Vol.8 No.6 November 1989

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

televisio... Text via Ceefax & pics BBC 9pm news.

words via BBC Ceefax pages 8/6/89

images via BBCtv 9pm News 8/6/89

v.2

© Dorian Marius Goring 23-7-89

CHINESE CRISIS

PM GIVES HK GOVERNOR STRONG ASSURANCES

SIR DAVID WILSON

The Governor of Hong Kong, Sir David Wilson, had talks lasting an hour and a half with Mrs Thatcher in London tonight.

Afterwards he said the PM had turned down his proposal that Hong Kong citizens be allowed the right to come to Britain - but had given "strong assurances" about her commitment to the colony.

The BBC's Diplomatic Correspondent says the Government believes the situation in China is unclear and ...e no hasty decisions on ...nd to Hong Kong's concerns.

British Nationality 1981

~~LI PENG ON TV~~

Chinese Premier Li Peng has appeared on television for the first time since the weekend massacre in Peking. Mr Peng ... encouraging troops who had been sent into the capital to put an end to the pro-democracy demonstrations.

He said he hoped the troops would continue to work hard to preserve peace and order in the capital.

A sound recordist and cameraman were roughed up by the police while filming in Peking this morning.

POLISH ELECTION RESULTS SENATE

Instant Publishing

- BASIC ROM IMAGE CREATOR
- BINGO CALLER
- DISC FILE IDENTIFIER
- A GOOD REPORT

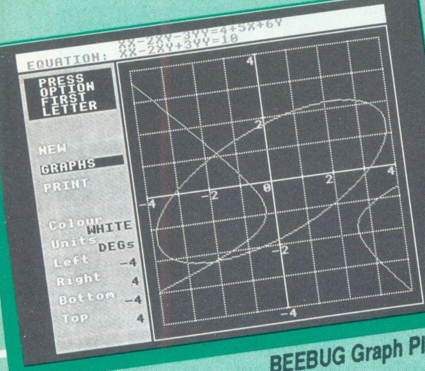
BEEBUG

Vol.8 No.6 November 1989

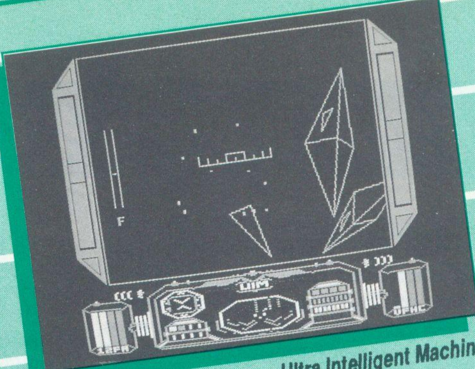
PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

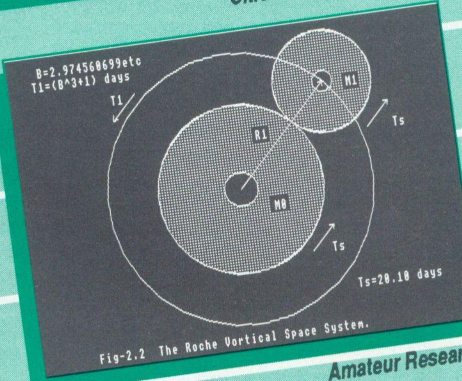
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



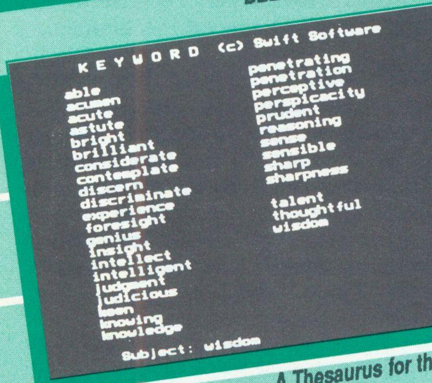
BEEBUG Graph Plotter



Ultra Intelligent Machine

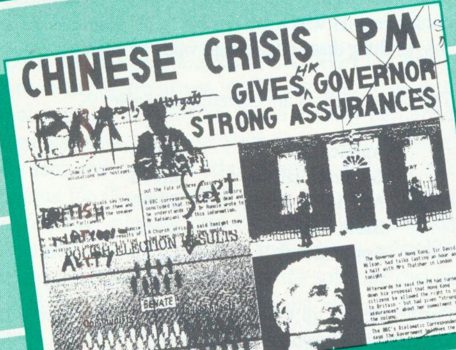


Amateur Research

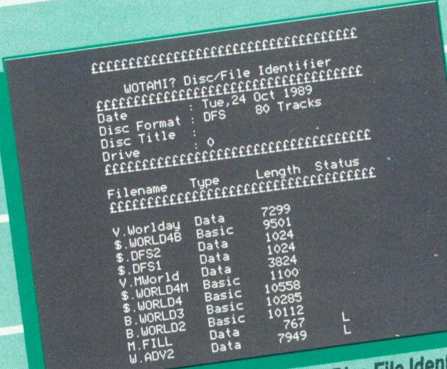


```
KEYWORD (c) Swift Software

able              penetrating
acumen            penetration
acute             perceptive
astute            perspicacity
bright            prudent
brilliant         reasoning
considerate       sense
contemplate       sensible
discern           sharp
discriminate      sharpness
experience
foresight         talent
genius            thoughtful
insight           wisdom
intellect
intelligent
judgment
judicious
keen
knowing
knowledge

Subject: wisdom
```

A Thesaurus for the Beeb



Instant Publishing



```
££££££££££££££££££££££££££££££££££££££££
     WOTAMI? Disc/File Identifier
££££££££££££££££££££££££££££££££££££££££
Date         : Tue,24 Oct 1989
Disc Format  : DFS    80 Tracks
Disc Title   :
Drive        : 0
££££££££££££££££££££££££££££££££££££££££
Filename     Type      Length  Status
££££££££££££££££££££££££££££££££££££££££
V.Worldday   Data       7299
$.WORLD4B    Basic      9501
$.DFS2       Data       1024
$.DFS1       Data       1024
V.MWorld     Data       3824
$.WORLD4M    Basic      1100
$.WORLD4     Basic     10558
B.WORLD3     Basic     10285
B.WORLD2     Basic     10112
M.FILL       Data        767    L
W.ADV2       Data       7949    L
```

Disc File Identifier

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

NEW BEEBUG OPEN DAY

We are now planning a further Open Day to take place on Sunday 3rd December starting again at 10am. We hope that this will not only enable us to meet even more members than before, but provide an excellent opportunity for your Christmas shopping. So far Acorn Computers, Acorn User, Colton Software, Computer Concepts, Computerware and Minerva Software have all confirmed that they will be present to demonstrate their latest products. Note the date in your diaries now - full details are being circulated with the magazine.

A feature of the Open Day will be BEEBUG's own Car Boot Sale where a wide variety of end-of-line products, demo equipment and other items will be on sale at low, low prices. This is a genuine sale and will cover books, manuals, software and hardware of all kinds.

Although the Open Day provides an excellent opportunity to purchase goods from us pre-Christmas, with all the benefits of staff on hand to advise you, our mail-order service will be fully available to satisfy the needs of those unable to attend in person.

We guarantee that all orders received by 14th December will be despatched in time for Christmas (stock permitting). Don't forget that you can also use our FAX number, (0727) 60263, for sending credit card orders to us.

BEEBUG SUMMER COMPETITION

In the July issue of BEEBUG we announced our Summer Competition with a first prize of an A3000 colour system, generously provided by Acorn. A number of other companies including Clares Micro Supplies, Colton Software, Computer Concepts and Minerva Software have also donated prizes for the runners up.

There were many entries from both BEEBUG and RISC User members, which have taken much longer to evaluate than expected, but we have now been able to select a winner whom we believe achieved the best program overall. He is Paul Warren of Bridgwater in Somerset who used a BBC model B. The runners up are Dr.R.Murphy (BBC model B), Mr.P.Dodgshon (model B), Mrs.A.Miskin (Master) and Mr.G.Hopkins (model B).

We hope to publish more details of the winning entries **in** due course, but it is interesting to note that although an equal number of entries was received from both BEEBUG and RISC User members, only four Archimedes owners reached the final short list of twenty.

BEST OF BEEBUG

Over the last twelve months BEEBUG magazine has produced a number of software products, under the general title Best of BEEBUG, the latest being Applications II (first announced last month). The aim of these products is to collect together a selection of previously published programs on a particular theme, and to supply these with documentation as appropriate to members at a low price (relative to any commercial value).

In each case, the programs are updated in the light of any improvements or extensions which have subsequently come to light, with any further extensions or enhancements which we believe to be desirable. As such, the resulting discs offer except value for money.

The following discs are currently available:

Applications	**£5.75**
General Utilities	**£5.75**
ASTAAD 3 (CAD)	**£9.95**
Basic Booster	**£6.00**
Applications II	**£5.75**

Further details on these products and how to order them are included elsewhere in this issue.

BEEBUG MAGAZINE DISC

As we approach Christmas we are aiming to increase your entertainment by including some additional items on the monthly magazine disc. For this issue, we have provided Clowns, an intriguing arcade style game that will test your keyboard skill to the utmost. The disc still remains at the same price of £4.75 (inc. VAT).

News News News News News News

STRANDED BY ROBICO

Already well known for its adventure games, Robico has launched a further game for the BBC micro called *Stranded*. In fact this is not completely new but an extensive rewrite of the former Heyley release of the same name. There is also a brand new Archimedes version as well. Prices (disc only) are £17.95 inc. VAT for the BBC and Master versions, £29.95 inc. VAT for the Archimedes version. Robico are at 3 Fairland Close, Llantrisant, Mid Glamorgan CF7 8QH, tel. (0443) 227354.

