the keyboard and these are represented by the (Fnx) convention where a x is a number between 1 and 8.

CLS	Clear Screen
HOME	Home cursor
CLR	Cursor Left
CR	Cursor Right
CU	Cursor Up
CD	Cursor Down
REV	Reverse video on
OFF	Turn it OFF
SP	Space
CTL	Control key
Fnx	Function key
BLU	Graphic Left
GRN	Graphic Right

Fig. 1. The complete set of cursor control codes and their program representations.



Fig. 2. The way we indicate block graphics on machines like the PET. The VIC system of Shift Left and Shift Right is shown in Fig. 1.

PET 2000/CBM 3000/CBM 4000

Screen memory	32768 to 32767
Colour memory	Not applicable
BASIC starts at	3234 (0400 Hex)

CBM 8000

Screen memory	32768 to 34067
Colour memory	Not applicable
BASIC starts at	3234 (0400 Hex)

Unexpanded VIC-20

Screen memory	7680 to 8185
Colour memory	38400 to 38905
BASIC starts at	4096 (1000 Hex)

VIC-20 plus 3K

Screen memory	7680 to 8185
Colour memory	38400 to 38905
BASIC starts at	3234 (0400 Hex)

VIC-20 plus 8K or 16K

Screen memory	4096 to 4607
Colour memory	32888 to 38399
BASIC starts at	4608 (1200 Hex)

Commodore 64

Screen memory	3234 to 3233
Colour memory	56296 to 56295
BASIC starts at	2048 (0800 Hex)

Table 1. The locations of the screen memory maps, colour memory maps and the start of BASIC programs for the various Commodore systems.

Bibliography

The PET Illustrated by Nick Humphrey
 PET Personal Computer Guide by Adam Osborne, Ian Strawn and Ellen Strawn
 PET Graphics by Nick Humphrey

IT'S HERE!



Buying or selling?
Now's the time to
take a look.

- **LATEST** computers featured and their performance assessed by Home Computing Weekly's team of resident experts.
- The **LATEST** news and views on personal computers.
- Inside information for the computer enthusiast.
- News coverage like you've never seen before.
- Over five pages of software reviews each week.

OUT NOW AND EVERY TUESDAY

Don't be left without — cut along dotted line and give this coupon to your newspaper.

Dear Newspaper,
Every Tuesday, please reserve me a copy of

Name:

Address:

.....

.....

Thank You



Let Commodore expand your horizons.

VIC 20 is the finest home computer that money can buy.

And the better you get to know it, the more confident, adventurous and ambitious you'll become.

You'll want to take advantage of the vast range of VIC software: a superb and constantly-growing selection of programs, embracing business systems, entertainment, education and many applications in the home.

Every program in the series has been designed by experts, and chosen for its quality and value for money.

VIC business software covers a wide range of applications, including spread-sheet analysis, stock control, information handling and word-processing.

A mind-blowing range of games including Scott Adams' world-famous 'Adventure' series.

Advanced space games, including the sophisticated 'Omega Race'.

Learn subjects as diverse as English Language, programming, and biology.

And 'home' software ranges from IQ tests to Robert Carrier menus.

In addition, there is a range of VIC software, like programmers' aids and graphics packages—

to add to your understanding and enjoyment of computers and computing.

There's even a special 'VicSoft' Club for VIC 20 enthusiasts, with many advantages including special offers to club members.





VIC software will expand your horizons. And your mind.

PRICES RANGE FROM \$4.99 to \$24.95 INC. VAT.

commodore
VIC 20

For more information, a catalogue of VIC software and details of your local retailers or dealers please phone or complete this coupon and send to:
The Commodore Information Centre,
875 Ayles Avenue, Slough, Berkshire SL1 4AG.
Telephone: Slough (0753) 75225.

Name

Address

Express

VIC 200 04 03

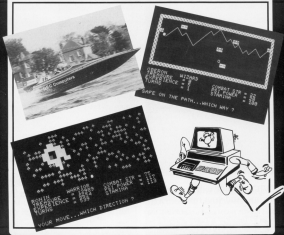
GAMES

Leapfrog — Our micro version of that old pub game played with coins. Originally published in **Computing Today**, January 1982.

Power Boat — Have all the fun of power boating without feeling seasick. Originally published in **Computing Today**, March 1982.

The Valley — Save the kingdom of the Valley by combatting Dragons, Balrogs and Wraiths. Choose your character with magic or physical strengths and do battle in our epic game, The Valley. Originally published in **Computing Today**, April 1982.

Towers of Braham — Moving rings from one pillar to another may sound easy but just try it. Originally published in **Computing Today**, August 1982.



LEAPFROG

The program Leapfrog is about a relatively simple but very interesting board game which is often played in

pubs with coins. In its usual form there are six coins and seven spots, as shown in Fig. 1. The purpose of the game is



Fig. 1. The basic of the Leapfrog game is to move the coins on the left to the right and those on the right to the left. It looks easy enough ...

VARIABLES

A\$,\$P1,\$2	Temporary variables
SP1,\$P(N)	Screen positions
C,C1,C2	Counters
A(N)	Record of where spot is, ie Left, Right or Middle
B(N)	Record of end status of points for checking
S	Character number for screen display
P1	Temporary variable for \$P(N)
Z	Number of spots in game
Z1,Z4,Z1	Left, Middle and Right specific values of Z
C8,N8	Current spot and New spot. That is, numbers of spot's last and new positions
R8	Record of moves made.

PROGRAM STRUCTURE

100-180	Introduction
170-930	Sets up example on screen, describes game and its rules
940-1280	Chooses game and sets up appropriate screen display
1290-1450	Spot to be moved and where to, and checks legality of move
1460-1560	Moves Left to Right
1570-1670	Moves Right to Left
1680-1730	Spot is empty message
1740-1750	Spots too far apart message
1800-1850	Spot is not empty message
1860-1910	Wrong direction message
1920-2140	Checks for solution
2150	Makes spots on screen
2220	Numbers spots on screen
2250	Removes a large message from top of screen
2300	Removes a small message from top of screen
2350	Removes a message from bottom of screen
2400	Holds screen until key is pressed.

A game, its concepts, planning and implementation for the PET.

to interchange the two sets of coins, and the rules are very simple:

1. The LEFT coins can move only to the RIGHT and the RIGHT coins can move only to the LEFT.
2. There are two possible types of move:
 - (a) to an adjoining empty spot;
 - (b) over another coin to an empty spot (like in checkers).

Of course, the number of spots can be any odd number so that, for example, there might be four coins on each side and a spot in the middle making nine spots altogether. The program allows you to choose between three, five, seven and nine spots.

GAME STRATEGY

There are two distinct advantages to playing the game on a computer VDU. In the first place, the initial stages of becoming clear about just what is permitted in a game are much more easily handled on a computer. Because, while it will obviously allow you to make wrong or foolish moves, it will not allow you to make illegal or incorrect moves. This means that the machine forces you to learn the rule structure of the game very quickly and forces you further into concentrating on strategy.

The second advantage is that it is possible for the computer to keep a record of all the moves made in any particular game. At the end of the game this record can be assessed and studied. In this way winning moves can be detected, winning patterns become clear and the structure of the game emerges.

Like all games the initial motivation for playing is pleasure, and this should not be minimised in any way. So, at this stage all readers ought to go off, type in the game and

play it for a while. But eventually a pattern of movements will begin to emerge and curiosity will develop about how the game might be symbolized and its structure analyzed. When that stage is reached the educational and, in this case mathematical, aspect of the game began to become more interesting and that is the ideal outcome. That is to say, not only is there pleasure to be gained, but eventually there is an educational payoff.

PROGRAM ANALYSES

This next section is an attempt to analyze and generalize this particular game and the program can be used at each stage to test the results and conclusions. But remember that this is a second-level activity and ought to follow some experience at playing the game. When you read this next part you will know how to succeed at the game and how to generalize and the interest will be purely academic.

The simplest version of the game looks like this:



A few quick trials will show that the successful strategy is very simple and can be coded as: LRL (RLR).

This means:

- First move the L from 1 to 2.
- Then move the R from 3 to 1.
- Then move the L from 2 to 3.

The next version of the game looks like this:



In this case the successful strategy of you start by moving 2) is coded as: L RR LL RR L. This means:

- First move an L piece.
- Then move two R pieces.
- Then move two L pieces.
- Then move two R pieces.
- Finally move an L piece.

```

100 REM ***** LEAPFROG *****
110 REM *****
120 REM *****
130 REM *****
140 REM *****
150 REM *****
160 REM *****
170 REM *****
180 REM *****
190 REM *****
200 REM *****
210 REM *****
220 REM *****
230 REM *****
240 REM *****
250 REM *****
260 REM *****
270 REM *****
280 REM *****
290 REM *****
300 REM *****
310 REM *****
320 REM *****
330 REM *****
340 REM *****
350 REM *****
360 REM *****
370 REM *****
380 REM *****
390 REM *****
400 REM *****
410 REM *****
420 REM *****
430 REM *****
440 REM *****
450 REM *****
460 REM *****
470 REM *****
480 REM *****
490 REM *****
500 REM *****
510 REM *****
520 REM *****
530 REM *****
540 REM *****
550 REM *****
560 REM *****
570 REM *****
580 REM *****
590 REM *****
600 REM *****
610 REM *****
620 REM *****
630 REM *****
640 REM *****
650 REM *****
660 REM *****
670 REM *****
680 REM *****
690 REM *****
700 REM *****
710 REM *****
720 REM *****
730 REM *****
740 REM *****
750 REM *****
760 REM *****
770 REM *****
780 REM *****
790 REM *****
800 REM *****
810 REM *****
820 REM *****
830 REM *****
840 REM *****
850 REM *****
860 REM *****
870 REM *****
880 REM *****
890 REM *****
900 REM *****
910 REM *****
920 REM *****
930 REM *****
940 REM *****
950 REM *****
960 REM *****
970 REM *****
980 REM *****
990 REM *****
1000 REM *****

```

[illegible]

Listing 1. The program for Hamilton

Notice that the number pattern is 1 2 3 2 1. The strategies for versions of the game with three, five and seven spots are as follows:

LAL
L. NUN. L. AR. L.
L. AR. L. L. AR. L. L. AR. L.

Turn these into number patterns as before:

111

and the nine spot version between, one would guess

1344371

This means, begin by moving one L piece, then two R pieces, then three L pieces, and so on. You can try this out on the nine-spot version on the computer and it does work.

At this stage the pattern is so simple and obvious that it is quite easy to succeed at any version of the game. But its analysis is not quite complete. The general case can be symbolised as the case with $2N + 1$ spots. That is, N spots on the left, N spots on the right, and one spot in the middle. So the pattern of movements can be shown as:

123 888 50

That is, one LEFT move, two RIGHT moves, three LEFT moves and this continues until N moves are necessary. This number of moves is, N occurs three times, and then the number begins to reduce by one each time until one move only is necessary. At this stage the moves are over.

The final question may well be, what is the total number of moves necessary at each level of the game, and this generalized pattern can be used to answer this. In the general case, the total number of moves is:

$$(1+2+3+\dots+N)+N+(N-1+\dots+3+2+1)$$

That is, $2\|M\| = \|M\| + \|M\| + 2\|M\|$.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 5 of this issue.



Photography courtesy Enterprise Pictures Limited.

```

640 PRINT
650 IF (1-((50-80) THEN 1000
660 GOTO 1000
670 IF (80-100) THEN 1000
680 IF (100-120) THEN 1000
690 IF (120-140) THEN 1000
700 PRINT "YOU TOOK THE CORNER FOR TOO WIDE!"
710 GOTO 1000
720 IF (140-160) THEN 1000
730 PRINT "YOU GOT THAT CORNER TOO CLOSE!"
740 GOTO 1000
750 GOTO 1000
760 IF (160-180) THEN 1000
770 IF (180-200) THEN 1000
780 IF (200-220) THEN 1000
790 IF (220-240) THEN 1000
800 IF (240-260) THEN 1000
810 IF (260-280) THEN 1000
820 IF (280-300) THEN 1000
830 IF (300-320) THEN 1000
840 IF (320-340) THEN 1000
850 IF (340-360) THEN 1000
860 IF (360-380) THEN 1000
870 IF (380-400) THEN 1000
880 PRINT "CONGRATULATIONS!! YOU WON!"
890 PRINT "FINISHED BUT VICTORY ISN'T THE END!"
900 GOTO 1000
910 IF (400-420) THEN 1000
920 IF (420-440) THEN 1000
930 IF (440-460) THEN 1000
940 IF (460-480) THEN 1000
950 IF (480-500) THEN 1000
960 IF (500-520) THEN 1000
970 IF (520-540) THEN 1000
980 IF (540-560) THEN 1000
990 IF (560-580) THEN 1000
1000 IF (580-600) THEN 1000
1010 IF (600-620) THEN 1000
1020 IF (620-640) THEN 1000
1030 IF (640-660) THEN 1000
1040 IF (660-680) THEN 1000
1050 IF (680-700) THEN 1000
1060 IF (700-720) THEN 1000
1070 IF (720-740) THEN 1000
1080 IF (740-760) THEN 1000
1090 IF (760-780) THEN 1000
1100 IF (780-800) THEN 1000
1110 IF (800-820) THEN 1000
1120 IF (820-840) THEN 1000
1130 IF (840-860) THEN 1000
1140 IF (860-880) THEN 1000
1150 IF (880-900) THEN 1000
1160 IF (900-920) THEN 1000
1170 IF (920-940) THEN 1000
1180 IF (940-960) THEN 1000
1190 IF (960-980) THEN 1000
1200 IF (980-1000) THEN 1000

```

Listing 1. The program for the power boat race game.

cause your boat to go straight. This has surprisingly been done to make the navigation easier, but be warned! The ratio between the rudder and the radius varies from 1 to 10 when stationary, to 1 to 5 at top speed just to make the game a little more difficult. To complete the course you must also regulate your speed to a value near to or, preferably, slightly above the maximum speed for the section.

The power to the drive is not the speed, but the amount of power you supply to the drive. The more power you supply to the drive the faster the boat will go. If you put too much power in, you run the risk of converting your boat into an aeroplane and flipping over. Life jackets are compulsory whilst using this program, in case you get a bit too excited!

The program is written in a standard PET Microsoft BASIC and should present no real difficulty in converting to other machines. Note that PRINT CHR\$(147) is a Clear Screen command and line 940 can be replaced with 940 INPUT A\$:—line 950 should obviously be deleted.

For those of you who wish to alter the game the following list of variables may help.

- C is the difference between actual and rated turning circles.
- D is the distance to the leader.
- E is the previous 'power to the main drive'.
- F is the present 'power to the main drive'.
- F is the amount of fuel remaining.
- M is the maximum speed for the section.
- P is your position.
- R is the rated turning circle.
- S is your present speed.
- T is the rudder position.
- W is the prevailing weather condition.
- X is the section number.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

Personal SOFTWARE

Personal Software is a quarterly publication dedicated to all aspects of software for the most popular micros.

Carrying on the trend of our system oriented issues the Autumn edition of Personal Software will contain a veritable host of material for the Apple II, Apple II+, and Apple IIfx. So if you've got an Apple system you simply can't afford to miss this great issue.

Personal Software can be ordered directly from us at £7.50 per annum or £1.85 per copy. To ensure a single copy or a complete year's supply, fill in the form below, cut it out and send it with your cheque or postal order (made payable to DSP Ltd) to:

Personal Software
Subscriptions,
513 London Road,
Thornton Heath,
Surrey CR4 6AR.

— you can even spread the load with
your credit card!

Don't miss out — subscribe now.



SUBSCRIPTION ORDER FORM

Cut out and SEND TO :

Personal
SOFTWARE

513 LONDON ROAD,
THORNTON HEATH,
SURREY,
CR4 6AR.

**POST FREE
SUBSCRIPTION**

Please use BLOCK CAPITALS and include post codes.

Name (Mr/Mrs/Ms)

Address

Signature

Date

Please continue my subscription to Personal Software with the same

SUBSCRIPTION RATES: (tick ☐ on appropriate)

☐ £7.50 for 4 issues
£15

☐ £1.85 for a single
copy/issue ☐ issue ☐

(All enclosing my cheque as necessary)
Cheques Please! (Order in International Money)

Order for £
(made payable to DSP Ltd)

OR

Debit my Access/Barclaycard*
(*delete as necessary)



Send me a...

--	--	--	--	--	--	--	--	--	--

THE VALLEY

Join the thousands of brave adventurers who have dared to enter our multi-scenario role-playing game.

The news spread quickly throughout the lands of Tybollen. Vourim, mightiest wizard of the Northern Reach, had arrived at the gates to the Princess Evanna's castle offering his aid against the Selric hordes which besieged her realm.

Her magical powers alone too weak to vanquish the foe, Princess Evanna eagerly accepted Vourim to her side. Together forming a psychic bond they wove a spell, powerfully constructed from the forces of light and darkness, to drive the savages from Tybollen's aid. Their combined magic scorched the Northern border-lands, scattering the enemy's host and laying waste the Selric threat forever.

In gratitude, Princess Evanna invited Vourim to make his home in her Kingdom and bestowed upon him the title 'Lord of the Valley Between Two Castles'. Knowing the land between the Princess' castle and her brother Xeron's to be most beautiful country, Vourim accepted the honour and began

plans to build two strongholds in the forests of the Valley.

Time passed... far away from the village settlements, Vourim's Lair (as his strongholds had become known) were often be the subjects of whispered conversation in the ale-houses of Tybollen. Even the Princess Evanna's counsellors felt that the Princess had closed her eyes to the changes that had overtaken Vourim during the years he had attended the castle as her chief advisor. He had become quiet and withdrawn, only visiting the castle at the dead of night. It was even rumoured he had entertained in his strongholds members of the White Order, an evil brotherhood of wizards from the Southern Slopes.

Following just one of these visits from the White Order, Vourim had begun building two temples dedicated to the worship of an obscure lizard-like god, Y'Naploth.

Shrouded by evil swamps, it seemed as though none could stop the wizard from carrying out his ancient sacrificial rituals. At first, the Princess listened to stories of livestock disappearing and of children running off with an air of humour; but soon even she could not deny her ears to the allegations of the high taste and cruelty of which her people complained. However, it was only when her war-like brother, Xeron, seemed to wither away in his sick-bed from the 'medicines' administered by

Vourim that Princess Evanna began to see the threat posed to her throne.

Arranging a Council of War with her neighbouring Lords, Princess Evanna asked them to pledge their allegiance and grant her the aid she needed to crush the evil wizard. There was much brave talk and long discussion but eventually the Lords decided not to intervene. The worship of Y'Naploth had spread and the peoples of Tybollen would likely as not support the wizard, High Priest of the lizard faith, rather than their over-lords.

The Lords quite clearly feared Vourim more than the Princess and rather than follow their heart's desires chose the easy route. The Princess was disheartened and, clearing them from her Council Chamber, slumped into her throne deep in thought. She could destroy this wizard, she mused, but at what cost?

As dawn broke, the Princess' meditations were interrupted by a young wizard by the name of Alarian. She recognized him instantly as the novice attached to Baron Ntall, Lord of the Eastern Plains — one of the Lords she had expected would grant her the aid she would need. Although far below the Princess in magical prowess, Alarian was able to offer the wealth of his experiences as a youth apprenticed to the mighty Vourim back in the Northern Reach. The young wizard also gave the Princess his copper amulet, studded with six precious gems — Alarian's amulet was a magical device, providing its wearer with the gift of life after mortal death.



Mounting her horse at the castle gates, Princess Evanna made one last desperate attempt to encourage her people to her side; the Lords looked away and her subjects feared. So, uttering a curse, the Princess Evanna set off to leave Youvain in his lair.

As she rode, she was saddened by the apparent sickness that hung over the Valley; nothing grew there now, save in the forests and swamps that surrounded Youvain's Lair and the Temples of Y Nagloth. Yet as she rode on she discovered, sheltered in the depths of the Valley floor, another building — a six-storey tower. She recognized the tower with sickening rapidity; she had once seen it in her youth — it was a replica of the Black Tower of Zaxxon, the home of the brotherhood of the White Order. Satisfying herself that the tower was empty, she spurred her mount and raced with renewed vigour towards the demon wizard, Youvain.

Catching the wizard amidst a ghastly blood rite, Princess Evanna began casting a spell of banishment on Youvain. Caught off guard, the Lord of the Valley, screaming vile obscenities, started to fade from sight. With a final blood-curdling scream, he made a final gesture at the Princess before passing from the mortal plane. The Princess, surrounded by dancing lights, fell to the floor writhing in pain. She had been poisoned by Youvain's magic and, with mounting horror, realised this would be a magical, and not a mortal death — the Anulet of Alarion would not help the Princess to cheat death.

Crumpled on the floor of Youvain's Lair, the Princess began to make her last magicks. She had the Anulet in one of the Temples of Y Nagloth and three of the stones she placed on the third floor of the Black Tower of Zaxxon, the fourth stone on the fourth floor, the fifth and the sixth stones coated on the two top floors. Struggling to keep her consciousness, the Princess made one last gesture at the

Heim and as she died, her magick passed into the Heim as it disappeared from sight forever.

The Valley buildings disappeared soon after Youvain's banishment, following him into the ethereal links in which the Princess had imprisoned him. Gradually over the years, the Valley returned to its former splendour. Alarion, satisfied that Evanna's spell was well cast, remained there for many years keeping his eye over the Kingdom. Then one fine morning, the first of Spring, Alarion, leaving a spell of watchfulness over the land, left for other adventures.



Concluding the story, he turned his attention to the Valley lying far beneath his window, blanketed in swirling mists shrouding all but the highest tree tops. On the horizon, clearly silhouetted against the morning sun above the silvery towers of Castle Xeron, nestling on the hill many leagues away. All was still... almost peaceful. "Listen, old man, I've heard your leerie story — just what is all this about?"

At the sound of the gruff voice, the hooded figure of the window swivelled around using his stick for support and, contemplating the six figures seated around his desk, began the slow and painful journey back to his chair.

"It is no leerie story, my friend" the old man muttered as he eased his back against the creaking carvings of the chair back. "I know the tale to be true for I was that young wizard, Alarion. It was I who, tens of thousands of years ago, sat with Princess Evanna helping her to prepare for her battle with Youvain."

Alarion lifted his hand weakly silencing the doubting questions of the company. "Please listen. You would not understand the ways I have prolonged my life, so do not ask. Accept simply that I am Alarion and all I speak of is true. The spell of watchfulness I cast so long ago has called me

here to protect your lands from great danger."

"With all due respect, Sirs, are you not a little late?" said another of the figures, a novice stirred by his appearance. "The Valley has been a place of rumour and mystery, concealed by strange mists for nigh on thirty years."

"I'm afraid" sighed Alarion "that you will find out one day soon that not all magick works as effectively as you would wish. I believe my spell of watchfulness was weakened in much the same way as Princess Evanna's spell of banishment. During my time of apprenticeship to Youvain, I too formed a psychic link with my master hoping to amass power before my time. As the tales would have it, Youvain, through the past mystic bonds with the Princess and I, was able to divert much of the strength of our spells, allowing him to attempt a return to the mortal plane unseen."

"Youvain, Lord of the Valley, is smashing a pathway from the chaos of his world of banishment through to our own. In his present situation, halfway between chaos and reality, he is almost visible to me; his followers and his buildings, the Lair, the Temples and the Black Tower, are already becoming reality again."

"As my wild-eyed barbarian friend pointed out earlier, I am but a frail old man. I can offer nothing but magical aid as I am all but restricted to this chair. Were I stronger, nothing would stop me fulfilling the quest alone but alas, it is to you I look for favour. Will any of you enter the Valley in search of the missing Heim of Evanna in my place?"

At the mention of the legendary Heim, the six figures moved closer around the wizard's desk.

"I can help whomever decides to go" continued Alarion "but I can help only one of you at a time. I can create a path of safety between this castle and Castle Xeron, both of which will prove safe havens during your quest; I can also make the buildings visible to you —

Although this means you will be seen and thus attacked by the inhuman creatures loyal to Youmim.

"You will need great experience to find the Helm of Evanna, such was Evanna's curse on her people — they spurned her when she needed their help to defeat Youmim. Princess Evanna had the means to conquer any threat to the Kingdom so that only the bravest Tyballeans could ever find it. To gain this experience, you would be wise to first search out my Amulet in one of the Temples of Y'Magloth and, once found, journey to the Black Tower of Saezon where you will find the six stones that fit the Amulet. However, care must be taken to find the stones in the correct order — if you don't, you will find they do not fit and will be useless to you.

"Although I have had little contact with my Amulet over past centuries, I am confident I can illuminate areas of residual magic within the Temples and the Black Tower indicating where magical items have been hidden at some time in the past. I will do my best to show you where the Amulet stones have been cached, but I have found that in my later years I have not the concentration I used to have and you may find only worthless baubles instead — I will do my best".

"Well that's not good enough for me" cried a frost-like character jumping to his feet, "I'm damned if I'll follow you through this Valley — I've heard stories of the creatures who dwell there. Sorry — bah!" He spat at Alarian's feet and departed.

As the stone chamber door slid away, Alarian surveyed the five remaining faces, a barbarian, a novice wizard, a cleric, a thief and a

warrior. Hand-picked and all native Tyballeans, Alarian wondered if one of these could achieve the impossible and bring back the lost Helm of Evanna.

"I would not blame you for following him," said Alarian, "the dangers he spoke of are all too real. Over the past weeks I myself have seen through my enchanted glass Dragons, Balrogs, Wraiths, even a creature with the very likeness of Y'Magloth himself, a Thunder Lizard, roaming the Valley.

"You will not, however, enter the Valley unprotected. I will teach you a potent sleep spell and, as you gain experience, will be able to bestow two other spells on you: a mind lance to attack creatures with a high psychic power and a spell which attacks using the very Fires of Hell. However, you will use these spells sparingly as they are extremely dangerous in the hands of the untutored and it takes many years of study before a spell can be cast with no loss of stamina.

"You will doubtless have realized that I am no fighting man" continued the wizard, nodding respectfully towards the warrior and the barbarian, "but if I may offer some advice on hand-to-hand combat. There are three effective ways to fight a creature of great physical strength either strike its head, body or limbs. Obviously an attack to the limbs or body will eventually lead to success, but it may initially cause little damage. A strike to the head may kill the beast in one blow, but will leave you open to return blows while striking. The decision will be yours, I cannot help. However, do not waste your time attacking a purely psychic creature with a sword, they can be defeated by spells alone.

"Care must be taken when approaching any building: the swamps and forests are dangerous — make sure you have the experience to cope. Also, beware of water, you will be considerably weakened by the weight of your armour".

Alarian, bringing his stick to the ground raised himself out of his chair.

"I cannot promise you riches, though treasure there be in the Valley, I ask only that you save the Kingdom. Find the Amulet. Fill it with the six stones and you will have the ability to cheat death, to resurrect yourself within these safe castle walls. It will also prove invaluable in your search for the Helm of Evanna in the dark Lord's lair".

"Alan, I cannot help you much in your search for the Helm. For although I can again guide you to the areas of residual magic, the Helm, on its mistress' instruction, will not reveal itself to you unless you have at least the power of a Warlord. It will be up to you to build up this experience. I can only provide an occasional aura of magic to boost your powers, yet you will find that Youmim also has a way of watching over his followers and may surround you in a circle of evil.

"There is little time for discussion; I have arrived thirty years too late. . . and I hear Youmim knows it. The choice must be made here and it must be made now. Will you go and find the Helm of Evanna and bring it back here?"

Each of the five heads nodded as, in turn, Alarian gazed deeply into their eyes. Setting his hands on the carved walking stick at his side, the elderly wizard spoke to the assembled company in a low rumbling voice.

"... who then will be first?"

The Valley is best described as a real-time adventure with graphics. It was developed as a direct result of reviewing and testing a large number of commercial offerings over Christmas 1990 and, we hope, overcomes many of the

failings of these various alternatives.

The program has always been based on a modular system to simplify both its production and its documentation; this approach also means that the game can

be tailored to suit the player's individual tastes. Again, because of the modular nature of the program it can easily be expanded provided you have more than the required 18K.

The published listing was developed and tested on a 32K

Commodore PET but will, if all non-essential spaces and REMs are removed, run in 16K. All graphics characters and cursor controls have been converted to C1's standards and character tables are provided. We are not suggesting that conversion to other systems is something that can be done in an evening but it is possible — we have versions running on Commodore HD32, TRS-80, Sharp MZ-80K, Sharp MZ-80A, NEC Model A and Model 5, VIC-20 (16K), Dragon 32, Atari 400 and 800 (32K), ZX Spectrum (48K), Apple II (disc + 48K) with others on the way soon!

The best way to implement the program on your system is to key it in one module at a time following the notes. We have broken the listing down into the separate modules, each with a

description of its main functions, to make this simpler. As each block is completed, SAVE it on tape before adding the next; 16K is a lot of programs to lose if you make a mistake!

PLAYING THE GAME

The objects of the game are explained in the introductory scenario, the actual mechanics of playing are described in the various sections that follow. Probably the best plan is to create a number of different characters, one of each type perhaps, and attempt to play each of them through the Valley and various scenes. The ultimate object of the game is to reach the highest rating level, 28, but along the way you will need to collect the various

special treasures to ensure that if you are unfortunate enough to be killed you stand a chance of re-incarnation.

Game tactics are dependent on the type of characters you have chosen and are best developed by the player as the game progresses. A couple of hints may be welcome, however. If you are in combat with a monster that is stronger than you and are suffering great damage, then Spell 1 is possibly the best option to select. The other important tip is to remember that once you enter a scenario other than the Valley you are committed for a number of turns, so ensure that your stamina level is high.

However, before you can play the game you must enter the program so now is the time to start pressing keys!

INITIALISATION

Although it would be regarded as 'proper' to declare all the variables used at the start we have only initialised those necessary to begin the program correctly. Arrays are all DIMensioned to their correct

sizes at this point and the vital positioning strings of cursor movements are also created.

The dummy READ routine between 300 and 320 moves the current position in DATA over the first block which will be used later for building castle-type scenes.

Monster data is loaded into three arrays; the monster name, its initial strength and its initial magical power. These starting values are modified according to the 'floor level' of the scene on which it appears. The monster details are presented in Table 1.

```
100 DIM ** DUMMY READ TABLE
110 DIM D(10),D(10),D(10),D(10),D(10)
120 DIM M(10),M(10),M(10),M(10),M(10)
130 DIM **DUMMY**DUMMY**
140 DIM DUMMY(10),D(10)
150 DIM **DUMMY**D(10)
160 DIM D(10),D(10),D(10)
170 DIM D(10),D(10),D(10)
180 DIM D(10),D(10),D(10)
190 DIM D(10),D(10),D(10)
```

```
200 DIM D(10),D(10),D(10)
210 DIM D(10),D(10),D(10)
220 DIM D(10),D(10),D(10)
230 DIM D(10),D(10),D(10)
240 DIM D(10),D(10),D(10)
250 DIM D(10),D(10),D(10)
260 DIM D(10),D(10),D(10)
270 DIM D(10),D(10),D(10)
280 DIM D(10),D(10),D(10)
```

CHARACTER INITIALISATION

This block of program allows the user to set up his character with a name and one of a number of options of character type: Wizard, Cleric, Barbarian, etc. Table 2 contains information on the various character types.

Alternatively, if the game has been played before, the user may have a character stored on tape so the option exists to load this instead of starting afresh. The selection is made in 1000 after the name has been entered. The maximum length of name is 16 characters (checked in line 1040) and,

because the string has to be entered as an INPUT, a simple bomb-proof trap is inserted at

line 1000. This works by forcing an asterisk to appear under the cursor so if you simply press

CHARACTER TYPES...CHOOSE CAREFULLY

```
WIZARD (4)
THUNDER (4)
BARBARIAN (4)
WARRIOR (4)
CLERIC (4)
```

WIZARD

GOOD LUCK

WIZARD WIZARD WIZARD

The first time you enter the game you can select your character type from the five available.

Monster	Physical	Magical	Code	Score
Thunder Lizard	50	0	V	V
Wolven	9	0	A	V,W,S,C
Hob-Goblins	9	0	A	V,W,S,C
Orc	9	0	A	V,W,S,C
Ogre	23	0	A	V,W,S,C
Balrog	50	50	A	V,W,S,C
Fire Imp	7	3	E	V,W,C,L,T
Harpy	10	12	E	V,W,C,L,T
Fire Giant	26	20	E	V,W,C,L,T
Rock Troll	19	0	G	V,C
Centaur	18	14	H	V,W
Wyvern	36	12	F	W,S
Water Imp	15	15	L	W,S
Kraken	50	0	L	W,S
Minotaur	35	25	C	C,L,T
Wraith	0	20	C	C,L,T
Hing Wraith	0	45	C	C,L,T
Narrow Wight	0	25	B	L,T
Dragon	50	20	B	L,T

Table 1. The monsters, their strengths and their habitats. The code letter is entered during the monster selection routine and matched against the current warrior's allowable monster rating. Note also that the two monsters coded with L can only be found in the lakes. The scores are coded on: V=Valley, W=Woods, S=Swamps, C=Cliff Tower, L=Wizard's Lair and T=Temple of T'Nagash.

Return this is entered rather than nothing.

If date entry is from tape, the stamina value is set to maximum and the program then jumps to 1400. If, however, this is the first time through the game, the tape loading section is skipped and the player offered the choice of five character types. According to your selection, the initial values of your character's physical and magical strengths are determined together with your two 'gain' factors. These determine how much you gain from finds and how well various sections of the combat work. The FI factor also acts as a limit on your character's growth; see Table 2.

Once the character has been selected, the Valley is then created for the first time and the initial status information displayed. The game now starts with your character located in the left-hand 'hole' code and control is passed to the Movement routine.

```

1000 DIM A% CHARACTER TYPES AND CODES
1010 PRINT "CHARACTERLINE A CHARACTER FROM TAPE (Y/N) ?"
1020 INPUT A%
1030 IF A%="Y" THEN GOTO 1000 ELSE GOTO 1040
1040 PRINT "TYPE CHARACTER'S NAME (1 CHAR) C/L/T/S"
1050 INPUT C%
1060 IF C%="C" THEN GOTO 1070
1070 IF C%="L" THEN GOTO 1080
1080 IF C%="T" THEN GOTO 1090
1090 IF C%="S" THEN GOTO 1100
1100 PRINT "PLACE PLACE NAME IN THE NAME BOX"
1110 INPUT NAME
1120 GOTO 1130
1130 PRINT "NAME"
1140 PRINT "NAME"
1150 PRINT "NAME"
1160 PRINT "NAME"
1170 PRINT "NAME"
1180 PRINT "NAME"
1190 PRINT "NAME"
1200 PRINT "NAME"
1210 PRINT "NAME"
1220 PRINT "NAME"
1230 PRINT "NAME"
1240 PRINT "NAME"
1250 PRINT "NAME"
1260 PRINT "NAME"
1270 PRINT "NAME"
1280 PRINT "NAME"
1290 PRINT "NAME"
1300 PRINT "NAME"
1310 PRINT "NAME"
1320 PRINT "NAME"
1330 PRINT "NAME"
1340 PRINT "NAME"
1350 PRINT "NAME"
1360 PRINT "NAME"
1370 PRINT "NAME"
1380 PRINT "NAME"
1390 PRINT "NAME"
1400 PRINT "NAME"
1410 PRINT "NAME"
1420 PRINT "NAME"
1430 PRINT "NAME"
1440 PRINT "NAME"
1450 PRINT "NAME"
1460 PRINT "NAME"
1470 PRINT "NAME"
1480 PRINT "NAME"
1490 PRINT "NAME"
1500 PRINT "NAME"
1510 PRINT "NAME"
1520 PRINT "NAME"
1530 PRINT "NAME"
1540 PRINT "NAME"
1550 PRINT "NAME"
1560 PRINT "NAME"
1570 PRINT "NAME"
1580 PRINT "NAME"
1590 PRINT "NAME"
1600 PRINT "NAME"
1610 PRINT "NAME"
1620 PRINT "NAME"
1630 PRINT "NAME"
1640 PRINT "NAME"
1650 PRINT "NAME"
1660 PRINT "NAME"
1670 PRINT "NAME"
1680 PRINT "NAME"
1690 PRINT "NAME"
1700 PRINT "NAME"
1710 PRINT "NAME"
1720 PRINT "NAME"
1730 PRINT "NAME"
1740 PRINT "NAME"
1750 PRINT "NAME"
1760 PRINT "NAME"
1770 PRINT "NAME"
1780 PRINT "NAME"
1790 PRINT "NAME"
1800 PRINT "NAME"
1810 PRINT "NAME"
1820 PRINT "NAME"
1830 PRINT "NAME"
1840 PRINT "NAME"
1850 PRINT "NAME"
1860 PRINT "NAME"
1870 PRINT "NAME"
1880 PRINT "NAME"
1890 PRINT "NAME"
1900 PRINT "NAME"
1910 PRINT "NAME"
1920 PRINT "NAME"
1930 PRINT "NAME"
1940 PRINT "NAME"
1950 PRINT "NAME"
1960 PRINT "NAME"
1970 PRINT "NAME"
1980 PRINT "NAME"
1990 PRINT "NAME"

```

Type	FI	CI	CS	PS	C	CS	PS
						Max	Max
Wizard	2.00	0.50	23	28	100	66	777
Thinker	1.50	0.75	24	25	113	72	341
Barbarian	0.50	2.00	28	22	125	77	89
Warrior	1.00	1.25	26	24	113	75	117
Cleric	1.25	1.00	25	25	113	74	157
Elf	1.00	1.00	30	30	113	75	117

Table 2. The six possible character types with their initial values and the maximums to which their physical and magical strengths can rise.

