

# DRAGON



# USER

February 1988

*The independent Dragon magazine*

## Contents

**Letters** 2  
Dragon teaches with maths ... disc fill error ... spig errors ... more projects please ... re-edit EDIT ... more spig ... WP guru fights back.

**News** 4  
Microdeal quits ... Harris has cheaper disc drives ... OS-9 user group ... new FORTH for OS-8 ... safety plug.

**Dragonsoft** 5  
Database from Pulver and Boulder Crash 2 from Giant Soft.

**Reboot** 6  
Same further thoughts on the BGGT command from Mike Hales and Julian Delverne.

**BREAKING the '84** 8  
Marlyn Amisage interrupts the Dragon 84's PRINT routine.

**Winners and losers** 9  
Gordon Lee helps September's competitors to sort out their lives and sizes.

**Pamcodes** 10  
Pam O'Leary submits to assembler and tackles the Yellow Book.

**Crossword** 12  
Last month's winners, this month's contest.

**Expert's Arcade Arena** 13  
The Expert engages with Arbut and redesigns the map.

**Adventure helpline** 13  
Plus the Communications coupon.

**Dragon drives direct** 14  
C J Walton describes how to use a bought interface to turn the Dragon into a driver for many kinds of common PC motors.

**Writer: ADVENTURE** 22  
Pete Gerrard puts real character into his characters.

**Adventure trail** 23  
Pete Gerrard tells a tale of Trekbot and tribulation.

**Down in the Dumps** 25  
A dump for the Brother HR-5.

**Competition** 26  
Gordon Lee tells circumstances suit daily with him.

**The Answer** 27  
Gordon Lee's own solution to the November competition.

Brian Cudge is on holiday.

## Editorial

HAPPY NEW YEAR. I've been waiting for weeks to say that. To you, it may be late January, cold, wet and miserable, but to us it's the 4th of January, cold, wet, etc. Best Press, most of British industry and the Post Office (nine days for Gordon Lee's copy to reach us, first class) stop for Christmas and the New Year, but not the Dragon.

John Penn Software has been in touch to announce a Dragon and Tandy show next April at Cardiff Wales Airport. The Penns are looking for demonstrators as well as retailers. See page 21 for further details. I was only mildly disappointed to discover that what I thought was Purple Car Painting was, in fact, Apple Car Parking, but perhaps Mr. Makin can arrange something ...

Requests for hardware projects are being met with vengeance this month with a blockbusting experimental interface project. Although the article is long, the individual sections are in reality quite simple and described in detail, so all would-be hardware enthusiasts can have a go.

Many thanks from everybody at and around Dragon User to our subscribers and advertisers, who are the real reason the Dragon keeps going. Here's to another year ...

**Telephone number**  
(all departments)  
4254343

**Editor**  
HELEN ARMSTRONG

**Production Editor**  
DRAGON EDITORIAL

**Administration**  
CAROL FRITH

**Advertisement Manager**  
DRAGON EDITORIAL

**Marketing Manager**  
HELEN PERRY

**Managing Editor**  
PETER KANE  
**Publishing Director**  
JERRY IRELAND

**Subscriptions**  
UK £14 for 12 issues  
Overseas (air freight) £28 for 12 issues  
ISSN 0955-1777. Tel: 286076  
Dragon User, 1253 Little Newport Street,  
London WC2H 9PP  
UK address: c/o Business Press International,  
205 East 41st St, New York, NY 10017  
Published by Best Press Ltd.  
© Best Press 1988  
Typesetting and Production by Adept Limited,  
London W8H.  
Printed by Hestley Brothers Ltd, Ashford, Kent  
Registered at the Post Office as a newspaper.  
Dragon and its logo are trademarks of  
Turnham Ltd.

### How to submit articles

The quality of the material we can publish in Dragon User each month will, to a very great extent, depend on the quality of the disclosures that you can make with your Dragon. The Dragon computer was launched on to the market with a powerful version of Basic, but with very poor documentation.

Articles which are submitted to Dragon User for publication should not be more than 3000 words long. All submissions should be typed. Please leave wide margins and a double space between each line. Programs should, wherever possible, be computer printed on plain white paper and be accompanied by a tape of the program.

We cannot guarantee to return every submitted article or program, so please keep a copy if you want to keep a program retained you must include a stamped addressed envelope.

# Letters

This is your chance to air your views — send your tips, compliments and complaints to Letters Page, Dragon User, 12-13 Little Newport Street, London WC2H 9PP.

## Not for killing

**Millenaria** Mike Hides's letter (December 1987) regarding the problem of quick disc filing, I have used the disc version of Selester since 1985 with a Dragon 64 and DragonDOS, and I came across the same problem.

However, the problem arose only after I had used the kill command in the Reisdisk program, so the fault must be there and not with Selester; and if the use of K.S.L. within the program is avoided you can get more files on the disc. I usually average about 80 files alongside Selester and Reisdisk on a single-sided disc.

Another problem I find with the Dragon 64, but not a '32 K' that if you wish to stop printing a file you have to turn off the printer as the print window on the 64 seems to be different to the 32. Other than these, I have had no trouble with Selester and feel that it is an excellent word processor and hope the hints given here are of use to other users.

Robert Haigh  
Gleadthorpe  
63 Gleadthorpe Bank  
Gleadthorpe  
Huddersfield

## Black and white case

THANK you for printing my screen dump in the November issue of Dragon User. I attempted to say that I left out one important piece of information, i.e. that the pens in the printer have to be rearranged so that they are in the order green, black, blue and red. Without this, users will get some very odd looking screen dumps.

Now for some information for fellow CoCo users: I don't know if you've noticed, but Tandy seem to have given up on the CoCo in favour of such trashy computers as the Spectrum (spit). As a matter of fact, my (ex-) nearby Tandy shops have about two CoCo games, and about fifty Spectrum and Amstrad games. I have been unable to

Every month we will be shelling out a game or two, courtesy of Microdeal, to the reader's who send the most interesting or entertaining letters. So send us your hints and your opinions, send us your hi-scoring and suggestions. Send us your best Dragon stories. What if you think we are, mind readers?

EXTRA  
PUFF

## Dragon maths are the right answer

THANK you for the reply to my letter concerning the non-arrival of a recent Dragon User. I hadn't actually expected replies to the points raised, which I subsequently reacted to fill up the page in the transfer of sending typed upon the writers in the above expectation that it would get all well, and kind.

Your point is taken about publishing lists of suppliers, and is one which many readers that were so concerned about the missing DU. The point I was trying to make is that the shows are not likely to come to Ireland, and those who in there like most rely on Dragon User as the only link to what is happening. To put it another way, you and the rest of the staff and contributors of DU are probably the only reason that the Dragon continues to exist as a viable computer. Since three years of consistently trying to have the school's Dragon replaced by a Mac, as a network have consistently failed, the Dragons have become an even thought they are becoming more and more difficult to maintain; for example, Pecosoft's joysticks are out of production, as I found when I tried to purchase them.

The kids are hard on joysticks, but when you are an 'inner-city' school, budget their teachers be allowed to go back after school to play computer games. I must be doing something right, in case you don't find that surprising. I receive screen dumps from two of the 'games' — they are standard ' joystick to right answer' type — is MATHS!

Keep up the good work.

Denis J. McCarthy  
of Commercially Services  
Dublin Co. Dublin, Ireland

MAGAZINES and special interest groups are rarely the only reason why something continues to exist, but they frequently provide a support system outside people's personal circle of friends and acquaintances and prejudices, even though the benefits may seem elusive at the time when it is not providing exactly what the consumer wants.

Any large group united by an interest is split by numerous factions with different angles. I recently saw a survey on what improvements Dragon User readers want from the magazine which was split very evenly between the three headings: more OS-recommends, 'more basic' and 'less rubbish'. The even split is far more significant than the headings, because every month I get letters praising DU's articles on those and other subjects, and every month I get letters discussing identical subjects as 'rubbish'.

If a computer needs 'usefulness' to justify its existence, then getting youngsters interested in maths should ensure the Dragon spirit that is not physical immortality.

'Dragon's Roar' December 1987, Ed. Simon Jones.

get any answer from Tandy about this even though I live near the UK headquarters. It seems the CoCo is going down the same road as the Dragon.

Finally, I have a complaint. Since going subscriptions, the quality of the printing in Dragon User has deteriorated, along with the spelling. The December volume of Adventure Italia full of spelling mistakes. I hope this is only a passing phase, as a detour from the employment of the magazine.

P.R. Marlow  
56 Lime Avenue  
Bentley  
Walsall  
W Midlands  
WS2 6UP

THANKS, P.R. This is the latest letter I've received all week apart from my gas bill. Perhaps if we can get the black and white right we can graduate to getting our colour scrambled, what of you?

Those aren't spelling mistakes, they are typos. Adventure Italia must have slipped the net in December. Please accept our apologies. I must disagree about the print quality; apart from wealthy magazines with large circulation, Dragon User has one of the best print standards about.

We are hoping to get more news on the progress of the CoCo, and in particular the CoCo Three in due course.

## Disc bug trap

I am writing in response to the letter from Mike Hides. The problem described is not a fault in Selester but one of at least 18 bugs in SuperDOS CB. While I would like to say the only cure is to get a copy of DOSPlus, the problem can be overcome by inserting the disc

on CLOOLD

www.program TELECOM BAS,  
Philip G. Scott  
4 Ridgeway Drive  
Frimley  
Camberley  
Surrey  
GU7 6UP

## Neophite whizzard

I SEE the cry is an again for more machine code. We are not all whizz kids, please help DU as it is, with something for everyone.

Only yesterday my son received a letter from one of his friends, asking where he can get a Dragon, as he has had four yrs, four Commodores, which have had to be reformed as they could not work properly!

So there is always someone out there likely to start from scratch.

Could anyone in Essex who has a spare Dragon, please contact my son's friend, Mike Wilson, Upper Mead, Boyles Court, Park Lane, Gt. Marley, Brentwood, Essex CM8 5LL? R.J. Ralph.

294 Grafton Way,  
Lynside, Leighton Buzzard,  
Bedk LU7 9GB

A happy story, which we hope will have a happy ending, and also a living, breathing example of points above. Why should the fact that some people want to do machine code mean that they are whizz-kids? You do not need genius to understand machine code, just a good guide, and practice.

Essex may find he likes machine code, but the wants to stick with starting space invaders, for one with him all the best. Groups of people need serious critical mass to provide resources which benefit them all, and if they spread into a closed outlook that mass evaporates. Despite his anxious words, Mr. Ralph seems like a good example of someone who doesn't mind sharing his Dragon world.

Perhaps somebody could also find Paul Harris (December 1987 64 Columns) and work out how to connect up a 6845 VGA chip — it's only 128 pins! VGA.

Also important still, we must keep sharing our Dragon knowledge. See if we can put more facts on Dragonair 88.

Alan Miller  
c Carter Gardens  
Lynside  
Leighton Buzzard  
Bedk LU7 9XE

If people ask for hardware long enough, we give them hardware. See page 14. This is not exactly an upgrade, but it is interesting.

## A is for Edit

I WOULD like to point out that if while using EDIT mode you accidentally delete the line(s) you can return EDITing from scratch by pressing R (not while in insert mode).

James Bonfield  
7 Water End  
Huntingworth  
Sandy  
Bedk SG9 9DA

## Could be Norse

still have just started a new Dragon magazine for all Scandinavian Dragon users. Write to us, and we will send you the first issue free. The magazine is written in Norwegian.

Dragon (Blaad)  
c/o Gør-Haaland  
Lundheim  
6943 Rensdal  
Norway

## Late one September

If November comes, can September be far behind?

Thank you for your letter, and it does appear that the Royal Mail is at least over the September issue. November is surely delivered, but no September, alas.

I hope you are successful in your recall of duplicated deliveries and if you are, please

remember me. Strangely enough, I had no problems in July.

ACF Wilson  
50 Goodlands  
Orton Goldsey  
Peterborough  
Cambs  
PE2 6LZ

## Obscure dream

I note that, as Pete Gernand commented the month previously, you have found my writing difficult to decipher last month, so I note that my tip (Dream never starts Letters, December 1987) was mis-spelt nonsense. To give you the correction, I PFFFF!

The start of symbol-table address is held at locations 545F8A and 545F8B. You printed the 5 as an S! (Silly typo, F.R. No wonder 545F8A twice in this letter, instead of 545F8B as the correct location! Only a spot of tape-changer for me to correct! I caught that in the next of time, I hope. Shouldn't it be after this month... (Ed.)

Also, typed from Dream, one has to press BREAK then Q and then ENTER. You omitted the Q in the last sequence. It was a quiet day. No Qs at the end. (Silly. Sorry.) These two points may have lost readers confused. Before entering a Dream file which has been saved in this way, the CLARAK address must be set to the usual address (where the file was saved). This is so that the interpreter allocates this space for Dream so that loading thereafter will recover the state. For those who don't know, this is followed by the command CLARAK where M is the address of the highest memory you want BASIC to have access to, all higher RAM is reserved for Dream or whatever else (machine code, data, etc.).

Apologies for the lack of readability.

F.R. Elph  
c/o G. Dave Street  
Haltax  
West Yorkshire  
HD7 2LF

THANK you for the prompt correction, F.R. One thing — what happened to the Word Processor? I hope your Dragonair notified us. My best wishes for its recovery if it is.

## Words in reply

I WOULD like to reply to some of the points made in Dragon User November 1987 about my article Three Little Words.

To Mr Grayson: OS-9 allows suppression of LRs when a CIP is used. Some should not inhibit this. Some printers do not allow CR without LF. CIP switches or no.

To Mr. Burnshaw: one of DragonDOS's few facilities is that it creates backup files — it is a shortcoming of the application that it does not support this.

I am impressed by references to 'home maintenance packages' which I see as an attempt to mystify. I'm sure Windows is better than some sort of data maintenance packages, and how well-do-you think Windows compares to Windows or one of the more modern PC wordprocessors?

To Mr. Rafter: OS-9 is ROM seemed like a good idea to Tandy, who use it with the CoCo3. You would see nothing, since additional commands would be on another input device. I can't imagine there would be advantage being able to power up in OS-9 rather than booting from disk. You yourself advocate the use of the RAM disk.

Typical about the display is not what I prefer, but whether colour set, background/blackground are over-selectable.

I'm impressed by an 88 column display that's in monochrome only, or a RAM disk that offers a nearly 64K and is not battery backed. But the price put me off from the start.

Reviewer right justify is undocumented — thank you for pointing it out.

The reference to MFFDCE? puzzles me in retrospect, too. If any non-OS-9 user is still useless, Style reserves its own screenshot automatically or as the user specifies.