GET INVOICED WITH APRICOTE STUDIOS

Apricote Studios has now made available version 3 of its popular and successful Account Book (reviewed in BEEBUG Vol.7 No.5). Moreover, Apricote has also released a new complementary package called simply *The Invoice Program*. This is menu driven like *the Account Book* with four sub-menus covering invoices, customer database, statements and utilities. Either package costs £27.95 each, or the two together for £49.95 (inc. VAT in both cases). versions of both packages are now available too for the Archimedes at the same prices. Contact Apricote Studios at 2 Purls Bridge Farm, Manea, Cambs. PE15 0ND, tel. (035 478) 432.

MORE THAN JUST A STOCKING FILLER

More budget business software has also been announced by Topologika, better known perhaps in the games market. Its new *Stock Pack* is claimed to be a fully featured business administration package with versions for PC compatibles, the Archimedes range and for BBC micros. *Stock Pack* deals with invoices, payments, customer balances and debtors, price lists, and stock control. All versions of the software cost just £29.95 inc. VAT from Topologika at P.O. Box 39, Stilton, Peterborough PE7 3RL, tel.(0733) 244682.

ACORN RELEASE NEW PRODUCT FOR THE MASTER

In an unexpected move Acorn Computers has announced the release of a new 1Mbit ROM for the Master 128 series. This is a plug-in replacement for the existing Acorn ROM. The new ROM provides support for international character sets, and incorporates an improved version of the ADFS disc system which runs at about twice the speed of the original. There is also auto-relocation for Basic and Edit when using co-processors, format and verify in ROM, and a faster version of Basic which is understood to be the same as that fitted to the Master Compact. The new ROM will retail at £44.85 inc VAT. Interestingly, Acorn will not (for the time being at least), be fitting the new ROM to new Masters. Acorn Computers are at Cambridge Technopark, 645 Newmarket Road, Cambridge CB5 8PD, tel. (0223) 214411.

MORE RESOURCES FOR SCHOOLS

At its third Conference, this time entitled IT Across the National Curriculum, RESOURCE (the Doncaster based consortium of local education authorities from South Yorkshire and Humberside) was due to show off its latest products. These include *Perfus*, a set of materials for secondary language work, *Moving*, a pack for lower primary science, and *Earth in Space*, a compilation of materials for secondary science classes. For more details contact RESOURCE at Exeter Road, Doncaster DN2 4PY.

SUPERIOR SOCCER

Strategically announced in the run up to Christmas, by Superior Software, a continuing supporter of the BBC micro, is its latest game *Superior Soccer* combining the best of the arcade soccer games with all the features of football management games. Superior expect this to be a top seller this year. Prices range from £9.95 for Electron and BBC micro cassette, £11.95 for BBC micro 5.25" disc, to £14.95 for 3.5" disc for the Master Compact. All prices include VAT. Superior are at P.O.Box 6, Brigg, South Humberside DN20 9NH.

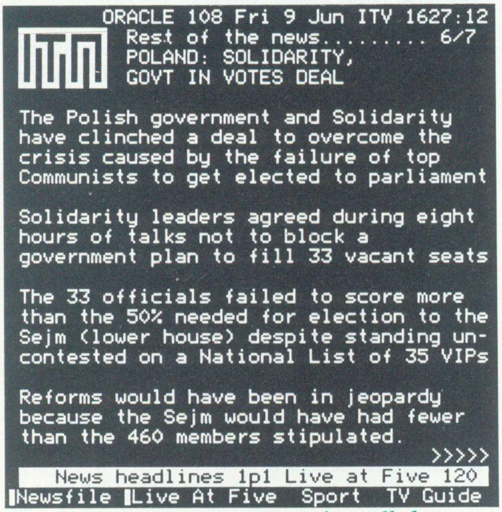
512 MICE

A mouse driver specially written for the 512 board has been released by a small company called Tull Computer Services. This driver will at last allow 512 users to use a mouse with many standard PC applications, instead of using the keyboard. It is particularly useful with some programs such as DeluxePaint II which will not work at all without a mouse attached. We hope to be able to review this in the future through our 512 Forum pages. The mouse driver with supporting manual costs £30 inclusive from Tull Computer Services, 49 Gammons Lane, North Watford, Herts WD2 5BY.

Instant Publishing

Dorian Goring explains how he electronically mixes readily available sources of text and graphics in instant publishing.

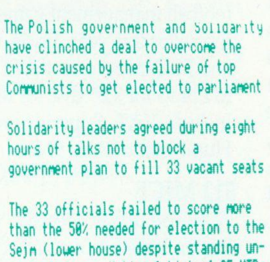
Making the most of what you've got in an innovative and imaginative way is nothing new, but applied to information technology it opens up a cluster of hybrid possibilities.



```
ORACLE 108 Fri 9 Jun ITV 1627:12
      Rest of the news........ 6/7
      POLAND: SOLIDARITY,
      GOVT IN VOTES DEAL

The Polish government and Solidarity
have clinched a deal to overcome the
crisis caused by the failure of top
Communists to get elected to parliament

Solidarity leaders agreed during eight
hours of talks not to block a
government plan to fill 33 vacant seats

The 33 officials failed to score more
than the 50% needed for election to the
Sejm (lower house) despite standing un-
contested on a National List of 35 VIPs

Reforms would have been in jeopardy
because the Sejm would have had fewer
than the 460 members stipulated.
                                 >>>>>
   News headlines 1p1 Live at Five 120
Newsfile Live At Five  Sport  TV Guide
```

A teletext page contents gives all the information you need

The 'Kida' front page emulation illustrated here was composed within a desktop publishing environment using a BBC Master. The words were downloaded from BBC2's Ceefax news pages, and the pictures were digitised from BBC1's Nine O'clock News on the same day in early June.

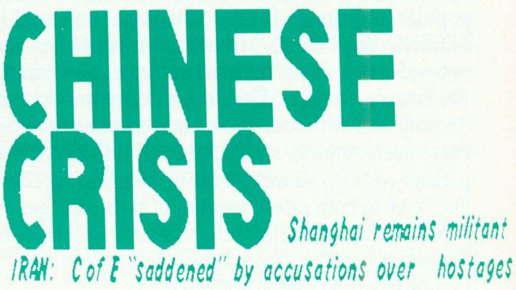


```
The Polish government and Solidarity
have clinched a deal to overcome the
crisis caused by the failure of top
Communists to get elected to parliament

Solidarity leaders agreed during eight
hours of talks not to block a
government plan to fill 33 vacant seats

The 33 officials failed to score more
than the 50% needed for election to the
Sejm (lower house) despite standing un-
contested on a National List of 35 VIPs

Reforms would have been in jeopardy
because the Sejm would have had fewer
than the 460 members stipulated.
```

The teletext file loaded into a word processor

This hybrid approach also highlights how computer technology makes repetition redundant. Apart from constructing the headlines and subheads (produced within AMX StopPress Felttip page font), no drawing or typing was involved - at least, not by me! From start to finish, it took a couple of hours to do.

Teletext data comes ready typed, so there's not much point re-typing it! Likewise, graphically presented summaries (e.g. Polish Election Results, and British Nationality Act 1981) won't gain by being re-drawn.



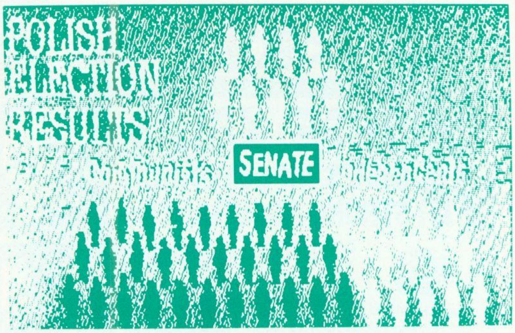
CHINESE CRISIS
Shanghai remains militant
IRAN: C of E "saddened" by accusations over hostages

You can use your favourite desk top publishing package

Teletext and viewdata systems are updated by the minute. This kind of instant publishing can form the basis of a school's pilot in publishing a daily newspaper. Local news and features are added hour-by-hour so that the paper could be ready for sale and distribution as students left the building at the end of the school day. You can't get more topical than that!

And, in Art, it is also a very efficient way of teaching page layout and design - the complex art of fitting page elements together such as rules, pictures, body-text, headlines, working white, clip-art, and tint panels, into a pleasing and harmonious composition.