FAST SUBROUTINES

UNGET: This is a universal GET routine and is designed to operate in conjunction with the string VGS. It will only return to the main program if the character keyed is one of those in VGS.

ANYKEY: This routine is used in the tape save and load routines to allow the player to ensure the cassette is ready in the tape machine before proceeding; it can be removed or replaced as required.

COMBAT GET: A special timed

GET routine for combat. It returns to the main program as soon as any key is pressed, assuming that this occurs within the time limit. The key pressed is held in CCS. If the time limit is exceeded the variable TV is set to 1. The routine also wipes

```

1400 REM ** MOVES ROUTINE
1400 GET POS-Y TO POSY** FROM 1400
1410 POS POS-Y TO POSY
1420 IF POSY<0 OR POSY>11:GOTO 1400:REM:RETRY
1430 REM Y
1440 GOTO 1400
1450 REM ** ATTACK ROUTINE
1460 REM ** (THIS IS ATTACK AND NOT TO CONFUSE **)
1470 GET POS-Y TO POSY** FROM 1410
1480 REM

```

```

1490 REM ** COMBAT GET ROUTINE
1500 FOR I=1 TO GETCOMB:GOTO 1400 ** DEFENSE ROUTINE
1510 REM
1520 FOR I=1 TO 24
1530 GET SPLIT:SPS** FROM 1510
1540 GOTO 1510
1550 REM
1560 REM **
1570 REM ** NO KEY PRESSURE
1580 REM ** (THIS IS (THIS IS) NO KEY PRESSURE)
1590 REM

```

away the text message
 ***STRIKE QUICKLY
 *** **

MOVEMENT

In many ways this represents the core of the whole program; it is certainly the most associated loop and controls access to all other major routines.

For such an important routine it occupies surprisingly little space, lines 2000 to 2260 in fact. Line 2000 is only used as an initial starting point when you first enter the Valley, either at the start of the game or when you return to the Valley from a scenario; all other calls are made to 2010. The POKE code 81 is the symbol used to display your current position in the Valley; Table 3 gives suggested alternatives for other systems.

The first operation is to give the character stamina a boost of 10, a dynamic refresh? Your current position is now examined to see if you are standing on a path or in the Valley and an appropriate message is printed requesting you to make a move. The player may move one square at a time in any direction; see Fig. 1. The choice of direction is made by keying one of the keys of the numeric keypad, the value of the key pressed then being inspected to find which direction it represents. In many programs of this type the checking is done by way of a



Fig. 1. The directions corresponding to the keys on a numeric pad together with their 8-directional displacements.

System	Object	PIT	MSX-K	TRS-80
Valley	Border	214	100	101
	Safe Castle	219	104	105
	Path Top	78	108	104
	Path Bottom	77	109	103
	Woods	218	80	87
	Swamps	173	42	83
	Towers	87	88	84
Woods	Character	81	202	79
	Border	86	-	128
	Tree	88	70	81
	Lake	204	183	181
Swamps	Yucca's	200	186	86
	Character	81	202	79
	Border	86	-	128
	Path	48	237	45
Towers	Lake	204	183	181
	Yucca's	200	238	88
	Character	81	202	79
Scenes	Border	183	87	181
	Walls	183	87	181
	Ships	172	128	183
	Coastline	48	237	178
	Towers	87	88	84
	Character	81	202	79

Table 3. These are the recommended POKE codes for a selection of the systems that the program has been converted onto, the problem that may arise, shown here by the TRS-80 ROM, is that certain symbols have to be printed into place rather than POKE'd. These are indicated by an * in the table.



Fig. 2. The mathematical sequence required to convert key values to row displacement.

look-up table which, although universal in operation and not restricted to numeric keypads is expensive in terms of memory.

Fortunately, one of the programming team had a mathematical background and produced the code between 2000 and 2095. Because each direction of movement corresponds directly to a screen displacement value, it must be possible to establish a simple mathematical relationship

between the numeric key pressed and the direction in which you wish to move.

The routine starts at 2000 by clearing out the keyboard buffer, an essential operation to prevent old keystrokes causing problems. A character is now read in by the GET command in 2000 and checked to see if it is an 'E'. If it is an 'E', control is passed to the rating routine which gives your current Ego. As we are only looking for

numeric inputs in this routine we can test for validity by finding the VAL of the character; if this is 0 it must have been non-numeric so the program loops back for another character.

Once a valid key has been pressed, its value is held in the variable A and testing starts at line 2000. The first piece of code repetitively subtracts 3 from the value of A to determine the horizontal displacement. If keys 1,4 or 7 have been pressed we must wish to move left; 3,6 and 9 indicate a move of the right and 2,5 and 8 maintain the current column position. As we are now left with a number between 1 and 3, we can determine the horizontal displacement by subtracting 2 from this remainder and this is done in 2009.

All we have to do now is to determine the correct vertical displacement, this also being computed in line 2009. Assuming a 40 column screen we must now establish the row we wish to move to; this is best explained by referring to Fig. 2. If we subtract 1 from the key

value (Fig. 2b) and then divide by 3 (Fig. 2c) the integer part of the remaining number is related to both the key value and its corresponding row (Fig. 2d). Subtract 1 and multiply by -40 and the resulting number is the vertical displacement.

Having computed the address of the position you wish to move to (held in variable W), we can now start to check what is at that position. These checks are preceded in line 2100 by incrementing the turns count and clearing away the movement message. The next block of lines inspects the contents of address W and is best shown as a small table:

Line	Character	Action
2110	Nothing	Jump to Movement
2120	Sale Castle	Jump to Out routine
2130	Solid object	Try again!
2140	Scene code	Jump to Scene Control
2150	Scene exit	Jump to Scene Control
2160	Stairs	Jump to

2170	Lakes	Stairs routine
		Reverse the character code
2180	Special finding to	Special Finds

After all these checks have been performed it only remains to move your character into its selected position; line 2190 does this and because you stepped on a blank square, the program now generates a random number to see if there is either a hidden 'treasure' or a waiting monster. If one of these in lines 2210 to 2230 and, depending on the value of the random number, control may be passed to either the Monster Selection routine or to the Finds routine.

If nothing is found a suitable message is printed and the program loops back to the beginning of movement at 2000. The DF value of 80 in line 2240 enables you to read the message; if you have stepped on the path, checked in line 2200, the delay is only set to 5 because there is no message to read.

```

2000 REM ** MOVEMENT ROUTINE
2001 REM:*****
2002 C=KEY$
2003 IF VAL(C) < 0 THEN GOTO 2000
2004 H=VAL(C)-3
2005 IF H=0 THEN GOTO 2009
2006 IF H=1 THEN GOTO 2009
2007 IF H=2 THEN GOTO 2009
2008 IF H=3 THEN GOTO 2009
2009 W=INT((VAL(C)-1)/3)*-40
2010 W=W+Y*40
2011 W=W+X*40
2012 W=W+1
2013 W=W+1
2014 W=W+1
2015 W=W+1
2016 W=W+1
2017 W=W+1
2018 W=W+1
2019 W=W+1
2020 W=W+1
2021 W=W+1
2022 W=W+1
2023 W=W+1
2024 W=W+1
2025 W=W+1
2026 W=W+1
2027 W=W+1
2028 W=W+1
2029 W=W+1
2030 W=W+1
2031 W=W+1
2032 W=W+1
2033 W=W+1
2034 W=W+1
2035 W=W+1
2036 W=W+1
2037 W=W+1
2038 W=W+1
2039 W=W+1
2040 W=W+1
2041 W=W+1
2042 W=W+1
2043 W=W+1
2044 W=W+1
2045 W=W+1
2046 W=W+1
2047 W=W+1
2048 W=W+1
2049 W=W+1
2050 W=W+1
2051 W=W+1
2052 W=W+1
2053 W=W+1
2054 W=W+1
2055 W=W+1
2056 W=W+1
2057 W=W+1
2058 W=W+1
2059 W=W+1
2060 W=W+1
2061 W=W+1
2062 W=W+1
2063 W=W+1
2064 W=W+1
2065 W=W+1
2066 W=W+1
2067 W=W+1
2068 W=W+1
2069 W=W+1
2070 W=W+1
2071 W=W+1
2072 W=W+1
2073 W=W+1
2074 W=W+1
2075 W=W+1
2076 W=W+1
2077 W=W+1
2078 W=W+1
2079 W=W+1
2080 W=W+1
2081 W=W+1
2082 W=W+1
2083 W=W+1
2084 W=W+1
2085 W=W+1
2086 W=W+1
2087 W=W+1
2088 W=W+1
2089 W=W+1
2090 W=W+1
2091 W=W+1
2092 W=W+1
2093 W=W+1
2094 W=W+1
2095 W=W+1
2096 W=W+1
2097 W=W+1
2098 W=W+1
2099 W=W+1
2100 W=W+1
2101 W=W+1
2102 W=W+1
2103 W=W+1
2104 W=W+1
2105 W=W+1
2106 W=W+1
2107 W=W+1
2108 W=W+1
2109 W=W+1
2110 W=W+1
2111 W=W+1
2112 W=W+1
2113 W=W+1
2114 W=W+1
2115 W=W+1
2116 W=W+1
2117 W=W+1
2118 W=W+1
2119 W=W+1
2120 W=W+1
2121 W=W+1
2122 W=W+1
2123 W=W+1
2124 W=W+1
2125 W=W+1
2126 W=W+1
2127 W=W+1
2128 W=W+1
2129 W=W+1
2130 W=W+1
2131 W=W+1
2132 W=W+1
2133 W=W+1
2134 W=W+1
2135 W=W+1
2136 W=W+1
2137 W=W+1
2138 W=W+1
2139 W=W+1
2140 W=W+1
2141 W=W+1
2142 W=W+1
2143 W=W+1
2144 W=W+1
2145 W=W+1
2146 W=W+1
2147 W=W+1
2148 W=W+1
2149 W=W+1
2150 W=W+1
2151 W=W+1
2152 W=W+1
2153 W=W+1
2154 W=W+1
2155 W=W+1
2156 W=W+1
2157 W=W+1
2158 W=W+1
2159 W=W+1
2160 W=W+1
2161 W=W+1
2162 W=W+1
2163 W=W+1
2164 W=W+1
2165 W=W+1
2166 W=W+1
2167 W=W+1
2168 W=W+1
2169 W=W+1
2170 W=W+1
2171 W=W+1
2172 W=W+1
2173 W=W+1
2174 W=W+1
2175 W=W+1
2176 W=W+1
2177 W=W+1
2178 W=W+1
2179 W=W+1
2180 W=W+1
2181 W=W+1
2182 W=W+1
2183 W=W+1
2184 W=W+1
2185 W=W+1
2186 W=W+1
2187 W=W+1
2188 W=W+1
2189 W=W+1
2190 W=W+1
2191 W=W+1
2192 W=W+1
2193 W=W+1
2194 W=W+1
2195 W=W+1
2196 W=W+1
2197 W=W+1
2198 W=W+1
2199 W=W+1
2200 W=W+1
2201 W=W+1
2202 W=W+1
2203 W=W+1
2204 W=W+1
2205 W=W+1
2206 W=W+1
2207 W=W+1
2208 W=W+1
2209 W=W+1
2210 W=W+1
2211 W=W+1
2212 W=W+1
2213 W=W+1
2214 W=W+1
2215 W=W+1
2216 W=W+1
2217 W=W+1
2218 W=W+1
2219 W=W+1
2220 W=W+1
2221 W=W+1
2222 W=W+1
2223 W=W+1
2224 W=W+1
2225 W=W+1
2226 W=W+1
2227 W=W+1
2228 W=W+1
2229 W=W+1
2230 W=W+1
2231 W=W+1
2232 W=W+1
2233 W=W+1
2234 W=W+1
2235 W=W+1
2236 W=W+1
2237 W=W+1
2238 W=W+1
2239 W=W+1
2240 W=W+1
2241 W=W+1
2242 W=W+1
2243 W=W+1
2244 W=W+1
2245 W=W+1
2246 W=W+1
2247 W=W+1
2248 W=W+1
2249 W=W+1
2250 W=W+1
2251 W=W+1
2252 W=W+1
2253 W=W+1
2254 W=W+1
2255 W=W+1
2256 W=W+1
2257 W=W+1
2258 W=W+1
2259 W=W+1
2260 W=W+1
2261 W=W+1
2262 W=W+1
2263 W=W+1
2264 W=W+1
2265 W=W+1
2266 W=W+1
2267 W=W+1
2268 W=W+1
2269 W=W+1
2270 W=W+1
2271 W=W+1
2272 W=W+1
2273 W=W+1
2274 W=W+1
2275 W=W+1
2276 W=W+1
2277 W=W+1
2278 W=W+1
2279 W=W+1
2280 W=W+1
2281 W=W+1
2282 W=W+1
2283 W=W+1
2284 W=W+1
2285 W=W+1
2286 W=W+1
2287 W=W+1
2288 W=W+1
2289 W=W+1
2290 W=W+1
2291 W=W+1
2292 W=W+1
2293 W=W+1
2294 W=W+1
2295 W=W+1
2296 W=W+1
2297 W=W+1
2298 W=W+1
2299 W=W+1
2300 W=W+1
2301 W=W+1
2302 W=W+1
2303 W=W+1
2304 W=W+1
2305 W=W+1
2306 W=W+1
2307 W=W+1
2308 W=W+1
2309 W=W+1
2310 W=W+1
2311 W=W+1
2312 W=W+1
2313 W=W+1
2314 W=W+1
2315 W=W+1
2316 W=W+1
2317 W=W+1
2318 W=W+1
2319 W=W+1
2320 W=W+1
2321 W=W+1
2322 W=W+1
2323 W=W+1
2324 W=W+1
2325 W=W+1
2326 W=W+1
2327 W=W+1
2328 W=W+1
2329 W=W+1
2330 W=W+1
2331 W=W+1
2332 W=W+1
2333 W=W+1
2334 W=W+1
2335 W=W+1
2336 W=W+1
2337 W=W+1
2338 W=W+1
2339 W=W+1
2340 W=W+1
2341 W=W+1
2342 W=W+1
2343 W=W+1
2344 W=W+1
2345 W=W+1
2346 W=W+1
2347 W=W+1
2348 W=W+1
2349 W=W+1
2350 W=W+1
2351 W=W+1
2352 W=W+1
2353 W=W+1
2354 W=W+1
2355 W=W+1
2356 W=W+1
2357 W=W+1
2358 W=W+1
2359 W=W+1
2360 W=W+1
2361 W=W+1
2362 W=W+1
2363 W=W+1
2364 W=W+1
2365 W=W+1
2366 W=W+1
2367 W=W+1
2368 W=W+1
2369 W=W+1
2370 W=W+1
2371 W=W+1
2372 W=W+1
2373 W=W+1
2374 W=W+1
2375 W=W+1
2376 W=W+1
2377 W=W+1
2378 W=W+1
2379 W=W+1
2380 W=W+1
2381 W=W+1
2382 W=W+1
2383 W=W+1
2384 W=W+1
2385 W=W+1
2386 W=W+1
2387 W=W+1
2388 W=W+1
2389 W=W+1
2390 W=W+1
2391 W=W+1
2392 W=W+1
2393 W=W+1
2394 W=W+1
2395 W=W+1
2396 W=W+1
2397 W=W+1
2398 W=W+1
2399 W=W+1
2400 W=W+1
2401 W=W+1
2402 W=W+1
2403 W=W+1
2404 W=W+1
2405 W=W+1
2406 W=W+1
2407 W=W+1
2408 W=W+1
2409 W=W+1
2410 W=W+1
2411 W=W+1
2412 W=W+1
2413 W=W+1
2414 W=W+1
2415 W=W+1
2416 W=W+1
2417 W=W+1
2418 W=W+1
2419 W=W+1
2420 W=W+1
2421 W=W+1
2422 W=W+1
2423 W=W+1
2424 W=W+1
2425 W=W+1
2426 W=W+1
2427 W=W+1
2428 W=W+1
2429 W=W+1
2430 W=W+1
2431 W=W+1
2432 W=W+1
2433 W=W+1
2434 W=W+1
2435 W=W+1
2436 W=W+1
2437 W=W+1
2438 W=W+1
2439 W=W+1
2440 W=W+1
2441 W=W+1
2442 W=W+1
2443 W=W+1
2444 W=W+1
2445 W=W+1
2446 W=W+1
2447 W=W+1
2448 W=W+1
2449 W=W+1
2450 W=W+1
2451 W=W+1
2452 W=W+1
2453 W=W+1
2454 W=W+1
2455 W=W+1
2456 W=W+1
2457 W=W+1
2458 W=W+1
2459 W=W+1
2460 W=W+1
2461 W=W+1
2462 W=W+1
2463 W=W+1
2464 W=W+1
2465 W=W+1
2466 W=W+1
2467 W=W+1
2468 W=W+1
2469 W=W+1
2470 W=W+1
2471 W=W+1
2472 W=W+1
2473 W=W+1
2474 W=W+1
2475 W=W+1
2476 W=W+1
2477 W=W+1
2478 W=W+1
2479 W=W+1
2480 W=W+1
2481 W=W+1
2482 W=W+1
2483 W=W+1
2484 W=W+1
2485 W=W+1
2486 W=W+1
2487 W=W+1
2488 W=W+1
2489 W=W+1
2490 W=W+1
2491 W=W+1
2492 W=W+1
2493 W=W+1
2494 W=W+1
2495 W=W+1
2496 W=W+1
2497 W=W+1
2498 W=W+1
2499 W=W+1
2500 W=W+1
2501 W=W+1
2502 W=W+1
2503 W=W+1
2504 W=W+1
2505 W=W+1
2506 W=W+1
2507 W=W+1
2508 W=W+1
2509 W=W+1
2510 W=W+1
2511 W=W+1
2512 W=W+1
2513 W=W+1
2514 W=W+1
2515 W=W+1
2516 W=W+1
2517 W=W+1
2518 W=W+1
2519 W=W+1
2520 W=W+1
2521 W=W+1
2522 W=W+1
2523 W=W+1
2524 W=W+1
2525 W=W+1
2526 W=W+1
2527 W=W+1
2528 W=W+1
2529 W=W+1
2530 W=W+1
2531 W=W+1
2532 W=W+1
2533 W=W+1
2534 W=W+1
2535 W=W+1
2536 W=W+1
2537 W=W+1
2538 W=W+1
2539 W=W+1
2540 W=W+1
2541 W=W+1
2542 W=W+1
2543 W=W+1
2544 W=W+1
2545 W=W+1
2546 W=W+1
2547 W=W+1
2548 W=W+1
2549 W=W+1
2550 W=W+1
2551 W=W+1
2552 W=W+1
2553 W=W+1
2554 W=W+1
2555 W=W+1
2556 W=W+1
2557 W=W+1
2558 W=W+1
2559 W=W+1
2560 W=W+1
2561 W=W+1
2562 W=W+1
2563 W=W+1
2564 W=W+1
2565 W=W+1
2566 W=W+1
2567 W=W+1
2568 W=W+1
2569 W=W+1
2570 W=W+1
2571 W=W+1
2572 W=W+1
2573 W=W+1
2574 W=W+1
2575 W=W+1
2576 W=W+1
2577 W=W+1
2578 W=W+1
2579 W=W+1
2580 W=W+1
2581 W=W+1
2582 W=W+1
2583 W=W+1
2584 W=W+1
2585 W=W+1
2586 W=W+1
2587 W=W+1
2588 W=W+1
2589 W=W+1
2590 W=W+1
2591 W=W+1
2592 W=W+1
2593 W=W+1
2594 W=W+1
2595 W=W+1
2596 W=W+1
2597 W=W+1
2598 W=W+1
2599 W=W+1
2600 W=W+1
2601 W=W+1
2602 W=W+1
2603 W=W+1
2604 W=W+1
2605 W=W+1
2606 W=W+1
2607 W=W+1
2608 W=W+1
2609 W=W+1
2610 W=W+1
2611 W=W+1
2612 W=W+1
2613 W=W+1
2614 W=W+1
2615 W=W+1
2616 W=W+1
2617 W=W+1
2618 W=W+1
2619 W=W+1
2620 W=W+1
2621 W=W+1
2622 W=W+1
2623 W=W+1
2624 W=W+1
2625 W=W+1
2626 W=W+1
2627 W=W+1
2628 W=W+1
2629 W=W+1
2630 W=W+1
2631 W=W+1
2632 W=W+1
2633 W=W+1
2634 W=W+1
2635 W=W+1
2636 W=W+1
2637 W=W+1
2638 W=W+1
2639 W=W+1
2640 W=W+1
2641 W=W+1
2642 W=W+1
2643 W=W+1
2644 W=W+1
2645 W=W+1
2646 W=W+1
2647 W=W+1
2648 W=W+1
2649 W=W+1
2650 W=W+1
2651 W=W+1
2652 W=W+1
2653 W=W+1
2654 W=W+1
2655 W=W+1
2656 W=W+1
2657 W=W+1
2658 W=W+1
2659 W=W+1
2660 W=W+1
2661 W=W+1
2662 W=W+1
2663 W=W+1
2664 W=W+1
2665 W=W+1
2666 W=W+1
2667 W=W+1
2668 W=W+1
2669 W=W+1
2670 W=W+1
2671 W=W+1
2672 W=W+1
2673 W=W+1
2674 W=W+1
2675 W=W+1
2676 W=W+1
2677 W=W+1
2678 W=W+1
2679 W=W+1
2680 W=W+1
2681 W=W+1
2682 W=W+1
2683 W=W+1
2684 W=W+1
2685 W=W+1
2686 W=W+1
2687 W=W+1
2688 W=W+1
2689 W=W+1
2690 W=W+1
2691 W=W+1
2692 W=W+1
2693 W=W+1
2694 W=W+1
2695 W=W+1
2696 W=W+1
2697 W=W+1
2698 W=W+1
2699 W=W+1
2700 W=W+1
2701 W=W+1
2702 W=W+1
2703 W=W+1
2704 W=W+1
2705 W=W+1
2706 W=W+1
2707 W=W+1
2708 W=W+1
2709 W=W+1
2710 W=W+1
2711 W=W+1
2712 W=W+1
2713 W=W+1
2714 W=W+1
2715 W=W+1
2716 W=W+1
2717 W=W+1
2718 W=W+1
2719 W=W+1
2720 W=W+1
2721 W=W+1
2722 W=W+1
2723 W=W+1
2724 W=W+1
2725 W=W+1
2726 W=W+1
2727 W=W+1
2728 W=W+1
2729 W=W+1
2730 W=W+1
2731 W=W+1
2732 W=W+1
2733 W=W+1
2734 W=W+1
2735 W=W+1
2736 W=W+1
2737 W=W+1
2738 W=W+1
2739 W=W+1
2740 W=W+1
2741 W=W+1
2742 W=W+1
2743 W=W+1
2744 W=W+1
2745 W=W+1
2746 W=W+1
2747 W=W+1
2748 W=W+1
2749 W=W+1
2750 W=W+1
2751 W=W+1
2752 W=W+1
2753 W=W+1
2754 W=W+1
2755 W=W+1
2756 W=W+1
2757 W=W+1
2758 W=W+1
2759 W=W+1
2760 W=W+1
2761 W=W+1
2762 W=W+1
2763 W=W+1
2764 W=W+1
2765 W=W+1
2766 W=W+1
2767 W=W+1
2768 W=W+1
2769 W=W+1
2770 W=W+1
2771 W=W+1
2772 W=W+1
2773 W=W+1
2774 W=W+1
2775 W=W+1
2776 W=W+1
2777 W=W+1
2778 W=W+1
2779 W=W+1
2780 W=W+1
2781 W=W+1
2782 W=W+1
2783 W=W+1
2784 W=W+1
2785 W=W+1
2786 W=W+1
2787 W=W+1
2788 W=W+1
2789 W=W+1
2790 W=W+1
2791 W=W+1
2792 W=W+1
2793 W=W+1
2794 W=W+1
2795 W=W+1
2796 W=W+1
2797 W=W+1
2798 W=W+1
2799 W=W+1
2800 W=W+1
2801 W=W+1
2802 W=W+1
2803 W=W+1
2804 W=W+1
2805 W=W+1
2806 W=W+1
2807 W=W+1
2808 W=W+1
2809 W=W+1
2810 W=W+1
2811 W=W+1
2812 W=W+1
2813 W=W+1
2814 W=W+1
2815 W=W+1
2816 W=W+1
2817 W=W+1
2818 W=W+1
2819 W=W+1
2820 W=W+1
2821 W=W+1
2822 W=W+1
2823 W=W+1
2824 W=W+1
2825 W=W+1
2826 W=W+1
2827 W=W+1
2828 W=W+1
2829 W=W+1
2830 W=W+1
2831 W=W+1
2832 W=W+1
2833 W=W+1
2834 W=W+1
2835 W=W+1
2836 W=W+1
2837 W=W+1
2838 W=W+1
2839 W=W+1
2840 W=W+1
2841 W=W+1
2842 W=W+1
2843 W=W+1
2844 W=W+1
2845 W=W+1
2846 W=W+1
2847 W=W+1
2848 W=W+1
2849 W=W+1
2850 W=W+1
2851 W=W+1
2852 W=W+1
2853 W=W+1
2854 W=W+1
2855 W=W+1
2856 W=W+1
2857 W=W+1
2858 W=W+1
2859 W=W+1
2860 W=W+1
2861 W=W+1
2862 W=W+1
2863 W=W+1
2864 W=W+1
2865 W=W+1
2866 W=W+1
2867 W=W+1
2868 W=W+1
2869 W=W+1
2870 W=W+1
2871 W=W+1
2872 W=W+1
2873 W=W+1
2874 W=W+1
2875 W=W+1
2876 W=W+1
2877 W=W+1
2878 W=W+1
2879 W=W+1
2880 W=W+1
2881 W=W+1
2882 W=W+1
2883 W=W+1
2884 W=W+1
2885 W=W+1
2886 W=W+1
2887 W=W+1
2888 W=W+1
2889 W=W+1
2890 W=W+1
2891 W=W+1
2892 W=W+1
2893 W=W+1
2894 W=W+1
2895 W=W+1
2896 W=W+1
2897 W=W+1
2898 W=W+1
2899 W=W+1
2900 W=W+1
2901 W=W+1
2902 W=W+1
2903 W=W+1
2904 W=W+1
2905 W=W+1
2906 W=W+1
2907 W=W+1
2908 W=W+1
2909 W=W+1
2910 W=W+1
2911 W=W+1
2912 W=W+1
2913 W=W+1
2914 W=W+1
2915 W=W+1
2916 W=W+1
2917 W=W+1
2918 W=W+1
2919 W=W+1
2920 W=W+1
2921 W=W+1
2922 W=W+1
2923 W=W+1
2924 W=W+1
2925 W=W+1
2926 W=W+1
2927 W=W+1
2928 W=W+1
2929 W=W+1
2930 W=W+1
2931 W=W+1
2932 W=W+1
2933 W=W+1
2934 W=W+1
2935 W=W+1
2936 W=W+1
2937 W=W+1
2938 W=W+1
2939 W=W+1
2940 W=W+1
2941 W=W+1
2942 W=W+1
2943 W=W+1
2944 W=W+1
2945 W=W+1
2946 W=W+1
2947 W=W+1
2948 W=W+1
2949 W=W+1
2950 W=W+1
2951 W=W+1
2952 W=W+1
2953 W=W+1
2954 W=W+1
2955 W=W+1
2956 W=W+1
2957 W=W+1
2958 W=W+1
2959 W=W+1
2960 W=W+1
2961 W=W+1
2962 W=W+1
2963 W=W+1
2964 W=W+1
2965 W=W+1
2966 W=W+1
2967 W=W+1
2968 W=W+1
2969 W=W+1
2970 W=W+1
2971 W=W+1
2972 W=W+1
2973 W=W+1
2974 W=W+1
2975 W=W+1
2976 W=W+1
2977 W=W+1
2978 W=W+1
2979 W=W+1
2980 W=W+1
2981 W=W+1
2982 W=W+1
2983 W=W+1
2984 W=W+1
2985 W=W+1
2986 W=W+1
2987 W=W+1
2988 W=W+1
2989 W=W+1
2990 W=W+1
2991 W=W+1
2992 W=W+1
2993 W=W+1
2994 W=W+1
2995 W=W+1
2996 W=W+1
2997 W=W+1
2998 W=W+1
2999 W=W+1
3000 W=W+1
3001 W=W+1
3002 W=W+1
3003 W=W+1
3004 W=W+1
3005 W=W+1
3006 W=W+1
3007 W=W+1
3008 W=W+1
3009 W=W+1
3010 W=W+1
3011 W=W+1
3012 W=W+1
3013 W=W+1
3014 W=W+1
3015 W=W+1
3016 W=W+1
3017 W=W+1
3018 W=W+1
3019 W=W+1
3020 W=W+1
3021 W=W+1
3022 W=W+1
3023 W=W+1
3024 W=W+1
3025 W=W+1
3026 W=W+1
3027 W=W+1
3028 W=W+1
3029 W=W+1
3030 W=W+1
3031 W=W+1
3032 W=W+1
3033 W=W+1
3034 W=W+1
3035 W=W+1
3036 W=W+1
3037 W=W+1
3038 W=W+1
3039 W=W+1
3040 W=W+1
3041 W=W+1
3042 W=W+1
3043 W=W+1
3044 W=W+1
3045 W=W+1
3046 W=W+1
3047 W=W+1
3048 W=W+1
3049 W=W+1
3050 W=W+1
3051 W=W+1
3052 W=W+1
3053 W=W+1
3054 W=W+1
3055 W=W+1
3056 W=W+1
3057 W=W+1
3058 W=W+1
3059 W=W+1
3060 W=W+1
3061 W=W+1
3062 W=W+1
3063 W=W+1
3064 W=W+1
3065 W=W+1
3066 W=W+1
3067 W=W+1
3068 W=W+1
3069 W=W+1
3070 W=W+1
3071 W=W+1
3072 W=W+1
3073 W=W+1
3074 W=W+1
3075 W=W+1
3076 W=W+1
3077 W=W+1
3078 W=W+1
3079 W=W+1
3080 W=W+1
3081 W=W+1
3082 W=W+1
3083 W=W+1
3084 W=W+1
3085 W=W+1
3086 W=W+1
3087 W=W+1
3088 W=W+1
3089 W=W+1
3090 W=W+1
3091 W=W+1
3092 W=W+1
3093 W=W+1
3094 W=W+1
3095 W=W+1
3096 W=W+1
3097 W=W+1
3098 W=W+1
3099 W=W+1
3100 W=W+1
3101 W=W+1
3102 W=W+1
3103 W=W+1
3104 W=W+1
3105 W=W+1
3106 W=W+1
3107 W=W+1
3108 W=W+1
3109 W=W+1
3110 W=W+1
3111 W=W+1
3112 W=W+1
3113 W=W+1
3114 W=W+1
3115 W=W+1
3116 W=W+1
3117 W=W+1
3118 W=W+1
3119 W=W+1
3120 W=W+1
3121 W=W+1
3122 W=W+1
3123 W=W+1
3124 W=W+1
3125 W=W+1
3126 W=W+1
3127 W=W+1
3128 W=W+1
3129 W=W+1
3130 W=W+1
3131 W=W+1
3132 W=W+1
3133 W=W+1
3134 W=W+1
3135 W=W+1
3136 W=W+1
3137 W=W+1
3138 W=W+1
3139 W=W+1
3140 W=W+1
3141 W=W+1
3142 W=W+1
3143 W=W+1
3144 W=W+1
3145 W=W+1
3146 W=W+1
3147 W=W+1
3148 W=W+1
3149 W=W+1
3150 W=W+1
3151 W=W+1
3152 W=W+1
3153 W=W+1
3154 W=W+1
3155 W=W+1
3156 W=W+1
3157 W=W+1
3158 W=W+1
3159 W=W+1
3160 W=W+1
3161 W=W+1
3162 W=W+1
3163 W=W+1
3164 W=W+1
3165 W=W+1
3166 W=W+1
3167 W=W+1
3168 W=W+1
3169 W=W+1
3170 W=W+1
3171 W=W+1
3172 W=W+1
3173 W=W+1
3174 W=W+1
3175 W=W+1
3176 W=W+1
3177 W=W+1
3178 W=W+1
3179 W=W+1
3180 W=W+1
3181 W=W+1
3182 W=W+1
3183 W=W+1
3184 W=W+1
3185 W=W+1
3186 W=W+1
3187 W=W+1
3188 W=W+1
3189 W=W+1
3190 W=W+1
3191 W=W+1
3192 W=W+1
3193 W=W+1
3194 W=W+1
3195 W=W+1
3196 W=W+1
3197 W=W+1
3198 W=W+1
3199 W=W+1
3200 W=W+1
3201 W=W+1
3202 W=W+1
3203 W=W+1
3204 W=W+1
3205 W=W+1
3206 W=W+1
3207 W=W+1
3208 W=W+1
3209 W=W+1
3210 W=W+1
3211 W=W+1
3212 W=W+1
3213 W=W+1
3214 W=W+1
3215 W=W+1
3216 W=W+1
3217 W=W+1
3218 W=W+1
3219 W=W+1
3220 W=W+1
3221 W=W+1
3222 W=W+1
3223 W=W+1
3224 W=W+1
3225 W=W+1
3226 W=W+1
3227 W=W+1
3228 W=W+1
3229 W=W+1
3230 W
```

100) and 700, depending on FL and a random factor. The third and fourth finds are not monetary but physical and

magical; although different messages are printed, lines 2410 and 2440:

Each of the four subroutines

returns to line 2320 for a delay and update before control is returned to the Movement routine, line 2010.

SPECIAL FINDS

If the Movement routine establishes that you have stepped onto an asterisk, a jump is made to the Special Finds module at line 2800. Here you are placed on the marker which is then erased (PK = 32); once you've picked up a special find, it's gone for good! Next, a random number is generated to decide what you've found; the 'Movement' message is wiped and a series of tests begins.