64K of RAM is generally inadequate for smooth OS-9 operation in my view. The time you save with multitasking is lost in the large number of disk accesses that are required.

Roger Merick  
350 Gravelly Lane  
Croydon  
Surrey  
CR2 5DH

## Projects please

I would like to add my name to the list of readers who are interested in hardware projects. I am sure that there is a lot that we can do to upgrade our Dragons. There must be people out there who produced lots of interesting add-ons, unlike myself who, given a soldering iron, can fly off in the second day.

# Microdeal to leave the Dragon Market

MICRODEAL have now confirmed that they will be pulling out of the Dragon market as of 1st January 1988.

They stress that the decision is financial and owing to diminishing returns and lack of suitable product to publish.

Commenting on the lack of good new material, John Symes said: "This time last year we had fifty reasonable tapes to look at. This year, we only had ten. There just isn't the material about any more."

The remaining stock of Dragon software has been taken aboard by Computape. When asked if the list would be maintained, a representative of Computape said that for the present they would concentrate on

selling from stock, but if the newer Microdeal games sold well enough, they would consider maintaining them.

Harry Massey of Computape recently told Dragon User that most of his business came from his company mailing list of about 5,000. Microdeal has confirmed their list Dragon mailing list is around 12,000. These figures are large enough to provide an active user base, but it is possible that the bulk of prospective customers have already bought all that they require from the present list, although Computape's price are bound to be attractive.

Crutcher, subject of the Dragon, John Symes said: "The Dragon has been an exceptionally good

machine for us and I would like to take this opportunity to thank all our suppliers in the past and hope that in the future when they return from Dragon computing that they move onto one of the machines we now publish for, namely the Commodore Amiga and Atari ST.

"My thanks also go to all the staff at Dragon User who have kept this specialist journal and the Dragon market alive for much longer than expected."

Dragon User thanks John Symes for his kind words. But on behalf of many dedicated users, may we add our hope that, by the time the last of the Dragon Users has retired, Microdeal will be publishing games for machines yet to be dreamed of.

## Fourth Plus

PROGRAMMER P. D. Smith has completed a Forté compiler for the OS-9 operating system. Based on the Fig-Forté standard, the compiler includes extra commands from other systems to increase its power. It compiles into pure machine code, giving it a fast running speed.

As well as standard Forté commands it includes facilities for the handling allowing file handling without complicated block handling procedures. There is a 30-page manual which has a section detailing the operation of the compiler for advanced programmers.

The compiler is available from P. D. Smith at University Hall, Birchwood Road, Penland, Cardiff CF5 5YB, price £14.95. The price includes software support and corrected updates if any bugs emerge.

## OS-9 User group is here

An introductory sheet for the OS-9 User Group has recently fallen into the hands of Dragon User. It has been warmly recommended by those who joined.

The group was set up to help Dragon OS-9 users in 1985, and has since expanded into additional hardware. It has a library with 40 volumes of public domain software, much of which is in Basic OS, Pascal and C, compatible at source level.

The group's bulletin is published on-line, called 'NewsDesk', allowing them to send software as well as news and advertisements. Members are supplied with advice on which to be concerned with the group.

The group is run on a non-profit basis, and memberships £25 per year. For further details contact the OS-9 User Group (European), 4 Reservoir Court, Llandudno, Gwynedd LL30 1TP.



## Power protection plug preferred

The SJPR.7 is a new power filter mains protection plug for computer equipment. Rated at 7 amps, the plug prevents voltage spikes from reaching equipment, and filters out radio frequency interference. Both these perils can be caused by electrical appliances switching, as well as by more obscure causes such as C-B radio and tele-

phone disturbances.

The SJPR.7 can be used to protect word processors, electronic tills, telephones and fax systems, videos and laser printers, as well as computers. The plug costs £14.95 plus VAT and 50p post and packing from Warwick Products, 60 Marsh Wall, West India Dock, London E14 9XJ. Tel. 01 528 5325.

## Disc drive deal

HARRIS Micro Software are proud to introduce a new range of fast, medium high capacity dual and single switchable 4860 track disc drives, complete with SuperDisk cartridges, for virtually the same price as their previous 40-track, single sided drives.

The drives, manufactured by 'Viglen', were originally designed for the BBC Micro, and have the advantage of connecting directly to SuperDisk or Curatadisc cartridges without modification.

Prices for complete systems

including cable, manual, cartridge and drive are £188.95 for the single 180 — 720K version, and £269.95 for the 360 — 1440K version. A reduction of £70 from the prices quoted covers the drive alone with cable.

Bob Harris can also supply a 'Viglen' Datadisk signal splitter for joining two separate drives together if required. The cable decides the priority of the drives, avoiding the need for configuration at the computer, and allows the drives to be switched at will. The Datadisk costs £15.



## Snip database cuts down paperwork

**Title:** Database  
**Supplier:** Pulser Software, 38 Powell, High Common, Stan, Chesham OL7 7NQ  
**Price:** £2.99 disc only

I CAN remember when my software collection comprised a handful of cassettes, which were left lying around by my old 32 and all I had to do was quickly glance through them to find the program I wanted. Now I have got three trays of cassettes and two boxes of discs and it used to take me hours to find a particular game or utility program.

Having typed up my cassette collection into several databases only to find that they were full before I had entered everything and then they used to print out one under the other wasting three quarters of my paper, I despaired of even starting to enter my entire disc collection.

Pulser Software have come up with the answer in Database, written by Bill Tarrant. On Sunday morning I loaded in the Help Screen and then followed with the Menu. Starting with the Create screen I was prompted to input whether I was using one drive or two.

As I have two drives I was able to leave my blank disc in Drive 2 while I inserted my program discs into Drive 1. After 20 minutes the information was on the working disc for all my 58 full discs. I then used the alphabetical sort, which took another 40 minutes. This may sound a long time but there were 647 different entries to be sorted, not including Menus, which are automatically left out by Database.

During this time you can go away and do other jobs while the computer carries on with the 'shell sort'. When sorted, I printed out the full list which

filled up three sheets of printer paper even though they were laid out over four columns. This has to be better than twelve sheets with most databases. Whenever I wish I can add to the database, and there are numerous options open to me. My printout is alphabetical with the name of the disc after it (you can use five characters including spaces to name your disc) but I would list out only Basic or only Binary if I wished.

A search can be made for any letters in it, eg PAC would find SPACE WARS and PAC-MAN, or you can list what is on any particular disc. All listings can be made either to the screen or the printer (it took me 1 hour 10 minutes to enter everything into the database and print out an alphabetical listing. Other databases would need keyring, and I do not know another database for the Dragon that could handle

25600 entries (800 entries for each letter and each number).

At £2.99 this has got to be a real snip and I would advise all disc owners to purchase it. I would certainly not recommend piracy at all! Quite apart from the usual reason, you are told on-screen to use the original only, and there is a reason: illegal copying will lead to the loss of your copy. Do not worry, no problems are incurred when you use the copy you have bought, and anyway, the original is duplicated on the other side of the floppy on which it comes.

This program has got to be worth five Dragons at this price.

Mike Stott



## A rolling stone gathers an Editor

**Title:** Boulder Crash 2  
**Supplier:** Giant Soft, 58 Moorcock Road, Sheffield S10 4QJ  
**Price:** £3.50

QUITE a while ago I bought Boulder Crash from Slaby Computer Games and soon afterwards won a copy of Stone Master 2 in a competition. These turned out to be virtually the same game, although myself and my two sons preferred the Slaby version. For those of you who do not know these games, the idea is to collect diamonds from around the screen of which you can only take a small part at any one time. When you have collected a set number you can then go through the Exit to the next screen. Boulders fall when you remove the earth under them, and these can crush you if you are not careful. On some screens you need to drop the boulders onto the Flappers who chase you to turn them into diamonds, although this does not

work with the Classics. Beware also of the Stone which grows and tries to engulf you. Power is also available at certain places, and this can be fine — at parsing creatures.

Boulder Crash 2 is a screen editor for the original so you will need both programs before you can start designing your own screens. The instructions tell you to LOAD the file program from Side B and that this will then copy and compile the original taking about 15 minutes, but this is a 'once-only' task. I cannot comment on the ease of the compiling as this review copy was already done for me. When the revised program is loaded, there are two extra options from the main menu. Pressing 6 takes you to the Screen Editor and it allows you to select infinite lives or any number from 1 to 100. Extra screens can also be loaded from Paul's tapes, and will convert to these later.

When I first saw the rolling screen, I nearly gave up as the

graphics are abysmal but forget about that, as once you go back to playing mode everything appears exactly the same as in Boulder Crash. I have tried several of these 'edit your own screen' ideas and I must admit that this is about the best I have seen, although it is not really my cup of tea. The instruction sheet that I received as my comprehensive, and it is impossible to design your own screen and play on it within a very few minutes, but do not press reset if you are trapped somewhere, as I soon found out (I promise to read the instructions next time!).

Back to the extra screen provided with the program. On the whole they are very good, although several times my young seven-year-old has trapped himself between two walls and two boulders, and we have had to wait for the time to run out before we could start again due to not being able to reset. Still, I must admit that

Paul Bergen's screens are certainly better than mine, which are either too easy or just plain impossible. Any screen you devise yourself can be saved, so to speak so that you can play again whenever you wish, and there is also a built-in verify facility to ensure that they have been saved properly. The keys used in the edit mode are very logical. For instance, D is for diamond, W is for wall and SPACE is for a space.

To sum up, if you liked Boulder Crash and would like more screens or would like to design your own then I can recommend it. If you have not seen the original then contact H & A J Preston and you could very soon be enjoying this one as well. If the price were under 2.50 I would give it four dragons, but as Boulder Crash is not new I will give it three.

Mike Stott



# REBOOT

About the same time as we were publishing Auto-BOOT last month, another short listing emerged from the pile to shed some light on the non-operational Dragon-DOS BOOT. Then a revision on that listing turned up. Thus a revision of the original turned up . . . life isn't static, so we'll share them.

**Mike Hides** presents his solution to the *BOOT* dilemma

THE command BOOT allows one program or sector to be loaded and auto-run without the need to enter its name. Unfortunately, no method is provided into-DOS to enable users to make use of this facility. The following listing rectifies this, and allows one machine code program on a disc to be BOOTed. The listing starts with Dragon-DOS and SuperDOS, but not with CumanaDOS.

To use the program enter it using an assembler (written using Aldraw) and omit the line numbers. You need to enter the HEXC address of the program you intend to BOOT in line 1, and the file name in lines 33 and 35. Assemble the program and if free from errors, place a blank formatted disc in drive 1. BREAK K (in Aldraw) will execute the assembled program and write it to your disc. Now copy the program you have decided to BOOT onto

the same disc and all should now work.

If you wish to BOOT a basic program try the following modifications:

- a) delete lines 1 and 31
- b) after line 30 insert the following three lines:

```
JBR 00020
JBR 00070
JMP 00000
```

- c) make sure you changed the B14 to B45 in lines 35 and 35.

## How it works

On entering the BOOT command the DOS loads sector 016 of track 0 from the disc into memory starting at location 0000. It then checks to see if the first two bytes

contain the ASCII codes of "00", and if so execution is automatically started at 0000.

Lines 2 and 3 set the origin of the program to 0000 and put "00" into the first two bytes. Lines 4 to 15 do a CLS and put the file on the screen. The next part is to lower the program in memory so that it does not clash with the program it will be BOOTing. Lines 9 to 15 accomplish this movement. The filename is then moved into location 000 and LOADED using the routine at 00400 (lines 16 to 17). Finally, lines 22 to 31 reset a Ram lock, ensure the ram reset vector is correct, switch off the disc drive motors and jump to the execution address of your program.

The final part of the program (lines 38 to 40) writes all the above onto track 0 of the disc starting at sector 3. The label %START tells the assembler where to begin the evaluation of this part of the program.

### Listing one

```

1  EQUATE BOOT 0000000000000000 04 000 000
2  EQUATE BOOT 0000000000000000 04 000 000
3  EQUATE BOOT 0000000000000000 04 000 000
4  EQUATE BOOT 0000000000000000 04 000 000
5  EQUATE BOOT 0000000000000000 04 000 000
6  EQUATE BOOT 0000000000000000 04 000 000
7  EQUATE BOOT 0000000000000000 04 000 000
8  EQUATE BOOT 0000000000000000 04 000 000
9  EQUATE BOOT 0000000000000000 04 000 000
10 EQUATE BOOT 0000000000000000 04 000 000
11 EQUATE BOOT 0000000000000000 04 000 000
12 EQUATE BOOT 0000000000000000 04 000 000
13 EQUATE BOOT 0000000000000000 04 000 000
14 EQUATE BOOT 0000000000000000 04 000 000
15 EQUATE BOOT 0000000000000000 04 000 000
16 EQUATE BOOT 0000000000000000 04 000 000
17 EQUATE BOOT 0000000000000000 04 000 000
18 EQUATE BOOT 0000000000000000 04 000 000
19 EQUATE BOOT 0000000000000000 04 000 000
20 EQUATE BOOT 0000000000000000 04 000 000
21 EQUATE BOOT 0000000000000000 04 000 000
22 EQUATE BOOT 0000000000000000 04 000 000
23 EQUATE BOOT 0000000000000000 04 000 000
24 EQUATE BOOT 0000000000000000 04 000 000
25 EQUATE BOOT 0000000000000000 04 000 000
26 EQUATE BOOT 0000000000000000 04 000 000
27 EQUATE BOOT 0000000000000000 04 000 000
28 EQUATE BOOT 0000000000000000 04 000 000
29 EQUATE BOOT 0000000000000000 04 000 000
30 EQUATE BOOT 0000000000000000 04 000 000
31 EQUATE BOOT 0000000000000000 04 000 000
32 EQUATE BOOT 0000000000000000 04 000 000
33 EQUATE BOOT 0000000000000000 04 000 000
34 EQUATE BOOT 0000000000000000 04 000 000
35 EQUATE BOOT 0000000000000000 04 000 000
36 EQUATE BOOT 0000000000000000 04 000 000
37 EQUATE BOOT 0000000000000000 04 000 000
38 EQUATE BOOT 0000000000000000 04 000 000
39 EQUATE BOOT 0000000000000000 04 000 000
40 EQUATE BOOT 0000000000000000 04 000 000

```

## Dummy run

The previous program which allowed the use of the BOOT command to LOAD and RUN any file on a disc codes has a potential problem: this is how to prevent the DOS writing over the code which resides on track 0 sector 3 onto the disc. In May 1987 issue Paul Dagleish gave details of how the directory track is organised, and using this information the code can be protected.