Spin-off already means there's a local school piloting a newspaper jigsaw game where students compose pages from items in a box

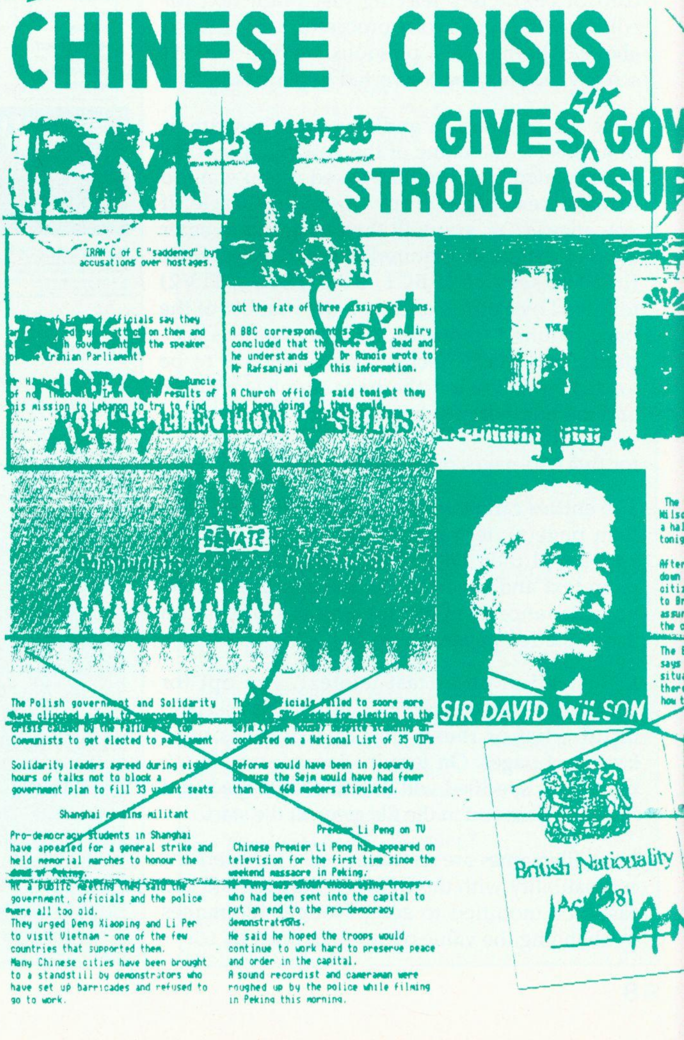


A digitised image ↑

Planning the page →

commercially available software. So, like many teachers, I researched BBC Basic and wrote my own 'grab and strip' program (called GrabIt) for spooling teletext pages for word processing.

This program (see end of article) accesses teletext pages via a GIS BBC Advanced Teletext Receiver, strips control codes, producing a file that can be imported straight into your favourite DTP software in galley strip form. You can then access television news and download pictures to go with the stories collected. This way, you can quickly build up an interesting, up-to-the-minute front page.


CHINESE CRISIS
HK
GIVES GOV
STRONG ASSU
POLISH ELECTION RESULTS
SENATE
SIR DAVID WILSON
British Nationality Act 81

IRAN C of E "saddened" by accusations over hostages.

out the fate of three missing Iranians.

A BBC correspondent said the inquiry concluded that the three were dead and he understands that Dr Runcie wrote to Mr Rafsanjani with this information.

A Church official said tonight they had been doing all they could.

The Polish government and Solidarity ~~have clinched a deal to overcome the crisis caused by the failure of top~~ Communists to get elected to parliament

Solidarity leaders agreed during eight hours of talks not to block a government plan to fill 33 vacant seats

The [illegible] officials failed to score more ~~than the 50% needed for election to the Sejm (lower house) despite standing un~~contested on a National List of 35 VIPs

Reforms would have been in jeopardy because the Sejm would have had fewer than the 460 members stipulated.

Shanghai remains militant

Pro-democracy students in Shanghai have appealed for a general strike and held memorial marches to honour the ~~dead of Peking.~~ ~~At a public meeting they said the~~ government, officials and the police were all too old.
They urged Deng Xiaoping and Li Per to visit Vietnam - one of the few countries that supported them.
Many Chinese cities have been brought to a standstill by demonstrators who have set up barricades and refused to go to work.

Premier Li Peng on TV

Chinese Premier Li Peng has appeared on television for the first time since the weekend massacre in Peking. ~~Li Peng was shown encouraging troops~~ who had been sent into the capital to put an end to the pro-democracy demonstrations.
He said he hoped the troops would continue to work hard to preserve peace and order in the capital.
A sound recordist and cameraman were roughed up by the police while filming in Peking this morning.

based on the 'Kida' idea, or coming through a computer teletype emulation. And, I've used Oracle's astrology pages for the school's parent-teacher magazine. Using this method it is easy to generate a valuable resource of electronic words and pictures, not just in news, for use across the curriculum.

From a Media Education view point, a hybrid approach to educational technology is an important attitude to establish in any educational establishment's policy on cross-curricular IT.

Here, it means thinking of each teletext page as a string of ASCII codes mixed with control codes. Think of a TV picture as a string of dots. Scanners and digitisers effectively remove picture controls so the data can be formatted to suit the graphics mode you're working in. Likewise, it is possible to remove controls from teletext pages leaving a file which can be loaded into a word processor and manipulated.

One drawback to the hybrid approach is that often there is no

Digitised images and spooled teletext have further advantages when used within a DTP environment. Having most of the work done for you concentrates attention on page composition.

No DTP environment can turn a novice into a successful page layout artist overnight, but the hybrid approach is a useful 'top-down' approach, putting the whole before its parts.

USING THE GRABIT PROGRAM

The program listed at the end of this article allows suitable text screens to be downloaded from Ceefax or Oracle, and then strips out all control characters leaving just plain text for editing with your word processor as described above. Alternatively, previously saved teletext screens may also be converted to plain text.

Type the program in and save it. When run, it first asks for the name of the new plain text file to be created. You are then asked if you want to load teletext pages from Ceefax or Oracle. If you answer 'Y', you will then be ásked to enter a list of the pages required. For each page enter first the service (BBC1, BBC2, ITV1, ITV2) followed by Return, and then the number of the page required (three digits). If a page has sub-pages you may specify how many are to be loaded by following the page number with 'S' and the number of sub-pages, for example:

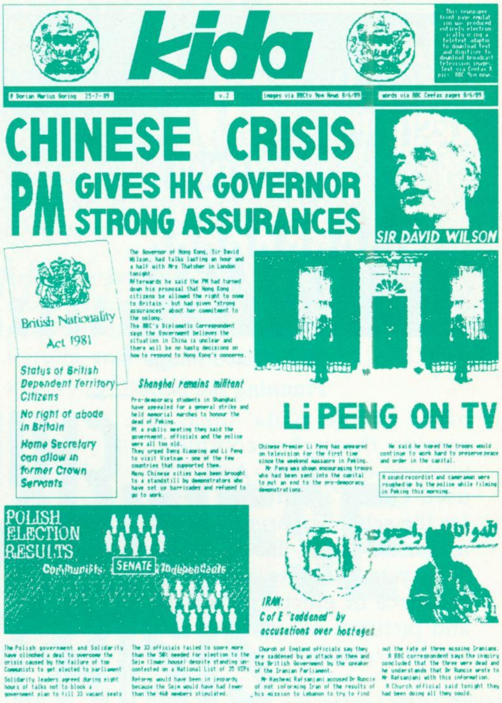
```
119S5
```

for five sub-pages of page 119.

The entries can be repeated, giving service first, then page(s), terminating input with '/'. The pages will then be downloaded, converted to plain text and saved in the file specified. Note, no allowance is made for any corruption on down-loaded pages.

If, instead of broadcast teletext, you opt for previously saved screens, you will be asked to input a list of these, one at a time, as with broadcast pages. On terminating input with '/', the screens specified will be loaded, stripped, and saved as plain text in the file given at the start.

Note, file names are restricted to 7 characters for compatibility with the DFS, but the program can easily be modified to accept longer file names by changing the value of numb in line 110.



kida

This newspaper front page emulation was produced entirely electronically using a teletext adaptor to download text and digitiser to download broadcast television images. Text via Ceefax & pics: BBC 9pm news.

@ Dorian Marius Goring 25-7-89 | v.2 | images via BBCtv 9pm News 8/6/89 | words via BBC Ceefax pages 8/6/89

CHINESE CRISIS

PM GIVES HK GOVERNOR STRONG ASSURANCES

SIR DAVID WILSON

British Nationality Act 1981

Status of British Dependent Territory Citizens

No right of abode in Britain

Home Secretary can allow in former Crown Servants

The Governor of Hong Kong, Sir David Wilson, had talks lasting an hour and a half with Mrs Thatcher in London tonight.
Afterwards he said the PM had turned down his proposal that Hong Kong citizens be allowed the right to come to Britain - but had given "strong assurances" about her commitment to the colony.
The BBC's Diplomatic Correspondent says the Government believes the situation in China is unclear and there will be no hasty decisions on how to respond to Hong Kong's concerns.

Shanghai remains militant

Pro-democracy students in Shanghai have appealed for a general strike and held memorial marches to honour the dead of Peking.
At a public meeting they said the government, officials and the police were all too old.
They urged Deng Xiaoping and Li Peng to visit Vietnam - one of the few countries that supported them.
Many Chinese cities have been brought to a standstill by demonstrators who have set up barricades and refused to go to work.

LiPENG ON TV

Chinese Premier Li Peng has appeared on television for the first time since the weekend massacre in Peking.
Mr Peng was shown encouraging troops who had been sent into the capital to put an end to the pro-democracy demonstrations.
He said he hoped the troops would continue to work hard to preserve peace and order in the capital.

A sound recordist and cameraman were roughed up by the police while filming in Peking this morning.

POLISH ELECTION RESULTS

Communists SENATE Independents

IRAN:
C of E "saddened" by accusations over hostages

The Polish government and Solidarity have clinched a deal to overcome the crisis caused by the failure of top Communists to get elected to parliament
Solidarity leaders agreed during eight hours of talks not to block a government plan to fill 33 vacant seats
The 33 officials failed to score more than the 50% needed for election to the Sejm (lower house) despite standing uncontested on a National List of 35 VIPs
Reforms would have been in jeopardy because the Sejm would have had fewer than the 460 members stipulated.