The first test (line 2820) succeeds if you're in Vornir's lair, $5=5$, have a full Amulet, $T(1)=8$, a rating greater than 25, and have not already found the Helm of Evarna, $T(2)=0$. You also have to very lucky, $RN>0.95$.

Line 2830 tests for the empty Amulet; this can only be found in the Temple of F'Magloth, $5=5$ with $RN>0.85$. You may only have one Amulet at a time,

$T(0)=0$. Note that although you need a full Amulet to obtain the Helm, losing the Amulet later (through reincarnation) means you are free to find another one; no problems arise if you already have the Helm.

The next line, 2840, checks for Amulet stones which only occur in the Black Tower, $5=4$. Not only must you first have the Amulet, $T(0)=1$, and space left in it, $T(1)<6$, but you must be on a sufficiently high floor. The first stones can be found low down the Tower but as you find each stone you must venture higher to find the remaining ones. $FL>T(1)$. Assuming you have found an Amulet stone, lines 2840-2850 decide whether it fits or not. Since RN must already be greater than 0.7 to get to these lines, the condition that $RN>0.85$ here gives a 50-50 chance of the stone being the right one.

If the tests in line 2830-2840

all fail then you have found either a precious stone or a worthless bauble. The random factor of 0.43 in line 2850 was chosen to get a long-term average of roughly 50% between precious stones and worthless baubles since RN may have been 'biased' by the previous tests. For example, if all three tests failed on factors other than the value of RN, it could be anything between 0 and 1 on line 2850 and the probability is 43% that you have a worthless bauble. On the other hand, if line 2840 failed only because $RN<0.7$ then the probability shifts to $0.43/0.7=61.4\%$.

As well as obtaining the objects themselves your treasure, TS, is updated in line 2900 by an amount which depends on the number of items you've already found. Baubles and wrong Amulet stones don't count and bypass this line.

```

2700 GOTO 11:GOTO 00:((11)END:GOTO 11
2710 GOTO 11:GOTO 00:((11)END:GOTO 11
2720 GOTO 11:GOTO 00:((11)END:GOTO 11
2730 GOTO 11:GOTO 00:((11)END:GOTO 11
2740 GOTO 11:GOTO 00:((11)END:GOTO 11
2750 GOTO 11:GOTO 00:((11)END:GOTO 11
2760 GOTO 11:GOTO 00:((11)END:GOTO 11
2770 GOTO 11:GOTO 00:((11)END:GOTO 11
2780 GOTO 11:GOTO 00:((11)END:GOTO 11
2790 GOTO 11:GOTO 00:((11)END:GOTO 11
2800 GOTO 11:GOTO 00:((11)END:GOTO 11
2810 GOTO 11:GOTO 00:((11)END:GOTO 11
2820 GOTO 11:GOTO 00:((11)END:GOTO 11
2830 GOTO 11:GOTO 00:((11)END:GOTO 11
2840 GOTO 11:GOTO 00:((11)END:GOTO 11
2850 GOTO 11:GOTO 00:((11)END:GOTO 11
2860 GOTO 11:GOTO 00:((11)END:GOTO 11
2870 GOTO 11:GOTO 00:((11)END:GOTO 11
2880 GOTO 11:GOTO 00:((11)END:GOTO 11
2890 GOTO 11:GOTO 00:((11)END:GOTO 11
2900 GOTO 11:GOTO 00:((11)END:GOTO 11
2910 GOTO 11:GOTO 00:((11)END:GOTO 11
2920 GOTO 11:GOTO 00:((11)END:GOTO 11
2930 GOTO 11:GOTO 00:((11)END:GOTO 11
2940 GOTO 11:GOTO 00:((11)END:GOTO 11
2950 GOTO 11:GOTO 00:((11)END:GOTO 11
2960 GOTO 11:GOTO 00:((11)END:GOTO 11
2970 GOTO 11:GOTO 00:((11)END:GOTO 11
2980 GOTO 11:GOTO 00:((11)END:GOTO 11
2990 GOTO 11:GOTO 00:((11)END:GOTO 11
3000 GOTO 11:GOTO 00:((11)END:GOTO 11

```

MONSTER SELECTION

If you have the misfortune to draw a monster at the end of the Movement routine, control is passed to the segment of code which starts at 3000. A random number is generated between 1 and 16 and tested to see if it is greater than 9. This test is made to ensure that the stronger monsters cannot occur too frequently; the checks and limits for this are established in line 3020.

If the character is currently standing or swimming in a lake the choice of monsters is limited by line 3030 to the Water Imp

and the Kraken, both prefixed L in the DATA.

The most unpleasant general monster is the Balrog and it is drawn from the array, a further check is made in line 3040 to ensure that he appears less frequently.

Some monsters live only in the rarified heights of the Black Tower or one of the two special castle-type scores and if these are drawn from the array, a further check is made in line 3050 to ensure that these conditions are met.

Once an acceptable monster

has been selected from the array, the left-hand character of its name is stripped off to see if it can exist in the current scenario; this character is then checked against FS in lines 3060 to 3090. If all is correct, the name of the chosen beast is displayed on the screen and combat commences. The base strengths of each monster are held in arrays MS0 and N00 and these values are further modified by the code between lines 3120 and 3170 to produce the actual strengths of the chosen monster.

```

1000 REM ** MONSTER REACTION ROUTINE
1010 PRINT DELTA * 60000...TIME MUST BE INCREASED ***
1020 REMARK:GOTO
1030 IF (RND*(360/PI)*PI)>PI THEN GOTO 1040 AND WRITE(10,10)
1040 REM 1040
1050 IF 30<=RND THEN GOTO 1060 THEN GOTO 1070(10,10)
1060 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1070 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1080 REMARK:GOTO
1090 REM 1090
1100 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1110 REM 1110
1120 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1130 REM 1130
1140 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1150 REM 1150
1160 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1170 REM 1170
1180 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1190 REM 1190

```

```

1000 REM 1
1010 GOTO 1010
1020 REMARK:GOTO
1030 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1040 REM 1040
1050 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1060 REM 1060
1070 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1080 REM 1080
1090 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1100 REM 1100
1110 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1120 REM 1120
1130 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1140 REM 1140
1150 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1160 REM 1160
1170 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1180 REM 1180
1190 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070

```

CHARACTER'S COMBAT

The action of fighting a monster can be broken down into three main sections; you hitting it, it hitting you and you casting a spell. The Spells are controlled by their own section of code that will be described later and can be simply treated as a jump out of the physical combat routines.

The Character's Combat section is located from 3570 to 3910 but before this can be executed we must determine whether you have surprised the beast or not. This is tested for in line 3600 where a random number is generated giving a 60/40 chance of you surprising the monster. If you do have surprise you are then offered the option of retreating from combat. The "R" key must be pressed within the time limit of the Combat Get routine or control passes directly to the Monster's Combat.

If you do choose to retreat a suitable message is displayed and the program goes back to the Movement routine.

Choosing to attack, the message " *** Strike Quickly *** " is

displayed and you have the choice of attacking its Head, Body or Limbs with the further option of trying to cast a Spell. If no key was pressed or the wrong one was chosen, control passes to the Monster's Combat via a suitable message. The control for this section of the combat is handled by lines 3670 to 3810.

Assuming that you have pressed a valid key the program checks in line 3680 to see if you wished to cast a spell and it so passes control to the Spell Control section. Before determining how much damage, if any, you have done to the beast, the program computes your current experience factor (in line 3690) and deducts one stamina point. If you have exhausted yourself attempting to fight, the program detects this in 3690 and passes control to the Death routine.

Because each of the three target areas of the monster have different levels of vulnerability the code between 3690 and 3710 determines whether you hit the beast or not and, if you did, sets the damage factor variable

Z to the appropriate level. As it is possible to strike a heavy blow which will leave the monster helpless we must first inspect the corresponding flag, HF, which tells us if the beast is certain to die on this attempt. If this flag has not been set in line 3750, we calculate the damage done to the monster and display it — it is possible to hit the monster yet do no damage! There are now several options available to us and these are sorted out by the rest of the routine from 3800 to 3910. The first alternative is that we have killed it, in which case we collect experience, reset the combat flag and go back to the movement routine (lines 3860 to 3890). Our second option is that we have done so much damage to the beast that it is unable to have another go at us. In this case we set the flag, HF, and go back to the " *** Strike Quickly *** " message. The remaining alternatives are that we either did no damage or insufficient to cripple the monster and in both cases, control now passes to the Monster's Combat routine.

```

1000 REM ** CHARACTER'S COMBAT ROUTINE
1010 IF RND*(1/3)>1/3 THEN GOTO 1020 ** MONSTER'S COMBAT
1020 PRINT DELTA * 60000...TIME MUST BE INCREASED ***
1030 REMARK:GOTO
1040 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1050 REM 1050
1060 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1070 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1080 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1090 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1100 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1110 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1120 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1130 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1140 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1150 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1160 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1170 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1180 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1190 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070

```

```

1000 REM 1000
1010 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1020 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1030 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1040 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1050 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1060 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1070 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1080 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1090 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1100 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1110 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1120 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1130 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1140 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1150 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1160 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1170 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1180 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070
1190 IF 30<=RND AND RND*(PI)>PI THEN GOTO 1070

```



```

1000  GOV=VAL+7*YCODE:MOV=VAL+YDEL+YDEL
1001  IF GOV=VAL+7*YCODE+YDEL THEN GOV=VAL+YDEL
1002  IF YDEL=0 THEN
1003    GOV=GOV+YDEL*(Y11)-YDEL*(Y11)
1004  IF GOV=VAL THEN GOV=VAL
1005  IF GOV=VAL THEN GOV=VAL
1006  GOV=GOV+YDEL*(Y11)-YDEL*(Y11)

```

```

1007  POINT GOV=7*YDEL+YDEL+YDEL+YDEL
1008  IF GOV=VAL THEN GOV=VAL
1009  GOV=GOV+YDEL*(Y11)-YDEL*(Y11)
1010  GOV=GOV+YDEL*(Y11)-YDEL*(Y11)
1011  GOV=GOV+YDEL*(Y11)-YDEL*(Y11)
1012  GOV=GOV+YDEL*(Y11)-YDEL*(Y11)

```

SCENE CONTROL

Having down the Valley, the Feh and the scenario positions, our hero is now free to wander where he chooses. A monster may kill him but with a little luck and fast reflexes, he will sooner or later enter one of the scenarios, or having entered may wish to escape!

To do this he simply moves onto the scenario or exit symbol. The part of the program concerned with movement, lines 2000-2260, detects the symbol and the program is directed to Scene Control at line 9000 or Scene Exit at line 9090.

A check is made at line 9000 to see if you are attempting to enter one of the secondary scenarios direct from the water, POKE code 224. If you are, the program is directed back to Movement via line 9110 which inhibits the turn count and the status refresh.

The two scene building arrays, P0 and M0, are dealt with in lines 9010-9050. Array P0 is used to fix the pattern of rooms on different levels of a scenario and if the scenario had more than one level, each level retains the same room pattern while you remain in that scenario. Array P0 has all previous values stored.

Array M0 has a series of random numbers, integers between 4 and 8, assigned to it (5 is not permitted as it can produce an unacceptable pattern). This array determines the depth of the rooms in the primary and secondary castle-type scenarios, Vossin's Lair and the Temple of Y Nagoth (pronounced Es-nag-ee-oth).

Entering a scenario from the Valley, tested for in line 9000, M0 is assigned your last position in the Valley, M, so that when you leave the scenario you will return to the same position in the Valley.

A random integer between 1

and 10 is generated in line 9070 and is assigned to P03. This is used in all scenarios to determine the position of lakes or the pattern of rooms. Line 9080 sets a temporary variable T0 to the number of turns you have had so far. T0. This is then used when you try to exit from a scenario testing whether you been 'in' for a certain number of turns. The line then directs the program past the initial portion of the scene exit to line 9130.

Moving on to line 9090, a test is made to see if you have remained within a scenario for a minimum number of turns. Thus if the turns count is equal to or larger than the generated random number, the program jumps to line 9130. If not you are barred from exiting the scenario and line 9100 inhibits the turns count and the status refresh. The program is then directed back to the Movement routine at line 9120.

With the Valley still displayed on the screen, line 9130 POKEs a space into your present position and POKEs your character symbol to the position of the scenario symbol. W is the position you are about to go to.

The lines 9140-9220 check to see what scenario or exit symbol has been stepped on thus determining the scene change required; Q1 represents the POKE code of position W.

If the symbol which frames the Swamp or Woods is found, a shifted space (POKE code 96 as specified in line 9140, then you must be leaving the scene and entering the Valley, S=1; FL=1. Line 9150 detects gateways, Q1=104, in the Black Tower, S=4, denoting that you are leaving the Tower and entering the Valley.

If a gateway is detected in the Temple, S=5, or the Lair, S=6, line 9160 directs the

program to allow you to enter the Swamps, S=2, or the Woods, S=3, respectively. These secondary scenarios are only listed on a lake in these scenarios but in both cases the scene number of the Swamp or Woods is 3 below their secondary scenario, S=S-3. FL is determined in a similar manner, FL=FL-4, and your position in the Swamp or Woods is reset to your original position outside the secondary scenario, M=MW.

A check is made at line 9170 to see if you have entered the Swamps, setting the scene number, S=2, and the level, FL=2, if you have 9180 does the check for the Woods.

Another check is made at line 9190 for either Swamps or Woods to assign two string variables, D25 and R25, if either of these scenarios are entered. Both D25 and R25 are cursor control movements: D25 being a random number of Cursor Downs between Q and 9 and R25 being a random number of Cursor Rights between 1 and 10; this is the first use of PG0.

In line 9210, there is a check of one of the secondary scenarios in the Swamps or the Woods. If the Lair or Temple symbol, POKE code 230 is recognised, the appropriate scene number is assigned, S=S+3 and level number, FL=FL+4. The temporary position variable, MW, is assigned your position, M, immediately prior to entering the secondary scenario.

The program is directed to the appropriate subroutine in line 9220 and then in line 9230, the program is directed to the delay and status update subroutine. On RETURNing, line 9240 takes you back to the Movement routine and the new scenario is displayed awaiting further exploration.

```

1000 REM ** DRAWING INITIAL SCENERY
1010 IF GO=34 AND PG=34 THEN PRINT GO:PG:GOTO 1100
1020 SCREEN MODE 128:CLS:SCREEN 110
1030 GOSUB 1000
1040 GOTO 1100
1050 G(1)=INT(RND*(70+1))
1060 IF G(1)=0 THEN GOTO 1070
1070 GOTO 1080
1080 IF INT(RND*(70+1))=0 THEN
1090 GOTO 1100
1100 IF INT(RND*(70+1))=0 THEN
1110 GOTO 1120
1120 IF INT(RND*(70+1))=0 THEN
1130 GOTO 1140
1140 IF INT(RND*(70+1))=0 THEN
1150 GOTO 1160
1160 IF INT(RND*(70+1))=0 THEN
1170 GOTO 1180
1180 IF INT(RND*(70+1))=0 THEN
1190 GOTO 1200
1200 IF INT(RND*(70+1))=0 THEN
1210 GOTO 1220
1220 IF INT(RND*(70+1))=0 THEN
1230 GOTO 1240
1240 IF INT(RND*(70+1))=0 THEN
1250 GOTO 1260
1260 IF INT(RND*(70+1))=0 THEN
1270 GOTO 1280
1280 IF INT(RND*(70+1))=0 THEN
1290 GOTO 1300
1300 IF INT(RND*(70+1))=0 THEN
1310 GOTO 1320
1320 IF INT(RND*(70+1))=0 THEN
1330 GOTO 1340
1340 IF INT(RND*(70+1))=0 THEN
1350 GOTO 1360
1360 IF INT(RND*(70+1))=0 THEN
1370 GOTO 1380
1380 IF INT(RND*(70+1))=0 THEN
1390 GOTO 1400
1400 IF INT(RND*(70+1))=0 THEN
1410 GOTO 1420
1420 IF INT(RND*(70+1))=0 THEN
1430 GOTO 1440
1440 IF INT(RND*(70+1))=0 THEN
1450 GOTO 1460
1460 IF INT(RND*(70+1))=0 THEN
1470 GOTO 1480
1480 IF INT(RND*(70+1))=0 THEN
1490 GOTO 1500
1500 IF INT(RND*(70+1))=0 THEN
1510 GOTO 1520
1520 IF INT(RND*(70+1))=0 THEN
1530 GOTO 1540
1540 IF INT(RND*(70+1))=0 THEN
1550 GOTO 1560
1560 IF INT(RND*(70+1))=0 THEN
1570 GOTO 1580
1580 IF INT(RND*(70+1))=0 THEN
1590 GOTO 1600
1600 IF INT(RND*(70+1))=0 THEN
1610 GOTO 1620
1620 IF INT(RND*(70+1))=0 THEN
1630 GOTO 1640
1640 IF INT(RND*(70+1))=0 THEN
1650 GOTO 1660
1660 IF INT(RND*(70+1))=0 THEN
1670 GOTO 1680
1680 IF INT(RND*(70+1))=0 THEN
1690 GOTO 1700
1700 IF INT(RND*(70+1))=0 THEN
1710 GOTO 1720
1720 IF INT(RND*(70+1))=0 THEN
1730 GOTO 1740
1740 IF INT(RND*(70+1))=0 THEN
1750 GOTO 1760
1760 IF INT(RND*(70+1))=0 THEN
1770 GOTO 1780
1780 IF INT(RND*(70+1))=0 THEN
1790 GOTO 1800
1800 IF INT(RND*(70+1))=0 THEN
1810 GOTO 1820
1820 IF INT(RND*(70+1))=0 THEN
1830 GOTO 1840
1840 IF INT(RND*(70+1))=0 THEN
1850 GOTO 1860
1860 IF INT(RND*(70+1))=0 THEN
1870 GOTO 1880
1880 IF INT(RND*(70+1))=0 THEN
1890 GOTO 1900
1900 IF INT(RND*(70+1))=0 THEN
1910 GOTO 1920
1920 IF INT(RND*(70+1))=0 THEN
1930 GOTO 1940
1940 IF INT(RND*(70+1))=0 THEN
1950 GOTO 1960
1960 IF INT(RND*(70+1))=0 THEN
1970 GOTO 1980
1980 IF INT(RND*(70+1))=0 THEN
1990 GOTO 2000

```

THE VALLEY

Let us start where our "alter ego" whatever his character type, will step out into this adventure... the Valley. Line 10000 clears the screen sets F5 to "VARGH" which determines what monsters may be found in the Valley, is monsters from groups V,A,E,G, and M (see Table 1). The difficulty level is set to 1 (FL has a bearing on how strong the monsters are) and finally sets the score number also to 1... the Valley.

First let us draw the bounds of the Valley; this consists of a rectangle 39 characters wide by 14 characters high. Lines 10100 and 10160 draw in the top and bottom frames of the Valley, between them a FOR...NEXT loop draws the vertical frames consisting of the appropriate character repeated by 37 spaces. These are drawn 12 times giving an internal playing area of 37 by 12.

Line 10200 determines the position, M, of the left-hand side castle. 32809 is the screen map position of the top left-hand corner of our frame.

Line 10300 assigns the position and character code for the left-hand side castle to array elements G(0) and G(1) respectively. L and MF are temporary position variables. M is used throughout as you position *see* and W is your position when you next move (the look-ahead variable). All are set to position of the side castle.

We now have to work out the course of the path; for the PET we use the two diagonal lines, POKE codes 77 and 76 (suggested symbols for other micros are given in Table 2). Using only these two symbols for the path enables us to make

a fairly simple decision on how the path may be drawn:

- 1) If the path is already slanting up to the right then, only two possibilities are acceptable; upwards to the North East or downwards to the East (remember the path must be continuous)
- 2) If the path is already slanting down to the right then only downwards to the South East or upwards to the East are permissible.

The choice of an upwards or downwards diagonal is made randomly in line 10100. Lines 10180 and 10190 initially set the POKE code, PC, for a downwards diagonal to be to the South East of the present path (L1=L+4) and for an upwards diagonal to be to the North East (L1=L-39). L1 is a temporary position variable.

In line 10130 we check to see if the path is trying to go through the top or bottom frames of the Valley. If it is, the program is directed back to line 10100 to choose again. If it is within the Valley, we allocate in line 10140 an element array G(L) to the POKE code for that

section of the path. Looking back to 1) and 2) above, you will notice that if the next path element is different to the previous element it must always be to the East. This condition is checked in line 10160 and the temporary variable, L1, is altered if necessary.

Line 10160 stores the position of the path in array G(L), assigns the starting position for the next section of the path to variable L and draws the section just computed on the screen.

This selection of path direction and position is repeated 36 times within a FOR...NEXT loop. Position 1 and 37 are side castles from which you may leave the Valley if you so wish. Even elements of G() are screen positions and odd elements are POKE codes for the side castles or the path. Line 10180 completes the "path draw" by placing the right-hand side castle on the last path position.

Going back to line 10060 we see that this tests to see if the path has already been



Observe the Wizard struggles across a Hog-Castle!

computed. If it has not then GND will be zero. If it has, the program skips the path computation and jumps to line 10190 which initiates a FOR...NEXT loop that draws the path using the information stored in array S(1). In fact if the path has been drawn for the first time (on entering the Valley) it is also redrawn in lines 10180 to 10210.

Having drawn the Valley path we now have to work out where the different scenarios are to be placed within the Valley. These positions are stored in array S(1). Line 10220 checks if they have already been assigned and if they have,

the program jumps to line 10280 where the scenarios are drawn on the map of the Valley.

Line 10240 generates two random numbers that represent row and column positions within the Valley. Line 10250 assigns this position to an element of array S(1). Line 10260 checks to see that the chosen position and the position immediately to its right are not the path or safe castle (ie an empty space POKE code 32). If the positions chosen are free then the selection is repeated for the next element of array S(1).

Lines 10280-10300 POKE the scenario symbols on the screen. There is a chance that some

symbols may be overwritten by other scenarios but then this is the last of the draw! Woods are drawn first (code 205) then Swamps (code 173) and finally the Black Tower of Zaena (pronounced Zeek-on), code 87. Woods and Swamps are represented by two symbols side by side and both scenarios are repeated. The Black Tower is a single symbol; only one such Tower is found in the Valley.

Line 10310 assigns your present position to that of temporary variable, NP. This is used to remember your last position in the Valley when you return to the Valley from one of the other scenarios.

```

10000 REM ** SUBROUTINE 1 (TREE ROUTINE)
10001 REM ** POKE(173,POKECODE*(N-1)+1)
10002 REM ** ON GND FOR RANDOM POKE
10003 REM ** POKE(POKE*(POKE*(N-1)+1))
10004 REM ** ON GND
10005 REM ** POKE(POKE*(POKE*(N-1)+1))
10006 REM **
10007 REM **
10008 REM **
10009 REM **
10010 REM **
10011 REM **
10012 REM **
10013 REM **
10014 REM **
10015 REM **
10016 REM **
10017 REM **
10018 REM **
10019 REM **
10020 REM **
10021 REM **
10022 REM **
10023 REM **
10024 REM **
10025 REM **
10026 REM **
10027 REM **
10028 REM **
10029 REM **
10030 REM **
10031 REM **
10032 REM **
10033 REM **
10034 REM **
10035 REM **
10036 REM **
10037 REM **
10038 REM **
10039 REM **
10040 REM **
10041 REM **
10042 REM **
10043 REM **
10044 REM **
10045 REM **
10046 REM **
10047 REM **
10048 REM **
10049 REM **
10050 REM **
10051 REM **
10052 REM **
10053 REM **
10054 REM **
10055 REM **
10056 REM **
10057 REM **
10058 REM **
10059 REM **
10060 REM **
10061 REM **
10062 REM **
10063 REM **
10064 REM **
10065 REM **
10066 REM **
10067 REM **
10068 REM **
10069 REM **
10070 REM **
10071 REM **
10072 REM **
10073 REM **
10074 REM **
10075 REM **
10076 REM **
10077 REM **
10078 REM **
10079 REM **
10080 REM **
10081 REM **
10082 REM **
10083 REM **
10084 REM **
10085 REM **
10086 REM **
10087 REM **
10088 REM **
10089 REM **
10090 REM **
10091 REM **
10092 REM **
10093 REM **
10094 REM **
10095 REM **
10096 REM **
10097 REM **
10098 REM **
10099 REM **
10100 REM **
10101 REM **
10102 REM **
10103 REM **
10104 REM **
10105 REM **
10106 REM **
10107 REM **
10108 REM **
10109 REM **
10110 REM **
10111 REM **
10112 REM **
10113 REM **
10114 REM **
10115 REM **
10116 REM **
10117 REM **
10118 REM **
10119 REM **
10120 REM **
10121 REM **
10122 REM **
10123 REM **
10124 REM **
10125 REM **
10126 REM **
10127 REM **
10128 REM **
10129 REM **
10130 REM **
10131 REM **
10132 REM **
10133 REM **
10134 REM **
10135 REM **
10136 REM **
10137 REM **
10138 REM **
10139 REM **
10140 REM **
10141 REM **
10142 REM **
10143 REM **
10144 REM **
10145 REM **
10146 REM **
10147 REM **
10148 REM **
10149 REM **
10150 REM **
10151 REM **
10152 REM **
10153 REM **
10154 REM **
10155 REM **
10156 REM **
10157 REM **
10158 REM **
10159 REM **
10160 REM **
10161 REM **
10162 REM **
10163 REM **
10164 REM **
10165 REM **
10166 REM **
10167 REM **
10168 REM **
10169 REM **
10170 REM **
10171 REM **
10172 REM **
10173 REM **
10174 REM **
10175 REM **
10176 REM **
10177 REM **
10178 REM **
10179 REM **
10180 REM **
10181 REM **
10182 REM **
10183 REM **
10184 REM **
10185 REM **
10186 REM **
10187 REM **
10188 REM **
10189 REM **
10190 REM **
10191 REM **
10192 REM **
10193 REM **
10194 REM **
10195 REM **
10196 REM **
10197 REM **
10198 REM **
10199 REM **
10200 REM **
10201 REM **
10202 REM **
10203 REM **
10204 REM **
10205 REM **
10206 REM **
10207 REM **
10208 REM **
10209 REM **
10210 REM **
10211 REM **
10212 REM **
10213 REM **
10214 REM **
10215 REM **
10216 REM **
10217 REM **
10218 REM **
10219 REM **
10220 REM **
10221 REM **
10222 REM **
10223 REM **
10224 REM **
10225 REM **
10226 REM **
10227 REM **
10228 REM **
10229 REM **
10230 REM **
10231 REM **
10232 REM **
10233 REM **
10234 REM **
10235 REM **
10236 REM **
10237 REM **
10238 REM **
10239 REM **
10240 REM **
10241 REM **
10242 REM **
10243 REM **
10244 REM **
10245 REM **
10246 REM **
10247 REM **
10248 REM **
10249 REM **
10250 REM **
10251 REM **
10252 REM **
10253 REM **
10254 REM **
10255 REM **
10256 REM **
10257 REM **
10258 REM **
10259 REM **
10260 REM **
10261 REM **
10262 REM **
10263 REM **
10264 REM **
10265 REM **
10266 REM **
10267 REM **
10268 REM **
10269 REM **
10270 REM **
10271 REM **
10272 REM **
10273 REM **
10274 REM **
10275 REM **
10276 REM **
10277 REM **
10278 REM **
10279 REM **
10280 REM **
10281 REM **
10282 REM **
10283 REM **
10284 REM **
10285 REM **
10286 REM **
10287 REM **
10288 REM **
10289 REM **
10290 REM **
10291 REM **
10292 REM **
10293 REM **
10294 REM **
10295 REM **
10296 REM **
10297 REM **
10298 REM **
10299 REM **
10300 REM **
10301 REM **
10302 REM **
10303 REM **
10304 REM **
10305 REM **
10306 REM **
10307 REM **
10308 REM **
10309 REM **
10310 REM **
10311 REM **
10312 REM **
10313 REM **
10314 REM **
10315 REM **
10316 REM **
10317 REM **
10318 REM **
10319 REM **
10320 REM **
10321 REM **
10322 REM **
10323 REM **
10324 REM **
10325 REM **
10326 REM **
10327 REM **
10328 REM **
10329 REM **
10330 REM **
10331 REM **
10332 REM **
10333 REM **
10334 REM **
10335 REM **
10336 REM **
10337 REM **
10338 REM **
10339 REM **
10340 REM **
10341 REM **
10342 REM **
10343 REM **
10344 REM **
10345 REM **
10346 REM **
10347 REM **
10348 REM **
10349 REM **
10350 REM **
10351 REM **
10352 REM **
10353 REM **
10354 REM **
10355 REM **
10356 REM **
10357 REM **
10358 REM **
10359 REM **
10360 REM **
10361 REM **
10362 REM **
10363 REM **
10364 REM **
10365 REM **
10366 REM **
10367 REM **
10368 REM **
10369 REM **
10370 REM **
10371 REM **
10372 REM **
10373 REM **
10374 REM **
10375 REM **
10376 REM **
10377 REM **
10378 REM **
10379 REM **
10380 REM **
10381 REM **
10382 REM **
10383 REM **
10384 REM **
10385 REM **
10386 REM **
10387 REM **
10388 REM **
10389 REM **
10390 REM **
10391 REM **
10392 REM **
10393 REM **
10394 REM **
10395 REM **
10396 REM **
10397 REM **
10398 REM **
10399 REM **
10400 REM **
10401 REM **
10402 REM **
10403 REM **
10404 REM **
10405 REM **
10406 REM **
10407 REM **
10408 REM **
10409 REM **
10410 REM **
10411 REM **
10412 REM **
10413 REM **
10414 REM **
10415 REM **
10416 REM **
10417 REM **
10418 REM **
10419 REM **
10420 REM **
10421 REM **
10422 REM **
10423 REM **
10424 REM **
10425 REM **
10426 REM **
10427 REM **
10428 REM **
10429 REM **
10430 REM **
10431 REM **
10432 REM **
10433 REM **
10434 REM **
10435 REM **
10436 REM **
10437 REM **
10438 REM **
10439 REM **
10440 REM **
10441 REM **
10442 REM **
10443 REM **
10444 REM **
10445 REM **
10446 REM **
10447 REM **
10448 REM **
10449 REM **
10450 REM **
10451 REM **
10452 REM **
10453 REM **
10454 REM **
10455 REM **
10456 REM **
10457 REM **
10458 REM **
10459 REM **
10460 REM **
10461 REM **
10462 REM **
10463 REM **
10464 REM **
10465 REM **
10466 REM **
10467 REM **
10468 REM **
10469 REM **
10470 REM **
10471 REM **
10472 REM **
10473 REM **
10474 REM **
10475 REM **
10476 REM **
10477 REM **
10478 REM **
10479 REM **
10480 REM **
10481 REM **
10482 REM **
10483 REM **
10484 REM **
10485 REM **
10486 REM **
10487 REM **
10488 REM **
10489 REM **
10490 REM **
10491 REM **
10492 REM **
10493 REM **
10494 REM **
10495 REM **
10496 REM **
10497 REM **
10498 REM **
10499 REM **
10500 REM **
10501 REM **
10502 REM **
10503 REM **
10504 REM **
10505 REM **
10506 REM **
10507 REM **
10508 REM **
10509 REM **
10510 REM **
10511 REM **
10512 REM **
10513 REM **
10514 REM **
10515 REM **
10516 REM **
10517 REM **
10518 REM **
10519 REM **
10520 REM **
10521 REM **
10522 REM **
10523 REM **
10524 REM **
10525 REM **
10526 REM **
10527 REM **
10528 REM **
10529 REM **
10530 REM **
10531 REM **
10532 REM **
10533 REM **
10534 REM **
10535 REM **
10536 REM **
10537 REM **
10538 REM **
10539 REM **
10540 REM **
10541 REM **
10542 REM **
10543 REM **
10544 REM **
10545 REM **
10546 REM **
10547 REM **
10548 REM **
10549 REM **
10550 REM **
10551 REM **
10552 REM **
10553 REM **
10554 REM **
10555 REM **
10556 REM **
10557 REM **
10558 REM **
10559 REM **
10560 REM **
10561 REM **
10562 REM **
10563 REM **
10564 REM **
10565 REM **
10566 REM **
10567 REM **
10568 REM **
10569 REM **
10570 REM **
10571 REM **
10572 REM **
10573 REM **
10574 REM **
10575 REM **
10576 REM **
10577 REM **
10578 REM **
10579 REM **
10580 REM **
10581 REM **
10582 REM **
10583 REM **
10584 REM **
10585 REM **
10586 REM **
10587 REM **
10588 REM **
10589 REM **
10590 REM **
10591 REM **
10592 REM **
10593 REM **
10594 REM **
10595 REM **
10596 REM **
10597 REM **
10598 REM **
10599 REM **
10600 REM **
10601 REM **
10602 REM **
10603 REM **
10604 REM **
10605 REM **
10606 REM **
10607 REM **
10608 REM **
10609 REM **
10610 REM **
10611 REM **
10612 REM **
10613 REM **
10614 REM **
10615 REM **
10616 REM **
10617 REM **
10618 REM **
10619 REM **
10620 REM **
10621 REM **
10622 REM **
10623 REM **
10624 REM **
10625 REM **
10626 REM **
10627 REM **
10628 REM **
10629 REM **
10630 REM **
10631 REM **
10632 REM **
10633 REM **
10634 REM **
10635 REM **
10636 REM **
10637 REM **
10638 REM **
10639 REM **
10640 REM **
10641 REM **
10642 REM **
10643 REM **
10644 REM **
10645 REM **
10646 REM **
10647 REM **
10648 REM **
10649 REM **
10650 REM **
10651 REM **
10652 REM **
10653 REM **
10654 REM **
10655 REM **
10656 REM **
10657 REM **
10658 REM **
10659 REM **
10660 REM **
10661 REM **
10662 REM **
10663 REM **
10664 REM **
10665 REM **
10666 REM **
10667 REM **
10668 REM **
10669 REM **
10670 REM **
10671 REM **
10672 REM **
10673 REM **
10674 REM **
10675 REM **
10676 REM **
10677 REM **
10678 REM **
10679 REM **
10680 REM **
10681 REM **
10682 REM **
10683 REM **
10684 REM **
10685 REM **
10686 REM **
10687 REM **
10688 REM **
10689 REM **
10690 REM **
10691 REM **
10692 REM **
10693 REM **
10694 REM **
10695 REM **
10696 REM **
10697 REM **
10698 REM **
10699 REM **
10700 REM **
10701 REM **
10702 REM **
10703 REM **
10704 REM **
10705 REM **
10706 REM **
10707 REM **
10708 REM **
10709 REM **
10710 REM **
10711 REM **
10712 REM **
10713 REM **
10714 REM **
10715 REM **
10716 REM **
10717 REM **
10718 REM **
10719 REM **
10720 REM **
10721 REM **
10722 REM **
10723 REM **
10724 REM **
10725 REM **
10726 REM **
10727 REM **
10728 REM **
10729 REM **
10730 REM **
10731 REM **
10732 REM **
10733 REM **
10734 REM **
10735 REM **
10736 REM **
10737 REM **
10738 REM **
10739 REM **
10740 REM **
10741 REM **
10742 REM **
10743 REM **
10744 REM **
10745 REM **
10746 REM **
10747 REM **
10748 REM **
10749 REM **
10750 REM **
10751 REM **
10752 REM **
10753 REM **
10754 REM **
10755 REM **
10756 REM **
10757 REM **
10758 REM **
10759 REM **
10760 REM **
10761 REM **
10762 REM **
10763 REM **
10764 REM **
10765 REM **
10766 REM **
10767 REM **
10768 REM **
10769 REM **
10770 REM **
10771 REM **
10772 REM **
10773 REM **
10774 REM **
10775 REM **
10776 REM **
10777 REM **
10778 REM **
10779 REM **
10780 REM **
10781 REM **
10782 REM **
10783 REM **
10784 REM **
10785 REM **
10786 REM **
10787 REM **
10788 REM **
10789 REM **
10790 REM **
10791 REM **
10792 REM **
10793 REM **
10794 REM **
10795 REM **
10796 REM **
10797 REM **
10798 REM **
10799 REM **
10800 REM **
10801 REM **
10802 REM **
10803 REM **
10804 REM **
10805 REM **
10806 REM **
10807 REM **
10808 REM **
10809 REM **
10810 REM **
10811 REM **
10812 REM **
10813 REM **
10814 REM **
10815 REM **
10816 REM **
10817 REM **
10818 REM **
10819 REM **
10820 REM **
10821 REM **
10822 REM **
10823 REM **
10824 REM **
10825 REM **
10826 REM **
10827 REM **
10828 REM **
10829 REM **
10830 REM **
10831 REM **
10832 REM **
10833 REM **
10834 REM **
10835 REM **
10836 REM **
10837 REM **
10838 REM **
10839 REM **
10840 REM **
10841 REM **
10842 REM **
10843 REM **
10844 REM **
10845 REM **
10846 REM **
10847 REM **
10848 REM **
10849 REM **
10850 REM **
10851 REM **
10852 REM **
10853 REM **
10854 REM **
10855 REM **
10856 REM **
10857 REM **
10858 REM **
10859 REM **
10860 REM **
10861 REM **
10862 REM **
10863 REM **
10864 REM **
10865 REM **
10866 REM **
10867 REM **
10868 REM **
10869 REM **
10870 REM **
10871 REM **
10872 REM **
10873 REM **
10874 REM **
10875 REM **
10876 REM **
10877 REM **
10878 REM **
10879 REM **
10880 REM **
10881 REM **
10882 REM **
10883 REM **
10884 REM **
10885 REM **
10886 REM **
10887 REM **
10888 REM **
10889 REM **
10890 REM **
10891 REM **
10892 REM **
10893 REM **
10894 REM **
10895 REM **
10896 REM **
10897 REM **
10898 REM **
10899 REM **
10900 REM **
10901 REM **
10902 REM **
10903 REM **
10904 REM **
10905 REM **
10906 REM **
10907 REM **
10908 REM **
10909 REM **
10910 REM **
10911 REM **
10912 REM **
10913 REM **
10914 REM **
10915 REM **
10916 REM **
10917 REM **
10918 REM **
10919 REM **
10920 REM **
10921 REM **
10922 REM **
10923 REM **
10924 REM **
10925 REM **
10926 REM **
10927 REM **
10928 REM **
10929 REM **
10930 REM **
10931 REM **
10932 REM **
10933 REM **
10934 REM **
10935 REM **
10936 REM **
10937 REM **
10938 REM **
10939 REM **
10940 REM **
10941 REM **
10942 REM **
10943 REM **
10944 REM **
10945 REM **
10946 REM **
10947 REM **
10948 REM **
10949 REM **
10950 REM **
10951 REM **
10952 REM **
10953 REM **
10954 REM **
10955 REM **
10956 REM **
10957 REM **
10958 REM **
10959 REM **
10960 REM **
10961 REM **
10962 REM **
10963 REM **
10964 REM **
10965 REM **
10966 REM **
10967 REM **
10968 REM **
10969 REM **
10970 REM **
10971 REM **
10972 REM **
10973 REM **
10974 REM **
10975 REM **
10976 REM **
10977 REM **
10978 REM **
10979 REM **
10980 REM **
10981 REM **
10982 REM **
10983 REM **
10984 REM **
10985 REM **
10986 REM **
10987 REM **
10988 REM **
10989 REM **
10990 REM **
10991 REM **
10992 REM **
10993 REM **
10994 REM **
10995 REM **
10996 REM **
10997 REM **
10998 REM **
10999 REM **
11000 REM **

```

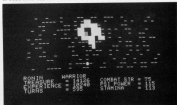
WOODS AND SWAMPS

These two scenarios are fundamentally the same, only the characters representing their contents and the special scene in the middle of the Lake differ. The starting point selected by the Scene Control routine is 12000 for the Swamp and 12040 for the Woods. The function of each of these entry points is to establish the valid monster string, F3, and to assign the POKE code, PC, to the correct value for the scene. The routine proper begins by re-assigning the POKE value of the square under your feet to 32, a space, so that when you move away you don't leave the character you were standing on in the Valley behind you.