The method is to insert a dummy entry which occupies the appropriate part of the disc and cannot be deleted. After putting the BOOT code onto an empty disc, run the Basic program in Listing two.

Paul Dagleish described how the directory track was arranged into slots 25 bytes long starting at sector 3 on tracks 16 and 20. This program writes directly to these tracks and sets up a file called leave ALLOCATE which appears to occupy the appropriate position (track 5, sector 3).

Line 30 reads the disc directory track using the SPREAD command and stores the current information in two string variables C0 and B0. Line 40 removes the information for the first directory slot from C0. Lines 50 to 110 read in the information for the dummy file from the DATA statements and a checksum is ascertained if everything is OK. It is now vitally important that only the disc with the BOOT information is in drive one, otherwise, when the rest of the program is run, directory will be incorrectly altered. Lines 120 to 150 give you the chance to abort. Finally the WRITE command is used in lines 160 and 160 to write the information onto the disc. By DIR to see the result. This dummy file cannot be deleted by the normal DOS commands and should protect your BOOT code.

```

Listing two
10 CLEAR 400
20 AA = "" : BB = "" : CC = "" : CC = 0
30 READ:1,20,3,CC,BB
40 CC = RIGHT(CC,10)
50 FOR A = 1 TO 20
60 READ:20
70 AA = AA + CHR$(VAL("6H" + BB))
80 CC = CC + VAL("6H" + BB)
90 NEXT A
100 IF CC > 1303 THEN PRINT " DATA ERROR " : END
110 AA = AA + CC
120 CLS : PRINT " ARE YOU READY TO CONTINUE?"
130 PRINT:PRINT"ENTER Y or N"
140 Q0 = INKEY$:IF Q0 = "" THEN 140
150 IF Q0 <> "Y" THEN END
160 PRINT:PRINT"PUT DISK WITH BOOT
FILE IN", "DRIVE 1 AND PRESS ANY
KEY"
170 REBO41194
180 WRITE:1,10,3,AA,BB
190 WRITE:1,20,3,AA,BB
200 DATA 02,6C,65,61,76,65,41,40,4F
210 DATA 4E,45,21,00,07,01,00,00,00
220 DATA 00,00,00,00,00,00,00,FF

```

### Julian Osborne has leapt, quite independently, to the same conclusion

SINCE the publication of my article *Auto BOOT* on the *BOOT* command in the October 1987 issue of *Dragon User*, it has been noticed that under certain circumstances the *BOOT* routine causes problems, where it occupies the first sectors of track 0 and can be overwritten by the DOS saving program on these sectors — not an ideal situation!

To get around this problem, the following program (to allocate) the first 16 sectors of track 0 by altering the directory bit maps say that these sectors are occupied.

The procedure to set up a *BOOT* disc would then be:

- 1) *DISKINIT* a fresh disc
- 2) Run the program given in **listing three** (20K) should then show 17008 free bytes
- 3) Run the *BOOT* program from last month's article. This should then create

the *BOOT* code on track 0 as normal, but it will prevent DOS from using track 0 for any more programs. This routine works because *READ* and *WRITE* do not refer to the directory contents when reading or writing to the disc.

The bit map for *DragonDOS* resides on track 20 sector 1 (for double sided 80-track drives it also takes up track 20 sector 2), and it represents the free space on a disc as follows:

Each of the bits in sector 1 or track 20 represents a disc sector, the first bit represents track 0 sector 0, the next bit represents track 0 sector 1 and so on.

If a bit is set to 1 then *DragonDOS* assumes the sector is free and if it is set to zero then it assumes that the sector is in use.

The *RESERVE* program reads the bit map into two strings (AA and BB) and using

the *RIGHT\$* function removes the first two bytes (as a 255-decimal which is 11111111 in binary). These two 'null' bytes (0 decimal is 00000000 binary) are added to the front of the string and the string is then written back to the directory track 20, and also to track 16 (file reserve directory track). The two 'null' bytes represent 16 bits which in turn represent the first 16 sectors of the disc (track 0, sectors 1 to 16) thus fooling *DragonDOS* into believing that they are occupied.

My apologies for not spotting this problem sooner but hopefully this program (and explanation) should solve it!

Anyway, as before I am willing to attempt to answer any queries about either the *BOOT* routine or the *RESERVE* program which may arise. My address is 8 Halseon Road, Malvern, Bristol BS29 2JA.

#### Listing three

```

10 REM RESERVE BOOT SECTORS ON DISK
20 REM RESERVES SECTORS 1 TO 16 ON TRACK 0
30 CLEAR 1000
40 READ:1,20,1,AA,BB : REM READ BIT MAP SECTOR
50 CC=RIGHT(BB,126) : REM CHOP OFF FIRST 2 BYTES
60 AA=CHR$(10)+CHR$(10)+CC : REM REPLACE WITH 2 NULL BYTES
70 WRITE:1,20,1,AA,BB : REM WRITE BACK TO BIT MAP ON DIRECTORY TRACK
80 WRITE:1,16,1,AA,BB : REM WRITE BACK TO BACKUP DIRECTORY TRACK
90 CLOSE
100 BIR : REM SHOULD SHOW 17008 FREE BYTES REMAINING

```

# BREAKing the '64

*Martyn Ammitage stops the 64's printer in its tracks*

In Pam O'Arcy's article *Exon Made Easy* in the February issue of *Dragon* (see the mention) the fact that the Dragon 64 will not allow the **PRINTED** routine to be interrupted by pressing the **BREAK** key as on the 32. She also says that the bug hasn't been mentioned to date. Well, unfortunately, the problem is a bug, but more of a case of poor forethought by the programmer. In the 32 ROM the **BREAK** key is checked for quite regularly, and if pressed a return to basic is the result. When the ROM for the 64 was written an extra routine was thrown in. This extra routine simply looks at the state of the keyboard, and if any key is pressed then the routine goes into a loop, waiting until the keyboard is clear, ie no key pressed. Pressing any key, including the **BREAK** key will cause the machine to wait for the key to be released. Once the key is released then a check is made to see if the **BREAK** key has been pressed, and if it has then the usual return to basic will ensue. Well then, why is it impossible to abort output to the printer? The answer is as said previously, poor forethought. Between the 'wait while key pressed'

routine and the 'check for **BREAK**' routine there are approximately 40 machine code instructions to be executed, at an average of four microprocessor cycles per instruction this gives a period of about 160 mps cycles or approximately 160 microseconds (0.000160 seconds), not a lot of time to either release and re-press the **BREAK** key, or to hit the **BREAK** key at the right time. Too early and the wait routine is entered, too late and the **BREAK** check is missed.

The problem is very infuriating, as the only way out is to wait for the **RESET** etc, to end naturally or press the **RESET** key. My answer to the problem is to put the computer into map 1 (all RAM) copying the ROM across and then to patch out the 'wait while key press' routine. The short machine code program that follows accomplishes this, and it also copies the CARTRIDGE memory (AHC0000 — AHFFFF) so that should a cartridge or cartridge/program be present then it can still be used as normal.

The machine code generated by the assembler listing sits in the first page of

graphics if *Dragon* Des is present, or the second graphics page if the Des isn't present. The code is position independent and so can be relocated anywhere in memory if you should need to do so, for example if you have something in the graphics memory that you wish to preserve.

If you don't own an assembler (shame on you) then the basic listing will install the machine code, which will then require saving, the addresses being:

```
Start AHC000
End AHC04
Exec AHC000
```

Remember that when the Dragon 64 is in map type 1 (all RAM) pressing the reset button causes the machine to revert to map type 0 (RAM and ROM), and doing so will cause the patch to be ineffective. If you do happen to press the reset button while in map type 1, you can type **POKE AHC000,1 ENTER**, which will put the Dragon into 64K RAM mode and the patch should be intact. If not then you will have to reload the program and execute it.

## Listing one

		ORG	\$C00	
		PUT	\$C00	
0C00	3401	PSHS	CC	SAVE CC REG.
0C02	1A50	ORCC	#250	DISEABLE IRQ'S
0C04	868000	LDA	\$8000	FIRST ROM BYTE
0C07	3402	PSHS	A	SAVE 'A'
0C09	7A8000	DEC	\$8000	ALTER FIRST ROM BYTE
0C0C	868000	LDA	\$8000	GET FIRST ROM BYTE
0C0F	A180	CHFA	,8+	COMPARE WITH OLD VAL
0C11	2612	BNE	RAM	NOT SAME-NAPI (RAM)
0C13	8E8000	LDE	\$8000	START OF ROM
0C16	D1FFDE ROM	STA	\$FFDE	SET MAP 0 (ROM)
0C19	A684	LDA	,1	GET ROM BYTE
0C1B	D1FFDF	STA	\$FFDF	SET MAP 1 (RAM)
0C1E	A780	STA	,2+	STORE IN RAM, INC X
0C20	8CFF00	CHPE	\$FF00	LAST ROM BYTE?
0C23	26F1	BNE	ROM	NO SO GET NEXT BYTE
0C26	867E RAM	LDA	#7E	OPCODE FOR JMP
0C27	B78000	STA	\$8000	RESTORE RAM BYTE

0C2A	#B0CFD	LDR	#B0CFD	SEND TO PRINTER ADDR
0C2B	B7BECF	STA	#B0CFD	OPCODE FOR JMP
0C2C	BFB0CE	STX	#B0CFE	PATCH OUT WAIT CODE
0C2D	15401	PULS	OC_PC	RESTORE 180'S RETURN

```

10 CLS
20 FOR I=8000 TO 80C4
30 READ A$
40 A=VAL("4H"+A$)
50 POKE I,A:CS=CS+A
60 NEXT
70 IF CS<8000 THEN PRINT "ERROR IN DATA"-END
80 PRINT"CODE INSTALLED..." :PRINT"REMEMBER TO SAVE IT"
90 DATA 34,1,1A,3C,3E,80,0,34,2,7A,80,0
100 DATA 84,80,0,A1,ED,2E,12,8E,80,0
110 DATA B3,FF,DE,A6,84,B3,FF,DF,A7,80
120 DATA 8C,FF,0,2E,F1,86,7E,B7,80,0
130 DATA 8E,DC,FD,B7,DE,CD,BF,8E,CE
140 DATA 35,81

```

## Winners and Losers

Every month  
Gordon Lee will  
look at some of the most unusual

THE competition was not in itself difficult, although many readers submitted up to six minor problems. To quote Phil Kaplan: "A very short program is all that is necessary . . . and to prove frequent *linguistic* was what he attacked. Paul Menzies worked along similar lines (*Spelling bee*). By taking the digits from groups of five (tailor the computer's mathematical capacities) the running time was effectively reduced five fold."

Almost without exception all of the winners adopted a version of these programs. However, this month, the non-winners also came up with a number of

surprise. An identical 5221775205 from Randy Longshore and E. A. Newman: the explanation was simpler. The programming and mathematics were perfectly correct, but they had both counted the number of digits from the decimal point and not from the left. A technical footnote.

Surprised also from Dave Lindber and Denis O'Malley, who submitted 436769628 and 520778874 respectively. One is almost the reverse of the other. Closer examination reveals that the first five digits of Mr. Mulloy's number, reversed, is the same as the first five digits of its

answer, while the last five digits of his  
Lardner's number is almost the same as  
the start of the answer. I suspect that the  
different digit is just a copying error and  
that both are five digits too soon in the  
series. Remember that though there are  
304343 cases in the division, there are only  
304333 digits in the result. This difference of  
five is due to six zeros needing to be added  
before the first digit of the answer appears  
— that is, the division will begin 111111  
304343. The number of digits in the answer  
is almost five short of the number of cases

A good safeguard against this sort of error is to print out (skipped portions of the) result. I have compiled the result into lines of 70 characters. The digits show whether the beginning, middle and end of the list appear. To determine where to find a specific digit, divide the number of the digit by 70. The SNOBOL digit is located by:

Copyright © 2004 John Wiley & Sons, Ltd.

The required digit (which begins the sequence of our answer) is therefore found on line 101, at the 101st position.

1000

```

[0] 0.000000e+00
[1] 0.00000000e+00 0.00000000e+00 0.00000000e+00
[2] 0.00000000e+00 0.00000000e+00 0.00000000e+00
[3] 0.00000000e+00
[4] 0.00000000e+00
[5] 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
[6] 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
[7] 0.00000000e+00

```



### Listing three

```

0001      * LISTING 3
0002      *
0003      * JORFREQ (FILENAME)
0004      *
0005      * EXAMPLE OF POSITION DEPENDENT
0006      * MACHINE CODE
0007      *
0008      *
0009      * USING DREAM ASSEMBLER
0010      * AFTER CLEARING ADDRESS
0011
0012
0013 00000000 00 JOR FREQ
0014 00000000 JOR HARRY
0015 20
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027 20
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300
0301
0302
0303
0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
0321
0322
0323
0324
0325
0326
0327
0328
0329
0330
0331
0332
0333
0334
0335
0336
0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358
0359
0360
0361
0362
0363
0364
0365
0366
0367
0368
0369
0370
0371
0372
0373
0374
0375
0376
0377
0378
0379
0380
0381
0382
0383
0384
0385
0386
0387
0388
0389
0390
0391
0392
0393
0394
0395
0396
0397
0398
0399
0400
0401
0402
0403
0404
0405
0406
0407
0408
0409
0410
0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481
0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529
0530
0531
0532
0533
0534
0535
0536
0537
0538
0539
0540
0541
0542
0543
0544
0545
0546
0547
0548
0549
0550
0551
0552
0553
0554
0555
0556
0557
0558
0559
0560
0561
0562
0563
0564
0565
0566
0567
0568
0569
0570
0571
0572
0573
0574
0575
0576
0577
0578
0579
0580
0581
0582
0583
0584
0585
0586
0587
0588
0589
0590
0591
0592
0593
0594
0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
0650
0651
0652
0653
0654
0655
0656
0657
0658
0659
0660
0661
0662
0663
0664
0665
0666
0667
0668
0669
0670
0671
0672
0673
0674
0675
0676
0677
0678
0679
0680
0681
0682
0683
0684
0685
0686
0687
0688
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700
0701
0702
0703
0704
0705
0706
0707
0708
0709
0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727
0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739
0740
0741
0742
0743
0744
0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
0775
0776
0777
0778
0779
0780
0781
0782
0783
0784
0785
0786
0787
0788
0789
0790
0791
0792
0793
0794
0795
0796
0797
0798
0799
0800
0801
0802
0803
0804
0805
0806
0807
0808
0809
0810
0811
0812
0813
0814
0815
0816
0817
0818
0819
0820
0821
0822
0823
0824
0825
0826
0827
0828
0829
0830
0831
0832
0833
0834
0835
0836
0837
0838
0839
0840
0841
0842
0843
0844
0845
0846
0847
0848
0849
0850
0851
0852
0853
0854
0855
0856
0857
0858
0859
0860
0861
0862
0863
0864
0865
0866
0867
0868
0869
0870
0871
0872
0873
0874
0875
0876
0877
0878
0879
0880
0881
0882
0883
0884
0885
0886
0887
0888
0889
0890
0891
0892
0893
0894
0895
0896
0897
0898
0899
0900
0901
0902
0903
0904
0905
0906
0907
0908
0909
0910
0911
0912
0913
0914
0915
0916
0917
0918
0919
0920
0921
0922
0923
0924
0925
0926
0927
0928
0929
0930
0931
0932
0933
0934
0935
0936
0937
0938
0939
0940
0941
0942
0943
0944
0945
0946
0947
0948
0949
0950
0951
0952
0953
0954
0955
0956
0957
0958
0959
0960
0961
0962
0963
0964
0965
0966
0967
0968
0969
0970
0971
0972
0973
0974
0975
0976
0977
0978
0979
0980
0981
0982
0983
0984
0985
0986
0987
0988
0989
0990
0991
0992
0993
0994
0995
0996
0997
0998
0999
1000

```

assembler itself occupies \$5000+ in memory, the assembler may contain the option to generate code as if it were in the required area. Using Dream, this can be achieved using the ORG (=ORGAN) and PUT directives. Listing four says generate code as if it is positioned at \$7000 (\$7000) but for the assembler, place it in its normal workspace (\$0000). If the PUT had been omitted, the generated code would be placed directly into address \$7000+ corrupting the Dream program with instructions, but no doubt catastrophic results. ORG and PUT are what are known as assembler DIRECTIVES. They are not machine code instructions, but the assembler's own means of allowing programmers to give it commands and this is one of the areas of greatest difference in assemblers.