Church of England officials say they are saddened by an attack on them and the British Government by the speaker of the Iranian Parliament.
Mr Hashemi Rafsanjani accused Dr Runcie of not informing Iran of the results of his mission to Lebanon to try to find out the fate of three missing Iranians.
A BBC correspondent says the inquiry concluded that the three were dead and he understands that Dr Runcie wrote to Mr Rafsanjani with this information.
A Church official said tonight they had been doing all they could.

Final copy

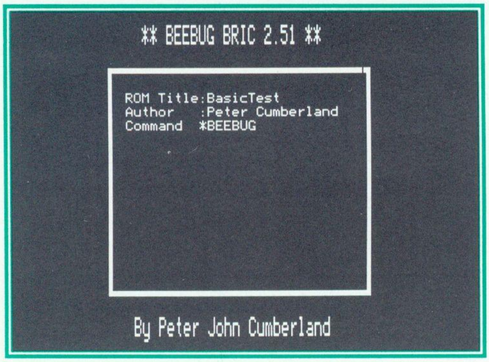
Do check the copyright of material taken from various sources. BEEBUG cannot be held responsible for the consequences of unauthorised use of any such material.

```
   10 REM Program GrabIt
   20 REM Version B1.1
   30 REM Author  Dorian Goring
   40 REM BEEBUG  November 1989
   50 REM Program subject to copyright
   60 :
  100 MODE7:ON ERROR CLOSE#0:MODE7:REPOR
T:PRINT" at line ";ERL:END
  110 VDU23,1,0;0;0;0;:numb=7
  120 A$="BBC1BBC2ITV1ITV2"
  130 B$=CHR$(&94)+CHR$(&B7)+STRING$(&25
,CHR$(&A3))+CHR$(&EB)
  140 C$=CHR$(&94)+CHR$(&F5)+STRING$(&25
,CHR$(&F0))+CHR$(&FA)
  150 D$=CHR$(&94)+CHR$(&B5)+CHR$(&86)
  160 E$=CHR$(&94)+CHR$(&EA):F$=CHR$(&8D
)+CHR$(&81)+"GrabIt"
  170 :
  180 REM Menu
  190 PROCbox
  200 PRINTTAB(0,9);"New filename ";:INP
UTname$:IF LEN(name$)<1 OR LEN(name$)>nu
mb GOTO 200
  210 REPEAT:CLS:INPUTTAB(0,9)"TV Telete
xt (Y/N)?"tv$:UNTIL INSTR("YyNn",tv$)
  220 tv$=CHR$(ASC(tv$)AND&5F)
  230 dBASE=OPENOUT("X"):sfile%=OPENOUT(
name$)
  240 :
  250 REM Enter data
  260 CLS:R%=1
  270 REPEAT
  280 IF tv$="N" PRINTTAB(0,9)"SCREEN na
me "+STR$(R%)+":"; ELSE PRINTTAB(0,9)"TE
LETEXT data "+STR$(R%)+":";
  290 M$="":P%=0:PRINTTAB(16,9)STRING$(n
umb,"."):PRINTTAB(16,9);
  300 REPEAT:G%=GET
  310 IF G%=&7F AND P%>0 VDU8,46,8:M$=LE
FT$(M$,P%-1):P%=P%-1 ELSE IF G%>=32 AND
G%<127 AND P%<numb M$=M$+CHR$(G%):P%=P%+
1:VDU G%
  320 UNTIL G%=&0D OR G%=&2F:IF G%=&2F G
OTO 430
  330 :
  340 REM check_data
  350 IF (tv$="N" AND LEN(M$)>0) GOTO 40
0
  360 IF LEFT$(M$,1)<"A" GOTO370 ELSE IF
 INSTR(A$,M$)<>0 GOTO400 ELSE GOTO290
  370 IF LEN(M$)<4 M$=M$+"S3F7F"
  380 IF (MID$(M$,4,1)<>"S") OR (LEN(M$)
<5) GOTO290
  390 :
  400 FOR count%=1 TO LEN(M$):BPUT#dBASE
,ASC(MID$(M$,count%,1)):NEXT
  410 IF tv$="N" BPUT#dBASE,&23 ELSE BPU
T#dBASE,&20
  420 PRINTTAB(25,R%);CHR$(R%+48);">";M$
:R%=R%+1
  430 UNTIL G%=&2F
  440 :
  450 REM keep it
  460 BPUT#dBASE,&7F:EXT#dBASE=PTR#dBASE
:CLOSE#dBASE
  470 PRINTTAB(0,15);CHR$(&83);"OK? Y/N
":VDU7:VDU31,9,17:IFGET$="N"CLOSE#0:RUN
  480 :
  490 REM Read database
  500 dBASE=OPENIN("X")
  510 M$=""
  520 IF EOF#dBASE GOTO 770
  530 G$=CHR$(BGET#dBASE):IF G$=" " GOTO
 570 ELSE IF G$="#"CLS:OSCLI("LOAD "+M$)
:PROCstrip:GOTO510
  540 M$=M$+G$:GOTO520
  550 :
  560 REM Grab it
  570 p$=M$
  580 :
  590 REM Get tetetext service
  600 IF LEN(p$)<5 OSCLI("TTXON "):OSCLI
(p$):OSCLI("TTXOFF "):GOTO510
  610 s$=MID$(p$,5,LEN(p$)):IF s$="3F7F"
 GOTO 670
  620 :
  630 REM Subpage
  640 count%=1
  650 IF count%<10 THEN zero$="000" ELSE
 zero$="00"
  660 sub$=STR$(count%):p$=LEFT$(p$,3)+z
ero$+sub$
  670 PRINT STRING$(7," ");TAB(0,0)"P"+p
$:REM print page addr
  680 OSCLI("TTXON "):OSCLI("HON 0 ")
  690 OSCLI("TPAGE "+p$)
  700 OSCLI("TRANSFER FFFF7000 ")
  710 OSCLI("DISPLAY FFFF7000 ")
  720 OSCLI("TTXOFF ")
  730 PROCstrip
  740 IF s$="3F7F" GOTO510 ELSE IF count
%<VAL(s$)count%=count%+1:GOTO650
  750 GOTO510
  760 :
  770 REM End
  780 EXT#sfile%=PTR#sfile%:CLOSE#sfile%
:CLOSE#0
  790 MODE7
  800 END
  810 :
 1000 DEF PROCbox
 1010 CLS:FOR r%=2 TO 3:PRINTTAB(28,r%);
F$:NEXT
 1020 PRINTTAB(0,4)B$;TAB(0,22);C$
 1030 FOR r%=5 TO 21:PRINTTAB(0,r%)D$;TA
B(38,r%)E$:NEXT
 1040 VDU28,3,21,37,5
 1050 ENDPROC
 1060 :
 1070 DEF PROCstrip
 1080 FOR row%=&7C00 TO &7FBF STEP &28
 1090 IF ?row%>&8E AND ?row%<&A0 GOTO980
 1100 FOR screen%=row% TO row%+&27
 1110 ?screen%=(?screen% AND &7F)
 1120 IF ?screen%<&20 ?screen%=&20
 1130 BPUT#sfile%,?screen%
 1140 NEXT screen%
 1150 BPUT#sfile%,&0D
 1160 NEXT row%
 1170 ENDPROC
```

Basic ROM Image Creator

Put your Basic programs into sideways RAM or ROM with this utility from Peter Cumberland.

This Basic ROM Image Creator (BRIC) allows you to create ROM images of Basic programs. It is also possible to convert assembler programs that use large chunks of Basic. The resulting ROM image may be loaded into Sideways RAM (SWR) if you have it, or blown into an EPROM.



```
** BEEBUG BRIC 2.51 **

ROM Title:BasicTest
Author   :Peter Cumberland
Command  *BEEBUG

By Peter John Cumberland
```

Converting a Basic program to a ROM image

BRIC also ensures a simple way of getting your programs into Sideways RAM or ROM. The method by which you load the image into Sideways RAM depends on which type of machine you are using. On the model B you should consult your SWR User Guide. If you are using a Master or a Compact then you should use the *SRLOAD command:

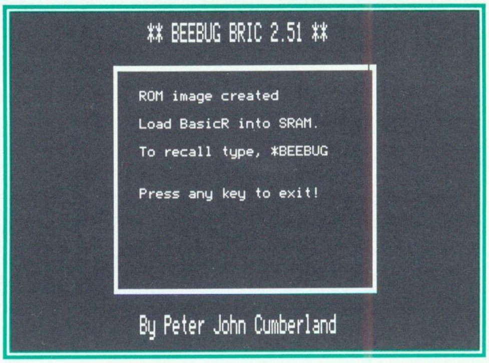
```
*SRLOAD <rom> 8000 <x> Q I
```

where <rom> is the ROM image filename stored on disc and <x> is the ROM slot.

Once the ROM image has been installed or loaded into sideways RAM, a simple user-selected star command is all that is required to reload the Basic program back into main memory and start immediate execution. Being in ROM format also ensures that the Basic program is always instantly available for use.

Type in the listing and save it as BRIC. Now run the program and you will be prompted to supply all the information required. BRIC first asks you to enter the name of the Basic program which you want to convert (and then press Return). The specified program will be *LOADed into memory at 3700.

The next question asks for the PAGE address. This is the address at which you would like the image to load back. On the model B with DFS this is normally &1900. On the Master and Compact it is &E00.



```
** BEEBUG BRIC 2.51 **

ROM image created
Load BasicR into SRAM.
To recall type, *BEEBUG

Press any key to exit!

By Peter John Cumberland
```

The program reports when the ROM image is complete

Next, the text window will clear and you will be requested to enter the ROM name. This can be up to 10 characters in length. The name will subsequently be displayed in response to the *ROMS command (and others).