The screen is now cleared in preparation for the construction of the scene. A pointer variable, L, is set to the value of the top left-hand corner of the scene

and a random FOR...NEXT loop inserts 200 appropriate scene characters into the screen area. This is all handled by the block of code from 12040 to 12070.

Having now constructed the inside of the scene, we have to print a border around it in order to detect an attempt at movement outside the scene



Again the Warrior steps bravely out in the Swamp. His objective, the Temple of T'Nagloth where he may be lucky enough to find the Amulet of Kharon.

THE BLACK TOWER

Our other primary scenario is the Black Tower of Zaanes. This is a six floor castle type scene and its construction is also used to produce the secondary single floor scenes found in the Woods and Swamps. The Tower has a stable floor pattern; once a floor has been entered it will remain the same as long as you are in the Tower.

Scenario Control directs the program to the routine at 14000 assigning the monster string, FI, setting the floor pattern variable, P, and setting the room depth variable, PL, to the current FLth element of array N(). The current position character is set to a space and the program jumps to 14020. The variables for the secondary scenes are initialized in 14010. One slight change is that the array FI is set to the value of P(2). This is done because the secondary scenes have initial PL values of 5 or 7 depending on type and three elements of FI are 0 which would cause the room pattern to be the same each time (see line 14020).

The frame of the Tower is printed first by lines 14030 to 14090 using a reversed space character. The vertical walls are drawn next by the somewhat complex routine found between 14070 and 14250. In order to assure that the pattern of rooms varies on each floor and on each visit to the scene, we use the 31 statement DATA statement at line 60000. These are READ sequentially for each new floor.

and represent the width of each room. To give variety to the pattern of rooms the starting point of the READ is determined randomly in the Semantic Control section and stored in P12. To start the drawing sequence the DATA pointer is RESTORED and then P10 dummy READs are made; V is used only as a temporary store. Once again we use the pointer variable, L, to hold the address of the top left-hand corner of the scene. We now read the next three DATA items from the list and store them in array D0; the number of sets of 3 is stored in the temporary variable, E.

The actual drawing of the vertical walls is done by lines 14150 to 14240 and their length is dependent on the value of H, the room depth variable. The wall characters are POEED into position as are the doors which occur at predetermined distances along them. Having drawn the first set of vertical walls the starting point is re-assigned in line 14250 and the next set is drawn in - this process is repeated until the walls have reached the bottom of the house.

The horizontal walls can now be drawn in, and their spacing is dependent on the value of the current element of array `NO`. The routine is located between lines 14220 and 14340.

As only the Black Tower has stairs, line 14360 causes the secondary series to skip over this section of the program. The

Black Tower has stairs located in opposite corners for each floor and these are POKED into position on lines 14360 and 14370. If you are on the ground floor of the Tower or in one of the secondary screens, a doorway is POKED into position by line 14380.

If you are stepping into the Tower or either of the secondary scenes for the first time, your character will be placed just inside the doorway; the check for this made in 14300 on P13 will only be 0 if you haven't moved in any other way.

The appropriate name for the castle-type scene is PRINTED into position by lines 14400 to 14490 and, in the case of the Tower, the floor number is also displayed.

Treasures can be found in the upper floors of the Tower and either of the two secondary scores, provided the value of PL is equal to or greater than 4 and a random factor is greater than 0.3. If these conditions are not met, control returns to the Scenario Control section and then back to the Movement routine. If both conditions are met, a random number of special treasure symbols are displayed; between 2 and 8 can appear and are shown as asterisks. They are positioned by the two temporary variables N1 and N2 which act as row and column co-ordinates. Provided the position selected is vacant an asterisk is POKED into place.

RATINGS

The ratings system used in the Valley program is based on a character achieving the maximum rating of 20, Master of Destiny, only after amassing 200,000 experience points.

Assigning a rating of 7 to an experience of 10,000 and a rating of 20 to 50,000 experience, the plotted curve began to show definite parabolic tendencies. After experimenting with the general equation of a parabola, $y^2 = 4ax$ or $y = c\sqrt{x}$ (where c is a constant), no simple values were found to fit. So... we compromised! Using the formula, $y = 0.067\sqrt{x}$, we managed to get y values of 6.7 for an x value of 10,000, 15 at 40,000 and 28 at 50,000.

Realizing that the rating should be based on experience and treasure, the x factor was defined as $x = \text{EX} + \text{TS}/3$. Then, in an attempt to penalize cowardice and rewarding those taking risks, a second factor, $\log(\text{ICO}/\text{TS} + 1) \div 1.50$, was added taking the number of turns to acquire your experience into the final equation.

1	Monster Food	15	Neonmancer
2	Peasant	16	Lovecrafter
3	Cadet	17	Paleid
4	Cannon Fodder	18	Supershero
5	Path Walker	19	Dragon Slayer
6	Novice Adventurer	20	Knight of the Valley
7	Survivor	21	Master of Combat
8	Adventurer	22	Dominator
9	Assassin	23	Prince of the Valley
10	Apprentice Hero	24	Guardian
11	Giant Killer	25	War Lord
12	Hero	26	Demon Killer
13	Master of the Sword	27	Lord of the Valley
14	Champion	28	Master of Destiny

Notes: The suffix numbers developed during the game and the character classifications that they represented in. They may be easily added into versions of the Rollenspiele as numbers with more than 100.

[illegible]

CONFIDENTIAL

If the adventure steps on either of the two side castles, one of each end of the path, he is offered the option of leaving the Valley. Regardless of his selection, his current rating is also computed and displayed at this point. If the player chooses to leave the Valley by leaving "Y", control is passed to the *Game procedure of level 00000*.

Because the castle is safe the player's character is "healed" of his wounds and readied for the Valley once more. This healing

[illegible]

consists of resetting the slowness to its maximum value and

ensuring a minimum combined strength of 30

Stepping on one of the two safe castles is the only way to leave the Valley in an upright position as the option to save your character on tape is then offered. Taking this option out of the Qut routine passes control to the Save routine at word

[illegible]

[illegible]

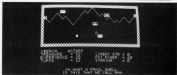
At the end of this routine, whether you reach it by saving the data on tape or by choosing not to save in the Quit routine and dropping through, all the current variables are cleared and a farewell message displayed.

RELTV

This routine is the one part of the program the player would rather not have executed! Mary and varied are the ways in which one can arrive at line 35000 and on all but one occasion the outcome is inevitable. The one exception is when you have been fortunate enough to collect the Amulet of Alarion and filled it with the six missing stones because this gives you a second life.

The test to see if you have the Amulet and its stones is made at \$5000 and if successful you are restored to life. The price is, however, high as you lose all your treasure together with the Amulet and its stones. Your combat strength and your psi power are both set to 30, the only value that remains the same after death is your intelligence.

If, as is most likely the case, you don't have the protection of

[illegible]

The end of the road for Channing was 1910, and there he left nothing but a shadow.

the Amulet and its six stores
then the same code with all the
variables being moved in line
store.

DATA

Rather than placing each data block with its relevant routine, we have chosen to lump it all together at the end of the program. The first block contains all the information needed to build the three cattle-type scenarios (see the relevant sections for more details on this). The second block of data holds the monitor information which is READ into the three arrays M\$1, M\$3 and N\$1 at the start of the program.

[illegible]

CREDITS

The Valley was devised and programmed by Peter Fowley, Peter Green, Ron Harris and Henry Budgett during 1981. The story of The Valley was written by Roger Mould.

Those wishing to avoid typing in the 16K of code or wishing to adapt the program for other Commodore machines will be interested to know that versions are available from ASP Software to run on CBM 2000/3000/4000, VIC-20, Commodore 64 and

8000 series systems. Tapes are priced at £11.45 all inclusive and the 8000 series version is available on an 8000 format disc for £13.95 inclusive. Orders should be sent to ASP Software at 145 Choring Cross Road, London, W17-6D 0RT.

TOWERS OF BRAHMA

The object of the game is to transfer rings from one pillar to another, although it's not as easy as you might think.

The following game is a simulation of the (supposed) project given by the supreme Hindu deity, Brahma, to his disciples. The object of the game is explained in the initial text and is not as simple as you might think from first inspection.

```

100 PRINT "WELCOME TO THE TOWER OF BRAHMA"
110 GOTO 120
120 PRINT "
130 PRINT "
140 PRINT "
150 PRINT "
160 PRINT "
170 PRINT "
180 PRINT "
190 PRINT "
200 PRINT "
210 PRINT "
220 PRINT "
230 PRINT "
240 PRINT "
250 PRINT "
260 PRINT "
270 PRINT "
280 PRINT "
290 PRINT "
300 PRINT "
310 PRINT "
320 PRINT "
330 PRINT "
340 PRINT "
350 PRINT "
360 PRINT "
370 PRINT "
380 PRINT "
390 PRINT "
400 PRINT "
410 PRINT "
420 PRINT "
430 PRINT "
440 PRINT "
450 PRINT "
460 PRINT "
470 PRINT "
480 PRINT "
490 PRINT "
500 PRINT "
510 PRINT "
520 PRINT "
530 PRINT "
540 PRINT "
550 PRINT "
560 PRINT "
570 PRINT "
580 PRINT "
590 PRINT "
600 PRINT "
610 PRINT "
620 PRINT "
630 PRINT "
640 PRINT "
650 PRINT "
660 PRINT "
670 PRINT "
680 PRINT "
690 PRINT "
700 PRINT "
710 PRINT "
720 PRINT "
730 PRINT "
740 PRINT "
750 PRINT "
760 PRINT "
770 PRINT "
780 PRINT "
790 PRINT "
800 PRINT "
810 PRINT "
820 PRINT "
830 PRINT "
840 PRINT "
850 PRINT "
860 PRINT "
870 PRINT "
880 PRINT "
890 PRINT "
900 PRINT "
910 PRINT "
920 PRINT "
930 PRINT "
940 PRINT "
950 PRINT "
960 PRINT "
970 PRINT "
980 PRINT "
990 PRINT "
1000 GOTO 100

```



Photos taken from the TV series *Samurai*

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

EDUCATIONAL

Micro Examination. — Test your friends and children with this multiple choice program. Originally published in **Computing Today**, July 1980.

Quiz Time — Assess your performance in terms of speed and accuracy with a multiple choice program.
Originally published in **Computing Today**, September 1982.

QUESTION NUMBER 1

NAME _____

DATE _____

1. 7-447 IS A

A. 7 TO 7 ELEMENT NUMBER

A. 800 COUNTER

A. 81000 COUNTER

A. 81000 COUNTER

ENTER A SINGLE LETTER, A OR B OR C OR D.

ENTER CORRECTLY. IF YOU ENTER SOMETHING ELSE, YOU WILL BE TOLD SO.

ENTER CORRECTED PAGE NUMBER HERE

Q. Q. THE UNIT OF MAGNETIC FLUX DENSITY
 YOUR ANSWER IS
 T. T. CORRECT REPLY
 NEXT QUESTION

Q. Q. THE UNIT OF MAGNETIC FLUX DENSITY
 YOUR ANSWER IS
 T. T. CORRECT REPLY
 NEXT QUESTION

```

QUESTION NUMBER 4
NAME _____ EL PINOCHET IS DEAD
74 45 00000 PLAN
HIGH POWER WITH LOW SPEED 0
LOW POWER WITH HIGH SPEED 0
HIGH POWER WITH HIGH SPEED 0
HIGH VOLTAGE WITH LOW FREQ 0
ENTER A SINGLE LETTER, A OR B OR C OR D.
THINK CAREFULLY! IF YOU CHOOSE YOUR
ANSWER INCORRECTLY ..... AND ANOTHER LETTER.
WHEN SATISFIED PRESS [ENTER] KEY

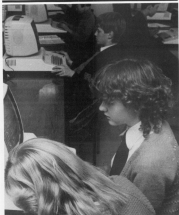
```

01-1918-2-22-70 04-1-25-1975-21
 THE ANSWERS WILL BE PRESENTED TO YOU
 IN A RANDOM ORDER
 YOU MUST ANSWER EACH QUESTION WITH A
 SINGLE WORD ONLY
 YOUR SPELLING MUST BE CORRECT
 IF YOU DO NOT KNOW THE ANSWER TO A
 QUESTION TYPE N
 IF YOU WISH TO END THE QUIZ TYPE N
 PRESS SPACE BAR TO CONTINUE

```
QUESTION NUMBER 7
=====
NAME          SSN 0987654321000000
=====
12-BIT BINARY COUNTERS CAN COUNT UP TO
=====
2048             X
2047             X
4096             X
4097             X
=====
ENTER A SINGLE LETTER A OR B OR C OR D.
(Enter carefully) IF YOU DO NOT WANT
ANSWER (A,B,C,D), AND NO-DOUBT ENTER
=====
WHEN ENTERED IS PAGE 000000 <E>
```

[illegible]

MICRO EXAMINATION



The multiple-choice question paper first vented its spite on thousands of the luckless "volunteers" who were trained during the last World War. The traditional essay type examination was too slow, favoured those who had the ability to disguise their ignorance with high-sounding jargon and, worst of all, the marking of the exam required some degree of professionalism. A gentleman by the name of Ballard is credited with the invention of presenting a question and four answers labeled A, B, C and D ... only

one of which is considered to be the "right" one. All the trainees had to do was place a cross in the "right" place. The technique was highly successful. A wide range of subjects could be covered in 100 question paper and could be marked by unskilled personnel in less than a minute by simply placing a prepared stencil over the paper. Although originally intended as a wartime expedient, the advantages were found to be so great that it has survived until the present day. The educational Establishment was naturally very critical, mumbled something like

Multiple choice exams represent an ideal entry point to the classroom for computers.

"... training a bunch of parrots etc etc" but the seal of respectability was finally given when technical colleges, and even universities, succumbed to the temptation. The computer is ideally situated as a tool in this area of education because it demands a minimum of keyboard interaction from the examinee. A question is flashed on the screen, demanding that ONE particular key is pressed. Traditional keyboard questions and answers suffer from the infuriating habit of marking you wrong even if a trivial spelling error is made or perhaps even an extra space.



GUIDING PRINCIPLES

Much of the criticism of multiple choice papers is due not to the method itself but the style of the questions. Too many of these questions are made up by small-minded individuals who often lack real knowledge of their subject and make up for it by composing, what they believe to be, "clever" tricks which are guaranteed to fool the poor student. The rules are simple:

- keep the question short and straightforward.
- make sure that all four of the answers are *superficially* correct.
- the correct answer expected should be that which is more universally true.
- don't make one of the

answers absurdly wrong because this is equivalent of reducing the number of choices by one.

I make sure you really know the correct answer yourself!

THE PROGRAMS

These have been in use for some time at a MOD Training Establishment (where I slave from dawn to dusk in return for the occasional bowl of rice).

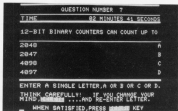
MULTIPLE CHOICE PREPARATION allows anyone to enter 25 questions, each with four answers and the right answer. In addition, the time allowed and the minimum pass mark can be entered. The end result of this activity is 'Data Tape' with the precious collection of mental sadism embedded within its magnetic bosom.



The second program, **MULTIPLE CHOICE EXAM**, is operated by the person being examined and begins with instructions for loading the data tape containing the questions. The questions are presented together with the four answers and the final score with percentages and a grading.



The examination program not only records the answers given but also computes a final analysis on the student's performance.



category appears on the screen. Should the examinee exceed the time allowed, the questions cease and the score page is presented immediately. Time lapses is given on each 'page'. Facilities exist during the preparation stage for producing the questions, displaying the questions, modifying them, saving them on the data tape and making additional copies from an existing tape. Subroutines are used to ensure that deficiencies in the data tape operating system (present in the 'old ROMS') are corrected by suitable patching. A subroutine for testing the keyboard as an INPUT FILE is also provided to prevent the program from breaking out should the operator inadvertently press RETURN before entering a character. The preamble for the questions are written as DATA/READ statements in order that modifications to suit local conditions are easy to

incorporate. It is reasonably 'idiot proof' but to fit the program into an 8K PET the ROM statements had to be curtailed. However, the program should be fairly straightforward to follow without these.



APPLICATION

Although the program is oriented towards the teaching profession, it could also prove useful in the home. It is educational in two senses: answering the questions which one member of the family has set with the aid of the 'PREPARATION' program and vice versa. It is probably harder to write a good set of 25 questions than it is to answer them. Some of the questions may of course be disputed (or rather the particular answer which is supposed to be correct) but even this is good. Plato and his followers spent most of their life learning by arguing. Because of the possibility of dispute, facilities are provided for modifying a question. Some


```

1000 INPUT "NAME: ";
1010 PRINT "NAME: ";
1020 GOTO 1470
1030 PRINT "DATE: ";
1040 PRINT "DATE: ";
1050 PRINT "DATE: ";
1060 PRINT "DATE: ";
1070 PRINT "DATE: ";
1080 PRINT "DATE: ";
1090 PRINT "DATE: ";
1100 PRINT "DATE: ";
1110 PRINT "DATE: ";
1120 PRINT "DATE: ";
1130 PRINT "DATE: ";
1140 PRINT "DATE: ";
1150 PRINT "DATE: ";
1160 PRINT "DATE: ";
1170 PRINT "DATE: ";
1180 PRINT "DATE: ";
1190 PRINT "DATE: ";
1200 PRINT "DATE: ";
1210 PRINT "DATE: ";
1220 PRINT "DATE: ";
1230 PRINT "DATE: ";
1240 PRINT "DATE: ";
1250 PRINT "DATE: ";
1260 PRINT "DATE: ";
1270 PRINT "DATE: ";
1280 PRINT "DATE: ";
1290 PRINT "DATE: ";
1300 PRINT "DATE: ";
1310 PRINT "DATE: ";
1320 PRINT "DATE: ";
1330 PRINT "DATE: ";
1340 PRINT "DATE: ";
1350 PRINT "DATE: ";
1360 PRINT "DATE: ";
1370 PRINT "DATE: ";
1380 PRINT "DATE: ";
1390 PRINT "DATE: ";
1400 PRINT "DATE: ";
1410 PRINT "DATE: ";
1420 PRINT "DATE: ";
1430 PRINT "DATE: ";
1440 PRINT "DATE: ";
1450 PRINT "DATE: ";
1460 PRINT "DATE: ";
1470 PRINT "DATE: ";
1480 PRINT "DATE: ";
1490 PRINT "DATE: ";
1500 PRINT "DATE: ";
1510 PRINT "DATE: ";
1520 PRINT "DATE: ";
1530 PRINT "DATE: ";
1540 PRINT "DATE: ";
1550 PRINT "DATE: ";
1560 PRINT "DATE: ";
1570 PRINT "DATE: ";
1580 PRINT "DATE: ";
1590 PRINT "DATE: ";
1600 PRINT "DATE: ";
1610 PRINT "DATE: ";
1620 PRINT "DATE: ";
1630 PRINT "DATE: ";
1640 PRINT "DATE: ";
1650 PRINT "DATE: ";
1660 PRINT "DATE: ";
1670 PRINT "DATE: ";
1680 PRINT "DATE: ";
1690 PRINT "DATE: ";
1700 PRINT "DATE: ";
1710 PRINT "DATE: ";
1720 PRINT "DATE: ";
1730 PRINT "DATE: ";
1740 PRINT "DATE: ";
1750 PRINT "DATE: ";
1760 PRINT "DATE: ";
1770 PRINT "DATE: ";
1780 PRINT "DATE: ";
1790 PRINT "DATE: ";
1800 PRINT "DATE: ";
1810 PRINT "DATE: ";
1820 PRINT "DATE: ";
1830 PRINT "DATE: ";
1840 PRINT "DATE: ";
1850 PRINT "DATE: ";
1860 PRINT "DATE: ";
1870 PRINT "DATE: ";
1880 PRINT "DATE: ";
1890 PRINT "DATE: ";
1900 PRINT "DATE: ";
1910 PRINT "DATE: ";
1920 PRINT "DATE: ";
1930 PRINT "DATE: ";
1940 PRINT "DATE: ";
1950 PRINT "DATE: ";
1960 PRINT "DATE: ";
1970 PRINT "DATE: ";
1980 PRINT "DATE: ";
1990 PRINT "DATE: ";
2000 PRINT "DATE: ";

```

Listing 1. The data preparation program used by the teacher or lecturer to produce the series of questions and their possible answers.

```

1000 INPUT "NAME: ";
1010 PRINT "NAME: ";
1020 GOTO 1470
1030 PRINT "DATE: ";
1040 PRINT "DATE: ";
1050 PRINT "DATE: ";
1060 PRINT "DATE: ";
1070 PRINT "DATE: ";
1080 PRINT "DATE: ";
1090 PRINT "DATE: ";
1100 PRINT "DATE: ";
1110 PRINT "DATE: ";
1120 PRINT "DATE: ";
1130 PRINT "DATE: ";
1140 PRINT "DATE: ";
1150 PRINT "DATE: ";
1160 PRINT "DATE: ";
1170 PRINT "DATE: ";
1180 PRINT "DATE: ";
1190 PRINT "DATE: ";
1200 PRINT "DATE: ";
1210 PRINT "DATE: ";
1220 PRINT "DATE: ";
1230 PRINT "DATE: ";
1240 PRINT "DATE: ";
1250 PRINT "DATE: ";
1260 PRINT "DATE: ";
1270 PRINT "DATE: ";
1280 PRINT "DATE: ";
1290 PRINT "DATE: ";
1300 PRINT "DATE: ";
1310 PRINT "DATE: ";
1320 PRINT "DATE: ";
1330 PRINT "DATE: ";
1340 PRINT "DATE: ";
1350 PRINT "DATE: ";
1360 PRINT "DATE: ";
1370 PRINT "DATE: ";
1380 PRINT "DATE: ";
1390 PRINT "DATE: ";
1400 PRINT "DATE: ";
1410 PRINT "DATE: ";
1420 PRINT "DATE: ";
1430 PRINT "DATE: ";
1440 PRINT "DATE: ";
1450 PRINT "DATE: ";
1460 PRINT "DATE: ";
1470 PRINT "DATE: ";
1480 PRINT "DATE: ";
1490 PRINT "DATE: ";
1500 PRINT "DATE: ";
1510 PRINT "DATE: ";
1520 PRINT "DATE: ";
1530 PRINT "DATE: ";
1540 PRINT "DATE: ";
1550 PRINT "DATE: ";
1560 PRINT "DATE: ";
1570 PRINT "DATE: ";
1580 PRINT "DATE: ";
1590 PRINT "DATE: ";
1600 PRINT "DATE: ";
1610 PRINT "DATE: ";
1620 PRINT "DATE: ";
1630 PRINT "DATE: ";
1640 PRINT "DATE: ";
1650 PRINT "DATE: ";
1660 PRINT "DATE: ";
1670 PRINT "DATE: ";
1680 PRINT "DATE: ";
1690 PRINT "DATE: ";
1700 PRINT "DATE: ";
1710 PRINT "DATE: ";
1720 PRINT "DATE: ";
1730 PRINT "DATE: ";
1740 PRINT "DATE: ";
1750 PRINT "DATE: ";
1760 PRINT "DATE: ";
1770 PRINT "DATE: ";
1780 PRINT "DATE: ";
1790 PRINT "DATE: ";
1800 PRINT "DATE: ";
1810 PRINT "DATE: ";
1820 PRINT "DATE: ";
1830 PRINT "DATE: ";
1840 PRINT "DATE: ";
1850 PRINT "DATE: ";
1860 PRINT "DATE: ";
1870 PRINT "DATE: ";
1880 PRINT "DATE: ";
1890 PRINT "DATE: ";
1900 PRINT "DATE: ";
1910 PRINT "DATE: ";
1920 PRINT "DATE: ";
1930 PRINT "DATE: ";
1940 PRINT "DATE: ";
1950 PRINT "DATE: ";
1960 PRINT "DATE: ";
1970 PRINT "DATE: ";
1980 PRINT "DATE: ";
1990 PRINT "DATE: ";
2000 PRINT "DATE: ";

```

```

1000 INPUT "NAME: ";
1010 PRINT "NAME: ";
1020 GOTO 1470
1030 PRINT "DATE: ";
1040 PRINT "DATE: ";
1050 PRINT "DATE: ";
1060 PRINT "DATE: ";
1070 PRINT "DATE: ";
1080 PRINT "DATE: ";
1090 PRINT "DATE: ";
1100 PRINT "DATE: ";
1110 PRINT "DATE: ";
1120 PRINT "DATE: ";
1130 PRINT "DATE: ";
1140 PRINT "DATE: ";
1150 PRINT "DATE: ";
1160 PRINT "DATE: ";
1170 PRINT "DATE: ";
1180 PRINT "DATE: ";
1190 PRINT "DATE: ";
1200 PRINT "DATE: ";
1210 PRINT "DATE: ";
1220 PRINT "DATE: ";
1230 PRINT "DATE: ";
1240 PRINT "DATE: ";
1250 PRINT "DATE: ";
1260 PRINT "DATE: ";
1270 PRINT "DATE: ";
1280 PRINT "DATE: ";
1290 PRINT "DATE: ";
1300 PRINT "DATE: ";
1310 PRINT "DATE: ";
1320 PRINT "DATE: ";
1330 PRINT "DATE: ";
1340 PRINT "DATE: ";
1350 PRINT "DATE: ";
1360 PRINT "DATE: ";
1370 PRINT "DATE: ";
1380 PRINT "DATE: ";
1390 PRINT "DATE: ";
1400 PRINT "DATE: ";
1410 PRINT "DATE: ";
1420 PRINT "DATE: ";
1430 PRINT "DATE: ";
1440 PRINT "DATE: ";
1450 PRINT "DATE: ";
1460 PRINT "DATE: ";
1470 PRINT "DATE: ";
1480 PRINT "DATE: ";
1490 PRINT "DATE: ";
1500 PRINT "DATE: ";
1510 PRINT "DATE: ";
1520 PRINT "DATE: ";
1530 PRINT "DATE: ";
1540 PRINT "DATE: ";
1550 PRINT "DATE: ";
1560 PRINT "DATE: ";
1570 PRINT "DATE: ";
1580 PRINT "DATE: ";
1590 PRINT "DATE: ";
1600 PRINT "DATE: ";
1610 PRINT "DATE: ";
1620 PRINT "DATE: ";
1630 PRINT "DATE: ";
1640 PRINT "DATE: ";
1650 PRINT "DATE: ";
1660 PRINT "DATE: ";
1670 PRINT "DATE: ";
1680 PRINT "DATE: ";
1690 PRINT "DATE: ";
1700 PRINT "DATE: ";
1710 PRINT "DATE: ";
1720 PRINT "DATE: ";
1730 PRINT "DATE: ";
1740 PRINT "DATE: ";
1750 PRINT "DATE: ";
1760 PRINT "DATE: ";
1770 PRINT "DATE: ";
1780 PRINT "DATE: ";
1790 PRINT "DATE: ";
1800 PRINT "DATE: ";
1810 PRINT "DATE: ";
1820 PRINT "DATE: ";
1830 PRINT "DATE: ";
1840 PRINT "DATE: ";
1850 PRINT "DATE: ";
1860 PRINT "DATE: ";
1870 PRINT "DATE: ";
1880 PRINT "DATE: ";
1890 PRINT "DATE: ";
1900 PRINT "DATE: ";
1910 PRINT "DATE: ";
1920 PRINT "DATE: ";
1930 PRINT "DATE: ";
1940 PRINT "DATE: ";
1950 PRINT "DATE: ";
1960 PRINT "DATE: ";
1970 PRINT "DATE: ";
1980 PRINT "DATE: ";
1990 PRINT "DATE: ";
2000 PRINT "DATE: ";

```

Listing 2. The examination program uses the prepared data tape, regardless of the type of subject under examination.

Because of the limited characters per line and number of lines on the PET screen, the following rules apply to preparing questions: The question must be limited to

TWO lines. Remember to use the Space key to turn the corner to the second line ... not the Return key. Each answer must be limited to one line.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

QUIZ TIME

PLEASE TYPE YOUR NAME NOW
 PLEASE TYPE YOUR COURSE CODE OR 0 FOR
 NO. YOU WANT TO BE GIVEN INFORMATION ON
 THE QUIZ.

Q.1 THE UNIT OF MAGNETIC FLUX DENSITY
 YOUR ANSWER IS: T
 THE CORRECT ANSWER IS: T
 I WILL REPEAT THE QUESTION

Q.2 THE UNIT OF MAGNETIC FLUX DENSITY
 YOUR ANSWER IS: T
 CORRECT REPLY
 NEXT QUESTION

Q.3 THE UNIT OF CONDUCTANCE
 YOUR ANSWER IS: A

THANK YOU VERY
 FOR ATTEMPTING THIS QUIZ
 YOU ATTEMPTED 20 QUESTIONS
 YOUR INTERNAL SCORE IS 22 OUT OF A
 POSSIBLE 100
 YOUR TOTAL RESPONSE TIME WAS 782.4
 SECONDS

PRESS SPACE BAR TO CONTINUE

A multiple choice test program that's been proved in the classroom.

While all know that the SI unit of current is the Ampere but do you know the name of the unit of luminous intensity and are you sure you can spell the name of the unit of conductance correctly? If the answer is no, then your friendly computer will prompt you with the answer so that by the second or third re-run, your score should have increased significantly until eventually you achieve a perfect score. If you are a high scorer anyway you can still use the quiz to help sharpen your keyboard response in a test time.

SCORING AND TIMING

Out of a possible 24 questions in this quiz, only 20 are asked at any one session and these are chosen and presented in a random order. This makes it impossible for you to obtain a high score and a fast time simply by remembering the order of the answers. A note is kept by the computer as each question is asked so that repetition is avoided; this is the function of the array A(). The method of scoring is as follows: you are allowed two attempts at each question and score five or three marks for a correct answer at your first or second attempts respectively. If you are wrong at your second go or if you reply with 'D' (for don't know) then you are given the correct answer, but are still asked the question again to help you to fix the answer in your mind. A correct reply at this stage is

Some typical illustrations from the program. If you get a question wrong it is asked again to ensure that you understood the correct answer.

worth one mark but if you are careless and give the wrong answer you will incur a penalty of minus one!

While that explains the scoring of a normal run you may, if you wish, exit from the program by typing 'X' and for that question you will score minus three marks. After the normal 20 questions or after an 'X' reply you will be shown a list of the unasked questions so that you can test yourself on these as well but without the benefit of the model answers. There is one further penalty which will be incurred by anyone asleep at the keyboard. After a don't know reply to a question you are told the correct answer — so if you give a 'D' reply a second time to the same question, you will be given a penalty of minus five marks to help wake you up!

The normal total score will be between 20 and 100 but you should get 80 or more after two runs. At this point your aim should be 100% in a fast time. The method of timing used in this quiz is to add up your individual response times to each question — the time taken by the computer to present each question is not included. The total time recorded is rounded to the nearest tenth of a second and you can use this, together with the score, to check your progress. If you have a printer then you can receive a certificate which includes a record of your score and speed.

The printer used to produce the certificate was the CRN 4022; you may have to modify subroutine SR6 if you use a different model. The printer is, of course, an optional extra in this program and may be omitted altogether by deleting lines 750 to 810 in the main program and deleting the whole of SR6.

A problem common to many interactive programs is that of a program crash which occurs if, in response to an INPUT statement, the Return key is hit before any other key. In a quiz this mistake would be very tiresome, particularly if the participant was half way through the questions. To

prevent this crash, line 3080 has been modified and line 3090 added in SR3. If now, in response to 'YOUR ANSWER?', the Return key is hit then the underline will prevent a program crash. If, however, a normal answer is typed then the first character will replace the underline and a simple comparison with the model answer can take place.