Having assembled the code in listing 4, that code is now fixed to execute successfully only from address \$7000 in the machine. Just before I leave that, as this may well be an area that newer machine coders may have difficulty understanding with their assemblers, a word on saving machine code assembled in a different place from where it will run (at least, for Dream assemblers, but perhaps it is also appropriate to other assemblers).

After assembly, the generated code is actually sitting at address \$0000+ in the machine. Its length is \$0000-\$0000 inclusive (first and last byte generated in the listing) = 8 bytes. If there needs to be saved with the likes of `CLASMEM JORFREQD", $H000, $H0000, $H0000`

### Listing five

```

0001      * LISTING 5
0002      *
0003      * JORFREQ (FILENAME)
0004      *
0005      * EXAMPLE OF POSITION DEPENDENT
0006      * MACHINE CODE
0007      *
0008      *
0009      * USING DREAM ASSEMBLER
0010      * AFTER CLEARING ADDRESS
0011
0012
0013 00000000 00 JOR FREQ
0014 00000000 JOR HARRY
0015 20
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027 20
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300
0301
0302
0303
0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
0321
0322
0323
0324
0325
0326
0327
0328
0329
0330
0331
0332
0333
0334
0335
0336
0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358
0359
0360
0361
0362
0363
0364
0365
0366
0367
0368
0369
0370
0371
0372
0373
0374
0375
0376
0377
0378
0379
0380
0381
0382
0383
0384
0385
0386
0387
0388
0389
0390
0391
0392
0393
0394
0395
0396
0397
0398
0399
0400
0401
0402
0403
0404
0405
0406
0407
0408
0409
0410
0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481
0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529
0530
0531
0532
0533
0534
0535
0536
0537
0538
0539
0540
0541
0542
0543
0544
0545
0546
0547
0548
0549
0550
0551
0552
0553
0554
0555
0556
0557
0558
0559
0560
0561
0562
0563
0564
0565
0566
0567
0568
0569
0570
0571
0572
0573
0574
0575
0576
0577
0578
0579
0580
0581
0582
0583
0584
0585
0586
0587
0588
0589
0590
0591
0592
0593
0594
0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
0650
0651
0652
0653
0654
0655
0656
0657
0658
0659
0660
0661
0662
0663
0664
0665
0666
0667
0668
0669
0670
0671
0672
0673
0674
0675
0676
0677
0678
0679
0680
0681
0682
0683
0684
0685
0686
0687
0688
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700
0701
0702
0703
0704
0705
0706
0707
0708
0709
0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727
0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739
0740
0741
0742
0743
0744
0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
0775
0776
0777
0778
0779
0780
0781
0782
0783
0784
0785
0786
0787
0788
0789
0790
0791
0792
0793
0794
0795
0796
0797
0798
0799
0800
0801
0802
0803
0804
0805
0806
0807
0808
0809
0810
0811
0812
0813
0814
0815
0816
0817
0818
0819
0820
0821
0822
0823
0824
0825
0826
0827
0828
0829
0830
0831
0832
0833
0834
0835
0836
0837
0838
0839
0840
0841
0842
0843
0844
0845
0846
0847
0848
0849
0850
0851
0852
0853
0854
0855
0856
0857
0858
0859
0860
0861
0862
0863
0864
0865
0866
0867
0868
0869
0870
0871
0872
0873
0874
0875
0876
0877
0878
0879
0880
0881
0882
0883
0884
0885
0886
0887
0888
0889
0890
0891
0892
0893
0894
0895
0896
0897
0898
0899
0900
0901
0902
0903
0904
0905
0906
0907
0908
0909
0910
0911
0912
0913
0914
0915
0916
0917
0918
0919
0920
0921
0922
0923
0924
0925
0926
0927
0928
0929
0930
0931
0932
0933
0934
0935
0936
0937
0938
0939
0940
0941
0942
0943
0944
0945
0946
0947
0948
0949
0950
0951
0952
0953
0954
0955
0956
0957
0958
0959
0960
0961
0962
0963
0964
0965
0966
0967
0968
0969
0970
0971
0972
0973
0974
0975
0976
0977
0978
0979
0980
0981
0982
0983
0984
0985
0986
0987
0988
0989
0990
0991
0992
0993
0994
0995
0996
0997
0998
0999
1000

```

### Listing four

```

0001      * LISTING 4
0002      *
0003      * JORFREQ (FILENAME)
0004      *
0005      * EXAMPLE OF POSITION DEPENDENT
0006      * MACHINE CODE
0007      *
0008      *
0009      * USING DREAM ASSEMBLER
0010      * AFTER CLEARING ADDRESS
0011
0012
0013 00000000 00 JOR FREQ
0014 00000000 JOR HARRY
0015 20
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027 20
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
02
```

# Listing six

- \*
  - \* YELLOWLOB (FILENAME)
  - \*
    - \* THE YELLOW BLOB - PAGE 5A
    - \* FROM "DRAGON MACHINE CODE"
    - \* BY JONERACINGILL (SHIVA)
    - \*
      - \* TYPED IN AS PER THEIR LISTING

```

44B4 LDH #0400 10C B4 00
      STX 4400 10F 64 00
      LDY #01FF 118 0C 01 FF
      LDA #02 105 00 00
      CLR 4403 11F 64 02
      CLR 4403 11F 64 03
      STA 1 1 1 1 1 1 1 1
      LEAY -1,Y 111 3F
      BNE CLEAR 126 F4
      LDA #FF 105 9F
      LDH 4400 10E 64 00
      STA 1 1 1 1 1 1 1 1
      JGR 0004 10D 00 05
      BGR 00D 127 00
    
```

```

      ORPA 00E 101 0E
      BNE DOWN 126 05
      JGR UP 10D 64 5D
      BBA K0D 120 F2
      DOWN ORPA 00A 101 0A
      BNE LEFT 126 05
      JGR HDOWN 10D 64 5A
      BBA K0D 120 09
      LEFT ORPA 000 101 00
      BNE RIGHT 126 05
      JGR LEFT 10D 64 55
      BBA K0D 120 08
      RIGHT ORPA 00F 101 0F
      BNE 126 05
      JGR RIGHT 10D 64 56
      BBA K0D 120 D7
      BREAK ORPA 003 101 03
      BGR END 127 02
      BBA K0D 120 D1
      END RTS 129
      UP RTS 129
      HDOWN RTS 129
      LEFT RTS 129
      RIGHT RTS 129
    
```

characters page of the Dragon manual, Appendix A — and we should be familiar with the 'get keypress' ROM call (JGR 0000) to read.

The working visual result is not very exciting — green screen with yellow blob top left and the program simply sitting

there until break is pressed — but the prospect of what you have achieved if you have used this listing (and not your own routine), particularly if your assembler is not geared to letting you generate code at break, should be very exciting indeed to you (in which case, you are probably

beyond reading this series at all). The only other thing that I will say is that if your assembler is like mine, you will need to add in an awful lot of dollar signs to get it to generate code as this listing suggests (shown as comments in the source lines). Good luck!

## Crossword

The third month of the Dragon Crossword. We have the results from the first crossword now, and the lucky winners whose correct entries were selected from the editor's list were Paul Simpson, who wanted an up to date *Telnet* software, and Mrs. H. Clarke, who gave us a list of choices, most of which are delicious! We will see what we can do — so promises.

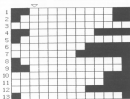
There will be a couple of free tapes from the Editor's Magic Bottomless Box for the first correct entries to reach us each month. You can even try telling us which tapes you'd like in an ideal world. It all depends on what we can find.

And you don't have to cut up your Dragon Over editor — heaven's forbid! Entries can be written out on a photostat or a plain piece of paper, as long as we can read them. The ingenuity practiced by DU readers to avoid mutilating their precious paper-work never ceases to astound.



by Terry and Derek Probyn

All this month's answers are names of Dragon software. When the crossword is complete, the columns marked with an arrow will spell out a phrase.



1. Plant this in noise (the devil) (5,6)
2. Grinder runs around the lines (4,6)
3. American President's formal place? (8,4)
4. He should not snatch! (7)
5. Tax for cowart whinpoat (8,6)
6. I mounted trouble when the pit was closed (8,3)
7. Organise some resistance — don't sit on it (7)
8. Britain's black gold? (5,3,3)
9. Hasty Huzzahs! (3,7)
10. If he did this slowly, he could not be seen (3,6)
11. He looks for holes in a chart venue (4,6)
12. ET met a broker on South African journey (8)
13. The red beast visits a castle with a bishop (5,5)



# Expert's Arcade Arena

HELLO, and welcome to this month's straight-faced column. Thanks for your letters, but I must at least twice as many entries to the software survey as I've received — it's no contest with just my entry! (Bilateral note: the Expert stacks you. Owing to the double time scheme operating here, you won't even have seen January's column at the time of writing.)

With the new year comes a new format for hints and cheats in the Arena. From now on they will have boxes round them, so that they are easier to spot when you search through your Dragon Users looking for a particular tip. The first words in each box will tell you to which game it refers and whether it's a hint or cheat.

Got that? Good. Let's go!

## Airball cheat

Load the game using Paul Bugen's program G (Arcade Arena, September 1988) and co-ordinated with the line:

20 PORE 3241,LIVES

where LIVES is an integer from 1 to 255.

First off the top of the pile this month is this handy PORE for Airball which has been passed up and down the country and eventually reached me. The earliest origin of the PORE I know is Mr. T. Williams. Thanks, Tom. Thanks also to James Bonfield who sent in the same PORE (but wasn't the first. Bad luck everyone who voted for

Airball to be hacked in the survey, but don't worry; your votes still count. The PORE's quite useful, but how about a PORE to stop the ball deflating?

Sticking to Airball for a minute, I've a few ideas from people stuck with two screens. For one, I provide a solution, but the other I leave to you. The screen in question is described as SE, SE, SW, SW, SW, NW. SW twice are compass directions from the starting screen. It's a screen with a high wall and some vicious spikes, and is stopping people's progress with the game. Please send your solutions to the usual address.

## Airball hints

To get past the screen which is SE, SE, SW, SW, SE, SE from the start screen, you must select eight directional movements. This enables you to jump into the room diagonally from the room below, and land on the two pedestals, not the deadly floor tiles. You must press two keys together (as well as Bounce) to obtain this diagonal movement, and jump straight towards the bottom of the screen.

Everyone out there who's having trouble getting from one screen to another in time, don't forget the keys S and F will alter the speed of the ball.

James B. also provides a few more POREs, some of which have already been printed and some of which I'll feature sometime in the future. To show some appreciation of

his work, perhaps someone can help him with a query. He would like to know what The Oracle is. No, not ITN's telecom service, but a utility which seems to leave messages such as "The Oracle is in Ridd", "Enter key" and "Program loads too low in RAM" hidden within the coding of a large number of Microdeal games. Any ideas? You know the address.

Before I go, I've got here a few advance details for a couple of new games for 1989. Firstly, Crazy Foss 2, which isn't available as I write, but will be by the time you read this. I haven't got copies of the new or old versions, but the letter says that it's now greatly improved, including different colours for each team, and options to allow a variety of games by changing the duration and speed of each game. The other new game I've been told about is the latest from Paul Bugen. This time he's stuck his routing clues into Space Monopoly. Space Monopoly 4 apparently includes a whole lot of new features, including holdings advice, controlling interests lists, psychic options, options to influence the creation of the universe and many others. The most interesting is an option to play against the computer, from 2 to 4 players. I gather that Paul is now trying to work out how he can sell the game without upsetting Microdeal. (Paul should be looking at the news page, and having a quiet word with Harry Massey.)

Once again, the end is nigh. Apologies for a short column this month, but we all want a couple of days off at Christmas. I'll say Goodbye now; see you next month.

## Adventure Contact

**Adventure:** Starship Destroyer, Dungeon Destroyer, Wild West Destroyer.  
**Problems:** Hints wanted.  
**Name:** Paul Lockington.  
**Address:** 40 Solihull Avenue, Park St, Helms, Macclesfield, Warrs 93Q.

**Adventure:** Galagun.  
**Problems:** How to get on his toes.  
**Name:** Gary Hunt.  
**Address:** 71 Fernside Road, Luptonstone, London E11 3DH.

**Adventure:** Sea Queen's Mystery of the Juro Star.  
**Problems:** Where's the mirror? Where is the Ruby hidden on the ship?  
**Name:** Craig Dillon.  
**Address:** 24 Glen Kinglass Road, Overton, Bournemouth, BH6 6BW.

**Problems:** Poets predictor program needed, must be able to alter teams in divisions and send division results to printer.  
**Name:** B. Tozer.  
**Address:** 10 Diamond Ave, Beverley High Road, Hull, Tel. 0482 443485.

## Communication

### Communication

Write down your problems on the coupon below (make it as brief and legible as possible) together with your name and address and send it to Communication, 1213 Little Margaret Street, London WC2H 7HT.

**Problem** .....

.....

.....