The next question asks for the author's name, so enter your own name here. Then BRIC asks for the star command which will activate your program. Enter just the name - you don't need the asterisk. Whenever this command is issued, your program will be instantly available, and you can also just list or run your program.

The final question asks for the disc file name for the ROM image. Enter a name and press Return. Your image will then be stored on disc, and the program will advise you on how to reload it.

And there you have it, the easy way to get your programs into Sideways RAM or ROM.

Note: An additional short program is included on the magazine disc called TESTB which you may use to test out BRIC.

```
   10 REM Program BRIC
   20 REM Version B2.51
   30 REM Author  Peter J Cumberland
   40 REM BEEBUG  November 1989
   50 REM Program subject to copyright
   60 :
  100 ON ERROR VDU22,7:REPORT:PRINT" at
line ";ERL:END
  110 *FX 229,1
  120 MODE7:VDU23,1,0;0;0;0;
  130 w$=CHR$135
  140 FOR x =0 TO 1
  150 PRINTTAB(1,x);CHR$141CHR$132CHR$15
7CHR$131;SPC2"** BEEBUG BRIC 2.51 **
  ";CHR$156
  160 NEXT x
  170 PRINTTAB(1,21);
  180 PRINTCHR$141CHR$132CHR$157CHR$131"
 By Peter John Cumberland      ";CHR$156
  190 PRINTTAB(1,22);
  200 PRINTCHR$141CHR$132CHR$157CHR$131"
 By Peter John Cumberland      ";CHR$156
  210 PROCbox(2,19,35,3,3,7)
  220 :
  230 K%=1:Q%=10:OSBYTE=&FFF4
  240 INPUTTAB(1,1);"Program Name :"F$
  250 OSCLI("LOAD "+F$+" 3700")
  260 S%=&C00:F%=&C20:$F%=F$+CHR$(13)
  270 S%?0=F%MOD256:S%?1=F%DIV256:A%=5
  280 X%=S%MOD256:Y%=S%DIV256:CALL&FFDD
  290 CALL&FFDD
  300 :
  310 J%=S%?10+256*S%?11
  320 DIM P(4):HIMEM=&3500
  330 J%=J%+&100
  340 INPUTSPC1"PAGE address &"Loc$
  350 r%=EVAL("&"+Loc$):o%=r% DIV 256
  360 :
  370 REM Main PROCedure's start here
  380 PROCinput
  390 PROCassemble
  400 PROCsaverom
  410 END
  420 :
 1000 DEF PROCinput:CLS:PRINT
 1010 INPUT"ROM Title:"T$
 1020 T$=LEFT$(T$,Q%):R%=LEN(T$)
 1030 INPUT"Author    :"A$
 1040 A$=LEFT$(A$,Q%):B%=LEN(A$)
 1050 INPUT"Command   *"C$
 1060 C$=LEFT$(C$,10):com%=LEN(C$)
 1070 ENDPROC
 1080 :
 1090 DEF PROCassemble:VDU 21
 1100 FOR begin%=0 TO 2 STEP 2
 1110 P%=&3500
 1120 [
 1130 OPT begin%
 1140 BRK:BRK:BRK:JMP &8030
 1150 ]
 1160 ?P%=&82:P%=P%+1:?P%=(9+R%):P%=P%+1
 1170 ?P%=00:P%=P%+1:$P%=T$:P%=P%+R%
 1180 ?P%=00:P%=P%+1:?P%=&28:P%=P%+1
 1190 ?P%=&43:P%=P%+1:?P%=&29:P%=P%+1
 1200 $P%=A$:P%=P%+B%:P%=&3530
 1210 [OPT begin%
 1220 CMP #&04:BEQ branch
 1230 RTS
 1240 .end
 1250 PLA:TAX:PLA:TAY:PLA:RTS
 1260 RTS
 1270 .branch
 1280 PHA:TYA:PHA:TXA:PHA
 1290 ]
 1300 Z%=P%+3
 1310 FOR bc%=1 TO com%
 1320 [
 1330 OPT begin%
 1340 LDA (&F2),Y:CMP #&00
 1350 BNE end:INY
 1360 ]
 1370 NEXT bc%
 1380 [
 1390 OPT begin%
1400 LDA (&F2),Y:CMP #&0D:BNE end
1410 LDA #o%:STA &71:LDA #&82:STA &73
1420 LDA #&00:STA &70:STA &72
1430 LDY #&FF
1440 .continue
1450 ]
1460 I%=P%
1470 [
1480 OPT begin%
1490 LDA &73:CMP #&00
1500 BEQ Lloop:INY
1510 LDA (&72),Y:STA (&70),Y
1520 CPY #&FF:BNE continue
1530 INC &71:INC &73
1540 CLC:BCC continue
1550 .Lloop
1560 INY:CPY #&00:BEQ fini
1570 LDA (&72),Y:STA (&70),Y
1580 CLC:BCC Lloop
1590 .fini
1600 LDX #&00:LDA #&82:LDY #&50
1610 JSR &FFF4
1620 LDY #&41:JSR OSBYTE
1630 LDY #&2E:JSR OSBYTE
1640 LDY #&3D:JSR OSBYTE
1650 LDY #&26:JSR OSBYTE
1660 ]
1670 L=LEN(Loc$)
1680 IF L=3 THEN Loc$="0"+Loc$
1690 FOR J=1 TO 4
1700 P(J)=ASC(MID$(Loc$,J,1))
1710 NEXT J
1720 [
1730 OPT begin%
1740 LDY #P(1):JSR OSBYTE
1750 LDY #P(2):JSR OSBYTE
1760 LDY #P(3):JSR OSBYTE
1770 LDY #P(4):JSR OSBYTE
1780 LDY #&D:JSR OSBYTE
1790 LDY #&4F:JSR OSBYTE
1800 LDY #&4C:JSR OSBYTE
1810 LDY #&44:JSR OSBYTE
1820 LDY #&0D:JSR OSBYTE
1830 LDY #&52:JSR OSBYTE
1840 LDY #&55:JSR OSBYTE
1850 LDY #&4E:JSR OSBYTE
1860 LDY #&0D:JSR OSBYTE
1870 PLA:TAX:PLA:TAY:PLA
1880 LDA #138:LDX #0:JSR OSBYTE
1890 LDY #10:JSR OSBYTE
1900 LDY #ASC"O":JSR OSBYTE
1910 LDY #ASC"L":JSR OSBYTE
1920 LDY #ASC"D":JSR OSBYTE
1930 LDY #13:JSR OSBYTE
1940 LDY #ASC"R":JSR OSBYTE
1950 LDY #ASC"U":JSR OSBYTE
1960 LDY #ASC"N":JSR OSBYTE
1970 LDY #13:JSR OSBYTE
1980 LDA #&00
1990 RTS
2000 ]
2010 NEXT begin%
2020 VDU6
2030 FOR begin%=1 TO com%
2040 ?Z%=ASCMID$(C$,begin%,1)
2050 Z%=Z%+7:NEXT begin%
2060 A=(J%DIV256)+&82
2070 B=(J%MOD256)
2080 I%?3=A:I%?24=B
2090 ENDPROC
2100 :
2110 DEF PROCsaverom:CLS:PRINT
2120 INPUT SPC1"Enter filename:"Fn$
2130 IF LEN(Fn$)>7THENVDU7:GOTO2120
2140 IF &200+J%<&4000 THEN I$="4000"
2150 IF &200+J%<&2000 THEN I$="2000"
2160 IF &200+J%<&1000 THEN I$="1000"
2170 IF &200+J%<&400  THEN I$="400"
2180 OSCLI("SAVE "+Fn$+" 3500+"+I$+" D9
CD 8000")
2190 CLS
2200 PRINT'SPC1"ROM image created"
2210 PRINT'SPC1"Load ";Fn$;" into SRAM.
"
2220 PRINT'SPC1"To recall type,";CHR$13
1;"*";C$
2230 PRINT''SPC1"Press any key to exit!
"
2240 *FX 229,0
2250 £$=GET$:CALL !-4
2260 ENDPROC
2270 :
2280 DEF PROCbox(a%,b%,c%,d%,c1%,c2%)
2290 VDU26
2300 FOR y%=d%+1 TO b%-1
2310 PRINTTAB(a%,y%)CHR$(144+c1%)"5"CHR
$(128+c2%);
2320 PRINTTAB(c%-2,y%)CHR$(144+c1%)CHR$
(106)w$;
2330 NEXT
2340 PRINTTAB(a%,d%)CHR$(144+c1%)CHR$(6
0)STRING$(c%-a%-3,",")CHR$(108)w$;
2350 PRINTTAB(a%,b%)CHR$(144+c1%)CHR$(4
5)STRING$(c%-a%-3,",")CHR$(46)w$;
2360 VDU28,a%+3,b%-1,c%-3,d%+1,12
2370 ENDPROC
```

A BEEBUG Graph Plotter (2)

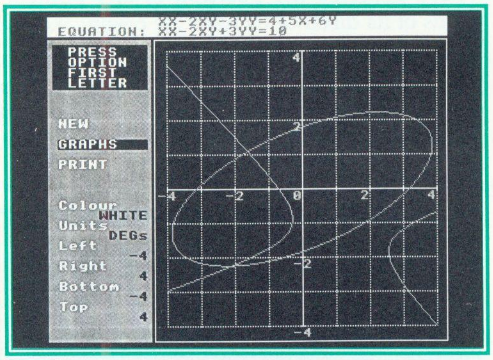
by Robin Murphy

In part 1 of this article, published last month, an easy-to-use graph plotting program was described which was capable of drawing the graphs of functions. This second part extends its repertoire to the graphs of relations, families of curves, and those defined parametrically and using polar co-ordinates.