CHANGING THE QUESTIONS

The questions in this SI quiz all require single word answers which must be spelt correctly. It is a simple task to modify the program so that questions are asked on an entirely different topic provided that you can keep to the same format of single word answers. An obvious alternative quiz of this type would include questions and answers such as:

WHAT IS THE CAPITAL OF FRANCE?

If you look at line 3040 you will see that each question is a concatenation of a fixed part (QNS) and a variable part (QV). The variable part of the questions and the model answers are stored in the DATA statements which are very easy to change. If, for some reason, it is necessary to present the questions in a predetermined order then line 3030 together with SR6 should be deleted. It would, however, be a relatively simple matter to change the program so that one of several predetermined selections was chosen at random on each run.

Suppose, however, you wish to accept two or more correct answers. Clearly all the acceptable answers must be included in your DATA statements and the HEAD statement in line 6030 of SR6 must be changed as well as the test for the correct response in line 3100. If some of your questions have one correct answer while others have two or more then a further modification is necessary to allow for these variations. Including null strings in the data would be one

of several ways of doing this.

A word of warning should be heeded by those of you who wish to create a new set of questions. Make sure that your questions are not ambiguous and also double check on the correctness of your model answers. It is a good idea to shelter behind the protection of an established authority on the

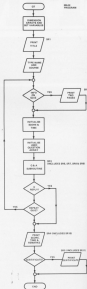


Fig. 1. The main flowchart.

subject. In the SI quiz, my references to the well known book **Physical Constants** by Kaye and Laby (14th Edition).

THE PROGRAM

Even if you are not interested in this kind of quiz I hope that you may find some useful programming ideas. First, you must decide precisely what it is you wish the program to achieve. The outline scheme for this quiz, but not all of the detail, has already been described. Now examine the main program flowchart (Fig. 1) and you will see that most of the detailed programming is put into the subroutines. A subroutine is a module which can be designed and tested as an entity and used in other programs if required. The subroutine will operate in any program subject only to the correct transfer of data to and from the subroutine. This method of design is to be recommended because it is easier to achieve high reliability and good maintenance, that is, the ability to modify the program to meet new conditions with no or few faults. Debugging a new program is easier too!

It is highly desirable that the user should be given sufficient information on the screen to enable him to use the program correctly. On the other hand, not all of the information need be repeated on each re-run and a choice is given to the user,

such as in line 700, which may save much frustration. The time taken to read a screen page of information varies considerably from user to user and depends, in part, on previous usage. For this reason, information should not be presented for a fixed time; instead, the user should be given a simple instruction which, when carried out, 'turns' the page. In this quiz the instruction used is 'PRESS SPACE BAR TO CONTINUE' but, as you will see from lines 1080 and 1090 in SR1, the program would continue if any key is pressed. The reason for giving the specific instruction rather than the more general 'PRESS ANY KEY' is that the latter may raise a doubt in the user's mind about the possibility of different effects resulting from pressing different keys.

The three line press space bar routine occurs a number of times throughout the program and could have been turned into a four line subroutine which was called by a GOSUB on each occasion as required. The message structure and the control structure used in this program could also be simplified if only a single reply to each question is permitted. This, however, is more appropriate to a competitive type of quiz than to a self-learning scheme with rewards and penalties. The variable P is used both for control and for scoring (see SR1).

The variable T is used to store the total time to reply to

each question. This is set to zero at the beginning of a run (line 300) and is measured in 'ticks' (ticks are the internal divisions of time on the PET and represent 50ths of a second, Ed.) while the increments of time are being added together. T is finally converted to seconds (rounded to the nearest tenth of a second) in line 1100.

The 4002 printer used in this quiz to produce a certificate has a wide variety of format controls. Use has been made of the control (CHFB 1) in lines 5120, 5130, 5240 and 5370, to change the print size. Clearly these lines may have to be modified to suit the printer available to you.

RUNNING THE PROGRAM

This is perfectly straightforward, provided that the user follows the screen instructions. Yes/no answers to questions such as do you wish to be given information on the quiz? may be given in full or abbreviated to Y or N. The program has been tested on many first year students who have used it without supervision. Reactions to the quiz have shown me that many students like this kind of learning process. It is, however wise to take note of the final message which is to remember that knowing the names of the units is not the same as understanding their meaning.

```
100 REM ** UNTHINKABLE AND DUMBEST QUIZ
110 DIM SQRN(1) ** DIM SQRN(1) TO STORE FOR QUESTION
120 DIM SQRN(1) ** NO OF QUESTIONS
130 DIM SQRN(1) ** NO OF QUESTIONS
140 DIM SQRN(1) ** NO OF QUESTIONS
150 DIM SQRN(1) ** NO OF QUESTIONS
160 DIM SQRN(1) ** NO OF QUESTIONS
170 DIM SQRN(1) ** NO OF QUESTIONS
180 DIM SQRN(1) ** NO OF QUESTIONS
190 DIM SQRN(1) ** NO OF QUESTIONS
200 DIM SQRN(1) ** NO OF QUESTIONS
210 DIM SQRN(1) ** NO OF QUESTIONS
220 DIM SQRN(1) ** NO OF QUESTIONS
230 DIM SQRN(1) ** NO OF QUESTIONS
240 DIM SQRN(1) ** NO OF QUESTIONS
250 DIM SQRN(1) ** NO OF QUESTIONS
260 DIM SQRN(1) ** NO OF QUESTIONS
270 DIM SQRN(1) ** NO OF QUESTIONS
280 DIM SQRN(1) ** NO OF QUESTIONS
290 DIM SQRN(1) ** NO OF QUESTIONS
300 DIM SQRN(1) ** NO OF QUESTIONS
310 DIM SQRN(1) ** NO OF QUESTIONS
320 DIM SQRN(1) ** NO OF QUESTIONS
330 DIM SQRN(1) ** NO OF QUESTIONS
340 DIM SQRN(1) ** NO OF QUESTIONS
350 DIM SQRN(1) ** NO OF QUESTIONS
360 DIM SQRN(1) ** NO OF QUESTIONS
370 DIM SQRN(1) ** NO OF QUESTIONS
380 DIM SQRN(1) ** NO OF QUESTIONS
390 DIM SQRN(1) ** NO OF QUESTIONS
400 DIM SQRN(1) ** NO OF QUESTIONS
410 DIM SQRN(1) ** NO OF QUESTIONS
420 DIM SQRN(1) ** NO OF QUESTIONS
430 DIM SQRN(1) ** NO OF QUESTIONS
440 DIM SQRN(1) ** NO OF QUESTIONS
450 DIM SQRN(1) ** NO OF QUESTIONS
460 DIM SQRN(1) ** NO OF QUESTIONS
470 DIM SQRN(1) ** NO OF QUESTIONS
480 DIM SQRN(1) ** NO OF QUESTIONS
490 DIM SQRN(1) ** NO OF QUESTIONS
500 DIM SQRN(1) ** NO OF QUESTIONS
510 DIM SQRN(1) ** NO OF QUESTIONS
520 DIM SQRN(1) ** NO OF QUESTIONS
530 DIM SQRN(1) ** NO OF QUESTIONS
540 DIM SQRN(1) ** NO OF QUESTIONS
550 DIM SQRN(1) ** NO OF QUESTIONS
560 DIM SQRN(1) ** NO OF QUESTIONS
570 DIM SQRN(1) ** NO OF QUESTIONS
580 DIM SQRN(1) ** NO OF QUESTIONS
590 DIM SQRN(1) ** NO OF QUESTIONS
600 DIM SQRN(1) ** NO OF QUESTIONS
610 DIM SQRN(1) ** NO OF QUESTIONS
620 DIM SQRN(1) ** NO OF QUESTIONS
630 DIM SQRN(1) ** NO OF QUESTIONS
640 DIM SQRN(1) ** NO OF QUESTIONS
650 DIM SQRN(1) ** NO OF QUESTIONS
660 DIM SQRN(1) ** NO OF QUESTIONS
670 DIM SQRN(1) ** NO OF QUESTIONS
680 DIM SQRN(1) ** NO OF QUESTIONS
690 DIM SQRN(1) ** NO OF QUESTIONS
700 DIM SQRN(1) ** NO OF QUESTIONS
710 DIM SQRN(1) ** NO OF QUESTIONS
720 DIM SQRN(1) ** NO OF QUESTIONS
730 DIM SQRN(1) ** NO OF QUESTIONS
740 DIM SQRN(1) ** NO OF QUESTIONS
750 DIM SQRN(1) ** NO OF QUESTIONS
760 DIM SQRN(1) ** NO OF QUESTIONS
770 DIM SQRN(1) ** NO OF QUESTIONS
780 DIM SQRN(1) ** NO OF QUESTIONS
790 DIM SQRN(1) ** NO OF QUESTIONS
800 DIM SQRN(1) ** NO OF QUESTIONS
810 DIM SQRN(1) ** NO OF QUESTIONS
820 DIM SQRN(1) ** NO OF QUESTIONS
830 DIM SQRN(1) ** NO OF QUESTIONS
840 DIM SQRN(1) ** NO OF QUESTIONS
850 DIM SQRN(1) ** NO OF QUESTIONS
860 DIM SQRN(1) ** NO OF QUESTIONS
870 DIM SQRN(1) ** NO OF QUESTIONS
880 DIM SQRN(1) ** NO OF QUESTIONS
890 DIM SQRN(1) ** NO OF QUESTIONS
900 DIM SQRN(1) ** NO OF QUESTIONS
910 DIM SQRN(1) ** NO OF QUESTIONS
920 DIM SQRN(1) ** NO OF QUESTIONS
930 DIM SQRN(1) ** NO OF QUESTIONS
940 DIM SQRN(1) ** NO OF QUESTIONS
950 DIM SQRN(1) ** NO OF QUESTIONS
960 DIM SQRN(1) ** NO OF QUESTIONS
970 DIM SQRN(1) ** NO OF QUESTIONS
980 DIM SQRN(1) ** NO OF QUESTIONS
990 DIM SQRN(1) ** NO OF QUESTIONS
```

```
1000 REM ** UNTHINKABLE AND DUMBEST QUIZ
1010 DIM SQRN(1) ** DIM SQRN(1) TO STORE FOR QUESTION
1020 DIM SQRN(1) ** NO OF QUESTIONS
1030 DIM SQRN(1) ** NO OF QUESTIONS
1040 DIM SQRN(1) ** NO OF QUESTIONS
1050 DIM SQRN(1) ** NO OF QUESTIONS
1060 DIM SQRN(1) ** NO OF QUESTIONS
1070 DIM SQRN(1) ** NO OF QUESTIONS
1080 DIM SQRN(1) ** NO OF QUESTIONS
1090 DIM SQRN(1) ** NO OF QUESTIONS
1100 DIM SQRN(1) ** NO OF QUESTIONS
1110 DIM SQRN(1) ** NO OF QUESTIONS
1120 DIM SQRN(1) ** NO OF QUESTIONS
1130 DIM SQRN(1) ** NO OF QUESTIONS
1140 DIM SQRN(1) ** NO OF QUESTIONS
1150 DIM SQRN(1) ** NO OF QUESTIONS
1160 DIM SQRN(1) ** NO OF QUESTIONS
1170 DIM SQRN(1) ** NO OF QUESTIONS
1180 DIM SQRN(1) ** NO OF QUESTIONS
1190 DIM SQRN(1) ** NO OF QUESTIONS
1200 DIM SQRN(1) ** NO OF QUESTIONS
1210 DIM SQRN(1) ** NO OF QUESTIONS
1220 DIM SQRN(1) ** NO OF QUESTIONS
1230 DIM SQRN(1) ** NO OF QUESTIONS
1240 DIM SQRN(1) ** NO OF QUESTIONS
1250 DIM SQRN(1) ** NO OF QUESTIONS
1260 DIM SQRN(1) ** NO OF QUESTIONS
1270 DIM SQRN(1) ** NO OF QUESTIONS
1280 DIM SQRN(1) ** NO OF QUESTIONS
1290 DIM SQRN(1) ** NO OF QUESTIONS
1300 DIM SQRN(1) ** NO OF QUESTIONS
1310 DIM SQRN(1) ** NO OF QUESTIONS
1320 DIM SQRN(1) ** NO OF QUESTIONS
1330 DIM SQRN(1) ** NO OF QUESTIONS
1340 DIM SQRN(1) ** NO OF QUESTIONS
1350 DIM SQRN(1) ** NO OF QUESTIONS
1360 DIM SQRN(1) ** NO OF QUESTIONS
1370 DIM SQRN(1) ** NO OF QUESTIONS
1380 DIM SQRN(1) ** NO OF QUESTIONS
1390 DIM SQRN(1) ** NO OF QUESTIONS
1400 DIM SQRN(1) ** NO OF QUESTIONS
1410 DIM SQRN(1) ** NO OF QUESTIONS
1420 DIM SQRN(1) ** NO OF QUESTIONS
1430 DIM SQRN(1) ** NO OF QUESTIONS
1440 DIM SQRN(1) ** NO OF QUESTIONS
1450 DIM SQRN(1) ** NO OF QUESTIONS
1460 DIM SQRN(1) ** NO OF QUESTIONS
1470 DIM SQRN(1) ** NO OF QUESTIONS
1480 DIM SQRN(1) ** NO OF QUESTIONS
1490 DIM SQRN(1) ** NO OF QUESTIONS
1500 DIM SQRN(1) ** NO OF QUESTIONS
1510 DIM SQRN(1) ** NO OF QUESTIONS
1520 DIM SQRN(1) ** NO OF QUESTIONS
1530 DIM SQRN(1) ** NO OF QUESTIONS
1540 DIM SQRN(1) ** NO OF QUESTIONS
1550 DIM SQRN(1) ** NO OF QUESTIONS
1560 DIM SQRN(1) ** NO OF QUESTIONS
1570 DIM SQRN(1) ** NO OF QUESTIONS
1580 DIM SQRN(1) ** NO OF QUESTIONS
1590 DIM SQRN(1) ** NO OF QUESTIONS
1600 DIM SQRN(1) ** NO OF QUESTIONS
1610 DIM SQRN(1) ** NO OF QUESTIONS
1620 DIM SQRN(1) ** NO OF QUESTIONS
1630 DIM SQRN(1) ** NO OF QUESTIONS
1640 DIM SQRN(1) ** NO OF QUESTIONS
1650 DIM SQRN(1) ** NO OF QUESTIONS
1660 DIM SQRN(1) ** NO OF QUESTIONS
1670 DIM SQRN(1) ** NO OF QUESTIONS
1680 DIM SQRN(1) ** NO OF QUESTIONS
1690 DIM SQRN(1) ** NO OF QUESTIONS
1700 DIM SQRN(1) ** NO OF QUESTIONS
1710 DIM SQRN(1) ** NO OF QUESTIONS
1720 DIM SQRN(1) ** NO OF QUESTIONS
1730 DIM SQRN(1) ** NO OF QUESTIONS
1740 DIM SQRN(1) ** NO OF QUESTIONS
1750 DIM SQRN(1) ** NO OF QUESTIONS
1760 DIM SQRN(1) ** NO OF QUESTIONS
1770 DIM SQRN(1) ** NO OF QUESTIONS
1780 DIM SQRN(1) ** NO OF QUESTIONS
1790 DIM SQRN(1) ** NO OF QUESTIONS
1800 DIM SQRN(1) ** NO OF QUESTIONS
1810 DIM SQRN(1) ** NO OF QUESTIONS
1820 DIM SQRN(1) ** NO OF QUESTIONS
1830 DIM SQRN(1) ** NO OF QUESTIONS
1840 DIM SQRN(1) ** NO OF QUESTIONS
1850 DIM SQRN(1) ** NO OF QUESTIONS
1860 DIM SQRN(1) ** NO OF QUESTIONS
1870 DIM SQRN(1) ** NO OF QUESTIONS
1880 DIM SQRN(1) ** NO OF QUESTIONS
1890 DIM SQRN(1) ** NO OF QUESTIONS
1900 DIM SQRN(1) ** NO OF QUESTIONS
1910 DIM SQRN(1) ** NO OF QUESTIONS
1920 DIM SQRN(1) ** NO OF QUESTIONS
1930 DIM SQRN(1) ** NO OF QUESTIONS
1940 DIM SQRN(1) ** NO OF QUESTIONS
1950 DIM SQRN(1) ** NO OF QUESTIONS
1960 DIM SQRN(1) ** NO OF QUESTIONS
1970 DIM SQRN(1) ** NO OF QUESTIONS
1980 DIM SQRN(1) ** NO OF QUESTIONS
1990 DIM SQRN(1) ** NO OF QUESTIONS
```

[illegible][illegible][illegible]

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 8 of this issue.

APPLICATIONS

Multipurpose Records — Set up your own filing system which enables you to store, search, edit and retrieve data.

Originally published in **Computing Today**, August 1980.

VIC Editor — Take one VIC-20 and add this program and what do you have? A VIC-20 with an 'enlarged' screen.

Originally published in **Computing Today**, November 1982.

Commodore Communications — Get Commodores talking to each other using this application.

Originally published in **Computing Today**, November 1982.

Address Book — Compile an address book, or any other similar list, and throw away those bits of paper.

Multicolumn Records — A multipurpose data base program for use at home or in the office.

Originally published in **Computing Today**, July 1981.




```

100 PRINT "PRINT 1"
110 PRINT "PRINT 2"
120 PRINT "PRINT 3"
130 PRINT "PRINT 4"
140 PRINT "PRINT 5"
150 PRINT "PRINT 6"
160 PRINT "PRINT 7"
170 PRINT "PRINT 8"
180 PRINT "PRINT 9"
190 PRINT "PRINT 10"
200 PRINT "PRINT 11"
210 PRINT "PRINT 12"
220 PRINT "PRINT 13"
230 PRINT "PRINT 14"
240 PRINT "PRINT 15"
250 PRINT "PRINT 16"
260 PRINT "PRINT 17"
270 PRINT "PRINT 18"
280 PRINT "PRINT 19"
290 PRINT "PRINT 20"
300 PRINT "PRINT 21"
310 PRINT "PRINT 22"
320 PRINT "PRINT 23"
330 PRINT "PRINT 24"
340 PRINT "PRINT 25"
350 PRINT "PRINT 26"
360 PRINT "PRINT 27"
370 PRINT "PRINT 28"
380 PRINT "PRINT 29"
390 PRINT "PRINT 30"
400 PRINT "PRINT 31"
410 PRINT "PRINT 32"
420 PRINT "PRINT 33"
430 PRINT "PRINT 34"
440 PRINT "PRINT 35"
450 PRINT "PRINT 36"
460 PRINT "PRINT 37"
470 PRINT "PRINT 38"
480 PRINT "PRINT 39"
490 PRINT "PRINT 40"
500 PRINT "PRINT 41"
510 PRINT "PRINT 42"
520 PRINT "PRINT 43"
530 PRINT "PRINT 44"
540 PRINT "PRINT 45"
550 PRINT "PRINT 46"
560 PRINT "PRINT 47"
570 PRINT "PRINT 48"
580 PRINT "PRINT 49"
590 PRINT "PRINT 50"
600 PRINT "PRINT 51"
610 PRINT "PRINT 52"
620 PRINT "PRINT 53"
630 PRINT "PRINT 54"
640 PRINT "PRINT 55"
650 PRINT "PRINT 56"
660 PRINT "PRINT 57"
670 PRINT "PRINT 58"
680 PRINT "PRINT 59"
690 PRINT "PRINT 60"
700 PRINT "PRINT 61"
710 PRINT "PRINT 62"
720 PRINT "PRINT 63"
730 PRINT "PRINT 64"
740 PRINT "PRINT 65"
750 PRINT "PRINT 66"
760 PRINT "PRINT 67"
770 PRINT "PRINT 68"
780 PRINT "PRINT 69"
790 PRINT "PRINT 70"
800 PRINT "PRINT 71"
810 PRINT "PRINT 72"
820 PRINT "PRINT 73"
830 PRINT "PRINT 74"
840 PRINT "PRINT 75"
850 PRINT "PRINT 76"
860 PRINT "PRINT 77"
870 PRINT "PRINT 78"
880 PRINT "PRINT 79"
890 PRINT "PRINT 80"
900 PRINT "PRINT 81"
910 PRINT "PRINT 82"
920 PRINT "PRINT 83"
930 PRINT "PRINT 84"
940 PRINT "PRINT 85"
950 PRINT "PRINT 86"
960 PRINT "PRINT 87"
970 PRINT "PRINT 88"
980 PRINT "PRINT 89"
990 PRINT "PRINT 90"

```

```

1000 PRINT
1010 INPUT "NAME: "; NAME
1020 INPUT "AGE: "; AGE
1030 PRINT "NAME AND AGE: " NAME; AGE
1040 INPUT "NAME: "; NAME
1050 INPUT "AGE: "; AGE
1060 PRINT "NAME AND AGE: " NAME; AGE
1070 INPUT "NAME: "; NAME
1080 INPUT "AGE: "; AGE
1090 PRINT "NAME AND AGE: " NAME; AGE
1100 INPUT "NAME: "; NAME
1110 INPUT "AGE: "; AGE
1120 PRINT "NAME AND AGE: " NAME; AGE
1130 INPUT "NAME: "; NAME
1140 INPUT "AGE: "; AGE
1150 PRINT "NAME AND AGE: " NAME; AGE
1160 INPUT "NAME: "; NAME
1170 INPUT "AGE: "; AGE
1180 PRINT "NAME AND AGE: " NAME; AGE
1190 INPUT "NAME: "; NAME
1200 INPUT "AGE: "; AGE
1210 PRINT "NAME AND AGE: " NAME; AGE
1220 INPUT "NAME: "; NAME
1230 INPUT "AGE: "; AGE
1240 PRINT "NAME AND AGE: " NAME; AGE
1250 INPUT "NAME: "; NAME
1260 INPUT "AGE: "; AGE
1270 PRINT "NAME AND AGE: " NAME; AGE
1280 INPUT "NAME: "; NAME
1290 INPUT "AGE: "; AGE
1300 PRINT "NAME AND AGE: " NAME; AGE
1310 INPUT "NAME: "; NAME
1320 INPUT "AGE: "; AGE
1330 PRINT "NAME AND AGE: " NAME; AGE
1340 INPUT "NAME: "; NAME
1350 INPUT "AGE: "; AGE
1360 PRINT "NAME AND AGE: " NAME; AGE
1370 INPUT "NAME: "; NAME
1380 INPUT "AGE: "; AGE
1390 PRINT "NAME AND AGE: " NAME; AGE
1400 INPUT "NAME: "; NAME
1410 INPUT "AGE: "; AGE
1420 PRINT "NAME AND AGE: " NAME; AGE
1430 INPUT "NAME: "; NAME
1440 INPUT "AGE: "; AGE
1450 PRINT "NAME AND AGE: " NAME; AGE
1460 INPUT "NAME: "; NAME
1470 INPUT "AGE: "; AGE
1480 PRINT "NAME AND AGE: " NAME; AGE
1490 INPUT "NAME: "; NAME
1500 INPUT "AGE: "; AGE
1510 PRINT "NAME AND AGE: " NAME; AGE
1520 INPUT "NAME: "; NAME
1530 INPUT "AGE: "; AGE
1540 PRINT "NAME AND AGE: " NAME; AGE
1550 INPUT "NAME: "; NAME
1560 INPUT "AGE: "; AGE
1570 PRINT "NAME AND AGE: " NAME; AGE
1580 INPUT "NAME: "; NAME
1590 INPUT "AGE: "; AGE
1600 PRINT "NAME AND AGE: " NAME; AGE
1610 INPUT "NAME: "; NAME
1620 INPUT "AGE: "; AGE
1630 PRINT "NAME AND AGE: " NAME; AGE
1640 INPUT "NAME: "; NAME
1650 INPUT "AGE: "; AGE
1660 PRINT "NAME AND AGE: " NAME; AGE
1670 INPUT "NAME: "; NAME
1680 INPUT "AGE: "; AGE
1690 PRINT "NAME AND AGE: " NAME; AGE
1700 INPUT "NAME: "; NAME
1710 INPUT "AGE: "; AGE
1720 PRINT "NAME AND AGE: " NAME; AGE
1730 INPUT "NAME: "; NAME
1740 INPUT "AGE: "; AGE
1750 PRINT "NAME AND AGE: " NAME; AGE
1760 INPUT "NAME: "; NAME
1770 INPUT "AGE: "; AGE
1780 PRINT "NAME AND AGE: " NAME; AGE
1790 INPUT "NAME: "; NAME
1800 INPUT "AGE: "; AGE
1810 PRINT "NAME AND AGE: " NAME; AGE
1820 INPUT "NAME: "; NAME
1830 INPUT "AGE: "; AGE
1840 PRINT "NAME AND AGE: " NAME; AGE
1850 INPUT "NAME: "; NAME
1860 INPUT "AGE: "; AGE
1870 PRINT "NAME AND AGE: " NAME; AGE
1880 INPUT "NAME: "; NAME
1890 INPUT "AGE: "; AGE
1900 PRINT "NAME AND AGE: " NAME; AGE
1910 INPUT "NAME: "; NAME
1920 INPUT "AGE: "; AGE
1930 PRINT "NAME AND AGE: " NAME; AGE
1940 INPUT "NAME: "; NAME
1950 INPUT "AGE: "; AGE
1960 PRINT "NAME AND AGE: " NAME; AGE
1970 INPUT "NAME: "; NAME
1980 INPUT "AGE: "; AGE
1990 PRINT "NAME AND AGE: " NAME; AGE

```

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

[illegible]

We can sort the information into order. Here we've arranged the stock list from the previous page into ascending order of stock code.

FILE CREATION

When an operator creates a file it is essential to guard against the inadvertent pressing of the Return key before hitting a data key. The subroutine at line 236 prevents this on `ATTNPUT` statements. The option entitled `SHORT COLUMN` employs a simple bubble sort and is rather slow for long records but the extra statements for the faster merge sort would encroach on the 6K memory capacity still further. It will be appreciated that sorting records into some order is not a simple case of sorting a column. When a sweep is necessary, every data item in the record must take part in the sweep which complicates the process. Owners of 6K models can change the bubble to a merge sort if they find the `sleepsort` irritating.

The option **RETRIEVE RECORDS** allows an operator to selectively display all records which fall into required category. Thus if one of the column headings is "AGE", we may ask of a display of all personnel of age 23. If, by chance, nobody in the file is of that age the message "RECORD NOT ON FILE" is given.

Files are stored on a 'DATA' tape and the relevant patching routines to compensate for the early ROM bugs are included in the program. It is possible to obtain further copies of a data tape by first loading in the tape by the option 'LOAD FILE' and then using 'SAVE FILE' with a blank tape. Because file creation is error prone provides assistance for modifying at the end of each record and, again when the file is complete by using the option 'MODIFY FILE'.

VIC EDITOR

The problem with processing text on the VIC-20 is that the screen is rather small. No problem. Just make it bigger!

The VIC-20 is a nice little computer: colour, sound and almost high resolution graphics at a reasonable cost. Only one problem, right? The display.

It's when you try to cram a decent amount of information into a 22 by 23 screen that you wish there was some way of easily getting a larger screen. Well, there is. In fact, you can alter the configuration of the screen by simple POKEs to give any practical size up to 26 columns and 32 rows.

The screen is controlled by registers in the VIC chip. There are 16 registers which also control the colour, sound and games capabilities of the VIC.

If we call the 'VIC' register (at 36864) 'VIC' — a tradition begun by ones older and wiser than myself — then the registers that we are interested in are VIC, VIC+1, VIC+2 and VIC+3. These registers control the screen's shape, size, location on the TV screen and the location of the screen RAM in memory.

Let's make some definitions:

VIC = 36864
 HP = horizontal screen position on the TV screen (9 to 33)
 VP = vertical screen position (10 to 43)
 VC = number of video columns (1 to 33)
 VR = number of video rows (1 to 27)

The figures in brackets are the practical limits for each value.

Choose some values and try them with:

```
POKE VIC,HP
POKE VIC+1,VP
POKE VIC+2,VC AND 155 OR VC
POKE VIC+3,VR AND 155 OR VR+1
```

Now try a few other values until you get the hang of what's going on.

If you experiment a little more, you may discover that:

— the cursor is totally bedevilled with VC not equal to 32
 — only the first 306 screen locations can be printed to and

```
100 VIC=36864
110 VC=26:REM ** COLS
120 VR=32:REM ** ROWS
130 DEF FPA(I)=1-128*(I-1)/32:REM ** INV CURS FOR CURSOR
140 DEF FPR(I)=1-128*(I-1)/32
150 POKE VIC,VR:REM ** SPDS OF SCREEN
160 POKE VIC+1,16:REM ** THOS OF SCREEN
170 POKE VIC+2,FPR(VIC-4) AND 155 OR VC
180 POKE VIC+3,FPA(VIC-1) AND 155 OR VR*2
190 REM ** MOVE SCREEN DOWN BY 512 BYTES
200 POKE 36864,FPR(36864) AND NOT 128
210 REM ** DEFINE SCREEN AND COLOR RAM ADDRESSES
220 SC=7168:CO=37888
230 POKE 340,35:REM ** TELL VIC WHERE SCREEN IS
240 HC=SC+256:REM ** MACHINE CODE START
250 CL=HC:REM ** START OF CLEAR CODE
260 ST=SC+31:REM ** START OF SAVE CODE
270 LS=SC+64:REM ** START OF LOAD CODE
280 POKE 54,37:POKE 32,27:REM ** PROTECT SCREEN FROM
BASIC BY LOWERING TOP OF MEMORY
290 REM ** LOAD CODE FROM DATA STATEMENTS INTO HC+1
300 FOR I=0 TO 63:READ A:POKE HC+I,A:NEXT
310 STS HC:REM ** CLEAR SCREEN
320 P=SC:REM ** TOP LEFT
330 RV=VR:REM ** REVERSE CHARACTER FLAG
340 REM ** BLINK CURSOR UNTIL A KEY IS HIT
350 FOR P=FPA(FPR(FI)):FOR I=0 TO 40
360 IF PEEK(156)<0 THEN POKE P,FPR(FPR(FI)):GOTO 380
370 NEXT I:POKE P,FPR(FPR(FI)):FOR I=1 TO 40:
380 IF PEEK(156)<0 THEN FOR I=350
390 NEXT:GOTO 370
400 A="":GET A
410 REM ** CLEAR SCREEN
420 IF A="(CLR)" THEN STS CL:P=HC:GOTO 350
430 REM ** HOME CURSOR
440 IF A="(DHR)" THEN P=SC:GOTO 350
450 REM ** REVERSE ON
460 IF A="(REV)" THEN RV=1:GOTO 350
470 REM ** REVERSE OFF
480 IF A="(OFF)" THEN RV=0:GOTO 350
490 REM ** JUDGE KEYS
500 IF A="(C)" THEN P=P+VC
510 IF A="(C)" THEN P=P+VC
520 IF A="(R)" THEN P=P+1
530 IF A="(CL)" THEN P=P-1
540 REM ** RETURN
550 IF A="(RS)" THEN P=SC+VC*(FPA(FPR(FI)-SC)/VC)
560 REM ** DELETE
570 IF A="(DS)" THEN FOR P=32:P=P-1
```

only the first 512 can be POKEd to.
— 'clr' only works on the first 506 locations.

THE OUTER LIMITS

The fact that we only have 512 bytes of screen RAM available imposes a limit on the screen dimensions of 22 by 22, 26 by 19 or thereabouts. As we want a larger screen (26 by 32) we have to find the room from somewhere.

A useful source of extra RAM is the user's 3.5K. You lose a little (512 bytes) of your program space, but look what

you gain. To move the screen down by 512 bytes, we clear bit 7 of VIC+2:

```
POKE VIC+1,PEEK(VIC+1) AND NOT 128
```

We must protect our new screen by lowering the top-of-memory pointers so that BASIC can't get at it:

```
POKE 56,57:POKE 51,57
```

We can now define the screen and colour addresses for our new screen:

```
SCREEN=56 ** TOP-512  
COLOUR=56 ** 256-512
```

As the VIC's operating system doesn't know of our scheming,

we have to tell it where the new screen is — you wouldn't want to miss any error messages, would you? The statement:

```
POKE 248,56/256:REM ** SET UPPER 164 OF SCREEN ADDRESS
```

stores the MSB of the new address.

We now have our big screen, ideal for games and any applications where we want to display a lot of information. The practical limits on the screen's dimensions depend on your TV set, but typically they are 26 columns and 32 rows.

The following program is a simple text editor in BASIC and machine code that allows you to:

— write to the screen using the cursor keys, Home, clr, Return, delete, reverse on and reverse off.

— SAVE a screenful of tape by pressing 11.

— LOAD a screenful of tape by pressing 13. (Note that as the messages PRESS PLAY ON TAPE and PRESS PLAY AND RECORD ON TAPE are suppressed to avoid messing up the screen, you will have to realize what's going on. The cursor will vanish when VIC-ED is waiting for you to press the appropriate keys on the tape recorder.)

— get a printout of the screen on a VIC printer by pressing 15.

THREE IN ONE

The program uses three machine code routines: to save, to load and to clear the screen. The 'clear' routine is necessary because VIC can only clear the first 506 bytes of the screen. Using machine code to store the screen on tape means that we can save it in binary which is about 20 times faster than a normal data save.

The machine code is loaded into a 256 byte block of RAM just below the screen address by reading values from the data statements at the end of the program. The entry points defined in the program for the routines are: CL for 'clear', SV for 'save' and LD for 'load'.

```
570 IF P=0 THEN P=P+(26*32)
580 IF P>=56*(26*32) THEN P=P-(26*32)
590 IF AS="CL" OR AS="C" OR AS="CLR" OR AS="CL3"
    THEN 580
600 IF AS=CHR(22) OR AS=CHR(13) THEN 580
610 IF AS="(P)" THEN GOSUB 840:GOTO 358:REM ** P1
    SAVED TO TAPE
620 IF AS="(P)" THEN GOSUB 930:REM ** P1 LOADS FROM
    TAPE
630 IF AS="(P)" THEN GOSUB 710:GOTO 358:REM ** P5
    PRINTS SCREEN
640 REM ** GET SCREEN CODE FROM BASIC INPUT
650 A=ASC(AS):IF A>31 AND A<14 THEN GOTO 700
660 IF A=43 AND A=0 OR A=128 AND A=129 THEN A=A-64:
    GOTO 100
670 IF A=255 THEN A=0:GOTO 700
680 IF A=151 AND A=32 THEN A=A-128:GOTO 700
690 GOTO 358:REM ** UNPREDICTABLE
700 IF NV=1 THEN A=PEEK(AS):REM ** REVERSE FLAG ON
710 POKE P,A:IF P=0 THEN P=P+1
720 GOTO 358:REM ** NEXT CHARACTER
730 REM ** SCREEN COPY
740 AS=CHR(145):IF (PEEK(34449) AND 3)=3 THEN
    AS=CHR(17)
750 OPEN 4,4:IF AS=CHR(17) THEN CLOSE 4:OPEN 4,4,7
760 PRINT#4,AS:GOTO 800:FOR B=0 TO 255:AS=AS+B*255:
    FOR C=A TO A+255:
770 B=PEEK(32):IF B=128 THEN B=B-128:SV=128+B*255:
    IF B=32 THEN B=B+64:GOTO 820
780 IF (B=35) AND (C=43) THEN 820
790 IF (B=35) AND (C=128) THEN B=B-32:GOTO 820
800 IF (B=35) AND (C=13) THEN B=B-64
810 IF (B=35) AND (C=129) THEN B=B+64
820 B=B+CHR(10):IF NV=1 THEN B=B+CHR(145):SV=0
830 NEXT C:PRINT#4,B:NEXT C:PRINT#4,CLOSE 4:RETURN
840 POKE 848,16:PRINT "SAVE"[(17 C$)*255 SV:
    POKE 848,16:RETURN
850 POKE 848,16:PRINT "LOAD"[(15 C$)*255 LD:
    POKE 848,16:RETURN
860 REM ** DATA FOR MACHINE CODE ROUTINES
870 REM ** STARTING AT CL,SV AND LD
880 DATA 162,0,169,32,157,0,38,197,0,39
890 DATA 157,0,38,157,0,39,169,0,157,0
900 DATA 140,157,0,149,157,0,158,157,0
910 DATA 151,222,288,223,99,234,162,1
920 DATA 168,255,32,198,295
930 DATA 169,0,32,189,255,169,38,133,2,169,0,133,1
940 DATA 169,1,162,64,168,33,32,318,295,94,234
950 DATA 162,1,168,255,32,168,255,169,0,32,189
960 DATA 255,162,255,168,255,32,313,295,94,234
```

Listing 1. The program for increasing the size of the VIC-20 screen.