.....

.....

**Name** .....

**Address** .....

.....

.....

# Dragon drives direct

C.J. Walton describes an experimental interface project for DC motors

In order to link the Dragon to motors to drive them, it is necessary to use some form of interface. There have been a number of articles in *Dragon User* detailing experiments in various aspects of interfacing, and also on the production of interface units. Such units generally have mechanical relays built into them, or may be connected to suitable relays so that external devices, such as lights, heaters, motors, etc. may be switched on and off.

It is usual practice in the area of robotics to use stepper motors to obtain accurate positioning of arms and grippers. Such

control would then enable the buggy to be operated. The interface unit obtained was that produced by HCU Electronics of Hail, whose address is included at the foot of this article. It is the type C401, and at the time of purchase cost £14.95 inclusive. It could be as well to enquire about cost before placing an order. The interface was reviewed in the December 1984 edition of *Dragon User*. It connects to the printer socket of the Dragon and has eight lines available to connect to external circuits. The interface may be programmed in Basic using the PRINT statement or via

## Using the interface

Interface C401 is built on a small printed circuit board and connected to the computer by a 28-way ribbon cable and plug. To enable connections to be made easily to its outputs, the board was mounted in a small circuit box with a heta made at the side for the ribbon cable. The box was obtained from the local Tandy store (type 270-2842 metal mini-rack enclosure, price £2.60 in a sale) and the eight line outputs and the common connection were taken to 4mm sockets which were mounted on the top of the case. Leads with mini-plugs were then used to connect the interface to the motor circuits, which were also mounted inboxes with 4mm sockets.

The transistor outputs are open-collector and work with inverted logic. This can cause slight problems when driving motor circuits, as will be detailed later. The general connection and use of the interface is as shown in **Figure one**.

Note that the use of the Dragon's power supplies for external equipment is not advised. Batteries or a separate (preferably stainless) power supply should be used. Under no circumstances should mains voltage be connected to any part of the interface.

Initial tests with a number of DC motors, without the interface, indicated that currents greater than 200mA could occur

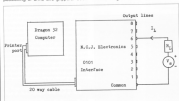


Figure one: The general interface connections.

motors rotate in a series of very short but very precise angular steps. Thus a typical motor may have 40 steps per revolution (7.5 degree step angle) and a more precise but more expensive type may have 200 steps (1.8 degrees step angle). Stepper motors are considerably more expensive than common DC motors, and require quite complex circuitry to control them. The motors found in model railways, model racing car systems, in Fischer-Technic kits and in Lego kits are all DC motors.

This article presents experiments done to enable a measure of control to be obtained over the simple DC motors that are generally in hand. Such control cannot be as precise as with stepper motors but being able to direct a buggy, for example, to behave rather like the classic Lego turtle is possible with the system.

To simplify construction a ready-built interface unit was purchased. (Electronic circuits were then built to enable the interface to run motors both forwards and in reverse. Two motors each, capable of independent

machine code using the print subroutine DS44. All the work done in this project involved Basic. The interface uses a 3A152223 8-bit latch with a transistor on each output line to enable low current devices to be drive directly. This interface does not contain relays, and it is quite compact (about 60mm x 50mm).

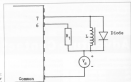


Figure two: Connections for an inductive load.

However, although the interface might be used to drive a small motor directly, it was decided not to risk damaging it, and to construct suitable motor drive circuits. **Figure one** shows an external circuit connected between line 6 and the common connection. Any or all of the lines could be connected, with the external power supply

Table one: variable values to set outputs ON.

Output line to be closed	1	2	3	4	5	6	7	8
Value to be added to V	1	2	4	8	16	32	64	128

negative terminals allowing returned to the common connection. Generally, a single power supply would be used, with the load for each line being taken to the positive side of the supply.

If the load is inductive, for instance a relay, then it is normal to include a diode across the load, as in **figure two**. When the current in an inductor is switched off, the back-emf pulse can destroy the transistor driving current through the inductor. To protect the transistor the diode is included and carries the unswitched current pulse away from the transistor.

**Figure two** shows a non-inductive load  $R$  connected to output 1, and an inductive load, such as a relay, connected to output 2. A diode is connected across  $L$ , and both loads are supplied from power source  $V$ .

## Interface programming

To program the interface in Basic, the format `PRINT #2,CHR$(V)` is used. The semi-colon is necessary to prevent a carriage return code being sent after the desired command. The variable  $V$  is the value needed to set a particular series of outputs to on/1 to act as a closed switch. **Table one** indicates the required values. For example, to switch on lines 1, 3, 5 and 6, the value of  $V$  would be given by

$$V = 1 + 4 + 16 + 32 = 53$$

The program line would then be

```
PRINT #2,CHR$(53)
```

The other lines would be left open. When a line is closed there is effectively a complete circuit from the external power supply through the load, through the line output and common connection back to the power supply. The interface acts like eight separate single pole single throw (single pole on-off) switches. Any combination of the switches may be operated by the `PRINT` command. As it stands, the interface may be used to control (small) motor directly by connecting the motor as the load  $L$  in **figure two**. Rather than risk damaging the interface by driving too much current through it, it is better simply to use the interface as a switching device and to operate the motor via an external transistor as in **figure three**.

An alternate circuit is shown in **figure four**. Here control of the motor is effected via the output line side of the interface. This becomes necessary if it is desired to operate a number of motors, with (a) equal relevant to a typical medium power NPN transistor. If the interface is switched off

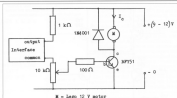


Figure three: operating the motor through an external transistor.

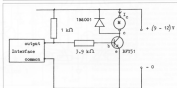


Figure four: an alternative control circuit.

open the transistor conducts (approximately  $I_c = 100\text{mA}$ ) and the motor runs. When the interface is switched on/closed, the transistor base  $b$  is effectively shorted to the  $5V$  line and the transistor is switched off ( $I_c = 0$ ). The  $100\Omega$  resistor protects the interface from excessive current when it is switched on (otherwise the power supply would be shorted across the interface).

Using a circuit similar to **figure four** a number of motors may be driven via the interface, each from one of the line outputs and with the transistor emitters all taken to the interfaced common line.

To operate the circuit of **figure three**, we need to close the interface as a switch and thus apply bias to the transistor base to cause current to flow in the collector. If we open the interface, then there is no bias to the transistor and hence the 80T51 (a typical medium power transistor) will be off. **Listing one** gives the commands to

switch the motor on and off ten times. Output line 1 is used. Thus the command

```
PRINT #2,CHR$(1)
```

closes line 1 and opens all the other lines.

```
PRINT #2,CHR$(0);
```

or any other value for  $V$  will open all lines, including line 1.

The circuit of **figure four** has to operate in an opposite way to that of **figure three**. Here we need to open the interface switch in order to apply bias to the transistor. **Listing two** gives an example to achieve this. Again we use line output 1. By setting the value of  $V$  to 1 (line 10 of the listing) we close the interface switch; this switches the motor off initially in line 60,  $V = 0$ . This opens the interface switch and causes the motor to run. Operation of the motor is

```
10 FOR I=1 TO 10
20 PRINT#2,CHR$(1):: REM MOTOR ON
30 FOR T=1 TO 2000: NEXT T
40 PRINT#2,CHR$(0):: REM MOTOR OFF
50 FOR T=1 TO 2000: NEXT T
60 NEXT I
```

Listing one: command for ten switchings.

```
10 PRINT#2,CHR$(1):: REM MOTOR OFF
20 K=INKEY$
30 IF K#"" THEN 50 ELSE IF K#"" THEN 50 ELSE 20: REM ON TO RUN, GO OFF
50 PRINT#2,CHR$(0):: REM GO 20
60 PRINT#2,CHR$(1):: GOTO 20
```

Listing two: command to open the interface switch.

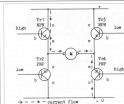


Figure five: circuit to operate the motor in either direction.

controlled from the keyboard, with key 0 to run and key 1 for off.

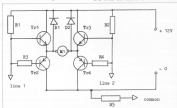
Since the remainder of the circuitry testbeds of the type shown in **figure four**, it will be necessary to open the interface switch on a particular line in order to turn a transistor on. Sometimes we must have a number of lines open while others are closed. Care must then be taken with the final value of *V* used in the PPRINT statement. Both circuits **figure three** and **figure four** were assembled on TDEC boards for testing. Resistor values may then be easily changed, as may the transistors used. Any kind of prototyping or breadboard will do the job.

A major problem with circuits of the form shown in **figures three** and **four** is that both types can only switch a motor on or off. When running it will drive in one direction only. The most versatile circuit allows when a motor can be switched to run in either direction. This can be achieved by using two of the interface outputs, two transistors and two relays. Suitable connection of the motor to the relays can then result in motion in one direction controlled by one transistor and motion in the other direction from the second transistor. A more elegant solution (and one which is cheaper and more reliable because mechanical relays are avoided) involves using a full-bridge transistor bridge of *Tr* circuit.

**Figure six** illustrates the principle. Two NPN and PNP transistors are used (ideally they will be matched complementary pairs). When the inputs to *T1* and *T4* are set high, current flows through them and drives the motor *M* in one direction. *T2* and *T3* are switched off by setting their inputs low, that is to the zero line.

*T1* and *T4* are then set low, with *T2* and *T3* set high, current flows in *T2* and *T3*, and the motor runs in the opposite direction. Setting all inputs low switches off the transistors and the motor stops. Setting all the inputs high would also switch the motor off, but since some current may flow in the transistors, it is preferable to switch the motor off by setting all inputs low.

**Figure six** gives a typical bridge circuit. Component values are not critical, and it is worth experimenting on a DEC board before soldering up a final version.



Resistors *R1* to *R4* should all have the same value. The 2N2905 is a PNP transistor with similar characteristics to the NPN 2N2904 type. As before, diodes (1N4001 type) are used to protect the transistors when the motor is switched off or reversed. *R5* is used to protect the interface should there be a fault in the circuit allowing the full supply voltage across the interface line and common terminals. It will restrict the current in the interface to a safe level (that is, more than 200mA).

## Single motor drive unit

It was decided to construct the circuit of **figure six** on a small circuit board and enclosed it in a suitable box. Tandy stores supply a variety of plastic and metal boxes; a plastic box 81mm x 51mm x 35mm was chosen. This came complete with a board which had a regular array of holes drilled in it, and copper laminate connections to the holes. **Figure nine** gives the component layout on the board. Small metal pins (obtainable from electronics suppliers) were inserted into the board and short lengths of stripped wire used to link the pins to act as 'bus-bars' for the individual components. Thus the component layout is directly comparable with the circuit diagram of **figure six**. Lead connections for the diodes and

transistors are given in **figure seven** and **figure eight**.

Although diodes and transistors are fairly rugged mechanically and electrically, they can be damaged by excess heat. A useful hint when soldering semi-conductors is to grip the lead between component body and soldering point with a pair of long-nosed pliers. To avoid the necessity for a third hand when soldering, wrap a strong elastic band around the handles of the pliers so that they are pulled shut. When heated open and placed on the component lead they will grip it firmly without the need to be held by hand. The pliers also hold the component in place on the board if you carefully arrange things prior to soldering. Allow the heat to dissipate before removing the pliers.

4mm sockets were mounted on the box and wires connected to these from the

circuit board for the two lines, common, motor and power supply. To assist in making connections to the unit the following colour coding was used for the sockets — line inputs: green, common: blue, motor: white, power supply positive: red, power



Figure seven: diode lead connections.



Figure eight: transistor lead connections, both sides.

# 5 REM SINGLE MOTOR CONTROL

```

10 PRINT#3, CHR$(33); REM LINES 1 & 2
20 RETURN ON, MOTOR OFF
30 MS=MSK14
40 IF MS=0 THEN 50 ELSE IF MS=L THEN
50 ELSE IF MS=D THEN 10
50 PRINT#3, CHR$(33); GOTO 20: REM LINE
1 ON, LINE 2 OFF, MOTOR RUNS RIGHT
60 PRINT#3, CHR$(33); GOTO 20: REM LINE
1 OFF, LINE 2 ON, MOTOR RUNS LEFT
70 REM LINE 1 HAS U=1, LINE 2 HAS U=2

```

Listing three: A program to run the motor left and right from the keypad.

Figure nine: A component layout for the circuit board.

supply negative black. This single motor drive unit could then be connected to the interface on the one hand and to a motor on the other. A Lego 12 volt 500 motor was used to test the unit. This took 0.26A when running and drawing some power. In another arrangement, it took 0.8mA with less loading. When the motor was off, the circuit current dropped to 1mA. With the motor running, the voltage across one of the conducting transistors was about 1.5V so that the power dissipation in the transistor was some 360mW, well below the transistor maximum (approx. 70mW).

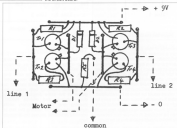
Listing three gives a short program to control a motor to run right and left from the keypad. To switch the motor off we need the 'V' value to be such that both lines are ON from the interface. This sets the two bridge inputs low. To turn the motor on, we want one input low and one high, i.e. one line to be ON and the other OFF. To have line 1 ON we set V=1, which automatically puts line 2 OFF. To have line 2 ON we set V=2, which puts line 1 OFF. Any other value for V will put both lines OFF. The program uses V=3. V values for other lines are given in Table one.

If the motor runs left when the R key is pressed, simply reverse the motor connections. To control motor direction by the cursor keys, replace the R and L of line 30 by CH-R(1) and CH-R(2) respectively.

## Double motor drive unit

In order to run two or more motors independently we must have a separate four transistor bridge circuit for each motor. Each bridge circuit then requires two lines from the interface unit. Two motors operating independently enables a variety of two-motor devices to be controlled. Examples could be two model trains (using separate power supplies to the two sets of tracks), and X-Y plotter or a buggy.

The circuit for the double motor drive unit is simply the combination of two arms single motor units, operating from one power supply. Figure ten gives the circuit. An extra component, D6, is used to protect the circuit (with its eight transistors) from inadvertently connecting the power supply with the wrong polarity. With no motor operating the quiescent current for a 12V supply was approximately 20mA. Operating values of current would depend on the



motors being driven (see the section on use with a Lego buggy).