Before typing in the new listing it is prudent to rename last month's program GRPLOT1 by typing:

```
    *RENAME GRPLOT1 GRPLOT0
```

so that it continues to be available if things go badly wrong. It should then be loaded, the additional program lines entered and the whole re-saved as GRPLOT1. Once again care should be taken to use the same line numbers as used in the listing.



```
XX-2XY-3YY=4+5X+6Y
EQUATION: XX-2XY+3YY=10
PRESS
OPTION
FIRST
LETTER
NEW
GRAPHS
PRINT
Colour
     WHITE
Units
      DEGs
Left
        -4
Right
         4
Bottom
        -4
Top
         4
```

Plotting Quadratics

When GRPLOT is run it should behave exactly as before except that a greater variety of equations can now be processed.

CARTESIAN RELATIONS

Provided that they can be evaluated at most values of x and y, and they represent, at worst, a quadratic in either x or y, the program will plot the graphs defined by relations between x and y. Equations suitable for the default scales are:

`X+2Y=10`	linear in X and Y
`XXY=12`	linear in Y
`XX-3XY-2YY=9`	quadratic in X and Y
`4-XX=X/Y`	quadratic in X

Note that:

`X/Y+Y/X=3` would be "TOO HARD!"

but rearranged into:

`XX+YY=3XY` it becomes quadratic.

The numerical methods used to solve the relations are prone to rounding errors and this can spoil the 'smoothness' of some graphs. That of:

```
    (XX-25)(YY-25)=0
```

is a typical example.

Plotting will always be much quicker if the equation is entered in the form X=... or Y=... permitted last month.

FAMILIES OF GRAPHS

There are many common 'one parameter' families of graphs that we meet in mathematics. These represent a set of curves or lines which include a single letter in their definition. Give different values to the letter and you get a different graph of the family. Common examples are:

$y=mx$	lines of gradient m
$y=x^n$	different powers of x
$x^2+y^2=r^2$	circles of radius r

The program will plot such families as long as the letter used is K. You will be asked to enter the starting value of K (K1), the stopping value (K2) and the step (dK), before any plotting takes place.

Entries suitable for the default scales are:

```
Y=KX       -2 to 2 by  .5
Y=X^K       0 to 4 by  1
XX+YY=KK  10 to 4 by -3
```

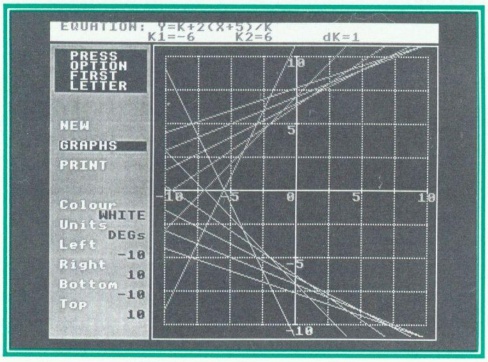
Remember that the Escape key is available if you want to interrupt the plotting.

PARAMETRIC GRAPHS

Parameters are also used to describe points on a particular curve. The program will plot such curves provided the parameter used is 't'. A very simple pair of parametric equations is:

$x=t$
$y=t^2$

Choose any value for t and these give the co-ordinates of a point on the curve. This curve is in fact the parabola $y=x^2$. Parametric form provides the only way of defining some curves.Such equations must be entered giving the x formula first, a colon, and then the y formula. You should type '@' to enter a 't' in the equation.



```
EQUATION: Y=K+2(X+5)/K
   K1=-6     K2=6        dK=1
PRESS OPTION FIRST LETTER
NEW
GRAPHS
PRINT
Colour WHITE
Units DEGs
Left -10
Right 10
Bottom -10
Top 10
```

Plotting Families of curves

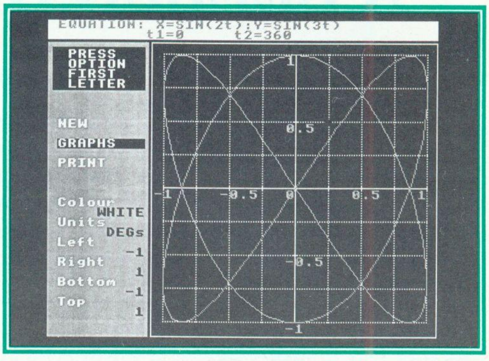
You will also be asked to enter the starting and stopping values of t before any plotting can take place.

Suitable examples for the default screen (using degrees) are:

```
X=10COS@:Y=5SIN@                0 to 360
X=8SIN(2@):Y=8SIN(3@)           0 to 360
X=10@/(1+@+@@):Y=10/(1+@@)     -5 to 5
```

Note that the program automatically substitutes 't' in place of '@' for the screen display. The parametric form may also be used to draw a graph which is defined differently on different intervals. For example, plot:

```
y=( 1       if   -10<x<=1
  ( x       if     1<x<=5
  ( 25/x    if     5<x<=10
```



```
EQUATION: X=SIN(2t):Y=SIN(3t)
      t1=0      t2=360
PRESS OPTION FIRST LETTER
NEW
GRAPHS
PRINT
Colour WHITE
Units DEGs
Left -1
Right 1
Bottom -1
Top 1
```

Plotting equations in parametric form

by:

```
X=@:Y=1           -10 to 1
```

then:

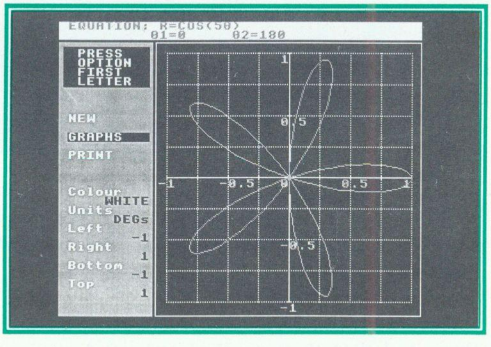
```
X=@:Y=X             1 to 5
```

then:

```
X=@:Y=25/X          5 to 10
```

POLAR CO-ORDINATES

These describe the position of any point in terms of its distance (r) from the origin, and the direction (Θ) measured anticlockwise from a reference line (see figure). If r is negative, it is measured in the opposite direction so that (5,0) and (-5,180) correspond to the same point Q in the figure.

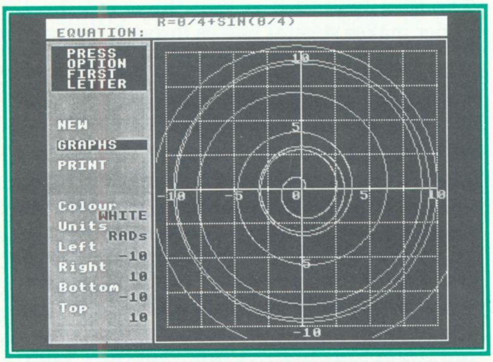


```
EQUATION: R=COS(5θ)
      θ1=0      θ2=180
PRESS OPTION FIRST LETTER
NEW
GRAPHS
PRINT
Colour WHITE
Units DEGs
Left -1
Right 1
Bottom -1
Top 1
```

Figure 1. Use of polar co-ordinates

The angle 'theta' is entered into the equation by pressing the '@' key (this is again automatically replaced for the screen display). Some entries suitable for the default scales are:

```
R=3+6COS@        0     to  360 degrees
R=10SIN@         0     to  180    "
R=8COS(3@)       0     to  180    "
R=5SIN@^2        0     to  360    "
R=@+SIN(5@)      0     to  4π radians
```


EQUATION: R=θ/4+SIN(θ/4)

Colour WHITE
Units RADs
Left -10
Right 10
Bottom -10
Top 10

A typical plot using polar co-ordinates

PROGRAM DETAILS (PART 2)

NEW PROCEDURES

draw	join points on graph of quadratic relation if close enough.
harder	draw parametric or polar graphs.
horiz	draw horizontal line at y value.
quadratic	initiate the solution of the quadratic relation.
range	allow input of the range of a parameter (and step in the case of K).
solve	use method of differences to form a quadratic eqution, then find its solutions v and V.
vert	draw a vertical line at x value.