Thus, the statement **STX CL** will clear the screen.

When you have typed the program, first check very carefully for mistakes in your copying, then **SAVE** it *before* running — machine code has a nasty habit of killing itself.

When it runs correctly, you will find that for a few moments the screen is full of rubbish — this should vanish as soon as the program loads the machine code and clears the screen. If there are any glitches at this point, look for errors in the machine code.

The program will run on a 3.5K or 6.5K VIC with any practical screen size and will fit into the basic VIC if you remove some of the REMs. It won't run on a 1K machine. It's all to do with the shifting screen and RAM conflicting —

blame Commodore, not me! My advice is to remove the expansion RAM pack and 'downgrade' your VIC.

References: *The VIC Programmer's Reference Manual* — Commodore and *VIC Revealed* — Nick Humphreys

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

■ ■ VIC-ED BY P. HINTJENS ■ ■

EDITOR IS A SCREEN EDITOR FOR THE COMMODORE VIC-20 — IT GIVES THE USER A 20*32 SCREEN AND VARIOUS CURSOR FUNCTIONS.

VIC-ED ALLOWS THE USER TO SAVE SCREENS OF TEXT TO A TAPE IN 10*ARRY. AN AVERAGE SAVE TAKES ABOUT 40 SECS.

THE USER CAN ALSO MAKE A HARD COPY OF THE SCREEN IF A VICPRINTER IS AROUND.

THE FUNCTION KEYS USED ARE

■■■■ SAVE-TO-TAPE
■■■■ LOAD-FROM-TAPE
■■■■ PRINT THE SCREEN

THE PROGRAM DEMONSTRATES WHAT CAN BE DONE AND ISN'T AN END IN ITSELF. IT WILL LEAD ITSELF TO SOME USEFUL EXPANSION.



COMMODORE COMMUNICATIONS



This article describes a machine-code utility to let two PETs, a PET and VIC or two VICs exchange programs via a link between the user ports. This link involves no additional circuitry, merely a 10 core screened cable with the appropriate connector on each end (see Fig. 1).

The general procedure would be to load the machine code, via a BASIC loader, into both machines. Programs may then be exchanged at speeds in excess of disc loads by giving a SYS command on each machine:

```

VIC to VIC or PET      SYS 7600,0  VIC to PET
                        SYS 7600,0  VIC to PET
VIC to VIC or PET      SYS 7600,0  VIC to PET
                        SYS 7600,0  VIC to PET

```

As the machine code is loaded into high memory in each case, it needs protecting. If loaded in Hex using the monitor, it contains code to protect itself, but if loaded from a BASIC loader it will need protecting

from BASIC prior to running with:

```

PET to VIC or PET      ROM 11,0:ROM 12,0
                        ROM 13,0:ROM 14,0
                        CLR
VIC to VIC or PET      ROM 14,0:ROM 15,0
                        ROM 15,0:ROM 16,0
                        CLR

```

It is vital to start the receiver first, both out of consideration for the hardware, as both user ports could be configured as outputs, and for the need to catch the first byte of the transfer.

THE IDEA

The idea and requirement for these routines arose after the acquisition of a VIC computer without a cassette player. Therefore, this computer had to share a cassette player with a PET system and it always seemed that the cassette was on the wrong machine. The power should be turned off before plugging in or unplugging the cassette, or anything else for that matter, as the VIA lines are

If you thought that loading software from disc was fast you've never downloaded from another machine!

unbuffered and the VIA chip is easily blown by accidental shorts. Turning the power off also loses the programs. Using the Communicator program loaded into the VIC from the cassette and into the PET from disc, it is possible to exchange programs at will, very quickly. This is not network software but is a little way towards this very popular subject.

Listing 1 gives both the source and object code for the PET version of the program. As it is symbolic, it may be easily relocated by readers with an assembler. The program calls three ROM routines that would have to be changed for other than New ROMs (3.0): what I believe to be BASIC 4.0 versions are given in the listing. The VIC version is structurally identical, though the user port is mapped to a different address and in the 'B' side of the VIA, whereas in the PET the 'A' side and CB2 are used. This entails some changes to the handshaking of the data across the link. Listing 2 gives the VIC version assembled for a VIC with a 3K memory cartridge, though this again is easily relocated for other VIC configurations. Listing 3 gives a BASIC loader version for the VIC described above and Listing 4 gives the BASIC loader for a 4.0 ROM 8K PET.

HOW THE PROGRAM WORKS

The two machine code programs might look fairly lonesome but they are structured for simplicity. The main sections will be discussed in more detail below.

PROTECT This section alters the top of memory pointers. This could be done from BASIC to save a few bytes if memory is short.

START This section looks beyond the SYS for the 'S' or 'E' signifying Send or Receive. If neither are present it will jump to the BASIC Syntax Error routine. Depending on whether an 'S' or an 'E' follows the SYS, the VIA data lines are set up for input or output.

SETUP This section first moves the CHRGET pointer past the 'S' or 'E' and then sets up the ACR (Auxiliary Control Register) and PCR (Peripheral Control Register) with the correct bit pattern for using CB2 as Data Valid/Busy and CA1 (CB1 on VIC) as data strobes.

SETBASPOINT Takes a copy of the 'Start of BASIC' vector and saves it at BASPOINT. It then checks the DDRA (Data Direction Register) and branches to either LISTEN or TRANSMIT.

LISTEN The first part signals to the other CBM machine that it is ready to receive data. The next subroutine waits for the data strobe from the other CBM and then loads the data from the VIA into the BASIC test area pointed to by BASPOINT. Next, a check is made to see if the end of the program has been

received. (The end of a program is marked by three consecutive zeros.) If the end is found, the program jumps to the ENDFOUND routine. If it is not the end, the program loops back to the beginning of the LISTEN routine.

TRANSMIT This is similar to LISTEN but in reverse. It waits for the receiver to signal ready and then loads the data from the BASIC test area into the VIA. A strobe then indicates to the receiver that the data is valid. A similar endcheck to LISTEN is performed and if it has not reached the end, it loops back for the next character. On finding the end the program branches to SENDEND.

ENDFOUND This section realigns the pointers so that BASIC knows the length of the transmitted program. A similar jump to CLR completes the LISTEN routine.

The rest of the program consists of the various subroutines called by the main program above.

USREADY Sends a pulse on the CB2 line to the CA1/CB1 input of the other machine, to signal Busy Clear or Data Valid.

THEMREADY Waits for a pulse on CA1/CB1 as above. **INITXT** Increments the CHRGET pointer and loads the next character from the BASIC test.

INCPTR Increments the BASPOINT pointer. **ENDCHECK** This routine looks back over the received characters to check for three zeros and sets the end flag at 500 accordingly.

DECPTR Decrements the pointer used by ENDCHECK to look back through the BASIC test.

CHECKEND Similar to ENDCHECK but the routine looks forward to trap the end condition before it is transmitted.

SENDEND Transmits three zeros.

TESTING THE PROGRAM

When using machine code, all the protection built into BASIC is lost and it is easy to set the computer into a condition where it is stuck in a loop, 'crashed' in the vernacular. In order to be able to interrupt the program

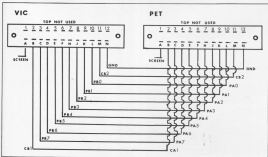


Fig. 1. A detailed illustration of the 10-core screened cable connection necessary to link a VIC-20 to a PET.


```

100 0000 1, 0
101 0001 10, 70, 0
102 0002 00
103 0003 1, 00
104 0004 0
105 0005 0000+000
106 0006 0000+0000
107 0007 0000+0000
108 0008 0000+0000
109 0009 0000+0000
110 0010 0000+0000
111 0011 0000+0000
112 0012 0000+0000
113 0013 0000+0000
114 0014 0000+0000
115 0015 0000+0000
116 0016 0000+0000
117 0017 0000+0000
118 0018 0000+0000
119 0019 0000+0000
120 0020 0000+0000
121 0021 0000+0000
122 0022 0000+0000
123 0023 0000+0000
124 0024 0000+0000
125 0025 0000+0000
126 0026 0000+0000
127 0027 0000+0000
128 0028 0000+0000
129 0029 0000+0000
130 0030 0000+0000
131 0031 0000+0000
132 0032 0000+0000
133 0033 0000+0000
134 0034 0000+0000
135 0035 0000+0000
136 0036 0000+0000
137 0037 0000+0000
138 0038 0000+0000
139 0039 0000+0000
140 0040 0000+0000
141 0041 0000+0000
142 0042 0000+0000
143 0043 0000+0000
144 0044 0000+0000
145 0045 0000+0000
146 0046 0000+0000
147 0047 0000+0000
148 0048 0000+0000
149 0049 0000+0000
150 0050 0000+0000
151 0051 0000+0000
152 0052 0000+0000
153 0053 0000+0000
154 0054 0000+0000
155 0055 0000+0000
156 0056 0000+0000
157 0057 0000+0000
158 0058 0000+0000
159 0059 0000+0000
160 0060 0000+0000
161 0061 0000+0000
162 0062 0000+0000
163 0063 0000+0000
164 0064 0000+0000
165 0065 0000+0000
166 0066 0000+0000
167 0067 0000+0000
168 0068 0000+0000
169 0069 0000+0000
170 0070 0000+0000
171 0071 0000+0000
172 0072 0000+0000
173 0073 0000+0000
174 0074 0000+0000
175 0075 0000+0000
176 0076 0000+0000
177 0077 0000+0000
178 0078 0000+0000
179 0079 0000+0000
180 0080 0000+0000
181 0081 0000+0000
182 0082 0000+0000
183 0083 0000+0000
184 0084 0000+0000
185 0085 0000+0000
186 0086 0000+0000
187 0087 0000+0000
188 0088 0000+0000
189 0089 0000+0000
190 0090 0000+0000
191 0091 0000+0000
192 0092 0000+0000
193 0093 0000+0000
194 0094 0000+0000
195 0095 0000+0000
196 0096 0000+0000
197 0097 0000+0000
198 0098 0000+0000
199 0099 0000+0000
200 0100 0000+0000

```

Listing 1. BASIC loader for the VIC-20.

```

100 0000 1, 0
101 0001 10, 70, 0
102 0002 00
103 0003 1, 00
104 0004 0
105 0005 0000+000
106 0006 0000+0000
107 0007 0000+0000
108 0008 0000+0000
109 0009 0000+0000
110 0010 0000+0000
111 0011 0000+0000
112 0012 0000+0000
113 0013 0000+0000
114 0014 0000+0000
115 0015 0000+0000
116 0016 0000+0000
117 0017 0000+0000
118 0018 0000+0000
119 0019 0000+0000
120 0020 0000+0000
121 0021 0000+0000
122 0022 0000+0000
123 0023 0000+0000
124 0024 0000+0000
125 0025 0000+0000
126 0026 0000+0000
127 0027 0000+0000
128 0028 0000+0000
129 0029 0000+0000
130 0030 0000+0000
131 0031 0000+0000
132 0032 0000+0000
133 0033 0000+0000
134 0034 0000+0000
135 0035 0000+0000
136 0036 0000+0000
137 0037 0000+0000
138 0038 0000+0000
139 0039 0000+0000
140 0040 0000+0000
141 0041 0000+0000
142 0042 0000+0000
143 0043 0000+0000
144 0044 0000+0000
145 0045 0000+0000
146 0046 0000+0000
147 0047 0000+0000
148 0048 0000+0000
149 0049 0000+0000
150 0050 0000+0000
151 0051 0000+0000
152 0052 0000+0000
153 0053 0000+0000
154 0054 0000+0000
155 0055 0000+0000
156 0056 0000+0000
157 0057 0000+0000
158 0058 0000+0000
159 0059 0000+0000
160 0060 0000+0000
161 0061 0000+0000
162 0062 0000+0000
163 0063 0000+0000
164 0064 0000+0000
165 0065 0000+0000
166 0066 0000+0000
167 0067 0000+0000
168 0068 0000+0000
169 0069 0000+0000
170 0070 0000+0000
171 0071 0000+0000
172 0072 0000+0000
173 0073 0000+0000
174 0074 0000+0000
175 0075 0000+0000
176 0076 0000+0000
177 0077 0000+0000
178 0078 0000+0000
179 0079 0000+0000
180 0080 0000+0000
181 0081 0000+0000
182 0082 0000+0000
183 0083 0000+0000
184 0084 0000+0000
185 0085 0000+0000
186 0086 0000+0000
187 0087 0000+0000
188 0088 0000+0000
189 0089 0000+0000
190 0090 0000+0000
191 0091 0000+0000
192 0092 0000+0000
193 0093 0000+0000
194 0094 0000+0000
195 0095 0000+0000
196 0096 0000+0000
197 0097 0000+0000
198 0098 0000+0000
199 0099 0000+0000
200 0100 0000+0000

```

Listing 4. BASIC loader for a
CERN 2000 2000+4000.

ORIC-1, VIC-20

The
meanest
game of
space
invaders
you'll
ever
play!

FEATURES

- Three invader types
- 1000000 scoring border drops
- Bonus bonus every 1000000
- Multiple scoring border drops
- Highscore table
- Full sound effects and music

You must defend earth from the hordes of invading space invaders, using your swift-fingered hand. The more invaders you hit, the better they score - leading to a big explosion when you lose.

A great shoot-'em-up, all action arcade game, for the 1KB or 4KB ORIC-1 or unexpanded VIC-20.

Send check or P.O. to: 100/1000000 (software) post, always enclosed 1990 and memory size to:



100/1000000
SOFTWARE

FREEMONT, SWANSEA, SA1 4EE

THE INSTITUTION OF ANALYSTS & PROGRAMMERS



An association which enhances the status of its members, encourages their high standards, makes their careers and promotes their interests in the essential foundation of every profession.

The Institution of Analysts & Programmers is the leading association for those engaged in systems analysis or computer programming for Commerce, Industry or Public Services. Membership of the Institution is shown by the designatory letters Comp.I.A.P., F.I.A.P., M.I.A.P. and A.M.I.A.P. is widely recognised and respected. The Institution is the supervising authority for the Copyright Register whose protective legal service is available to all (members and non-members) who write original programs.

If your computer practice could make you eligible to join the Institution or if you wish to secure your right to register through the Copyright Register write or telephone

01-898 2385

The General Secretary

The Institution Of Analysts & Programmers
87E, ROUTE, TANGIER ROAD, BIRMINGHAM, WEST MID

ADDRESS BOOK



For a change from games, try this useful and instructive data base program on your VIC-20. Using the excellent screen editor, you can enter, alter, delete, store on

tape, list and print entries in an address book. The entries are automatically stored in alphabetical order by name and each entry can be up to 88 characters long (four screen lines).

Find out how to address the problem of creating a useful and instructive data base program.

To adapt the size of the address book to your memory size, alter the variable MAX at the start of the program. On the unexpanded VIC-20, set MAX = 25; with a 3K expansion, set MAX = 50 and with a 16K expansion set MAX = 250. If you are using entries that are all quite short, you can increase these limits accordingly.

When you run the address book program, you will appear to be in normal operating mode; the cursor will flash and all the editing keys will work. However, the program is in control and only the following commands will be understood:

Entry

The 'Y' tells the program to enter the entry into the address book. The entry must be in the format: (J.A. SMITH) . . . where the name has exactly four initials and is followed by a space. If the name only has two initials, spaces should be entered. The entry is ordered as SMITHJ.A. . . . ie by surname and then initials. The data you put after the name is not used in the ordering.

@entry

Using the same principles of surname and initials for ordering, the program will search for the name and address entered and substitute the new data for the data that was previously attached to the name.

*entry

Again using the same ordering principles, this command will delete the entry from the address book.

END

Exits the program and reverts the normal operation of the Stop key.

HELP

Lists these commands on the screen.

LIST (start)

Lists the address book entries starting at the first or (start). After each entry is listed, the program waits for you to press a

key; if you press Stop the list will stop, otherwise the next entry will be listed. (start) can be any number of significant letters and the same surname/initials order applies. Eg LIST SM will list from the first surname starting with 'SM'.

LOAD (name)

Loads the address book called (name) off tape.

SAVE (name)

Saves the address book to tape and calls it (name).

MERGE (name)

Finds the address book called (name) on tape and merges the entries in it with the address book in memory.

PRINT

Prints the address book on the printer. The format of the printed entries depends on the special control characters used. (See later.)

FREE

This command will tell you how many entries existed there are and the amount of free memory available.

In all the above commands, the parts in brackets are optional. When giving a name for a LOAD, SAVE or MERGE, no quote marks are necessary. To use a command, simply type in the command and press Return. All normal facilities for editing, moving the cursor over an old command and re-entering it, etc are available.

Some examples of use are:

/J.A.SMITH /TEL 01 300 0323

(return)

@J.A.SMITH /TEL 01 300 0323

(return)

'J.A.SMITH /TEL 01 300 0323

(return)

..... enters the name and telephone number in the address book. The name must be followed by a space.

..... searches for the entry starting with J.A.SMITH and changes the telephone number. If there are two names with the same initials and surname, the first is acted on.

..... will delete the entry from the address book. The command must have exactly the

LOAD (return)

MERGE FAMILY (return)

PRINT (return)

same entry as that in the address book. In *J.A.SMITH (return) will have no effect here.

..... will find the first address book on tape, whatever it is called and load that into memory.

..... searches for the address book FAMILY and merges the entries in it with the ones in memory. It is always faster to merge a small address book into a large one than vice-versa.

Outputs the address book to the printer.

An obvious application for the program is to store names and addresses so that you can print address labels quickly. To do this, you need to be able to print a (return) after every line of an entry and perhaps a (comma) in the entry. To let you do this, the program will interpret the character 'Y' when it occurs inside an entry as (comma)-(return) and 'A' as (comma). So, to have the entry: J.A. SMITH, 1, ELM TREE AVENUE, BIRMINGHAM B1 A 2BD printed out in a normal address

format, it should be entered as:

/J.A. SMITH /1, ELM TREE AVENUE BIRMINGHAM B1 A 2BD(return). This will result in:

J.A. SMITH
1, ELM TREE AVENUE,
BIRMINGHAM B1A 2BD
being printed on the printer.

To provide more compatibility with VIC's normal operation, the Stop key is disabled by the command POKE 768, 194 in line 20. Then, when LISTING,

the Stop key can be used as normal to stop the listing. The key is reactivated by POKE 768, 191 in line 110.

Error messages produced by the program are:

SYNTAX ERROR — the command you used was not recognised by the program.
BOOK FULL — there are no entries entered.
NAME NOT FOUND — you specified a name for deletion or deletion that was not in the address book.

```

100 PRINT "LOADING ADDRESS BOOK. ...."
110 PROCEDURE
120 POKE 768,194 REM ** KILL 'STOP' KEY
130 GOTO 1000
140 OPEN "A:"
150 GET# 1,0
160 GET# 1,0
170 GET# 1,0
180 GET# 1,0
190 GET# 1,0
200 GET# 1,0
210 GET# 1,0
220 GET# 1,0
230 GET# 1,0
240 GET# 1,0
250 GET# 1,0
260 GET# 1,0
270 GET# 1,0
280 GET# 1,0
290 GET# 1,0
300 GET# 1,0
310 GET# 1,0
320 GET# 1,0
330 GET# 1,0
340 GET# 1,0
350 GET# 1,0
360 GET# 1,0
370 GET# 1,0
380 GET# 1,0
390 GET# 1,0
400 GET# 1,0
410 GET# 1,0
420 GET# 1,0
430 GET# 1,0
440 GET# 1,0
450 GET# 1,0
460 GET# 1,0
470 GET# 1,0
480 GET# 1,0
490 GET# 1,0
500 GET# 1,0
510 GET# 1,0
520 GET# 1,0
530 GET# 1,0
540 GET# 1,0
550 GET# 1,0
560 GET# 1,0
570 GET# 1,0
580 GET# 1,0
590 GET# 1,0
600 GET# 1,0
610 GET# 1,0
620 GET# 1,0
630 GET# 1,0
640 GET# 1,0
650 GET# 1,0
660 GET# 1,0
670 GET# 1,0
680 GET# 1,0
690 GET# 1,0
700 GET# 1,0
710 GET# 1,0
720 GET# 1,0
730 GET# 1,0
740 GET# 1,0
750 GET# 1,0
760 GET# 1,0
770 GET# 1,0
780 GET# 1,0
790 GET# 1,0
800 GET# 1,0
810 GET# 1,0
820 GET# 1,0
830 GET# 1,0
840 GET# 1,0
850 GET# 1,0
860 GET# 1,0
870 GET# 1,0
880 GET# 1,0
890 GET# 1,0
900 GET# 1,0
910 GET# 1,0
920 GET# 1,0
930 GET# 1,0
940 GET# 1,0
950 GET# 1,0
960 GET# 1,0
970 GET# 1,0
980 GET# 1,0
990 GET# 1,0

```

```

1000 PRINT "ADDRESS BOOK FULL"
1010 GOTO 100
1020 PRINT "NAME NOT FOUND"
1030 GOTO 100
1040 PRINT "SYNTAX ERROR"
1050 GOTO 100
1060 PRINT "BOOK FULL"
1070 GOTO 100
1080 PRINT "NAME NOT FOUND"
1090 GOTO 100
1100 POKE 768,191
1110 GOTO 100
1120 PRINT "LOADING ADDRESS BOOK. ...."
1130 GOTO 100
1140 PRINT "ADDRESS BOOK FULL"
1150 GOTO 100
1160 PRINT "NAME NOT FOUND"
1170 GOTO 100
1180 PRINT "SYNTAX ERROR"
1190 GOTO 100
1200 PRINT "BOOK FULL"
1210 GOTO 100
1220 PRINT "NAME NOT FOUND"
1230 GOTO 100
1240 PRINT "SYNTAX ERROR"
1250 GOTO 100
1260 PRINT "BOOK FULL"
1270 GOTO 100
1280 PRINT "NAME NOT FOUND"
1290 GOTO 100
1300 PRINT "SYNTAX ERROR"
1310 GOTO 100
1320 PRINT "BOOK FULL"
1330 GOTO 100
1340 PRINT "NAME NOT FOUND"
1350 GOTO 100
1360 PRINT "SYNTAX ERROR"
1370 GOTO 100
1380 PRINT "BOOK FULL"
1390 GOTO 100
1400 PRINT "NAME NOT FOUND"
1410 GOTO 100
1420 PRINT "SYNTAX ERROR"
1430 GOTO 100
1440 PRINT "BOOK FULL"
1450 GOTO 100
1460 PRINT "NAME NOT FOUND"
1470 GOTO 100
1480 PRINT "SYNTAX ERROR"
1490 GOTO 100
1500 PRINT "BOOK FULL"
1510 GOTO 100
1520 PRINT "NAME NOT FOUND"
1530 GOTO 100
1540 PRINT "SYNTAX ERROR"
1550 GOTO 100
1560 PRINT "BOOK FULL"
1570 GOTO 100
1580 PRINT "NAME NOT FOUND"
1590 GOTO 100
1600 PRINT "SYNTAX ERROR"
1610 GOTO 100
1620 PRINT "BOOK FULL"
1630 GOTO 100
1640 PRINT "NAME NOT FOUND"
1650 GOTO 100
1660 PRINT "SYNTAX ERROR"
1670 GOTO 100
1680 PRINT "BOOK FULL"
1690 GOTO 100
1700 PRINT "NAME NOT FOUND"
1710 GOTO 100
1720 PRINT "SYNTAX ERROR"
1730 GOTO 100
1740 PRINT "BOOK FULL"
1750 GOTO 100
1760 PRINT "NAME NOT FOUND"
1770 GOTO 100
1780 PRINT "SYNTAX ERROR"
1790 GOTO 100
1800 PRINT "BOOK FULL"
1810 GOTO 100
1820 PRINT "NAME NOT FOUND"
1830 GOTO 100
1840 PRINT "SYNTAX ERROR"
1850 GOTO 100
1860 PRINT "BOOK FULL"
1870 GOTO 100
1880 PRINT "NAME NOT FOUND"
1890 GOTO 100
1900 PRINT "SYNTAX ERROR"
1910 GOTO 100
1920 PRINT "BOOK FULL"
1930 GOTO 100
1940 PRINT "NAME NOT FOUND"
1950 GOTO 100
1960 PRINT "SYNTAX ERROR"
1970 GOTO 100
1980 PRINT "BOOK FULL"
1990 GOTO 100

```

```

000 RETURN
001 REM ## OUT (NAME) AFTER EXTENDS CL. LOAD (NAME)
002 GOTO 1
003 FOR J=0 TO LENDS:
004 IF BEEPING(1,1) THEN THEN NEXT RETURN
005 GOTO 1
006 REM ## BEEP
007 REM ## BEEP
008 REM ## BEEP
009 REM ## BEEP
010 REM ## BEEP
011 REM ## BEEP
012 REM ## BEEP
013 REM ## BEEP
014 REM ## BEEP
015 REM ## BEEP
016 REM ## BEEP
017 REM ## BEEP
018 REM ## BEEP
019 REM ## BEEP
020 REM ## BEEP
021 REM ## BEEP
022 REM ## BEEP
023 REM ## BEEP
024 REM ## BEEP
025 REM ## BEEP
026 REM ## BEEP
027 REM ## BEEP
028 REM ## BEEP
029 REM ## BEEP
030 REM ## BEEP
031 REM ## BEEP
032 REM ## BEEP
033 REM ## BEEP
034 REM ## BEEP
035 REM ## BEEP
036 REM ## BEEP
037 REM ## BEEP
038 REM ## BEEP
039 REM ## BEEP
040 REM ## BEEP
041 REM ## BEEP
042 REM ## BEEP
043 REM ## BEEP
044 REM ## BEEP
045 REM ## BEEP
046 REM ## BEEP
047 REM ## BEEP
048 REM ## BEEP
049 REM ## BEEP
050 REM ## BEEP
051 REM ## BEEP
052 REM ## BEEP
053 REM ## BEEP
054 REM ## BEEP
055 REM ## BEEP
056 REM ## BEEP
057 REM ## BEEP
058 REM ## BEEP
059 REM ## BEEP
060 REM ## BEEP
061 REM ## BEEP
062 REM ## BEEP
063 REM ## BEEP
064 REM ## BEEP
065 REM ## BEEP
066 REM ## BEEP
067 REM ## BEEP
068 REM ## BEEP
069 REM ## BEEP
070 REM ## BEEP
071 REM ## BEEP
072 REM ## BEEP
073 REM ## BEEP
074 REM ## BEEP
075 REM ## BEEP
076 REM ## BEEP
077 REM ## BEEP
078 REM ## BEEP
079 REM ## BEEP
080 REM ## BEEP
081 REM ## BEEP
082 REM ## BEEP
083 REM ## BEEP
084 REM ## BEEP
085 REM ## BEEP
086 REM ## BEEP
087 REM ## BEEP
088 REM ## BEEP
089 REM ## BEEP
090 REM ## BEEP
091 REM ## BEEP
092 REM ## BEEP
093 REM ## BEEP
094 REM ## BEEP
095 REM ## BEEP
096 REM ## BEEP
097 REM ## BEEP
098 REM ## BEEP
099 REM ## BEEP
100 REM ## BEEP

```

```

000 GOTO 100
001 REM ## LOAD
002 REM ## LOAD
003 REM ## LOAD
004 REM ## LOAD
005 REM ## LOAD
006 REM ## LOAD
007 REM ## LOAD
008 REM ## LOAD
009 REM ## LOAD
010 REM ## LOAD
011 REM ## LOAD
012 REM ## LOAD
013 REM ## LOAD
014 REM ## LOAD
015 REM ## LOAD
016 REM ## LOAD
017 REM ## LOAD
018 REM ## LOAD
019 REM ## LOAD
020 REM ## LOAD
021 REM ## LOAD
022 REM ## LOAD
023 REM ## LOAD
024 REM ## LOAD
025 REM ## LOAD
026 REM ## LOAD
027 REM ## LOAD
028 REM ## LOAD
029 REM ## LOAD
030 REM ## LOAD
031 REM ## LOAD
032 REM ## LOAD
033 REM ## LOAD
034 REM ## LOAD
035 REM ## LOAD
036 REM ## LOAD
037 REM ## LOAD
038 REM ## LOAD
039 REM ## LOAD
040 REM ## LOAD
041 REM ## LOAD
042 REM ## LOAD
043 REM ## LOAD
044 REM ## LOAD
045 REM ## LOAD
046 REM ## LOAD
047 REM ## LOAD
048 REM ## LOAD
049 REM ## LOAD
050 REM ## LOAD
051 REM ## LOAD
052 REM ## LOAD
053 REM ## LOAD
054 REM ## LOAD
055 REM ## LOAD
056 REM ## LOAD
057 REM ## LOAD
058 REM ## LOAD
059 REM ## LOAD
060 REM ## LOAD
061 REM ## LOAD
062 REM ## LOAD
063 REM ## LOAD
064 REM ## LOAD
065 REM ## LOAD
066 REM ## LOAD
067 REM ## LOAD
068 REM ## LOAD
069 REM ## LOAD
070 REM ## LOAD
071 REM ## LOAD
072 REM ## LOAD
073 REM ## LOAD
074 REM ## LOAD
075 REM ## LOAD
076 REM ## LOAD
077 REM ## LOAD
078 REM ## LOAD
079 REM ## LOAD
080 REM ## LOAD
081 REM ## LOAD
082 REM ## LOAD
083 REM ## LOAD
084 REM ## LOAD
085 REM ## LOAD
086 REM ## LOAD
087 REM ## LOAD
088 REM ## LOAD
089 REM ## LOAD
090 REM ## LOAD
091 REM ## LOAD
092 REM ## LOAD
093 REM ## LOAD
094 REM ## LOAD
095 REM ## LOAD
096 REM ## LOAD
097 REM ## LOAD
098 REM ## LOAD
099 REM ## LOAD
100 REM ## LOAD

```

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

Eccleston Electronics announce 25% off all this VIC 20 software whilst stocks last

VIC 20		CASSETTES		VIC 20	
001	001	001	001	001	001
002	002	002	002	002	002
003	003	003	003	003	003
004	004	004	004	004	004
005	005	005	005	005	005
006	006	006	006	006	006
007	007	007	007	007	007
008	008	008	008	008	008
009	009	009	009	009	009
010	010	010	010	010	010
011	011	011	011	011	011
012	012	012	012	012	012
013	013	013	013	013	013
014	014	014	014	014	014
015	015	015	015	015	015
016	016	016	016	016	016
017	017	017	017	017	017
018	018	018	018	018	018
019	019	019	019	019	019
020	020	020	020	020	020
021	021	021	021	021	021
022	022	022	022	022	022
023	023	023	023	023	023
024	024	024	024	024	024
025	025	025	025	025	025
026	026	026	026	026	026
027	027	027	027	027	027
028	028	028	028	028	028
029	029	029	029	029	029
030	030	030	030	030	030
031	031	031	031	031	031
032	032	032	032	032	032
033	033	033	033	033	033
034	034	034	034	034	034
035	035	035	035	035	035
036	036	036	036	036	036
037	037	037	037	037	037
038	038	038	038	038	038
039	039	039	039	039	039
040	040	040	040	040	040
041	041	041	041	041	041
042	042	042	042	042	042
043	043	043	043	043	043
044	044	044	044	044	044
045	045	045	045	045	045
046	046	046	046	046	046
047	047	047	047	047	047
048	048	048	048	048	048
049	049	049	049	049	049
050	050	050	050	050	050
051	051	051	051	051	051
052	052	052	052	052	052
053	053	053	053	053	053
054	054	054	054	054	054
055	055	055	055	055	055
056	056	056	056	056	056
057	057	057	057	057	057
058	058	058	058	058	058
059	059	059	059	059	059
060	060	060	060	060	060
061	061	061	061	061	061
062	062	062	062	062	062
063	063	063	063	063	063
064	064	064	064	064	064
065	065	065	065	065	065
066	066	066	066	066	066
067	067	067	067	067	067
068	068	068	068	068	068
069	069	069	069	069	069
070	070	070	070	070	070
071	071	071	071	071	071
072	072	072	072	072	072
073	073	073	073	073	073
074	074	074	074	074	074
075	075	075	075	075	075
076	076	076	076	076	076
077	077	077	077	077	077
078	078	078	078	078	078
079	079	079	079	079	079
080	080	080	080	080	080
081	081	081	081	081	081
082	082	082	082	082	082
083	083	083	083	083	083
084	084	084	084	084	084
085	085	085	085	085	085
086	086	086	086	086	086
087	087	087	087	087	087
088	088	088	088	088	088
089	089	089	089	089	089
090	090	090	090	090	090
091	091	091	091	091	091
092	092	092	092	092	092
093	093	093	093	093	093
094	094	094	094	094	094
095	095	095	095	095	095
096	096	096	096	096	096
097	097	097	097	097	097
098	098	098	098	098	098
099	099	099	099	099	099
100	100	100	100	100	100

Commodore 64 and Cassette Deck in stock. Ring us for our competitive prices



Announcing Speeder Software for the Commodore 64:
Machine Code Monitor — Tape Programmer's Toolkit —
Get and Mouse Game

RING US TODAY ON 021-336-6336/1326
BANCLYCARD & ACCESS ACCEPTED

ECCLESTON ELECTRONICS
Eccleston & Hart Limited.
8 Legge Lane, BIRMINGHAM B1 3LG.

MULTICOLUMN RECORDS

PRIMARY OPTIONS

CREATE NEW FILE	1
SEARCH FOR RECORD	2
FILE MANIPULATION	3
COLUMN SEARCH	4
SAVE FILE	5
LOAD FILE	6
EXIT PROGRAM	7

DESIGNER'S INTENTIONS

This issue contains a program (of mine) called 'Multipurpose Records'. Each record was limited in length to the width of one screen line, allowing a practical maximum of four columns. It would have been easy to replace this arrangement with a system allowing one screen page per record but the advantage of more space would have meant the loss of column comparison. Running the eyes down a column of data can provide valuable secondary information. In order to combine the advantages of both, a fresh approach was needed and this program is the result.

It allows a choice of serial or parallel (column or page) presentation of each record in a file. It was achieved by presenting the KEYFIELD as a stationary item but allowing each column to be reviewed into view from the left or right. Once the desired column appears in the window, the file can be sequenced up or down through the various records. Alternatively, a complete record can be displayed in full page

detail when required. Although there is no absolute restriction on the number of columns in each record, the file array has been dimensioned for a maximum of 10 columns in order to keep the memory cost down. While on the subject of memory it would be fair to mention that the program consumes an embarrassing amount of it. As it stands, it will not reside in an 8K PET. However, the screen messages lack the staccato shorthand often found in silicon vocal chords. The REM statements are sprinkled liberally and are equally verbose. It would be easy to get rid of the REMs, cut down on the textual material and slice out the disc-SAVE and disc-LOAD lines if unwanted. Extensive surgery of this kind could eventually slim the program down to 8K capacity although the residual memory would not hold many records.