Construction of the circuit was again on a protoboard, this time sized about 160mm x 60mm (out to slot into a circuit box of dimensions 160mm x 80mm x 50mm obtained from Tandy). A diagram of the component layout is given in figure eleven. As with the single motor unit, holes were drilled into the circuit box to take 4mm sockets. To avoid confusion during connection, I use the colour coding: inputs, green; motor outputs white; power supply positive; red; power supply negative; black; common connection to interface; blue. The transistors have maximum ratings as follows: BFY51:  $V_{ce} = 30V$ ,  $I_c = 1A$ ,  $P_{diss} = 60W$ ; 2N2905:  $V_{ce} = 40V$ ,  $I_c = 0.8A$ ,  $P_{diss} = 60W$ . When operating,  $V_{ce}$  is approximately half the supply voltage. Continuous operation could cause overheating if a motor takes more than about 0.1A. Heat-sinks on the transistors may then be advisable. The present unit has not been run for more than a few minutes continuously, and the transistors barely appear warm to the touch with a 12V supply and 0.2A maximum current per motor.

The double motor drive unit was used to control a model robotic arm built from a Flacko-Technic kit (Model 30654. Com-

puting). This uses two motors - one to rotate the arm left and right (motor 1) and one to raise the arm up and down (motor 2). The first four lines of the interface were linked to the drive unit, with lines 1 and 2 for motor 1, and lines 3 and 4 for motor 2. Motor 1 was controlled by keys R, L, and O (left), with motor 2 controlled by U, D, and S (stop).

To enable one motor to run whether or not the other was on at the time meant that some method of indicating the state of the motor was needed. This was achieved by having indicators or flags in the program. I, U=0 means motor 1 is not running left, I, U=1 means that it is running left. L2=0 means motor 2 is not running left; L2=1 means that it is running left (or down in the

Tab. 3	BFY51	D1,2	2N2905
Tab. 4	2N2905	D3,4	1 kOhm
		R5	100 Ohm

case of motor 2). Thus, every keypress sets an indicator.

Listing four gives lots of REMS to indicate what is happening in the program. The V values are given in the PRINT statements and were selected so that whether motor 1 is running or not, motor 2 can be switched on or off. Likewise, motor 1 can be operated irrespective of the state of motor 2. The V values are listed in table two for this arrangement. Since the interface has eight output lines, a total of four separate motors, each with its own power supply, could be independently operated. If such one is to be operated independently of the others, then suitable V values would need to be tabulated, again with indicators to report the state of each motor. Each motor would need one bridge circuit.

One multiple motor application could involve a model railway. Up to four trains could be programmed to run quite independently along separate tracks. A suitable program could be devised which would switch particular trains on or off at certain times. They could run forwards or backwards. The Dragon could control a set

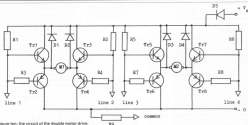


Figure ten: the circuit of the double motor drive.

timetable of train arrivals and departures from a number of stations for each train. One could even simulate British Rail by having a random element built into the program so that each train suffered a number of late arrivals and early departures, but exactly when would rail be kind?

To add two more motors, with an additional motor drive circuit for each line, new V values would be have to be calculated.

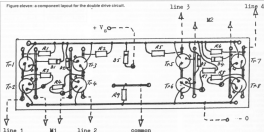
Table 3 shows the basic values in pairs for right and left rotation for each motor. Combining these values as in table two would then be required. A comparison between the tables should enable this to be done.

## The Lego buggy

The Lego buggy is assembled from a Lego kit (price about £20 as carriage and 1607) containing two 4.5V DC motors. Each motor may be used to drive one large wheel, while a small rear wheel (undriven) provides a third contact to the ground. Running both motors together in the same direction can produce forwards and reverse motion. Keeping one motor stationary while running the other, or running the two in opposite directions, causes the Buggy to turn.

It is emphasised by Lego that the motors should not be supplied with more than 4.5V, otherwise they may burn out. Initial tests indicated that the motors/timings take a fairly constant current. Over the voltage range 2 to 4.5V, a single motor took 0.38 while the two in parallel took 0.34A. With a motor stalled (ie trying to drive against an obstacle) the current taken by a single motor rose to about 0.25A. The stall current may rise up to 1A, if the motor alone, is prevented from running at maximum supply voltage. With the Buggy motors connected to the motor drive unit, a supply

Figure eleven: a component layout for the double drive circuit.



Tr1,3,5,7	18T51	D1-5	1N4001	R9	100 ohm
Tr2,4,6,8	2N2905	R1-8	1 kohm	M1-2	d.c. motor

Listing four: the control program with flags and REMs.

```

5 PRINT#2,CHR$(15): REM ALL 4 OUTPUTS
CLOSED: MOTORS OFF
10 L1=0: L2=0: R1=0: R2=0: REM MOTOR 1 AND
MOTORS SET TO 0
15 CLS: PRINT:PRINT"    SS CONTROL, NOT:
ON: PRINT"    USE THE KEYS:"
20 PRINT:PRINT"    ON: MOTOR 1 RIGHT":
PRINT:PRINT"    Q: MOTOR 1 LEFT"
30 PRINT:PRINT"    O: MOTOR 1 OFF: PR
INT:PRINT"    Q: MOTOR 2 UP: PRINT:P
RINT"    O: MOTOR 2 DOWN: PRINT:PRIN
T"
40 MOTOR 2 STOP"
50 KB=INKEY$
60 IF KB="R" THEN 100
70 IF KB="L" THEN 100
80 IF KB="O" THEN 100
90 IF KB="Q" THEN 100
100 IF KB="U" THEN 100
110 IF KB="D" THEN 100
120 IF KB="" THEN 20
130 R1=0: IF R2=0 THEN PRINT#2,CHR$(5):
ELSE IF L2=0 THEN PRINT#2,CHR$(3): EL
S E PRINT#2,CHR$(1): REM SWITCHES MOTOR
1 ON RIGHT
135 GOTO 30
140 L1=0: IF R2=0 THEN PRINT#2,CHR$(3):
ELSE IF L2=0 THEN PRINT#2,CHR$(1): EL
SE PRINT#2,CHR$(2): REM SWITCHES MOTOR
1 ON LEFT
145 GOTO 30
150 R1=0: L1=0: IF R2=0 THEN PRINT#2,CH
R$(7): ELSE IF L2=0 THEN PRINT#2,CHR$(1)
155 ELSE PRINT#2,CHR$(5): REM SWITCH
5 MOTOR 1 OFF
160 GOTO 30
170 R2=0: IF R1=0 THEN PRINT#2,CHR$(5):
ELSE IF L1=0 THEN PRINT#2,CHR$(3): EL
S E PRINT#2,CHR$(4): REM SWITCHES MOTOR
2 ON RIGHT (UP)
180 GOTO 30
190 L2=0: IF R1=0 THEN PRINT#2,CHR$(3):
ELSE IF L1=0 THEN PRINT#2,CHR$(1): EL
SE PRINT#2,CHR$(2): REM SWITCHES MOTOR
2 ON LEFT (DOWN)
200 GOTO 30
210 R2=0: L2=0: IF R1=0 THEN PRINT#2,CH
R$(7): ELSE IF L1=0 THEN PRINT#2,CHR$(1)
215 ELSE PRINT#2,CHR$(3): REM SWITCH
ES MOTOR 2 OFF
220 GOTO 30
230 REM COMMANDS FOR ONE MOTOR ALLOW OTH
ER MOTOR TO OPERATE INDEPENDENTLY

```

Listing five: the Buggy control program.

```

5 REM BUGGY CONTROL
10 PRINT#2,CHR$(15): REM ALL 4 OUTPUTS
CLOSED: MOTORS OFF
20 CLS: PRINT:PRINT"    TO CONTROL B
UGGY: PRINT"    USE THE KEYS:"
30 PRINT:PRINT"    F: FORWARD"
40 PRINT:PRINT"    B: BACKWARD"
50 PRINT:PRINT"    L: LEFT"
60 PRINT:PRINT"    R: RIGHT"
70 PRINT:PRINT"    S: STOP"
80 KB=INKEY$
90 IF KB="F" THEN 100
100 IF KB="B" THEN 100
110 IF KB="L" THEN 100
120 IF KB="R" THEN 100
130 IF KB="S" THEN 200
140 IF KB="" THEN 30
150 PRINT#2,CHR$(5): PRINT#2,CHR$(3)
160 GOTO 50: REM R1 ON R, R2 ON R
170 PRINT#2,CHR$(5): PRINT#2,CHR$(1)
180 GOTO 50: REM R1 ON L, R2 ON L
190 PRINT#2,CHR$(5): PRINT#2,CHR$(3)
200 GOTO 50: REM R1 ON L, R2 ON R
210 PRINT#2,CHR$(5): PRINT#2,CHR$(1)
220 GOTO 50: REM R1 ON R, R2 ON L
230 PRINT#2,CHR$(5): GOTO 50: REM BOT
H OFF

```

Listing six: an alternative control program.

```

5 REM BUGGY CONTROL2
10 PRINT#2,CHR$(15): REM ALL 4 OUTPUTS
CLOSED: MOTORS OFF
20 CLS: PRINT:PRINT:PRINT"    TO CONT
ROL BUGGY"
30 PRINT:PRINT"    USE CURSOR KEYS"
40 PRINT:PRINT:PRINT:PRINT"    TO STO
P"
50 PRINT:PRINT"    USE SPACE BAR"
60 KB=INKEY$
70 IF KB=CHR$(24) THEN 100:REM F
80 IF KB=CHR$(18) THEN 120:REM B
90 IF KB=CHR$(8) THEN 100:REM L
100 IF KB=CHR$(3) THEN 100:REM R
110 IF KB="" THEN 200
120 IF KB="" THEN 30
130 PRINT#2,CHR$(5): PRINT#2,CHR$(3)
140 GOTO 50: REM R1 ON R, R2 ON R
150 PRINT#2,CHR$(5): PRINT#2,CHR$(1)
160 GOTO 50: REM R1 ON L, R2 ON L
170 PRINT#2,CHR$(5): PRINT#2,CHR$(3)
180 GOTO 50: REM R1 ON L, R2 ON R
190 PRINT#2,CHR$(5): PRINT#2,CHR$(1)
200 GOTO 50: REM R1 ON R, R2 ON L
210 PRINT#2,CHR$(5): GOTO 50: REM BOT
H OFF

```

To run Motor 1	With Motor 2 OFF	OB R	OB L	OFF
		1	2	15
		OB R(1)	6	7
		OB L(2)	10	11
To run Motor 2	With Motor 1 OFF	OB R(3)	OB L(4)	OFF(MOP)
		4	8	15
		OB R	9	15
		OB L	10	14

Table two: P values for Listing four.

Motor 1	Motor 2	Motor 3	Motor 4
R L	R L	R L	R L
1 2	4 8	14 32	64 128

Table three: values for left and right rotation.

	M1	M2	P
To move forward P needs both motors ON R	R 1	R 4	5
To move backward R " " " ON L	L 2	L 8	10
To move left L needs M1 L and M2 R	L 2	R 4	6
To move right R needs M1 R and M2 L	R 1	L 8	9
To stop Buggy needs both motors OFF	1+2	4+8	15

Table four: values to drive the Buggy.

of 1V was needed to ensure that each motor received 5.5V, or a little less, when the Buggy was operating.

The Basic program needed to control the Buggy's movements requires suitable values of variable V. These will depend on how the motors are wired with regard to the polarity of the outputs from the motor drive unit. It will be necessary to experiment a little here. Viewing the Buggy from above and denoting the left-hand motor M1 and the right-hand motor M2, we may setup the table shown. A motor running right R is deemed to be in the forward direction, running left L, is taken as backward/reverse.

Make that in order to turn to L or R both motors are running. Turns may be effected by stopping one motor and driving the other. However, if this is done the power supplied to the Buggy is halved and the single motor running has difficulty in driving the Buggy. Hence the speed falls. On the other hand, by stopping one motor, the Buggy will pivot about the point of contact of the stopped wheel. This entitles a more precise description of the path of the Buggy. In this case, the third and fourth lines of the previous table need modifying to:

To move left L needs M1 OFF and M2 R	OFF 1+2	R 4	7
To move right R needs M1 R and M2 OFF	R 1	OFF 4+8	15

## Extra points

To stop the Buggy when it collides with an obstacle, one technique uses micro-switches at the front and rear of the Buggy. These are in series with diodes and the motor as shown in figure twelve. The switches should be of the normally closed type and when contact is made with the obstacle the switch opens and cuts off current to the motor via the relevant connector. Two diodes and two switches are used because current has to flow in two directions, depending on which way the motor is running.

If more powerful motors are to be used, the transistors used here will be overloaded. In this case, two transistors of the same type may be doubled up in form a Darlington pair as in figure thirteen.



The Buggy viewed from above.

The Buggy control program given in listing five operates with both motors running for a turn. Motion direction is selected by the R, L, and H keys, with STOP by S. When a new key is pressed the command to stop both motors is issued prior to starting the new motion. In view of the speed of the Dragon's signals this command will have little effect. However, if a time delay is inserted after the command (in lines 180 to 190), the Buggy can be made to pause for as long as required. This may be useful in some situations.

An alternate control program is given in listing six. The Buggy will now respond to the cursor keys for direction control and is stopped by the space bar. This may be found to be a more convenient method of control, particularly if an extended series of direction changes is needed. Again both motors operate on a turn.

Having obtained control of the Buggy, a felt-tip pen could be attached to it (eg clamped by a clothes peg) and Turtle-type graphics done on paper. For true LOGO control one needs to adapt a suitable program (perhaps Mike Harder's Mini-LOGO in Dragon (June/September 1988)) or write one's own.

Alternately, it is now possible to buy devices containing one Darlington pair in a single package, or multiple pairs in a single package. Recently complementary transistor arrays have appeared. One contains two PNP and two PNP transistors in matched pairs in a 14 pin DIL plastic package (600mW dissipation per transistor). This would be ideal for producing one motor drive unit circuit for, say, the Lego motor on a printed circuit board (but rather expensive, at 45p).

Finally, please note that I have no business connection with MCI Electronics, the producer of the G801 Interface Unit used for the experiments described in this article. It was good luck in the first place that acquainted me with the interface and enabled me to devise the experiments for myself over a period of time.

## Component sources

The address of MCI Electronics is 38 Marryfold Road, Chatterlands Ave, Hull HU6 4DS.

Two suppliers of electronic components to the amateur constructor are given below. Others, of whom the largest and best known are probably Maplin Electronics, may be found in the pages of any electronics magazine such as Practical Electronics, Electronics Today International, Everyday Electronics, Practical Wireless or Radio and Electronics World.



**Rapid Electronics**, Hill Farm Industrial Estate, Boxed, Colchester, Essex CO4 5PD.

**Magenta Electronics**, 135 Hunter Street, Burton-on-Trent, Staffs DE14 2ST.

The Lego Buggy can be obtained from **Econometrics Ltd.**, Epic House, Drygreen Road, Handsworth, Sheffield S18 6UG, **Unilever Ltd.**, Clarendon Road, Blackburn, Lancs BB1 9SS, and **Magenta Electronics**. It is also available in some large toy stores.