NEW FUNCTIONS

degree	use method of differences to find degree of x, or y, in the equation. There are four possibilities: 0, 1, 2, or >2.
eval	calculate left-hand-side minus right-hand-side of equation for current x and y values.

```
 1030  IFINSTR($I%,M$)OR("R"=CHR$?I%):PR
OCharder:GOTOD
 1040  K=TRUE:L=0:d=1
 1050  F%=INSTR($I%,"K"):IFF%X$=M$:M$="K
":PROCrange:M$=X$:a$=$I%:L=U:d=w:K=u-d
 1230  W=FNdegree(.1,0):Z=FNdegree(0,.1)
 1240  IFW=0ORZ=0:ENDPROC
 1250  IFZ=1:F=1.5:ENDPROC
 1260  IFW=1:F=1:ENDPROC
 1270  IFZ=2:F=2.5:ENDPROC
 1280  IFW=2:F=2:ENDPROC
 1320  I=FNeval(X,z):G=FNeval(X,z+1):IFI
=0ANDG=0:PROCvert(f*X)
 1330  =z+I/(I-G)
 1360  I=FNeval(z,Y):G=FNeval(z+1,Y):IFI
=0ANDG=0:PROChoriz(g*Y)
 1370  =z+I/(I-G)
 1380  DEFFNeval(X,Y)
 1390  =EVAL$K%-EVAL$J%
 1420  IFF:W=u-4:IF INT(F)=1THEN1600
 1430  IF INT(F)=2THEN1700
 1440  IFW=0THEN1480
 1450  IFW=1:PROCvert(FNx(1)*f):ENDPROC
 1460  IFW=2:F=1:Z=z:PROCquadratic:IFc:P
ROCvert(v):PROCvert(V):ENDPROC
 1470  F=-1:M$="Can't find X":ENDPROC
 1480  IFZ=0:F=-1:M$="SILLY!":ENDPROC
 1490  IFZ=1:PROChoriz(g*FNy(1)):ENDPROC
 1500  IFZ=2:F=.5:Z=z:PROCquadratic:IFc:
PROChoriz(v):PROChoriz(V):ENDPROC
 1510  F=-1:M$="Can't find Y":ENDPROC
 1520  DEFPROCvert(X)
 1530  MOVEX,k:DRAWX,H:ENDPROC
 1540  DEFPROChoriz(Y)
 1550  MOVEj,Y:DRAWJ,Y:ENDPROC
 1700  l=TRUE:c=0
 1710  C=1710:IFW>U:C=D:ENDPROC
 1720  FORW=W+4 TOU STEP4
 1730  Z=W/t:PROCquadratic
 1740  IFc=0ANDl>0PROCdraw(p,q,m,n):l=0:
NEXT:ENDPROC
 1750  IFc=0l=0:VDU7:NEXT:ENDPROC
 1760  IFF=2P=v:M=V:Q=W:N=W:ELSEQ=v:N=V:
P=W:M=W
 1770  IFl=0PROCdraw(P,Q,M,N):ELSEIFl>0P
ROCdraw(p,q,P,Q):PROCdraw(m,n,M,N):IFc<1
:PROCdraw(p,q,M,N):PROCdraw(m,n,P,Q)
 1780  p=P:m=M:q=Q:n=N:l=1:NEXT
 1790  ENDPROC
 1800  DEFPROCquadratic
 1810  IFF=INT(F):PROCsolve(FNeval(z-1,Z
),FNeval(z,Z),FNeval(z+1,Z))ELSEPROCsolv
e(FNeval(Z,z-1),FNeval(Z,z),FNeval(Z,z+1
))
```

Peacock Printer

Reviewed by Dorian Goring

Product	**Peacock Printer**
Supplier	**Datassistance** **83 Main Street, Great Boughton,** **Cockermouth,** **Cumbria, CA13 0YJ.** **Tel. (0900) 825503**
Price	**£8.95 inc.VAT**

Colour separation is a very interesting and creative computer graphics topic (see my article *Multi-Colour Printing* in BEEBUG Vol.8 No.4), and it provides a cheap way of getting colour from a standard 'black & white' dot matrix printer.

HARDWARE

You need a set of coloured ribbons to fit your printer (about £5 each, and available for most dot-matrix printers), and software including a printer driver to turn your colour graphics into separate printable screens.

The maximum number of ribbons you will need is four - yellow, magenta, cyan, and black. These are the industry standards in colour printing. So, for a total outlay of about £30 (including software) you could be printing professional colour, and save the cost of updating your printer!

SOFTWARE

There are several pieces of software available for colour separation using a BBC computer, if you are prepared to search. Two I've previously used were downloaded from Ceefax's Telesoftware service before its sudden demise. Now, Datassistance are marketing *Peacock Printer*, which most timely fills a vacant technology gap.

Peacock Printer is a fairly simple yet powerful and effective means of translating screen colour in any graphics mode into multi-coloured hardcopy for under £30 (including ribbons, available separately).

HOW IT WORKS

The idea behind it is that the same sheet of paper makes several passes through the printer, each time with a different coloured ribbon and colour screen. It is not a difficult process to understand and has a wide number of applications.

The program uses the red function keys to enable you to select which parts you want printed in a particular colour, with pauses to allow you to insert the appropriate coloured ribbon. It then prints these parts, and automatically re-aligns the paper (using reverse-feed on Epson compatibles) and centres the print head ready for the next colour.

HIDDEN DIFFICULTIES

Technically, the software works well (apart from a couple of tiny bugs in the printer driver code, but only single density printing is possible, more's the pity). It is reliable, and fairly simple in operation, but there are important 'hidden' difficulties, and omissions in the documentation (A5 format, 8 pages, folded). These can cause misunderstandings regarding operation (printer set-up and specification), and limit creative exploitation.

In order for the function keys to work (and to exit from the program!), your printer must be on-line with paper installed. This is an unnecessary limitation. Initially, I wanted to explore colour possibilities, and experiment with different colour combinations in different modes just to see how it all worked and the effects I could create.

However, being forced into using my printer, I discovered my Citizen 120D Epson compatible does not have *reverse feed*, and that the software off-centred the printhead. I therefore had to remember to draw registration marks on the paper each time so that I could accurately re-align for the next pass. No mention of possible incompatibility and what to do about it is made in the documentation.

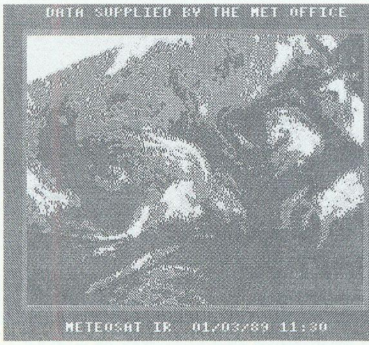
If reverse-feed doesn't work, press the form-feed button several times to eject the paper rather than manually turning the platten. This should also correctly centre the printhead.

PEDAGOGY OVERLOOKED

The ability to explore a package is very important because it allows you to get the 'feel' of the process through trial and error before using it in earnest. *Peacock Printer* takes some time to understand, particularly the use of the function keys. This important pedagogical aspect appears to have been entirely overlooked.

POOR DOCUMENTATION

Unfortunately, the document's wording makes the process sound more complicated than it is. Apart from function key f6 (used to set the printer working) and f9 (quit program), four other keys do all the work. Keys f0 and f1 are toggle switches used to identify physical and logical colours, while f2 is used to cycle through logical colours, and f3 assigns a particular physical colour to a particular logical colour.



Multi-coloured printout (in monochrome here) from Peacock Printer

Confused? I was! Sentences such as "Pixels of the current LOGICAL colour remain unblanked, regardless of any other pixels with the same physical colour" are quite difficult to understand first time round and should be avoided.

I couldn't help feeling that the documentation could have clarified this confusion rather than adding to it. Also, some attempt should have been made to provide a context or background to understanding what colour separation actually is, how it works, and why it is important. And it would be helpful if the software contained some demonstration examples with which to experiment.

COLOUR THEORY

It is helpful to know, for example that the printing industry has its own long established traditions, generally employing four standard inks as the primary colours - yellow, magenta (light red), cyan (light blue), and black - on standard white paper.

In-between colours (the secondaries orange, green, violet) are created by over-printing primary colour on primary colour (yellow + cyan = green). Colours can be darkened by printing more of the same colour on top (i.e. magenta over-printed with magenta makes red).

SILK-SCREEN PROCESS

Peacock Printer closely emulates silk screen print-making (generally associated with colour artwork and posters in advertising), and could provide an effective introduction to the process. This important educational area is not mentioned in the documentation.

The software effectively creates these different colour screens automatically, leaving you to cycle through the colours looking for realistic or creative colour combinations.

CONCLUSIONS

Peacock Printer is an imaginative piece of programming that should be snapped up by education (Art, and Science for theory of colour and light demonstrations to name but two), and those interested in colour graphics but unable to afford a colour printer. Seriously, it is a fascinating topic and highly creative. Give it a whirl!

Unfortunately, this package is let down by poor documentation, though I understand that there's a revised and enhanced version on the way. So, if you want something a little different, try *Peacock Printer* and colour separation. Maybe you'd like to send us your results? B

Disc File Identifier

Alan Mothersole describes a comprehensive utility for identifying and analysing the contents of your discs.

Have you ever rummaged through boxes of discs and come across some that you had forgotten all about? Even doing a *CAT leaves you none the wiser. If you use a Master or BBC converted to the ADFS it becomes even more of a problem to sort through the directories and files.

There have been several utilities to catalogue discs and attempts to identify files, but these have been predominantly for DFS users. This two-part series will ultimately cover the needs of both filing systems by constructing a disc and file analysing program called WOTAMI. This month we start with the skeleton program and code to handle DFS format discs.

By inserting a disc into a drive, WOTAMI will attempt to identify the types of files on it as follows:

BASIC
ROM Image
VIEW
ViewSheet
ViewStore
Text
Directories (ADFS)
Machine Code/Data

THE PROGRAM

The program listed here should be typed in with the line numbers exactly as shown and saved to disc as 'WOTdfs'. This program will only work with the DFS at present. The additional routines for ADFS users will be given in Part 2.

OPERATION

Run WOTdfs and wait for the menu to be displayed. Insert the disc to be analysed into a drive. If any disc surface other than 0 is required use <O> from the main menu to change it.

Pressing <R> will read the filenames and identify them. All the files on the selected side will be identified. Once the disc has been read a summary of the files will be shown beneath the menu.

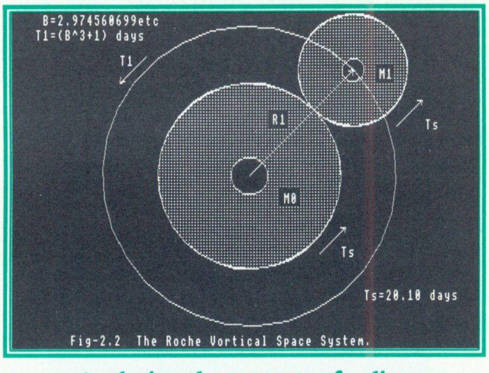
As of now there is only one way of displaying the information. <S> will display a list of the files together with their identification, length, lock status, disc size and title. However, an additional facility will be presented in Part 2.