Before keying in a program of this length, it would be wise to examine the facilities offered and judge whether or not the labour involved would be justified. This information could, of course, be gleaned by

A fully updated version of the Multipurpose program with greatly increased facilities. Ideal for small business or personal information.

study of the listing but in the interests of personal pride the following commercial break may be of interest.

PRIMARY OPTIONS

Create File enables a new file to be set up and the column headings and data entered. After each record is complete, the amount of memory left is displayed ... a necessary warning to deter those of a glibulous nature.

Save File can be used to store on cassette tape or disc with either drive 'D' or drive 'I' choice. **Load File** is the complementary function.

Search for Record allows any individual record to be accessed by asking for the key field or the record number.

Column Search can be used to examine the entire file and output the key field and record number of all records which have parameters equal to or within a given range of the search parameter. For example, if the file was on transistor specifications, it is possible to ask for, say, all transistors with a power-max rating less than 600 mW. Similarly, a file on employees may be examined for those under the age of 50 etc.

File Manipulation is a subset of the Primary Options and is examined later.

Exit Program although superficially a trivial option is necessary because of the RUN/STOP key is inhibited at the start of the program with POKE 144,49 (POKE 144,88 for BASIC 4 users). The program is therefore locked in an endless loop until the Exit Program option is executed and the RUN/STOP is released with POKE 144,46 (POKE 144,86 for BASIC 4 users).

FILE MANIPULATIONS

Twelve options are available (if we include the return to **Primary Options** as one of them).

View Columns allows the various columns to be revolved into the window. Key '<' will revolve left and key '>' right.

Next Record slides the next record into view (at least the key-filled and the chosen column). **Last Record**, as before but moving backwards.

Modify Data allows the data belonging to the column shown

to be changed.

Modify Column Heading will not be a frequent requirement but was included (after its original omission) because of criticism from a colleague; apparently he makes frequent mistakes when creating a new file!

Modify Key Field allows changes to the key field. This is the one that doesn't move when the columns are rotated.

Change File Name is handy if a file has been loaded from disc and modified in some

way.

Add Record allows additional records to be included in the file and the next highest record number allocated.

View Record displays the complete record with all its columns.

View File is a simple scrolling action of all records. Pressing the 'Space Bar' at any time during the scroll will return control to the window display made with the particular record captured.

Totalise Column causes the

```

REMEMBER THAT COLUMN 1 IS THE
ENTER FILE NUMBER IN NEXT RECORD
STATE OF COL 1
  
```

Taking the first stage in creating a new file.

```

YOU WANT: NAME DATE SEX DOB
ENTER 'END' IN EACH COLUMN TO
TERMINATE FILE
ENTER NAME
  
```

Once all the required column headings have been set up information can be entered as records.

```

NAME      AGE
DOB      SEX
1985-01-15  M
1985-02-20  F
1985-03-10  M
1985-04-05  F
1985-05-12  M
1985-06-18  F
1985-07-22  M
1985-08-25  F
1985-09-30  M
1985-10-01  F
  
```

Once the file is completed a secondary option menu is provided to allow you to manipulate the information.

```

NAME      AGE
ENTER AGE OF PERSON IN
DO YOU WANT ALL AGE 1-
EQUAL TO 10 10  KEY 'E'
LESS THAN 10 10  KEY 'L'
GREATER THAN 10 10  KEY 'G'
  
```

We've decided to search the file for any records where the recorded age is 10. The program also allows us to perform a variety of selective searches at this stage.

```

FOLLOWING TESTS IN FILE
AGE = 10
YES      RECORD NO. 1
1985-01-15
  
```

Simple! There was indeed a record listed in the file that matched our search parameters and we can now inspect the rest of the information stored for Fred.

[illegible]

An adjustment may be required in line 200 depending on memory size. For M6 PETs, no change is required. For an 8K system, M = 15359 should be changed to M = 7167. For a 32K PET, it should be M = 31743. This option is used when creating a file to warn how much memory is left after each entry.

The INPUT subroutine at line 3382 to 3710 is peculiar to the Connors PFT and can be replaced by simple INPUT for most other types.

KEYING IT IN

It is always a daunting task to key in a long program such as this. Some people just sit down and keep bashing the keys relentlessly until it is finished... a Herculean task. I would never have the courage for this because there would almost certainly be multiple mistakes. My plan (being a pessimistic, cautious type of person) would be based on the modular system. Enter about 10 or 20 lines at a time, stopping at some logical module-end and stick in a temporary STOP then RUN the program to that point to see the results. For example, enter lines 100 to 450 inclusive which covers the Primary Options' page and put STOP at lines 3080, 470, 590, 7350, 3860, 4450 and 5830. When this is RUN and the various options tried out, it is easy to check that the correct linkage from the ON... GOTO statement is established. Then, proceed to enter each primary option part separately and RUN before proceeding with the next one. One final warning - keep on loading each module on to tape (or disc) as each part is proved. In this way the tape will gradually grow and will act as an insurance policy if you do something daft during the current editing session.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 8 of this issue.

Il termine *il* precede un sostantivo maschile singolare, e si usa anche con i pronomi *lui* e *lei*.

THE GAMES YOU'VE BEEN DYING TO SEE!

For the Unexpanded VIC-20

TVIC 3 Skramble!	£7.95
TVIC 4 Terminal Invaders	£5.95
TVIC 5 Meteor Blaster	£5.95

This Month Only, Reduced from £7.95

TVIC 6 Gridder	for just £4.95
TVIC 7 Line Up 4/ Reversi	£7.95
TVIC 8 Get Lost! (3D Maze)	£5.95

Adventures for VIC-20 with 16K Expansion

TVIC 9 The Curse of the Werewolf	£9.95
TVIC 10 Rescue from Castle Dread	£9.95

Commodore 64 Software

TC641 Super Skramble	£9.95
----------------------	-------

For the Dragon 32

TDRAG 1 Line Up 4	£4.95
-------------------	-------

DEALER ENQUIRIES WELCOME

Machine-code programmers wanted! We will pay up to £1000 for good, original programs for any of the popular minis.

Demand our games at all good computer shops or buy mail order from:

TERMINAL SOFTWARE, DEPT. PST
26 Church Lane, Freshwick, Manchester M20 6AJ

MAIL ORDER PROTECTION SCHEME

If you order goods from Mail Order Advertisers in this magazine and pay by post in advance of delivery, this publication will consider you for compensation if the advertiser should become insolvent or bankrupt, provided:

1. You have not received the goods or had your money returned; and
2. You write to the publisher of this publication explaining the position not earlier than 28 days from the day you sent your order and not later than 2 months from that day.

Please do not wait until the last moment to inform us. When you write, we will tell you how to make your claim and what evidence of payment is required.

We guarantee to meet claims from readers made in accordance with the above procedure as soon as possible after the advertiser has been declared bankrupt or insolvent to a limit of £1,000 per annum for any one advertiser, or affected, and up to £5,000 p.a. in respect of all insolvent advertisers. Claims may be paid for higher amounts, or when the above procedures have not been complied with, at the discretion of this publication, but we do not guarantee to do so in view of the need to set some limit to this commitment and to learn quickly of reader's difficulties.

This guarantee covers only advance payment sent in direct response to an advertisement in this magazine (not, for example, payments made in response to catalogues, etc. received as a result of answering such advertisements).

CLASSIFIED ADVERTISEMENTS ARE EXCLUDED.

GIVE YOUR VIC20 & 64 IEEE PLUS RS232

VIC and 64 users

Would you like to be able to access any of these peripherals from your computer?

- 16 megabyte disks (Commodore 8040 drive)
- 1 megabyte disks (Commodore 8050 drive)
- 10 megabyte disks (Commodore 9090 hard disk)
- Printers including a wide range of inexpensive IEEE and RS232 matrix and quality printers
- IEEE instruments such as volt meters, plotters etc.

Now you are no longer limited by the VIC or the 64's serial bus. Simply by attaching INTERPOD you can vastly increase the power of your VIC-20 and when used with the new 64, INTERPOD

turns the computer into a really powerful system.

With INTERPOD the VIC and 64 become capable of running really professional quality software such as Word-processing, Accounting, Instrument control and many more.

INTERPOD will work with any software. No extra commands are required and INTERPOD does not affect your computer in any way.

Using INTERPOD is as easy as this:

Simply plug INTERPOD into the serial port of your computer, power up and you are ready to communicate with any number of parallel and serial IEEE devices and any RS232 printer.



INTERPOD

Oxford Computer Systems (Software) Ltd.
Hemington Road, Woodstock, Oxford OX2 1JH, England Tel: (0800) 842700

£125

UTILITIES

Subroutine Library — A library of BASIC subroutines.
Originally published in **Computing Today**, October 1983.

Toolkit Program — A simple toolkit program for the Commodore 64.

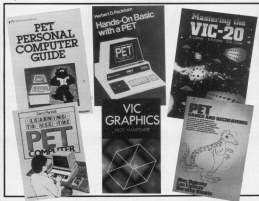
Tailoring VIC's Characters — Create your own characters on your VIC-20.
Originally published in **Computing Today**, February 1983.

Maxi-Mander — Bomb-proofing your software against unskilled fingers.
Originally published in **Computing Today**, July 1981.

VIC Blow Up — Find out how characters are made up and generate giant version on the VIC-20.
Originally published in **Computing Today**, January 1983.

Program Protection — Simple tips on how to protect your programs from being easily copied.
Originally published in **Computing Today**, June 1982.

Bibliography — A brief perusal of some of the multitude of books on Commodore computers.



we must re-assign as follows before using GOSUB:

```
PRINT "OK"
GOTO 1000
```

These pre-assignments constitute the calling sequence.

On return from the subroutine, it may be necessary to apply the reverse procedure:

REMARK

The calling sequence in this example is a case of 'parameter-passing'. It may be argued with some justification that changing variable names should be unnecessary. Why couldn't the subroutine be written with variable names to suit the demands of the calling program? This is not always possible. For example, the same subroutine may be called up to perform the process at different times on different variables. The subroutine may have been written within a collection of general purpose subroutines in which case there could have been no prior knowledge of the variable names employed in subsequent programs.

WHAT KIND?

This question is similar to the 'length of the piece of string' caper. It is sufficient to remark that anything purporting to be general purpose cannot be specific in aim. The best attitude to adopt in choosing the list is to consider the task as an upgrading of the operating system and the BASIC interpreter. A high level language itself (such as BASIC) is in reality nothing more than a complex conglomerate of subroutines. These are, of course, written in machine code but nevertheless, they are still subroutines. Many of the facilities offered are taken for granted and we only take notice of the defects or omissions. Anyone who can write a high level interpreter or compiler would certainly have my respect.

In spite of this, for reasons of ROM space, any high level language cannot cover more than a sprinkling of keywords.

A set of general purpose subroutines on tape or disc is equivalent to a language upgrading. Although no new keywords are added, the effect is the same. For example, most BASICs include SIN, COS, TAN and ATN but few offer the other inverse trig functions ASN (arc sine) and ACS (arc cosine). However, if GOSUB 5000 changes X into arc sine X then the BASIC is upgraded providing, of course, you have written such a subroutine and it is resident in memory. Naturally if your inclinations are such that arc sine X would only be used sparingly, if at all, then its inclusion would be foolish. General purpose subroutines are to some extent subjective to the individual.

Apart from augmenting the BASIC vocabulary, subroutines can be used to remedy defects in existing keywords. The INPUT statement in the PET has a truly diabolical feature. If the Return key is pressed (by accident) before the requested data has been keyed in, the program breaks out into COMMAND mode. A subroutine which replaces INPUT can easily be installed to remedy this effect.

Some subroutines can be very complex and others so simple as to be open to a charge of triviality. A simple subroutine, however, can still be a time and memory saver. For example, nothing could be more simple than a row of 16 across the screen in order to produce a line — but if many lines had to be drawn in the program, it would certainly justify a subroutine. On the other hand, it may be that a subroutine which requires an elaborate calling sequence each time might very well be counter productive. In such a case, it would be easier and more efficient to splice in the code each time it was used — an 'open loop' solution.

CHOICE OF VARIABLE NAMES...

It is good practice to choose variable names which suggest

the data item (insofar as it is possible with the two-character limit imposed by BASIC). When writing any form of general purpose subroutine it may be politic to ignore the rules in order to avoid clashes between the names. Unless care is taken, the subroutine may use working variables which are already used in the calling program with disastrous results. To act as a safeguard, a good plan is to use 'unusual' names in the subroutines in the belief that it would be too much of a coincidence to choose such names in the calling program. Using a high digit as the second character is also a good idea (thus M0 or K0).

...AND LINE NUMBERS

If subroutines are placed at the head of the program with low line numbers there is, in most BASICs, a speed advantage because the GOSUB search commences from the top downwards. However, it is 'tidier' to place them down the bottom end. Where subroutines of the general purpose variety are concerned there is another reason why they should be at the bottom with high line numbers, BASICs which include the ability to APPEND programs together (or 'Tools') normally require the APPENDED code to be at higher line numbers than the already resident code. Thus, it is possible while developing a program to become aware of the need for the subroutines, in which case they can be appended.

USING GENERAL PURPOSE SUBROUTINES

A key way and one to which I am addicted, is to list load in the list before beginning any program. If any are found to be useful then use them; any not used at the end can be erased.

After the program is more or less finished and tested, it may be possible to dispense with pre-assignments in the calling sequence by changing the

TOOLKIT PROGRAM

When programming your Commodore 64, you'll soon want features like function key assignment, search and others available only on an expensive toolkit expansion. Until now, because this program gives you a fairly comprehensive machine code toolkit that offers features including all the above and more.

Briefly, for the small price of 1K of memory and a few hours programming, you can:

- assign up to 32 characters to the eight function keys F1 to F8
- search a program for a string, printing out any 'bride'
- step through screen and border colour combinations at will
- stop the execution of any program so that you can inspect the screen, then continue running it as if nothing had happened
- have a bell ring whenever your typing nears the end of the line.

When in operation, the toolkit becomes part of the system I/O routine which is called 40 times every second to update the clock, see if the stop key is pressed and so on. When called, the toolkit can check whether a certain key is pressed and if so, do some action. For example, if some character string has been assigned to the key F1 and the toolkit sees that F1 is pressed, it will output that string.

THINGS TO KNOW

Before you can use the toolkit, you must enter and check the BASIC loader program. The machine code is contained in the data statements in lines 100

to 470. While entering the loader, which is quite large, you should make frequent backup copies so that if a jealous pet trips the wire, you don't have to start all over again. Check the program and have a copy saved before you run it.

When run, the program loads data into memory, checking all the time for errors in the data statements. Should it spot one, it will tell you the line number and abort the load. You should find and correct the error and then try again. There is also a check made on the overall sum of the data, so that an error not found by the checksum at the end of every data line may still be spotted. However, in this case you will not be told the line number.

When the toolkit is initialized (the loader does this for you but the way to do it is to type 'SYS 196'256 (return)') the message "' CBM 64 TOOLKIT (c) 1983 P. HINTIENS '" will be printed. The first time the toolkit is initialized, the function key assignments are cleared and the bell tab reset. However, after that it can be switched on and off and the function key assignments will not be wiped.

The first function provided is a 'bell tab'. This means that while you are typing a line, a bell will be rung when you reach the 75th character. The reason for this, apart from knowing where you are in that the CBM 64 (mine at least) has a ROM bug which crashes rather messily if you over-run the last screen line and try to delete your way back up. If you do this, the computer is lost and has to be switched off.

IN COMMAND

The second function provided is

Here is a fairly comprehensive machine toolkit program for use on a Commodore 64

the access to eight special commands using the Ctrl key in conjunction with one of the eight function keys (shifted and unshifted). These commands are:

F1 — FIND — Using the first program line as the search string, the program in memory is searched and any occurrences of the data are listed on the screen. After a line is listed, the toolkit waits for you to press a key. If you press Stop, the search is cancelled. If you press L, the toolkit starts listing from the point of the last find. If any other key is pressed, the search goes on. When the L option is chosen, you also press a key for the next line or Stop to stop. To use this facility, you enter the data to be looked for as the first program line (I keep line 0 for this purpose) and then press Ctrl and F1.

F3 — ASSIGN — This facility allows you to assign up to 32 characters to one of the eight function keys. Pressing Ctrl, Shift and F1 together will display a 'F' followed by a cursor. Enter the number (1 to 25) of the key and a space, followed by the command string. In some cases it can be useful to incorporate a carriage-return character into the command string — to do this, the '=' character should be entered. For example: (the computer's output is underlined):

☒ LIST THIS LINE = (return)

Now whenever the key F7 is pressed, 'LIST THIS LINE (return)' will be printed. Special provision is made so that if you are in quotes mode (when the function keys produce those strange characters) no command string will be output. ▶

F3 - PAUSE - It is often useful to be able to stop a program's execution for a time so that you can inspect the screen and then continue, for example during a list. Pressing Ctrl and F3 will do this; pressing them again will continue whatever was being done. While the toolkit is waiting for you to press the keys again, you may use the Stop key - a single push will do the same as the Ctrl-F3 combination; however, if you hold it down, a 'break' will be caused. Again this is useful during a list.

F4 - KILL - If you should need to disable the toolkit, use this command. The alternative is Stop-Restore, but this will reset the whole system, screen colours and all by way of a warm start. A message will tell you that all is well and the toolkit will no longer function. By installing it (STX 156 +256) it will come back to life, though any assigned command strings will be kept as they were.

F5 - TAB MOVE - It may be desirable to move the tab position along the line to suit a table or purpose; this command will move it one position to the right; after 20 it is moved back to 00.

F6 - TAB DISPLAY - To see where the tab position is, other than moving the cursor along until you hear a ring, use this command. A line of dots will be printed up to the tab position.

F7 - SCREEN COLOUR - Rather than continually POKEing some obscure location I know if's 55281 and you may know that, but don't spoil this, this command will step through the 16 available screen colours, starting at 0 (black) when it goes past 15.

F8 - BORDER COLOUR - This does the same for the border colour. The beauty of this command and Screen Colour is that you can set the colours while a program is running.

The toolkit program, in its machine code form, uses the top 1k of memory from BASIC, starting at \$9C00 up to \$9F00 -

a 256 byte chunk is used to store the function key assignments. The free 4k block from \$C000 to \$D000 was avoided, because every other machine code program will use it. Once you have a working copy of the toolkit, it would be, better and more efficient to make a saved copy of the actual machine code rather than the indirect BASIC loader. To do this, you tool the 64 into

thinking that there is a BASIC program starting at \$9C00 and ending at \$9F00 (156 +256 to 159 +256).

```
POKE 16,170:POKE 16,0
POKE 16,171:POKE 16,0
SAVE "TOOLKIT"
```

Then, to load the machine code directly into the same area of memory, use:

```
LOAD "TOOLKIT",1
POKE 16,170:POKE 16,170:POKE 16,170:POKE 16,170
```

```
100 GOTO 120,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,346,347,348,349,350,351,352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,373,374,375,376,377,378,379,380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,397,398,399,400,401,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,423,424,425,426,427,428,429,430,431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,448,449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,481,482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,499,500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521,522,523,524,525,526,527,528,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,549,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,574,575,576,577,578,579,580,581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597,598,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,618,619,620,621,622,623,624,625,626,627,628,629,630,631,632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,648,649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665,666,667,668,669,670,671,672,673,674,675,676,677,678,679,680,681,682,683,684,685,686,687,688,689,690,691,692,693,694,695,696,697,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721,722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,746,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,771,772,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,820,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,845,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,894,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,919,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,968,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,993,994,995,996,997,998,999,1000
```

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

TAILORING VIC'S CHARACTERS

Here is a utility program that allows you to create your own symbols on the VIC.

The VIC-20 is provided with an excellent set of graphics characters stored within its ROM. But like all the best of today's micro the VIC is not limited to just these 'off-the-peg' characters. The character generator on the VIC can be redefined, enabling it to produce a virtually limitless set of 'user-created' character shapes, like 'proper' Space Invaders, Lunar landers, tanks, Rockets(?) and even people.

Whilst the creation of user-defined characters is not especially difficult, the processes involved can be extremely tedious, usually requiring copious quantities of squared graph paper and furious thumbing of pocket-calculator buttons.

This program provides you with a high-resolution, on-screen character editor, together with automatic calculation of the POKE values of the new character for incorporating into other programs. Furthermore, a shape defined within the boundaries of just one character position is bound to be somewhat limited. The program is therefore arranged to work on a 3 by 2 character matrix, characters being formed within 1, 2, 3, or 4 character positions.

EIGHT TO GENERATE

But let us first quickly recap on how the VIC's character generator works. Each character that the VIC can PRINT can be considered as being made up of eight horizontal rows of eight pixels per row. Each row of pixels is defined in a separate address in memory, which contains the information as to which of the eight pixels in the row will be 'set', and which will be 'not set'.

These 'row-addresses' will contain a binary number with a

value between 00000000 and 11111111, ie between 0 and 255. The pattern of 1s and 0s that this binary number represents is directly related to the pattern of pixels 'set' and 'not set' in the row. This is illustrated in Fig. 1.

PIXELS SET	NUMBERS STORED IN ROW ADDRESSES	
	10 - BINARY	DECIMAL
	00011000	24
	00111000	56
	01111000	128
	00111000	56
	00011100	28
	01111100	120
	00111100	68
	00011000	24

Fig. 1. The correspondence between pixel sets and binary numbers stored in row addresses is illustrated.

Since each full character is made up of eight rows of pixels, the complete character shape can be defined by the numbers stored in eight consecutive row-addresses. The block of addresses used for holding the shape information for the whole character set is normally referred to as the 'character generator'. On the VIC-20 this is located in ROM starting at address 32768. Thus, the POKE code 0 (an 'M' symbol) uses the information in addresses 32768 to 32775; code 1 (an 'A' used) addresses 32776 to 32783, and so on.

However, there is essentially no difference between an address containing character shape information and any other address. All are eight bits wide, all will contain numbers between 0 and 255, so any of them could be used to represent one row of pixels. We cannot change the ROM character generator itself, but if we can tell the VIC to refer to a new

pointer to the start of the character generator. Normally it contains 240, which is used to point to address 32768. If we change the contents of 36869 we can redirect the VIC to a new character generator start address. For example, POKE 36869,255 will point to a character generator starting at address 7188, a RAM address.

This explanation of the VIC character generator is somewhat over-simplified, so please do accept my apologies now. It is nevertheless accurate as far as it goes — more detailed explanations can be found in the **VIC Programmer's Reference Guide**.

Having got the VIC to look somewhere else for its character generator all POKE statements to the screen will reference the new row-addresses.

That is

POKE (any screen address), 0
will use the information in the

first eight addresses of the new character generator. Strangely enough, the VIC will still regard character code 0 as an '@' symbol, code 1 as an 'A', etc., so the statement PRINT "0" is still perfectly valid even though the character that appears on the screen looks nothing like an '@'.

The VIC will merely assume that by '@' we mean the contents of the first eight addresses of the character generator, regardless of where in memory the character generator is located.

THE PROGRAM

The program is written in BASIC for the unexpanded VIC-20, and is, as previously stated, a 'character editor'.

The main screen display consists of a 16 by 16 'grid' of full size character positions at normal resolution. The grid represents the rows of pixels forming four full characters arranged in the formation:

0	1
2	3

A flashing cursor can be moved around the grid using the normal cursor control keys, and individual elements of the grid can be 'set' using function key F1 or 'unset' using function key F3. This setting or unsetting of elements at high resolution is reflected at normal resolution in an area immediately below the grid. So for every grid element which is set, a single pixel will appear at the bottom of the screen in exactly the same relative location.

Thus, as you move around the main display, turning elements on and off, four actions are mirrored in high resolution — showing you exactly how your finished character will look.

Included in the program are 'paint' and 'unpaint' routines. Paint, called by function key F2, will begin setting each pixel following the current cursor position until either you stop it by pressing any key, or it fills in the whole area remaining. Unpaint — function key F4 —

```

10 POKE 34,39:REM ** LOWER BANTON TO 'PROTECT' NEW
   CHARACTER GENERATOR
20 DATA 1,1,7,13,7,1,1,1:REM ** FIRST CHARACTER
30 DATA 88,188,8,388,204,220,234,250:REM ** SECOND
40 DATA 3,139,207,241,127,184,83,43:REM ** THIRD
50 DATA 199,134,34,42,282,171,244,288:REM ** FOURTH
60 DATA 200:REM ** END OF LIST POINTER
70 RESTORE
80 G=0
90 READ A:IF A=999 THEN GOTO 110
100 POKE 1288+G,A:G=G+1:GOTO 90
110 POKE 14889,255:REM ** POINT TO NEW CHARACTER
   GENERATOR AT 1288
120 PRINT "PROGRAM ...

```

Fig. 3. A program showing one method of incorporating character shape values.

```

3 REM ** INITIALISE NEW CHARACTER GENERATOR
10 POKE 34,39:FOR I=7168+32 TO 1488+39:POKE I,255:NEXT
20 FOR I=7168 TO 7168+40-1:POKE I,8:NEXT
30 PRINT "(CLR)":POKE 10489,255
40 REM ** MAIN DISPLAY
40 PRINT "[REY]";3 SPC;"7894321078543210"
50 PRINT "[REY]";3 SPC;"16 08"
60 FOR I=1 TO 3
70 FOR J=1 TO 8:PRINT "[REY]";1;"01234567";16 SPC;
   GOTO 100:NEXT J
80 PRINT "[REY]";5 SPC;"16 Gct";0:GOTO 100
90 REM ** INITIALISE MAIN VARIABLES
100 TL=7168:CL=8844:FF=71:CF=0:R=0:Y=1:MX=128:OL=188
110 REM ** PRINT NEW CHARACTER
120 PRINT "DOWN";15 C;"C";8:PA(CD)3 OL:80"
130 REM ** KEYBOARD INPUT
140 GET AS:IF AS="" THEN AP=0:GOTO 500:GOTO 110
150 AP=0
160 IF AS="(C)" THEN GOTO 330:GOTO 500:GOTO 110
170 IF AS="(C)" THEN GOTO 330:GOTO 500:GOTO 110
180 IF AS="(C)" THEN GOTO 330:GOTO 500:GOTO 110
190 IF AS="(C)" THEN GOTO 330:GOTO 500:GOTO 110
200 IF AS="(C)" THEN GOTO 330:GOTO 500:GOTO 110
210 IF AS="(C)" THEN GOTO 330:GOTO 500:GOTO 110
220 IF AS="(C)" THEN GOTO 330:GOTO 500:GOTO 110
230 IF AS="(C)" THEN GOTO 330:GOTO 500:GOTO 110
240 IF AS="(C)" THEN GOTO 330:GOTO 500:GOTO 110
250 GOTO 110
260 REM ** MOVE DOWN
270 IF T>15 THEN RETURN
280 T=T+1:AP=33
290 RETURN
300 REM ** MOVE UP
310 IF T<3 THEN RETURN
320 T=T-1:AP=33
330 RETURN
340 REM ** MOVE RIGHT
350 IF X<15 THEN RETURN
360 X=X+1:AP=0
370 RX=MX/16:IF MX<1 THEN RX=128
380 RETURN
390 REM ** MOVE LEFT
400 IF X<3 THEN RETURN
410 X=X-1:AP=3
420 RX=MX/16:IF MX<16 THEN RX=0
430 RETURN
440 REM ** SET PIXEL
450 CH=(INT(X/8)*2)+(INT(Y/8))
460 IF Y=8 THEN PY=Y-8:GOTO 480
470 PY=Y
480 CY=1288+(CH*8)+(PY-8)
490 POKE PP,8:POKE CY,8
500 POKE CY,PIXEL(CY) OR RX
510 RETURN
520 REM ** UNSET PIXEL
530 CH=(INT(X/8)*2)+(INT(Y/8))
540 IF T=8 THEN PY=Y-8:GOTO 470

```

```

500 PT=0
510 CV=160*(CV%5)+PT-1
520 POKE PP,160:POKE CV,PEEK(CV)-((PEEK(CV) AND MX)
530 RETURN
540 REM ** FLASHING CURSOR
550 PP=PP+PP:CF=CF+CF
560 BR=PEEK(PF):CR=PEEK(CF)
570 POKE PP,4:POKE CF,1
580 FOR I=1 TO DL-1ST
590 POKE PP,8:POKE CF,CR
600 FOR I=1 TO DL-1ST I
610 SP=0
620 RETURN
630 REM ** OUTPUT CHARACTER DATA
640 PRINT "[ROW][22 CO][REV][CONFIRM-ANY KEY][OFF]";
650 CF=0
660 CF=CF+1:IF CF>160 THEN PRINT "[ROW][22 CO][REV]
    (17 SPC)[OFF]";RETURN
670 GET CF:IF CF=" " THEN GOTO
680 PRINT "[CLS]"
690 FOR I=0 TO 1:PRINT "[REV]"*TAB(2):PEEK(7160+I);
    TAB(14):PEEK(7160+I+8):NEXT I:PRINT
700 FOR I=16 TO 23:PRINT "[REV]"*TAB(2):PEEK(7160+I);
    TAB(14):PEEK(7160+I+8):NEXT I:PRINT
710 PRINT "[ROW][4 CO][CR][17 CRLF]"
720 PRINT "[ROW][13 CO][CR][13 CRLF]"
730 PRINT "[5 CO][9 CR][14 CO][3 CRLF]"
740 PRINT "[CO][REV][TYPE 'C' TO CLEAR]";
750 GET CO:IF CO=" " THEN GOTO
760 PRINT "[CLS]"*POKE 56000,140:END
770 REM ** PAINT
780 DL=16
790 PRINT "[ROW][22 CO][REV][ANY KEY TO STOP][OFF]";
800 FOR TY=1 TO 1
810 FOR CX=1 TO 16
820 GOSUB 370:GOSUB 290:GOSUB 500
830 GET IS:IF IS=" " THEN GOTO
840 GOTO 870
850 NEXT CX
860 GOSUB 370:GOSUB 500
870 NEXT TY
880 DL=160
890 PRINT "[ROW][22 CO][REV][17 SPC][OFF]";
900 RETURN
910 REM ** UNPAINT
920 DL=16
930 PRINT "[ROW][22 CO][REV][ANY KEY TO STOP][OFF]";
940 FOR TY=1 TO 1
950 FOR CX=1 TO 16
960 GOSUB 440:GOSUB 290:GOSUB 500
970 GET IS:IF IS=" " THEN GOTO
980 GOTO 1000
990 NEXT CX
1000 GOSUB 370:GOSUB 500
1010 FOR CX=1 TO 16
1020 GOSUB 440:GOSUB 330:GOSUB 500
1030 GET IS:IF IS=" " THEN GOTO
1040 GOTO 1060
1050 NEXT CX
1060 GOSUB 370:GOSUB 500
1070 PRINT "[ROW][22 CO][REV][17 SPC][OFF]";
1080 RETURN

```

Listing 1. A program for character generation on the unexpanded VIC-20.

does the reverse, it it wipes out any pixel already set.

The high resolution character at the bottom of the screen is, in fact, a composite of the first four characters in a new character generator, starting at address 7160. These characters — character POKE codes 0, 1, 2, and 3 or, if you wish, PRINT symbols @, A, B, and C — are arranged on the screen in the same formation as the main grid. Editing the grid causes corresponding changes in the contents of the row-addresses of these characters, and hence changes in their shape.

When you are satisfied with your new character, a read-out of the values now stored in these 32 new row-addresses can be called via function key 18, but since displaying these values will effectively destroy the main grid display, two safety mechanisms are built in. First, the key 18 can only be operated using the SHIFT key at the same time, i.e. 18 is Shifted 17, so you can't press it accidentally. Secondly, even after 18 has been pressed, the program requires you to 'confirm' your request. If you do nothing, the program will, after a short pause, revert to the normal editing mode with the display intact.

Assuming that you have actually finished editing, and now require a readout of the character shape values, the values will be displayed on the screen in four blocks, each block corresponding to one character in the 2 by 2 matrix. Alongside each block of eight numbers the row character shape formed by them is displayed, and, at the bottom of the screen, the full 2 by 2 character is recreated.



Fig. 2. 8x Long range.

All you have to do now is to record the values displayed for inclusion in your object program. Remember, however, the relative position of each of the individual characters within the 2 by 2 matrix.

is 0 1 or 2 A
3 or B C

This must be preserved in your object program.

One method of incorporating the character shape values into a new program is shown below. In this example the shape values of a 2 by 2 character are POKE'd into the first 32 addresses of a new character generator starting at 7168. Generally you will want a variety of different characters in your program, so the use of a counter, I, together with a dummy DATA value at the end of the DATA list, 999, enables you to include extra characters as required without having to worry about the controlling variable in a FOR...NEXT loop.



Fig. 1. BSCR firing range.

LISTING NOTES

The listing given is for the unexpanded VIC-20, and uses the standard *Computing Today* conventions throughout.

The program could have been compacted quite a lot with further use of subroutines but has been left open for the sake of clarity.

For those of you who don't have a VIC, the following notes should enable the program to be converted to other machines with a similarly organized character generator.

POKE 56,38 lowers the top of RAM to protect a character generator starting at 7168.

The VIC's screen display

comprises 23 rows of 23 character positions per row, giving 529 positions in all. The screen is mapped into two separate areas of memory:

- 1) 7680 to 8192 is used to hold the code for each character displayed.
- 2) 38400 to 38908 hold the colour code for each character displayed. (I'm oversimplifying again but it's close enough)

Thus, POKEing any character onto the screen requires two separate instructions:

POKE 7680,I : POKE 38400,O

causes a black letter 'A' to be displayed at the top left corner of the screen, I being the POKE code for 'A', and O being the colour code for Black.

The characters shown as printed in Reverse video in the listing do not actually appear in their reverse form. Without going into details, this particular location of character generator enables normal resolution and high resolution character (Yes, I know they're both high resolution really) to be mixed on the same screen.

without copying from the ROM character generator into the new one. You simply PRINT the normal characters in their reversed form — handy ah?

The graphics characters listed in standard *Computing Today* format are:

Line 50 G @ is a horizontal

bottom bar. [□]

Line 20 G H is a vertical right-

hand bar. [□]

G G is a vertical left-

hand bar. [□]

Line 80 G T is a horizontal top

bar. [□]



Fig. 4. RT Special range.

Finally, I have included some of my own masterpieces using the program. Sorry about the rather feeble joke in the title — I just wanted to give British Telecom's Special Range phones a plug. (PUN).



NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

TIRED OF TYPING?

Why type in thousands of bytes of BASIC when you can buy all these programs ready to LOAD?



Micro Examination

Test your friends and children with this multiple choice program.

Multicolumn Records

Set up your own filing system which enables you to store, search, edit and retrieve data.

Multipurpose Records

A multipurpose data base program for use at home or in the office.

Quiz Time

Assess your performance in terms of speed and accuracy with a multiple choice program.



Communications

Get Commodore talking to each other using this application.

Address Book

Compile an address book, or any other similar list, and throw away those bits of paper.

VIC Blow Up

Find out how characters are made up and generate giant versions on the VIC-20.

Toolkit Program

A simple toolkit program for the Commodore 64.

Tailoring VIC's Character

Create your own characters on your VIC-20.

VIC Editor

Take one VIC-20 and add this program and what do you have? A VIC-20 with an 'enlarged' screen.

8 CM TAPE 1 ☐

8 CM TAPE 2 ☐

I enclose my Cheque/Postal Order/International Money Order for: (delete as necessary)

£..... (made payable to ASP Ltd)

OR debit my Access/Barclaycard (delete as necessary)



--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Tapes are priced at £9.99 each (all inclusive) or you can order two tapes for £19.99

ASP Software
ASP Ltd
145 Charing Cross Road
London WC2H 0EE

Please use BLOCK CAPITALS

Name (Mr/Mrs/Miss)

Address

Postcode

Signature Date

Please allow 21 days for delivery

MAXI-MANDER

MAXI-MAN Data Entry Routine
NB. ALL VALID CHARACTERS COUNT AS DIGITS

ENTER ANY NUMBER
BEHIND -9 AND +9
WITHIN TWO DECIMAL
OF SIX CHARACTERS.