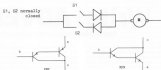


Figure twelve: bidirectional switching to stop the Buggy.

Figure thirteen: transistors doubled to form a Darlington pair.

### Parts Lists

#### Single motor drive

- 2 x 80V181 NPN transistor
- 2 x 2N2905 PNP transistor
- 2 x 1N4001 1A, 50V diode
- 4 x 1 kilohm 0.25W resistor
- 1 x 100 ohm 0.25W resistor

- 1 plastic box to suit
- 1 x pinboard to fit box
- 7 x 4mm sockets
- Pins and connecting wire

Approx. cost of unit £4.08

#### Double motor drive

- 4 x 80V181 NPN transistor
- 4 x 2N2905 PNP transistor
- 4 x 1N4001 1A, 50V diode
- 8 x 1 kilohm 0.25W resistor
- (8 x TO18 package heatsink)

- 1 plastic box to suit
- 1 pinboard to fit box
- 11 x 4mm sockets
- Connecting wire

Approx. cost of unit £8.50

While *Dragon User* makes every reasonable effort to ensure that published projects are viable, it cannot be held responsible for any loss or damage arising from such projects. These projects do not employ mains voltages directly and are not difficult to build, but require care and attention to detail to achieve success. If in doubt, ask the advice of someone with greater experience of electronics construction. Check that all the components you want to use are available from suppliers, and the current prices, before starting.

## THE DRAGON AND TANDY SHOW AT CARDIFF — WALES AIRPORT

on Saturday  
27th April 1988  
— 10.00 — 3.00

Adults £1.00

Children 0.50p

Don't miss the opportunity to meet the major retailers and to watch demonstrations by Dragon Users.

Ample car parking      Easy access  
Regular bus service      via M4  
from Cardiff Centre.      Refreshments.

If you would like to take part as a demonstrator, showing how you use your Dragon or Tandy in an interesting or unusual way, contact John Penn on 842653 6976

0211

### CLASSIFIED ADS

**APOLOGY:** Owing to the vagaries of the Christmas post, we are without our classified advertisements this month. Normal service will be resumed as soon as possible. Meanwhile, an announcement from one of our customers:

Herberts (0433) 363476 is offering Dragon software for sale for the price of £100.00 and not £120.00 as previously stated. A Dragon 32 and hardware are also available.

#### HERE'S MY CLASSIFIED AD:

(please write your copy in capitals on the lines below)


Name .....

Address .....

Tel. ....

Classified rate: 25p per word.

Please cut out and send this form to: Classified Department, Dragon User, 12-13 Lifford Newpark St, London WC2H 9PP.

# Write: ADVENTURE

Pete Gernard comes up with some codes for characters

SO far in this, the only miniseries in the world not to feature Joan Collins (thank heavens), we've managed to introduce fairly simple characters who do little more than be around, and gone on to consider more advanced characters who are capable of contributing significantly to the enjoyment of the game. They are also, in most cases, rather important in the solving of a was was.

This month we'll be looking at extracts from a real game, and as usual I'll have to make an apology for talking in expanded bits of Basic rather than the real things themselves. These, sadly, would make little or no sense if used as stand alone listings, so I hope you'll make allowances for that and I'll try and point out where to make the necessary changes.

We're going to be meeting, once again, living legend, Dmili Gicing the Wonder Dwarf! Plus a band of his pals, and we're going to consider the introduction of specific characters (and the corresponding verbs required) into an adventure game. They're all different, with varying attributes depending on their status in the adventure in question, some are useful, some are just decoration, but we'll sort all that out when we come to it.

By considering a real adventure you should be able to get a better idea of how characters are used and controlled, certainly better than I would talk in general terms. So, in order of importance we'll be meeting Stormdrigger the Grey, Legless the Elf, a goodlooking, Prof Harris/How did he get in here? and Rainier. You, of course, take on the role of Chris Gicing, master of them all, and squishy eyed introduction. Oh all right then, a brief word. You're a dwarf, and in the game have a specific mission to complete. No treasures to collect, just a series of bizarre solutions. Also for you it is impossible to complete the adventure without the help of several accomplices, and the first one you'll meet is Stormdrigger the Grey.

Stormdrigger is the hero of a couple of stories in a magazine that shall be nameless, and all you need to know is that his parents were dyakic, hence the unusual name. Being a wizard, he is capable of casting spells and generally sorting out one or two things that you can't manage, but he does have this thing for a drop of ale or two. The little, and here I cannot forget his hands shaking and cannot cast a spell to save his life. Too much, the obvious result occurs, and he still cannot cast a spell. So one of your tasks is to ensure that he manages to pay at least once to the pub, but no more than four. To add to your problems, he's a stubborn old goat, and will sometimes need a bit of persuading in order to comply with any request you might give him.

Figure one shows him making his first, and subsequent, appearances in the game. As this is an idea for a forum rather than a programming course we won't go into too much detail, but you should be able to get my drift, as that strange saying has it.

Two things to note. The subroutine `wait2400` is a simple delay loop to give you time to read a message, and the subroutine at line 1080 extracts and prints a message dictated by the variable 'vw'. Right then, let's have a look at Stormdrigger.

Line 1082 first of all. This is used if the wizard has never wandered into the adventure before, and you are in the correct location for meeting him. CP holds the player's current position, and location 12 being the heart of the pub means that the wizard here. Then we set the 'wizrd' worded 'flag', 'vw', the 'wizrd following' 'wf', and print up descriptions.

Lines 1084 and 1085 introduce a couple more variables, namely 'sa' and 'vp'. 'vp' is used to keep track of the number of visits that he makes to the pub, and 'sa' is used to keep an eye on how long Stormdrigger has stayed with you without being given something to do. Now then, if the wizard's following you and the random-number table with a given range and 'sa' is less than 11 then a random message about Stormdrigger is printed on the screen. Unlike Thorin, the wizard sits down and sips about beer, or other things. However, if the

variable 'sa' is greater than 10 then old Stormdrigger gets a bit of the idea of being given nothing to do and, provided that you're not already in the pub, he wanders off back to it, saying something along the lines of "Well, I'm off to find a decent ale house". Various flags and variables are set as a result of this.

Line 1088 is only used when you go off to retrieve the wizard, and if you're in location 12, the wizard's already been found once (`vw=1`). He's not following you at the moment (`wf=0`), then set the 'wf' flag and print up a message about him reappearing by your side and being ready to join in the game again.

Just four lines of code to give the wizard a real slice of character. He does random things, he sometimes sulk and stomps off to the pub, but when you find him again the characterised old boy is ever ready to lend help to you. If the characters, there is a lot more code involved with Stormdrigger than this, because you can talk to him and ask him to do things, but as I've said this is not a programming course. Instead, we're just discussing characters and how they can easily be introduced to your games.

Stormdrigger is a meaningful character, in that the adventure cannot be finished without him doing things for you. Before next month we'll look at Legless the Elf. This is Legless the Elf, always to be found in the pub, and as the game progresses he

```
Listing one
1082 IF CP=12 AND VW=0 THEN VW=1:WF=1:FO
R Q=1 TO 3:NR=Q:GOSUB 5000:PRINT:GOSUB 2
400:NEXT
1084 IF WF=1 AND RND(100)>75 AND SS=11 T
HEN NR=RND(3)+8:GOSUB 5000:GOTO 1086
1085 IF WF=1 AND RND(100)>75 AND SS=10 A
ND (CP=11 OR CP=17) THEN NR=4:VP=VP+1:G
O(17):NR:WF=0:SS=0:GOSUB 5000
1086 IF CP=12 AND VW=1 AND VP=0 THEN VP=
1:NR=5:GOSUB 5000
```

```
Listing two
1088 IF (CP=11 AND CP=18) THEN LL=LL+1:P
RINT:IF LL>10 THEN LL=10:NR=99+LL:GOSUB
5000:GOTO 1082
1081 IF (CP=11 AND CP=18) THEN NR=99+LL:
GOSUB 5000
1052 IF (CP=11 AND CP=18) AND SA=31 THEN
NR=102+LL:GOSUB 5000
```

```
Listing three
2457 IF SA=01 AND SS=1 THEN PD=0:LO=1:PR
INT:IF DECIDE8 TO FOLLOW YOU.1:OR(01)=1
:SS=SS+1:GOTO 10
2458 IF SA=01 THEN NR=204:GOSUB 5000:GOT
O 10
```

replay) ends up being *Legless* the elf, and the messages used reflect this (Figure two).

A character should never be used in a game unless he, she, or it adds something to it. You may feel, therefore, that *Legless* is a little redundant, but when the game was being play tested one of the 'testers' said that she kept looking around the pub in order to see what *Legless* was getting up to next, completely forgetting about getting on with the adventure in order to keep track of our friendly elf. So he fulfills his purpose by adding enjoyment to the game. So much so that someone else (hello Gandra!) told me that she wanted to talk to him as well, and in a very simple way we can take care of that possibility also.

Line 1080 checks first of all to see that you're in the pub, and if so the 'legless' variable is incremented by one. Since the chap can only handle so much beer we then see how many he's had, and print up a suitable message informing the player of the elf's current state of health. Once he's had enough he becomes totally incapable of doing anything at all, and line 1101 sorts out the final message in the sequence. There are ten messages used in total, and that seemed to be enough to keep people amused while playing the game.

Line 3452 is the 'talk to legless' line, which again checks to see whether you're

in the pub. If you are, and you're talking to noun number 31 ('legless') then we use the 'legless' variable 'l' again in order to avoid a suitable message from our file. Again, you're not really talking to him in the sense that the responses are pre-programmed, but it serves to enhance the feeling of genuineness about the elf.

The game is capable of being solved without *Legless* being in there at all, but it would be a poorer game without him.

We'll close a chapter up on the activities of *Rollin* (and also *Hey!*) where to start marketing the things that you can meet them yourself and finish off by talking to the guide dog. As I've started off by being like the elf, just there for decoration, but at a suggestion from someone I gave to play a much more active part in the game. My friend said, reasonably enough, that people are fed up with looking for lanterns to light their path in a cave, so why not use the guide dog in that respect, to guide you through the darkness of the cave? Good idea, and so that is what this particular character now does.

Before using the guide dog you have got hold of it, of course, and that is the purpose of Figure three. As with *Chimney* there is plenty more code concerning the dog, but this should give you some idea anyway. The noun variable 'm' is equal to 61 if the player is trying to get the dog. If the 'dog given bone' variable 'db' and the 'player

poof' variable 'gp' are both equal to one then you can safely take the dog. Thus we turn off the 'pick up' variable 'p' (turn on the 'light on' variable 'o' to show it) which are checked when the player is deep underground in theinky blackness) and put the dog in the player's possession after printing up a suitable message. If either of the two variables 'gp' and 'db' are not set (and if you want to know why it's necessary to play pool before getting the dog then you'll just have to play the game) then line 3456 prints up a suitable message.

## Conclusion

We've only looked at nine lines of code governing three very different characters, but that should be enough to give you an idea of what they're about, how they work, and how they affect the game. Also about how various words are needed to talk, for example. No character should be put into it doesn't, in some way, make for a better game, and in their own individual ways *Chimney*, *Legless*, and the guide dog, all contribute to the enjoyment of the adventure as a whole.

Well, maybe I've managed to give you an insight into at how I write adventures, and I trust, as a player reviewer, if looking for another people's games. Characters are all important, don't neglect them.

Bye for now.



PICTURE the scene if you will, it is a frosty morning in Wigan two weeks before Christmas, and Our Beloved Editor is already deranting the copy that you're now reading! There's a black and white stain in next door's garden, there's an enormous alarm sounding around two doors away from the stable, but the animals haven't met each other yet. This is not an old concentration, and I shall continue to report on the situation as the column progresses.

Some of you may recall my acquaintance Professor Deadrock from an earlier *Dragon User* July issue, for the curious amongst you, well, it looks like he's been out and about exploring again...

## Dear Diary

Found myself taking the unusual step of arranging a winter break courtesy of the textbook Adventure Club. What a strange

place to begin, though, people leaving things lying about all over the place. Found a manual, decided to open it to read it and see if I could make any sense of my surroundings, and a key popped out. Most odd, naturally enough I then tried unlocking a nearby cabinet, his one side around, so no-one to see my somewhat futile actions. Have no wish to bring shame on the family name. Looked in the cabinet and took everything out of it. A strange assortment of items. Suddenly (was it something I had done?) I received a strange message, and on further exploration found a tiny looking robot and read my message. Something about the co-ordinates, which I suppose I had better take a note for future use.

In a burst of inspiration I set the co-ordinates to the number that I had been given, 8053, and pressed a distinctly dangerous looking red button. Explored

still further, and found myself in the telegraph room. Naturally my curiosity got the better of me and I pressed the button in there, too, and wearing my head suit and carrying a beaker for any strange bits and pieces that might be waiting for me I entered the window that appeared before me. I was in a planet at last.

By going south and east I stumbled across an unusual ship lying in the middle of nowhere. By the simple expedient of going to the hatch I found myself inside it, with many objects to take and examine. I managed to get a map, a shovel, a lantern (so much for the beakery) and a cartridge from a helpful robot. Thought that was enough to be going along with, and after going north and filling my beaker with acid (might as well use it for something) I teleported myself back to the warmth and security of my own ship. What a peculiar day's exploration.

Had the most fitful of night's sleep, but awaking feeling rather refreshed and all that and ready to tackle the next stage of my adventure, whatever that might entail. The carriage looked as if it were designed to be inserted into the device in front of me, and sure enough it was. I pressed the button, and indeed in the dawn until I was empty.

A ladder, eh? I went up it, but by the merest chance I stumbled at the top and managed to grasp the acid all over a panel. It dissolved, of course, and I looked around cautiously to see if anyone would notice this accident and come and reprimand me, but there must be dashed exclusive holidays for nobody else appears to be around at all. Was beginning to feel lonely at the thought of it, but I realised that I must continue and as pressing buttons had not done me any harm so far I pressed the whole one that lay before me. So many buttons, so little dangerous effects to date.

Through I'd better go back and replace the thought that had been coming, you never know with stuff like this, if one person uses it then they'll all want it sooner or later, so made a quick trip back to the desert planet and filled up again. Didn't like to drop the basket anywhere, it looked extremely fragile, but I suppose a cushion or a pillow would break it all if I needed it. I decided to persevere.

Reached my map, which told me that the co-ordinates for the intriguingly named ice planet were 0416. I entered them into my favourite device and pressed the friendly red button as before, wearing my trusty suit, which I was now beginning to feel smart at home in, although it is in truth a far cry from the smoking jacket that I normally favour, and carrying a shovel and a basket I made my way to the teleport room and blundered down onto the surface of the planet.

## Holiday on the sun?

One cannot help wondering, dear diary, where all these pretend planets come from, and how these holiday agencies find them and manage to obtain them exclusively for us holidaymakers. A desert planet, an ice planet, whatever next, one wonders? A garden planet perhaps? We shall see as I find more co-ordinates no doubt. Meanwhile, I must carry on with my explorations of this ice planet and see what we can find, too, I shouldn't wonder.

After much digging I managed to obtain what appeared to be a sufficient quantity of ice for whatever possible means I might have for it, and the basket seemed to be the worst of things to carry in all the while it slipping away or melting into an embarrassing puddle that would be awkward to explain away at best. On returning around I found an interesting rockpool, and although not the sphere that I once was I stumbled and found my way to the top of it. Lo and behold I found an amulet, and couldn't help but think what a splendid souvenir that would make for Mrs. Armstrong at the Bridge Club, but I digress.

From the rockpool I deduced yet another set of co-ordinates. What a lot of numbers to remember for sure, and I am fortunate that despite my advancing years

I still have a stentive memory. But, time was getting on, and although it might have been thirty or forty sunrises ago I again die need of some sleep to assist in vigorous for the challenges of what will hopefully be my last journey to another planet. In some haste I made it back to the comfort of my own ship and once more settled down for the night.

Had the strangest dream, all about spiders, but cannot make head or tail of it for there are no such beasts on the ship. I have examined it thoroughly.

## Garden centre

As carrying all this was beginning to get awkward I decided to wear the amulet, and felt at once strangely reassured by its presence. Having set the co-ordinates in my usual way and pressed the red button I decided that the basket was a burden that I could not do without, and I dropped it and its contents on the desert floor. So much for that! Carrying my basket carefully this time, I also took along with me a capsule and a rope, wearing as ever my trusty suit and with the amulet firmly in place. I teleported myself onto the surface of this brave new world, and what did I find, dear diary? A garden planet, just as predicted.

On arrival I went east from the bridge and tied the rope to the tree, could now cross in safety and almost immediately found myself blundering into a forcefield. Something seemed to be protecting me, however, and I glanced reassuredly at the amulet, convinced that it had done the trick. Might as well keep it for myself rather than giving it to Mrs. Armstrong. She can have the space suit when I've done with it.

Good lord, a spider! My dreams are all coming true, and in desperation I tried feeling the beast with the capsule. To my relief it appeared to explode such a blast, although heaven only knows I'm going to send a stiff epidemic to the holiday company on my return for the dreadful food that I personally have had to suffer. Fortunately it would appear that I have not suffered as much as the spider, who ate the capsule and promptly became unconscious. I will admit to feeling a pang of sorrow, one doesn't like to see even the loveliest of creatures in pain, as in my fit of remorse I latched onto a strange room and left it there. This was after carefully removing

what I deemed to be a Xantos plant, which will surely win many a prize at the next horticultural show in the village. That will put a few noses out of joint, I can assure you.

Outside this strange room I pressed a button, and I think that I've somehow managed to hit the spider. This is indeed unfortunate, but one cannot stop to ponder the fate of every animal and insect in the world, and I must hurry on. I stumbled across a grate, but unlike every other grate I've ever met in an adventure holiday this one didn't take too kindly to being opened with a key, or so I deduced. It is a bit of a puzzle jumped up and down, and by the moment of accidents managed to fly home as soon as I stepped aside hastily and watched in amazement as the grate dissolved just like the panel earlier however firstly I got I made a view lower to even think of drinking the acid. Or entering the plant with it, hardly usual or to handy amulet.

## Rope trick

On going through the remains of the grate I found myself very much informed of a pain of loss. Fortunately there were some pieces of rubble scattered about and I could walk over with ease. Then I had to give myself a pat on the back for my foresight. The rope had nothing to do with the bridge, but was now available for me to climb and make my journey back to my ship easier and safer than a pleasant surprise, and one up for (Dear diary, oh diary?)

Once back in my ship I planted the Xantos flower before setting the co-ordinates to 1042 and the journey home is worth. Taking the Xantos flower and going over to and through the teleport window see the end of my adventure and winter weekend break. Perhaps it's give the flower to Mrs. Armstrong after all. She probably likes flowers.

And this is me again? What unusual adventures the old chapras, eh? Well, no time for dawdling, must get this in the post to Our Beloved Editor. Time only to tell you that the rabbit has now settled down under a large bush, presumably eating it away but from the inside, and the situation has settled into the distance somewhere. Rabbit lives again in light another day. Bye for now.

## Adventure Contact

To help puzzled adventurers further, we are instituting an Adventure Helpline — simply fill in the coupon below, stating the name of the adventure, your problem and your name and address, and send it to Dragon User Adventure Helpline.

10/13 Little Newport Street, London WC2H 7TF. As soon as enough entries have arrived, we shall start printing them in the magazine.

Don't worry — you'll still have Adventure! That to write to as well!

Advertiser	.....
Problem	.....
Name	.....
Address	.....

# Down in the dumps

*This month's screen dump is for a Brother HR-5 printer*

This program will copy the contents of the screen to a Brother HR-5 printer. It can be incorporated into your own programs as a subroutine in the manner shown, or used as a stand-alone utility to dump screens from other programs.

To use the program in the second way, DELETE lines 12 to 25, change line 109 to END, DELETE lines 303 onwards, RETURN and save the new version to tape. Load and run a program until you see a screen you wish to copy and press BREAK. Load the screen dump and RUN. The screen will now be printed out.

Being in BASIC, the execution time is rather slow, so have a cup of coffee while you're waiting.



Note on lines 122 to 125

In theory, these should read as follows:

```
122 PRINT C-2,CHR$(A);  
145 NEXT X  
152 PRINT C-2,CHR$(B);
```

However, if you substitute these lines for the equivalent ones in the program listing, you will find that when dumping a pattern or picture that fills the entire screen, as in the second example given, unwanted characters may be printed at the end of some lines. I don't know the reason for this, and the only way I've found to avoid it happening is by sacrificing the final column of pixels. Advice please, anyone?

K. Posthead

```
10 REM PROGRAM: SCREEN DUMP SUBROUTINE  
20 REM SYSTEM: DRAGON TO BROTHER HR-5 PRINTER  
30 REM AUTHOR: K. POSTHEAD, 1987.  
40 REM  
50 GOTO 250  
60 PRINT C-2,CHR$(27);"1";:REM SET LINE PITCH TO 1/9TH. INCH  
70 PRINT C-2,CHR$(27);"R";:REM ENGAGE ELITE CHAR. SET  
80 PRINT C-2;Y=0  
90 PRINT C-2,CHR$(H);CHR$(H);CHR$(H);:REM TAB TO CENTRE  
100 PRINT C-2,CHR$(27);"T";CHR$(255);CHR$(0);:REM ENGAGE PRINTER GRAPHICS MODE  
110 FOR X=0 TO 255  
120 A=PPROB(X,X)+128+PROB(X,Y+1)+64+PROB(X,Y+2)+32+PROB(X,Y+3)+16+  
PROB(X,Y+4)+8+PROB(X,Y+5)+4+PROB(X,Y+6)+2+PROB(X,Y+7)  
130 IF X=255 THEN A=0:GOTO 150  
140 PRINT C-2,CHR$(A);:NEXT X  
150 PRINT C-2,CHR$(13);  
160 Y=Y+8:IF Y<91 THEN GO  
170 PRINT C-2,CHR$(27);"2";:REM RESET LINE PITCH TO 1/6TH. INCH  
180 PRINT C-2,CHR$(13);:REM DISengage ELITE CHAR. SET  
190 RETURN  
200 REM DEMO PICTURE 11  
210 MODE 4,1:SCREEN 1,0:CLS  
220 R=50:P1=R+INT(R/3)  
230 FOR T=1 TO 360 STEP 20  
240 A=PI*T/180  
250 X=130+R*SIN A;Y=100+R*COS A  
260 CIRCLE (X,Y),R  
270 NEXT T  
280 GOSUB 60  
290 REM DEMO PICTURE 12  
300 MODE 4,1:SCREEN 1,0:CLS  
310 FOR I=0 TO 255 STEP 3  
320 LINE (I,0)-(255-I,191),PSET  
330 IF I<192 THEN LINE (0,I)-(255,I+1-I),PSET  
340 NEXT I  
350 GOSUB 60  
360 REM PROGRAM CONTINUES....  
370 REM  
380 END
```

# Faster, faster, faster!

Gordon Lee wants a quicker run-time. You tell him.

## Prize

DRAWING conclusions from a Gordon Lee puzzle is a well-known art of giving a programmer a pain, but drawing anything else can be nearly painless with a copy of David Heiser's top-of-the-duplicator graphics program, *Picture Maker*. Jane Peen Software, David's publisher, are offering 10 *Picture Makers* and ten discount vouchers for this month's winners.

## Rules

When you have replaced the Lee Listing in Lightning duration, print out your program, add any notes (notes will be considered if not considered), don't forget the forecaster, put it all in an envelope marked FEBRUARY COMPETITION and send it to the usual address (on the front cover, if any of you are still searching).

This month's tie-breaker is something close to the heart of many of us: send us any suggestion or suggestions for improving the running times of all or any British Mail train. Quality rather than quantity is the key factor points for anything we can't think of any easy British Rail-type excuse for.

## November winners

THE PC's nothing like a lot of literature is empty dance floor, is there? All we ask you do make up your mind in later words, then of a definition, and get them accepted at the Oxford English Dictionary by last Thursday, and what happens? Everyday chickens out, except a small and select band whose winning representations are retransmitted here in their glory. They are: Mark Twain of Long Eaton, Phil Sapiro of Buxton, Terry Rawson of Heaton, Paul Wexler of Milton-under-edge, Andrew Marshall of Blackley Gleason Barber of Clifton Coldwell, Keith Davis at Crawley (Three Bridges, actually, which sounds a lot more) and SA Sotopuk Czerwinski. Inevitable middle ring away and copies of *Space Translators* donated by R & AJ Preston inevitably.

"All the winners came up with at least one answer," says Gordon. "Top of the list with 28 words (all in Chambers!) is Mark Twain, for exclaiming out such gems as 'mug', 'muzzle' and 'red-eye' though 'bogey' would probably be a more appropriate term after such a splendid attempt. He also included a list of observations, adding up to 265, including PUBDOG, a poon on the tooth, ETCHER and upper-class sweaters, and LINECA, an automatic goal-scoring machine."

## Solution

This month's solution should be optimistic.

LAST month on this page we were considering an improved method of performing string multiplication. Such a method is given in Listing 1, but before examining it in detail there are one or two points to be mentioned. In this program there are only three main string variables: A\$ and B\$ — the two strings holding the numbers to be multiplied — and Z\$, the final product. The basis of this method depends on 'loading' into string Z\$ each digit as it is computed, thus eliminating the need to have an array of sub-products.

In order to do this the first requirement is to define Z\$ of the subset as a string of zeros long enough to contain the final product.

### Listing 1

```
10 TIMER=0
20 A$="1123456789"
30 B$="123456789"
40 C=0:G=LEN(A$)+B$:D=STRING$(1,"0")+A$+B$
50 FOR M=LEN(B$) TO 1 STEP -1:D=LEN(B$)+B
60 A$=VAL(MID$(A$,M,1))
70 FOR N=LEN(A$) TO 1 STEP -1:D=LEN(A$)+B
80 A$=VAL(MID$(A$,N,1))+C
90 C=A$+C+C
100 IF C>9 THEN A$=VAL(MID$(A$,1,1)+C-9)
110 A$=VAL(MID$(A$,2,1)+C-9)
120 IF C=0 THEN A$=VAL(MID$(A$,1,1)+C-9)
130 A$=VAL(MID$(A$,1,1)+C-9)
140 A$=VAL(MID$(A$,1,1)+C-9)
150 NEXT M
160 NEXT N
170 IF LEN(A$) < G THEN A$=A$+MID$(A$,1,1)+C-9
180 PRINT A$
190 PRINT "TIME TAKEN: ";TIMER/56;"seconds"
```

For any two numbers having *n* and *m* digits respectively will, when multiplied together, result in a product having either *n* plus *m* digits, or one less than the number. Consequently, having defined variables A\$ and B\$ in lines 20 and 30, the total length of Z\$ can be defined in line 40 in the listing, as both numbers have ten digits each, the value of L will be given as 20, and the STRING\$ command which follows (line 40) will produce an initial string for Z\$ as twenty zeros. The addition of the extra zero to the end of A\$ in this line simply to facilitate the handling of any final 'carry' at the end of the calculation.

To clarify the working of the program, the variables used are listed below:

- A\$, B\$ The two numbers to be multiplied
- Z\$ The final product
- C The carry variable
- L Length of A\$ and B\$ combined, ie length of Z\$
- M Position of digit in B\$ being operated on
- N Position of digit in A\$ being operated on
- A Value of a digit from A\$
- B Value of a digit from B\$
- Z Value of a digit from Z\$
- D Relative magnitude of current value of B (ie 9-units, 1-tens)
- E Relative magnitude of current value of A
- V Product of A and B, plus any 'carry'
- P Relative magnitude of current value of Z

In addition to these, the TIMER is set to zero at the very start of the program to give an indication of the time (in seconds) that any computation needs to be completed.

The method used by the program listing is fairly straightforward if followed against this outline of the method used in each of the lines:

- Lines 20 to 40 — Define initial variables
- Lines 50 to 60 — Take each digit in turn (starting from the right) from B\$ and convert to variable B
- Lines 70 to 80 — Take each digit in turn (starting from the right) from A\$ and convert to variable A; calculate position of cur-

rent value of Z in Z\$- Lines 80 and 90 — Find product of A times B (plus any carry) — Reset carry and check if V is greater than 9
- Lines 100 and 110 — Extract relevant digit from Z\$, and add value V (check to see if new value Z is greater than 9)
- Lines 120 and 130 — Convert digit Z to string V\$ and insert this character back in its correct place in string Z\$

If the string Z\$ contains a leading zero at the left hand end, this is removed at line 170 before the result is printed out, along with the time taken to complete the computation. On the Dragon the TIMER variable counts 56 times per second so if it is reset to zero at the start of the program the expression TIMER/56 will give the total running time in seconds.

We are now in a position to use the program to try out a much larger computation. In Listing 2 values at lines 20 and 30 (from Listing 1) have been redefined to two 50 digit numbers to be multiplied together. Because of the extra string space needed a will also be necessary to include a CLEAR 500 command in order to reserve more bytes for string storage (this can be added to line 10 as follows):

10 TIMER=0: CLEAR 500