9

ENTER ANY INTEGER
BEHIND -99 AND +99.

-10

ENTER
YOUR NAME.

HENRY

The recent rapid proliferation of microcomputers has meant that more and more people totally untrained in their use will find themselves trying to communicate with, or through a machine. Such applications, already clearly established in computer-aided learning (CAL), computer-aided medical diagnosis, or computerized help for the handicapped, will expand rapidly. What about computer-aided bank loans, insurance, travel, or even gardening? All of these situations rely on an intelligent dialogue between the computer and the user. Current technology usually demands that the user must reply to a number of program-generated questions. A reply is normally effected by typing an answer. It is the Maxi-MAN Data Entry process, the 'moment of truth' in computer-aided transactions, that is the subject of this article.

STRATEGIES

There are two complementary strategies for data entry. The first takes a data string, whether

numerical or alphabetical, and then tests it to check whether that particular input conforms to program requirements before accepting it. The second strategy takes each character, one at a time, as it is entered, checks it, and then concatenates it into a data string. BASIC supports both strategies and implements the first as the standard string INPUT statement. It is clear that this input data may be separated character by character and subsequently tested using string analysis techniques. An example is given below.

Statement	Function
90 INPUT A\$	Get an input
91 AL=LEN(A\$)	Find its length
92 FOR C=1 TO AL	
93 IF NOT ISNUM(VAL,C)	Test with right-hand character for numeric characteristic
94 I = I	
95 NEXT C	

Line 93 would normally check to see if the ASCII value of the character fell into some acceptable range, possibly numerical. In the latter case a letter would generate an error.

It is self-evident that, in

using the INPUT strategy, the whole of the data string has had to be entered before error detection can begin. Microcomputers such as the PET usually require that all characters to the right of the cursor on a screen line at the start of the BASIC string INPUT routine are read as part of the INPUT. If the INPUT occurs in the middle of a graphic display, driven perhaps to simulate a paper pro forma in which data entry 'windows' have been placed, then the length of the string must be checked. Characters in excess of the allowable character length must be deleted and the mutilated screen pro forma redrawn. In the example below the BASIC routine previously described has been modified to include such a routine.

Statement	Function
10 IF A\$=1 FROM 70 TO 11-selected input length	
11 FOR P=AL TO 11	
12 STR\$=STR\$+MID\$(A\$,P,1)	Delimit entire character
13 NEXT P	
90 INPUT C	
140 PRINT "*****"	Report problems with input

Line 80 merely replaces the PET Cursor control characters with printable ones.

BETTER STRATEGY

The second character input strategy is the better strategy because it enables remedial action to be taken as soon as a false input character is detected. It can be implemented in BASIC using the GET statement. The GET is performed almost instantaneously and will return a zero, or "", indicating a null string, even if no key is

pressed. During GET no cursor or characters are displayed and the input string is limited to one character. This means that a lengthy input string will have to be concatenated from a succession of characters before use. The ability to process individual characters before acceptance, despite cursor keyboard and display problems, makes the use of the GET statement very attractive for properly validated data entry routines.



Fig. 1. The simplified data input routine.

DATA ENTRY REQUIREMENTS

It is always good practice to think before acting. It is equally good programming practice to begin with an algorithm, often expressed in a flow chart, however simple. Let us begin by considering the algorithm given in Fig. 1. At least it is simple; it clearly indicates the order of events and divides the task into four separate parts.

The first box of the algorithm in Fig. 1 asked for the specification of the type of data entry required. Let us specify four types. Obviously further types could be added but the following input types (IT) should suit most applications.

- IT = 0 allows any character or symbol
- IT = 1 allows numbers only
- IT = 2 allows integers only
- IT = 3 allows letters only

Numerical input is often

required to fall between set upper and lower limits and it would be appropriate to specify these limits here. If the data string has to fit inside a particular graphics 'window', then this input length (IL) should also be specified at this point. A typical specification might read as follows:

```

IT = 2
IL = 3
UL = 40
LL = -2
    
```

This would mean a numerical integer input, two characters wide, having an upper limit (UL) of 40 and a lower limit (LL) of -2.

ENTRY ROUTINE

Let us now develop a data entry routine based on the preferred 'character input' strategy using the BASIC GET statement. Fig. 2 shows a possible algorithm.

The first point to notice in Fig. 2 is that a decision box is used to show the detection of null values of the input character C\$, which then loops control back to the GET box. Should C\$ get the 'RETURN' character, ASCII Code (13), then there must be a suitable method of ending the input.

A second decision box therefore shows the detection of CHR\$(13) and passes control to the end of the routine. The specification of the input is now relatively complex and a number of different error conditions could occur. It would be useful to tell an unsuccessful user why his data entry has failed to be accepted. Suitable error messages will need to be generated showing how the specified input conditions have been contravened. Notice that the only entry to the error handling routines is through the setting of an error flag. If the error flag (EF) is made to take more than the usual binary states (set and clear) then the flag itself will trigger the appropriate responses in the error handling subroutines.

A second method of terminating the input routine would be for the actual input

string length (AL) to equal the specified input length (IL). This technique avoids disturbing the display graphics. The PRINT statement is required because the GET statement, unlike INPUT, does not display characters as they are keyed in.



Fig. 2. A data entry routine based on the GET instruction.

ERROR DETECTION

The actual checking of each character (C\$) as it is entered is

relatively straightforward once the input type variable (IT) has been specified. The flow chart in Fig. 3 shows a suitable algorithm.

First of all, the ASCII code value of the character must be determined, and assigned as the value of an input variable (IV). Then the value contained in the IT variable must be used to direct program control to that area of code where IV can be compared with the ASCII codes of unacceptable characters. If a match is found then an error flag (EF) can be set. Code comparisons can be made using BASIC condition statements of the IF...THEN variety suitably combined with AND and OR operators. This in turn will assign a value representing the error to the error flag variable (EF), whenever the condition statements evaluate to TRUE.

If, on the other hand, the condition statements evaluate to FALSE, then the input character (C\$) will be concatenated into the string (IS). If the IT value represents numerical input the IV must be reassigned the numerical value of IS after which limit checks can be made.

The routine below shows a much simplified version of the technique. The numerical input given in line 300 would in reality need to be expanded to include the decimal point (.) as well as the plus (+) and the minus (-) signs as valid characters.

Statement	Function
100 GET C\$	Get a character
110 IF C\$="" THEN GOTO 120	Loop on C\$ until string input value (C\$) contains value of input
120 IF IV=1 THEN GOTO 330	This section directs different values of input type C\$ to
130 IF IV=2 THEN GOTO 330	relevant condition statements
140 IF IV=3 THEN GOTO 330	This only allows numbers as valid characters

The error flag variable (EF) can be clearly seen to take different values dependent on the input

type. Suitable values of EF are given below together with an appropriate error message.

Flag Value	Meaning	Error Message
0	NO ERROR (CLEAR)	—
1	DECIMALS DETECTED	—
2	NOT AN INTEGER	ONLY INTEGERS VALID... PLEASE RE-ENTER
3	NOT A NUMBER	ONLY NUMBERS VALID... PLEASE RE-ENTER
4	NOT A LETTER	ONLY LETTERS VALID... PLEASE RE-ENTER
5	OUT OF LIMITS	OUT OF LIMITS... PLEASE RE-ENTER

HANDLING ERRORS

The aim of error handling routines should be to retain program control of input and to restore the input display to the condition it was in before the error occurred. Error messages will make the program 'user friendly' if handled in a sympathetic way. Such messages are best removed once they have been read. The algorithm given in Fig. 4 summarises a suitable method.

Errors merely mean overprinting the display with spaces. There are two cursor resets and there are two ways of doing it: either by printing cursor control characters or alternatively by looking into the operating system and finding out where cursor position bytes are held and poking them back. Of course we have to PEEK into the right location before we start the input routine at all. So, we should amend Fig. 2 to show a 'FIND CURSOR' line between 'START' and the 'GET CHARACTER' box.

THE MOMENT OF TRUTH

The requirements of data entry are such that the program must be able to cope with idiosyncratic entries such as typing letters when numbers are required, or inadvertently keying RETURN or other control keys without aborting a program halfway through. Inevitably this involves

the checking and validation of all data entry characters, even though this may limit the speed

of typing if written in BASIC. In the final analysis an input, once accepted as reasonable by the program and subsequently accepted by the user, although actually incorrect, will be processed as valid data. The error may or may not be significant but garbage in nearly always results in garbage out. Well validated input will help reduce the incidence of garbage input but will not eliminate it.

SIZE

For a program segment to be commonly used it should be relatively short, especially as the pooled generation of microcomputers is limited in its free memory capacity. In its smallest available form (Tiny-Mander), the complete routine occupies less than 1K. The heavily commented (Maxi-Mander) listing offered in this article occupies a massive 5.9K and is intended only as a program for study. The removal of REM statements will reduce it to 1.3K, in which form it could be used directly.

PET CHARACTER SET

All the graphics and cursor control characters have been put into the Computing Taskay standard format.

PET OPERATING SYSTEM

The 6502 used in the PET allows

zero page addressing, a particularly fast and compact method for storing information, and it is used by Commodore for operating system variables. Zero page merely means the first 256 bytes of memory, which needs only one byte to address. Explanation of this area of the memory map is given in some versions of the **PET User Handbook** and in Nick Humphreys' book, **The PET Revealed**. The New ROM PET uses Page zero much more effectively than the old ROM machines in which many of the variables spill over into Page 2 (Page 1 being used for other things).

Examination of the memory map soon shows that certain new ROM Page zero addresses hold cursor and keyboard information. These are summarized below:

Old ROM	New ROM	Description
937-938	144-145	Hardware interrupt vector (IRQ)
939	159	No. of characters in keyboard buffer
944-945	163-164	Cursor log (row, column)
951	167	Cursor on 10 = flashing cursor else off
949	168	Cursor timing countdown
948	170	Cursor blink flag
224-225	196-197	Pointer to screen line
226	198	Position of cursor on above line (0-79)
248	216	Line where cursor lives

Owners of old ROM machines should use the alternative locations given above, taking care to check that variables take the same values as those given, which are for New ROM PETs.

During trials of this routine it was thought that the cursor log in 163-164 would be the best start point and routines using the log worked most of the time. For reasons unknown, inputs embedded in loops would unpredictably change the display line spacing, adding an unwanted cursor up after some types of error detection routines. This approach is therefore discarded for the moment. Variables in addresses 216 and 196-197 perform similar functions.

The screen line and cursor line need not carry the same information if cursor control characters are used. The cursor line is the actual line where the cursor lives. It may have been moved away from the screen line by cursor movement controls working relative to the screen line. If the edit keys are

A major problem is also encountered in switching off the cursor without leaving a pixel block permanently displayed. If the cursor blink is on, as the cursor is switched off then the pixel remains displayed. Much effort was devoted to looking into addresses 168 and 170 and trying to ensure the cursor was only switched off when the cursor pixel was in the 'off' part of the blink cycle. This was partially successful but differences always occurred under cursor removal under program control such as when the maximum length was met and cursor removal by entering RETURN. This method should lead to a solution, but termination of the input routine by printing a space solved the problem very neatly at the expense of making the programs 'window' at least one space longer than the specified input length. For presentation aesthetic an extra space at the beginning is also needed. The cursor is best displayed, clearly isolated, in the second character position of the 'window' when inviting data entry.

Addresses 144 and 145 are pointers to interrupt service routines. By changing the contents of the low byte, 144, to 49 (the high byte remaining unchanged) and skipping the STOP key detection routine, the STOP key is effectively disabled. The normal content of 144 is 46. Even when disabled the edit part of STOP key action still occurs and the cursor movement generates an error. This can be avoided by looking into the keyboard Peripheral Interface Adapter (PIA) which lives at addresses 50408-11. 50410 is the register address of the keyboard row input byte. If the fifth least significant bit is set to logical zero (which happens when the STOP key is pressed) with all the other bits remaining at logical 1, then the number 239 is generated. PEEK(50410), if equal to 239, detects STOP even when it is disabled. The conditional loop of line 640 thus avoids the edit problem.

One final point: the keyboard as a whole can be



The program as given demonstrates itself by asking for inputs within specified limits. Numbers are trapped and suitable error messages given.

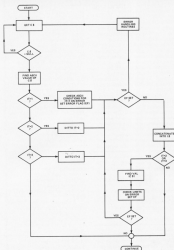


Fig. 1. How the program detects errors.

disabled by setting the least significant bit of PIA 1 register address 59411 (ICB 1) to logical zero. This can be done by a POKE 59411, 60. To enable the keyboard POKE 59411, 61. This option is not used in Mander.

CRASH PROOF INPUT

This section is divided into four subsections:

- 580-590 — Set up
- 600-750 — Enter valid data else no error
- 760-800 — RETURN (exit) routine
- 820-900 — DELETE (erase)

The setting up uses the Page-zero variable addresses to place the cursor, disable the STOP key a POKE 144, 48 must be included at the end of the main program or else a direct POKE must be used after the program has finished. The final way to reset is to switch off and back on. The action of all editor/control keys except DELETE is ignored by the statement in lines 660 and 670. The checks, except for the length of the input string (IS), are handled in other subroutines but the necessary GOSUBs appear in this section. Condition statements pointing to exit and erase are embedded in the

section 600-760. The code is best followed using the algorithm in Fig. 2.

ALPHANUMERICAL CHECKS

The program lines 920 to 1000 perform the alphanumerical checks. The series of condition statements at the beginning is used for clarity but in Tiny-Mander a single statement of type ON...GOTO occupies less space and executes more quickly. For numerical inputs the numbers 0-9, CHR\$(48) to (59), the decimal point CHR\$(46), the plus CHR\$(43) and minus signs CHR\$(45) are the only valid characters. The plus and minus signs must lead and only one decimal point can be allowed. When one decimal point has been detected then the error flag (EF) is given the value 1. If a second decimal point is detected then a 'non-numerical' flag is set: EF = 3. For alphabetic inputs the hyphen, CHR\$(45), the apostrophe CHR\$(39) and of course the full stop, CHR\$(46), together with normal letters CHR\$(65) to (90), must all be 'accepted' as valid. If an invalid character is detected then EF is set to 4 meaning the character is not a letter.

LIMIT CHECKS

The limit checks are very straightforward. If IV (on its second and final use as the numerical value of the input string) is outside the specified limits (LL & UL) then an 'out of limits' error flag is set: EF = 5. In practice two other conditions must be accommodated:

(a) Once any error is detected and EF set then further checks must be skipped or EF may be reset and generate an incorrect error message.

(b) The statement VAL returns the value zero (0) for alphabetic input as well as for the numerical value zero. The ambiguity can be resolved by assuming that the numerical value of the leading character must also be a zero. The penalty

[illegible]

Games of the unexpected for the unafraid...

THE WHITE BARRONS

Sometimes and the most of solid chambers look as evil
Barons when you need to stop. You're in, for a protected
by Gods, Demons, Spirits and the monstrous Dragon or two!
Your magic staff will block the tunnel to prevent the
escaping unless, that is, he survive you.
A real brain buster, White Barrons requires both brains
and brawn from its players. It's no good just hunking
your way through the Barons and hoping to fall over
the Barons. Eventually you'll meet a Dragon, and
they don't back easily! You'll need all your strength
and cunning to survive this one for long.
THE WHITE BARRONS Only £6.50 all inclusive!

CELLS AND SERPENTS

More mysterious than you ever thought could be
hidden your keyboard. Wander the hills in search
of gold and glory but be wary, very careful where
you tread! There are things here that will make
your wildest nightmares look like John Deere's
Pony-riding a Mind Flyer, for example? Or how
about shaking hands with an Demon? (You'll
only do that once!) Treasure is here to be found
though... the hard way.

Are just how good you really are at
adventuring with this practically unmovable
fantasy. Not for the faint of heart or the slow of
tread!

CELLS AND SERPENTS Only £6.50 all inclusive!

*** SPECIAL DEAL ***

Both programs for only £11.45 all inclusive!

Our Adventure Series programs are available on
tape for the following systems:
Commodore VIC 20 (not available for White
Barrons), Commodore PET, Sharp MZ 800 and
MZ 800, Tandy TWC 80 Model I, BBC Model B
or 3.01 Model A, Atari 800 and 800, Sinclair 486
ZX Spectrum.

ASP Software, ASP Ltd,
145 Charing Cross Road, London WC2N 6ES

Please send me repeat of the following programs:
The White Barrons @ £6.50 each
Cells and Serpents @ £6.50 each
Both tapes at special price only £11.45
My system is a Computer

TRADE ENQUIRIES WELCOME

I am enclosing my Cheque/Postal Order/Money Order
(delete as necessary) for £ (payable to ASP Ltd)
OR Debit my Access/Barclaycard (delete as necessary)

.....

Please use BLOCK CAPITALS and include your postcode
NAME (Mr/Ms)
ADDRESS
.....
..... (Postcode)
Signature Date

VIC BLOW-UP

One of the nicest things about Commodore's microcomputers is the ready-made graphics set that comes as standard with their machines. The VIC-20's character set is probably the best within the existing range, so I thought it might be interesting to have a look at exactly how the VIC sticks its pixel dots together to make up what is really a rather friendly set of characters.

THE SHAPE OF THINGS TO COME

As you doubtless already know, the VIC screen display is made up of 23 rows of 32 characters per row, giving a total of 500 character positions in all.

Each of these character positions can itself be divided up into eight rows of eight pixels per row — a pixel being the smallest area of the screen that can be individually controlled. On the VIC these pixels are rectangular, with a width to height ratio of approximately 3 to 2. Since the complete character position is,

as we said, built up in eight rows of eight pixels, then this too will be rectangular in the same proportions.

Thus, if you were able to magnify an individual pixel eight times you would find yourself looking at a normal size 'Reverse Space' character.

This technique is used by the program to 'blow-up' any of the characters within the VIC's ROM held character set by magnifying each of the 64 pixels making up the character.

THE PROGRAM TECHNIQUE

The program works by identifying the location in ROM character generator of a character typed in at the keyboard. The eight bytes holding the character 'shape' information are then retrieved from the character generator, and the 'bit-pattern' translated into eight rows of eight full character positions on the VIC's screen display. A 'Reverse Space' character is used to represent a pixel 'set'; a normal 'Space' for a pixel not set.

If you've ever wondered just how the characters your computer displays are made up then this program will reveal all.

The screen thus displays the exact arrangement of pixels as used in the formation of the normal size character — only eight times bigger. The screen also displays the memory locations of each of the eight bytes used for the character, and the values stored in those locations — both in decimal. All of the VIC character set is accessible, including all upper and lower case alpha-numerics and graphics, plus all 'reverse video' forms. The mode in which the program is currently operating, is:

- 1) Upper case/normal video
- 2) Upper case/reverse video
- 3) Lower case/normal video
- 4) Lower case/reverse video

is continuously displayed. Switches between the various modes — upper to lower case, reverse video on and off — are made in the usual ways, which if you're not sure about them, are:

- 1) Change from upper to lower case and back — press Shift and 'Commodore' keys together.
- 2) Change to reverse video — press 'CTRL' and 'VRSOFT' together.
- 3) Change back to normal video — press 'CTRL' and 'VRSOFF' together.

Invalid entries from the keyboard, eg. 'Return', 'INS/DEL', 'function keys', etc., are error-trapped. The program does not crash, but can be interrupted by pressing the 'RUN/STOP' key.

The listing is short enough not to tax even the most weary fingers, and the full Commodore Today standards have been



adhered to without.

If you're into re-defined character generators the program provides a quick and easy method of identifying the memory location of all the VIC's 'old the peg' characters for transferring into your own character generator as required.

Even if you've not a graphics geek, I think you'll find discovering how Commodore designed the VIC's excellent character set an entertaining experience in itself.

PROGRAM NOTES

The program is written in BASIC for the unexpanded VIC-20, but the following notes, together with the accompanying program description and Table 1, should enable the listing to be converted to any micro with a similar type of character generator.

The VIC screen display is controlled by two separate areas of memory. The first, starting at 7680 decimal, is used to contain the character or POKE code, whilst the second, starting at 38400 decimal, contains the code for the colour of the screen position. Thus:

POKE 7680,1:POKE 38400,0

causes a black A to be displayed at the top left corner of the screen — 1 being the character or POKE code for an A, and 0 being the colour code for Black. Address 36879 controls the background/ border colour combination. Address 36874 controls the pitch of the first tone generator of which the VIC has three, plus a white noise generator and address 36878 controls the volume of the sound generators.

Address 96669 is one of the busiest addresses in the VIC,

VARIABLES

A\$	Single character input by user
CH	POKE code of A\$
CR\$	Indicates normal or reverse video selected
CA	Indicates upper or lower case selected
CS	Pointer to character generator shape tables
CT	Character video RAM location
CO	Colour video RAM location
I	Counter used to point to each of the eight addresses containing the character shape information.
J	Counter used to point to each of the eight bits in each address containing the character shape information.
VL	Value held in each of the addresses containing the character shape information.
MX	Used as a reference value in conversion of the character generator bit patterns to the 'magnified' display
X	Delay loop counter in error trap routine.

PROGRAM STRUCTURE

Line	Function
Lines 10-20	Set border colour to Blue/background to White and clear screen.
Lines 30-110	Loop displaying current operating mode and testing for mode changes and/or single character inputs by user.
Line 120	PRINT single character input at top left of screen.
Lines 130-140	Get POKE code of character by PEEKing top left of screen; check input is a 'printable' character and if not, branch to error trap at line 330
Line 150	If input is a reverse video character, add 128 to POKE code.
Line 160	Locate start of shape tables for required character; initialize character and colour video RAMs.
Line 170	If a lower case character, redirect to lower case shape tables.
Lines 180-280	MAIN DISPLAY: Print magnified version of character by converting bit patterns from shape tables into patterns of full-size Reverse Spaces. Print the eight addresses containing the shape information and the values, in decimal, held in them.
Line 290	Print the input character at normal size within a Green border.
Line 300	Hand back to keyboard control loop.
Lines 310-360	Error trap — Clear screen, turn whole screen Red, sound audible alarm, advise of error, pause and resume program.

and a full description of everything it does would take up an article in itself. Suffice it to say that in this program, if it contains 240 decimal then the VIC is working in upper-case; if 242 then lower-case has been selected.

Finally, besides POKing characters onto the screen, the VIC has a particularly powerful PRINT command. The horrendous looking line 290 is a good example of the flexibility of the VIC PRINT statement. The listing uses the *Computing Today* standards for representing graphics and cursor control statements, so for those with the stomach for it, this is what line 290 actually does.

The cursor is first HOME'd to the top left of the screen as a starting reference point. It is then moved down 13 rows and across four columns. The cursor colour is set to Green and then

PROGRAM PROTECTION



Despite recent advances in microcomputer technology, unauthorised copying of programs remains a seemingly insoluble problem. Most existing methods attempt to prevent a return to the command level thus making copying impossible. This is achieved by amending appropriate addresses in the machine by using POKE. Such systems are, of course, ineffective if the program is not run.

This technique, although not a complete solution, has certain attributes which make it very effective. No attempt is made to prevent a copy being made and this can be verified against the original; if the copy is loaded into the PET, it will run and the listing is identical. However, when the user returns at a later date with the copy and attempts to use it, the copy appears corrupted and will not run.

CREATING PROTECTED COPIES

The technique listed here is suitable for use with 16K and 32K PETs.

1) Append an additional statement to the program to be protected. List the program and insert SYS 1000 near the beginning. Remember not to append it to a line which already contains a REMark statement because the entire content of a line following REM is ignored. Decimal 1000 is particularly suitable because it implies a small machine code subroutine resident in the second cassette buffer.

2) Determine the top of BASIC text. At this stage it is necessary to determine how long the program is. This is done by examining the pointer to the start of variables. Variables are stored in the RAM immediately following the BASIC text. Enter:

```
PRINT PEEK(40) PRINT PEEK(42)
```

When Return is pressed two figures will appear:

```
4
104
```

These are the high order byte and the low order byte, expressed in decimal, of the pointer to the start of variables.

This simple method of protecting your programs from being copied may not be the most elaborate but it can certainly cause considerable frustration to the unauthorised user.

They must now be converted to a four-digit hexadecimal number. Remember Hex is base 16 and the figures 10 to 15 are substituted by A to F. For example, decimal 200 becomes FF and the example above, therefore, becomes 0468 Hex.

3) Enter the Machine Language Monitor (TLM). TLM is resident in 16K and 32K PETs but a cassette version is required for 64K models. Enter:

```
SYS 1624
```

When Return is pressed the monitor is entered and a display of registers will appear.

4) Display the contents of decimal 1000 (Hex 03E8). After the full stop, type:

```
H 0000.00FF
```

When Return is pressed, the contents of the bytes from 03E8 Hex to 03EF Hex will be displayed on a single line as two-digit hexadecimal numbers starting with 03E8 Hex. If a '7' appears, the space between the M and the 0 of 03E8 Hex has been omitted. Although this is the standard format for using the monitor it seems designed to mislead! It is not the space that is important, any character can replace the space or comma without affecting the operation of the monitor. The start and end addresses must contain four digits and be in the correct position on the line.

5) Place the 6502 op-code 'Return from subroutine' in decimal 1000 (hexadecimal 03E8). Move the cursor up to the 03E8 Hex in front of the line of bytes previously displayed. Move across to the first two digit number, overwrite it with 60 and press Return.

6) Save the program using the Machine Language Monitor (TIM). TIM is used because the start and end address can be specified and it is necessary to save the op-code in 03E8 Hex with the BASIC text. After the full stop, enter:

S. "TITLE",01,03E8,0440

Where 'TITLE' is the program name, 01 is the device number to which the save is made (in this case the first cassette), 03E8 Hex is the start address and the last hexadecimal number (in this case 0440 Hex) is the figure previously calculated to be the top of BASIC text. The familiar 'PRESS PLAY and RECORD ON TAPE 1' should appear when Return is pressed. Operate the cassette deck as normal. If 'P' appears you have probably forgotten the space after the S.

To Exit from the Machine Language Monitor is done by typing X and pressing Return.

You now have a protected copy. The copy will load in the usual way because LOAD detects the start address placed on the tape header by TIM.

HOW IT WORKS

The protected copy contains the contents of the bytes from 03E8 Hex upwards as well as the BASIC text. When the program is LOAded and RUN, SYS 1000 passes control to decimal 1000 (03E8 Hex). The op-code 60HTS immediately passes control back to BASIC and the program RUNs as normal.

If an unauthorised copy is made using SAVE, only the BASIC text from 1004 upwards is recorded — the vital op-code in 03E8 Hex is not included. If the unauthorised copy is loaded to check it is correct, it will run as expected. This is because 03E8 Hex is in a protected area of RAM and is unaffected by command level instructions. The op-code from the original copy will still remain to pass control

back to BASIC, providing that the PET has not been switched off or the second cassette buffer used in the interim period.

When the user returns at a later date, the copy will LOAD but any attempt to RUN will cause the monitor to be entered followed by a break or a crash depending on the contents of 03E8 Hex at the time.

CONCLUSION

This technique offers a more subtle approach to program protection and attempts to confuse the copier rather than throw down a blatant challenge to defeat the protection.

NOTE: In order to adapt this program to work on other Commodore computers see the article on converting listings on page 6 of this issue.

LOW PRICE HIGH QUALITY SOFTWARE FOR COMMODORE 32K PET AND 64 PURCHASE AND SALES CONTROL £80 + VAT

Runs both purchase and sales ledgers with optional calculation of VAT from the goods or list prices, analysis by resourcing period, 'due for payment' report, totals for VAT and gross, etc.

INVOICE PRINT £80 + VAT

Prints invoices on your own stationery laid out according to your instructions. This programme is an optional add-on to be used in conjunction with purchase and sales control.

STOCK CONTROL £80 + VAT

Keeps detailed stock records including stock location, re-order levels, quantity on order, cost and selling prices and stock valuation.

NOMINAL LEDGER £80 + VAT

Produces trial balance and up to 20 reports in addition to profit and loss and balance sheet. This programme is intended for use on its own, but it can read files set up by our purchase and sales control and stock control programmes.

INTEGRATED ACCOUNTING SYSTEMS FROM £400 + VAT

Write or phone for details and complete software list

ELECTRONIC AIDS (TEWKESBURY) LTD

Myrtle Crest, The Nettle, Tewkesbury
Gloucestershire GL20 8JH
Tel: 0282 81 030/0342 296002

16 K RAM PACKS FOR VIC 20

To the trade only

PLUS 80 GUARANTEE

To beat any manufacturers price on 16K
and 24K RAM packs with full one year
guarantee.

This offer applies to a minimum order of 10 units.

Write or phone for details to:

PLUS 80

31-33 LOWER ROAD
HARROW
HA2 0DE

Tel: 01-423 6393

BIBLIOGRAPHY

When you consider the vast number of Commodore microcomputers that are in use in this country it should become fairly clear that this is not a comprehensive list of the books that have been written about them! Indeed, with new books coming out all the time it is unlikely that we could have ever put together a comprehensive list. So what we did was to look at a small selection of the books that were to hand and put together some of the more relevant details about them.

Commodore 64 Computing
Ian Sinclair
Ginn Press Publishing
132 pages; £5.95

This is an introductory guide and reference book for all Commodore 64 users. It covers the setting up and operation of the microcomputer and its many facilities in detail. BASIC syntax is comprehensively examined with examples to serve as a useful reference for the experienced user and a helpful guide for the less experienced.

PET/IBM BASIC
Richard Huxford
Pitman-Hill International
154 pages; £9.70

This is a practical manual that goes well beyond theory and shows you how to actually write programs in BASIC. Complete with examples illustrated by photographs from the computer's video screen, the book offers a step-by-step approach to top-down programming that will help you to apply fundamental concepts and program a computer easily and quickly.

The PET Index
Michael A. F. Ryan
Gower Publishing Company Ltd
194 pages; £12.90

The aim of this book is to provide an index to help the

reader locate references to the Commodore PET/IBM microcomputer. It covers 17 different publications (200 issues in all). All early references to the VIC-20 have also been included.

PET Games and Recreation
Mac Ogleby, Les Lindsay and Dorothy Kurian
Benton Publishing Company Inc.
245 pages; £12.95

The title describes the content of this book which includes plan-ahead games, deductive reasoning games, games of chance, language and counting skills games and recreations (purely for entertainment) emphasizing the PET's graphics capabilities.

Hands-on BASIC with a PET
Herbert D. Peckham
McGraw-Hill Book Company
267 pages; £10.95

This book, a modification of an earlier work by the author, is designed to be useful to anyone who wants to learn how to program the PET in BASIC. There are 10 chapters: The PET computer and BASIC; Getting acquainted with the PET; Introduction to BASIC; Computer arithmetic and program management; Input, output, and simple applications; Decisions, branching, and applications; Looping and functions; Working with collections of information; Do-it-yourself functions and subroutines; and Random numbers and simulations.

PET Personal Computer Guide
Adam Osborne, Jim Strauss and Ellen Strauss
McGraw-Hill Book Company
630 pages; £11.95

This book enables you to learn PET BASIC programming, highlight your programs with graphics, music and other special features, to evaluate and install the latest Commodore

A quick perusal of books for your Commodore microcomputer.

peripherals, to use the PET modem and voice synthesizer, and to write and edit drawings and programs with the PET's advanced screen editor.

Learning to use the PET Computer
Garry Marshall
Gower Publishing Company Ltd
67 pages; £6.95

This book provides a simple, down-to-earth, jargon-free introduction to the PET and its software. Many applications of the PET are described, including business, educational and hobby uses. Also, a simple and direct introduction to programming the PET is given.

VIC Graphics
Nick Hampshire
Dorset
125 pages; £6.95

This book provides the reader with an introduction to programming techniques used to generate graphics displays on a VIC-20. The topics covered include using colour, two dimensional shape plotting, shape scaling and stretching, shape movement, shape rotation, plotting using matrix manipulation and three dimensional shape plotting.

VIC Innovative Computing
Clifford Randolph
Melbourne House
151 pages; £6.95

This book provides a brief explanation and the listings for the following: Space flight, Hi-speed maze, High-octane drive, Warlock, Dragon's lair, Synthesizer, Ganyade, Hoppy, City bomber, Battleship, Maths, Duck shoot, Grass pits, Rat trap, Earth attack, Bomber attack, Blackjack, Squash, Save the shuttle, Hangman, Alien overan, Invasion, Demper, Siege, Snakes & ladders, Dungeon, Gold, Nuclear attack, Chess and Assassin.

CLIFFORD & MARK
RAMSHAW

COMMODORE 64 GAMES BOOK

Teach your Commodore 64 every trick in the book.

With the best software games book ever for your Commodore 64.

This mind-blowing collection of game programs, written by software writers Clifford and Mark Ramshaw, will turn your Commodore 64 into an entire grade of electronic fun and thrills. Experience action so fast and furious it will shatter all your concepts about computer game excitement.

The Commodore 64 Games Book is packed with un rivalling space and adventure games, and includes intriguing strategy games too. Some programs even contain machine language routines.

Every game maximises all the Commodore 64's sophisticated features, including its innovative Sprite capability. And you don't need complex programming skills, because all these programs are very easy to enter.

If you want to know every trick in our book, order your copy today. Book £5.95.
All programs from this book are also available on cassette. Each cassette contains 10 programs. £8.95.



We can teach the VIC 20 a few tricks too!



Vic Innovative Computing

38 highly creative game programs that will open a new dimension of thrills and excitement for every VIC 20 owner. Featuring many favourites like Koppo, Nuclear Attack, Space-Flight and Chess.

The specially designed format ensures that all the programs are very easy to understand and enter.

These programs are also available on a set of 3 cassettes. Book £8.95.
Each cassette £3.95.



Vic Games Pack

Five fantastic space games in one cassette, including Invaders, Storm and Ground Attack. Every game makes full use of the VIC 20's stunning graphics and sound.

Also included are two 100% machine language programs, Alienblast and Space Reflex. The Game Pack is the ultimate test of nerve and skill—unrivalled value for only £5.95.



Wizard and the Princess

In this multi-part medieval graphics adventure, you are a bold knight who must rescue the beautiful princess from the grasp of an evil wizard.

To succeed, you must storm the castle, slay fire-breathing dragons and fight the evil lord to the death. A challenge that pushes you and the capabilities of your VIC 20 to the limit.

Cassette for standard VIC 20 £5.95.

MELBOURNE HOUSE PUBLISHERS

100 Melbourne House, 100 The Arcade, London WC2N 6DB

Telephone 01-4039 4444

Write to: Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

or Mel House, 100 The Arcade, London WC2N 6DB

COMMODORE 64

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

STANDARD VIC 20

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

Standard VIC 20 Games Book

COMMODORE 64

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

Commodore 64 Games Book

MELBOURNE HOUSE PUBLISHERS



LIGHT PEN



So many uses, so easy and so good,
it's MINDEBoggling!!

now available for:
ATARI 400/800* VIC 20
BBC A&B* COMMODORE 64*

*allow high res graphics

only
£25.00^{inc}

STARTECH
STARTECH
STARTECH

COMPUTERS
sales & services

!!FREE GAME WITH EVERY PEN!!

Please rush me

Name _____
Address _____

I enclose Cheque P/B for
CREDIT CARD ☐ ACCESS ☐ BARCLAY CARD ☐

--	--	--	--	--	--	--	--	--	--

24 HOUR ANSWERPHONE SERVICE

208 Aigburth Rd, Liverpool L17.

051 727-7267

Black