Hard copy can also be obtained by using <P> from the main menu. The resulting printout could be cut out and slipped inside the disc sleeve for future reference.

NOTE: WOTdfs cannot guarantee to identify all files correctly. For example if a ROM image has been saved via a buffer at &3000 its load address may not be &8000.

FUTURE EXPANSION

In Part 2 of this series, additional routines will be added to allow both ADFS and DFS discs to be read.

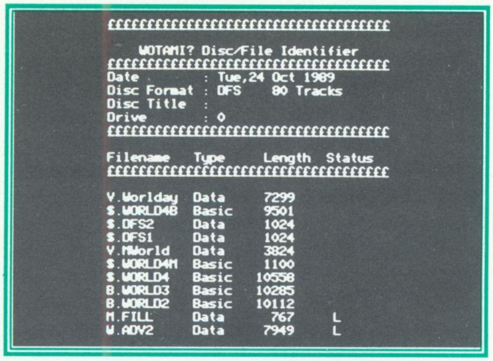


Analysing the contents of a disc

In addition, a powerful routine called 'Wotami' will be used to interrogate the identified files and give further useful information on the contents of Basic REMs, ROM Image header information, View comments and text file data.

TECHNICAL DETAILS

The program is in Basic and includes a number of O.S. calls to read parameters such as disc filing system, directory names, disc title, disc filenames and address details.



```
£££££££££££££££££££££££££££££££££££££

    WOTAMI? Disc/File Identifier
£££££££££££££££££££££££££££££££££££££
Date          : Tue,24 Oct 1989
Disc Format   : DFS    80 Tracks
Disc Title    :
Drive         : 0
£££££££££££££££££££££££££££££££££££££

Filename   Type     Length  Status
£££££££££££££££££££££££££££££££££££££

V.Worldday Data       7299
$.WORLD4B  Basic      9501
$.DFS2     Data       1024
$.DFS1     Data       1024
V.MWorld   Data       3824
$.WORLD4M  Basic      1100
$.WORLD4   Basic     10558
B.WORLD3   Basic     10285
B.WORLD2   Basic     10112
M.FILL     Data        767     L
W.ADV2     Data       7949     L
```

Disc contents displayed

MAIN PROCEDURES

PROCinit - Define arrays and variables.
PROCinit2 - Clear arrays and reset totals.
PROCmenu - Main screen menu.
PROCfs - Read computer filing system.
PROCreaddfscat - Read all filenames from selected DFS disc.
PROCreadcsd - Read currently selected directory. Default is $.
PROCdtitle - Read disc/directory title.
PROCreadad - Read load, execution addresses and file length.
PROClock - Find which attributes are set.
PROCchBasic - Check if file is Basic by examining if first byte is &0D. PROCchROM - Check if file is ROM image - is load address = &8000?
PROCcView - Check if file is in View format.
PROCcText - Check for Text file. If both load and execution addresses are not &FFFF or the execution address is &FFFF, or load address is &0 and execution address is &FFFF then file contains text.
PROCcVST - Check if file is ViewStore data file by reading from end of file for its end-of-data marker (&01).
PROCcVSHT - Check if file is ViewSheet data file (second byte is &03). PROCshow - Gives summary list of all files.
PROCcheck - Main routine for identifying files after checking for directories. The sequence chosen is ROM, Basic, View, ViewSheet, ViewStore, Text while any other files are declared as machine code. As soon as file is correctly identified the checking ceases and the next file is examined.

```
   10 REM Program WOTAMI
   20 REM Version B1.3
   30 REM Author  Alan Mothersole
   40 REM BEEBUG  November 1989
   50 REM Program subject to copyright
   60 :
  100 MODE7:master=INKEY(-256)=253
  110 ONERROR GOTO 10050
  120 PROCinit
  130 PROCmenu
  140 END
  150 :
 1000 DEF PROCinit
 1010 LOCAL I%
 1020 VDU23,1,0;0;0;:@%=6
 1030 buffer%=&7000
 1040 DIM block% 20,buf% 20,C% 18,S% 18,
blk% 20,block 18,name 11
 1050 DIM name$(47),load(47),ex(47),leng
th(47),lock$(47),type%(47),type$(8)
 1060 z$="**********"
 1070 FOR I%=1 TO 8:type$(I%)=z$:type$(I
%)="":NEXT
 1080 dr%=0:tt$="":csd$="$":read=FALSE:f
max%=47:namax%=10
 1090 RESTORE 10000:FOR I%=1TO8:READ typ
e$(I%):NEXT
 1100 sx$=CHR$133+CHR$157+CHR$131:sy$=CH
R$132+CHR$157:Date$=""
 1110 IF NOT master THEN INPUT'"Enter cu
rrent date: "Date$
 1120 ENDPROC
 1130 :
 1140 DEF PROCinit2
 1150 LOCAL I%
 1160 FOR I%=0TO550:buffer%?I%=0:NEXT
 1170 FOR I%=0 TO 46
 1180 name$(I%)=z$:name$(I%)="":lock$(I%
)="****":lock$(I%)="":type%(I%)=0
 1190 NEXT
 1200 ba%=0:ro%=0:vi%=0:sh%=0:st%=0:tx%=
0:mc%=0:di%=0:prt=FALSE:dlen%=0:tk%=0:fr
ee%=0
 1210 ENDPROC
 1220 :
 1230 DEF PROCmenu
 1240 LOCAL A%,T%:quit=FALSE
 1250 REPEAT
 1260 CLS:PROCtitle:PRINT
 1270 PROCcen(CHR$131+" Select Option:")
 1280 PRINT
 1290 RESTORE 10010
 1300 READ Y
 1310 FOR T%=1TO Y
 1320 READ A$
 1330 PRINTTAB(12)CHR$130;A$
 1340 NEXT
 1350 IF read THEN PROCupdate
 1360 A%=GET:A%=A% OR &20
 1370 IF A%=114 PROCreadcsd:PROCdtitle:P
ROCread:PROCcheck:read=TRUE
 1380 IF A%=115 CLS:VDU14:PROCshow:VDU15
 1390 IF A%=112 PROCprint
 1400 IF A%=111 PROCoptions:PROCreadcsd:
PROCdtitle
 1410 IF A%=42 PROCstar
 1420 IF A%=113 CLS:PROCcen(CHR$131+"Are
 you sure (Y/N) ?"):A$=GET$:IFA$="Y"ORA$
="y" THEN quit=TRUE
 1430 UNTIL quit
 1440 ENDPROC
 1450 :
 1460 DEF PROCtitle
 1470 LOCAL K%
 1480 CLS:FORK%=1TO2:PRINTsy$;:PROCcen(C
HR$131+CHR$141+"     W O T A M I ?"):NEXT
 1490 PRINTsy$;:PROCcen(CHR$130+"  Intel
legent Disc Reader"):PROCfs
 1500 f$="DFS"
 1510 PRINTsx$;"File System : ";CHR$134;
f$;TAB(23)CHR$131;"Drive : ";CHR$134;dr%
 1520 PRINTsx$;"Directory   : ";CHR$134;
csd$,'sx$;"Title       : ";CHR$134;tt$
 1530 ENDPROC
 1540 :
 1550 DEF PROCupdate
 1560 LOCAL I%
 1570 PRINTTAB(0,17)sy$;:PROCcen(CHR$131
+"Files on Disc")
 1580 b=19
 1590 FOR I%=1 TO 7 STEP 2
 1600 PRINTTAB(2,b)CHR$134;type$(I%);TAB
(10,b)":";CHR$131;TAB(22,b)CHR$134;type$
(I%+1);TAB(30,b)":";CHR$131;
 1610 b=b+1
 1620 NEXT
 1630 PRINT;TAB(13,19)ba%;TAB(33,19)ro%;
TAB(13,20)vi%;TAB(33,20)sh%;TAB(13,21)st
%;TAB(33,21)tx%;TAB(13,22)mc%;TAB(33,22)
di%
 1640 PRINT'sy$;
 1650 PROCcen(CHR$131+"      Total : "+ST
R$(nof%-di%)+CHR$11+CHR$11)
 1660 ENDPROC
 1670 :
 1680 DEF PROCfs
 1690 dfs=FALSE:A%=0:X%=0:Y%=0
 1700 IF (USR(&FFDA) AND &4)=&4 THEN dfs
=TRUE
 1710 ENDPROC
 1720 :
 1730 DEF PROCread
 1740 PROCinit2:CLS
 1750 PROCdfscat
 1760 ENDPROC
 1770 :
 1780 DEF PROCdfscat
 1790 LOCAL A%,B%,C%
 1800 PROCcen(CHR$131+"Reading DFS Filen
ames")
 1810 ?&70=dr%:X%=&70:Y%=0:A%=&7E:CALL&F
FF1
 1820 free%=!&70:IFfree%>&20000 THENtk%=
80 ELSE tk%=40
 1830 ?&70=dr%:!&71=buffer%:?&75=3:?&76=
&53:?&77=0:?&78=0:?&79=34
 1840 X%=&70:Y%=0:A%=&7F:CALL&FFF1
 1850 nof%=(buffer%?&105)/8
 1860 C%=buffer%+8
 1870 FOR A%=0 TO nof%-1
 1880 IF ?C%<>32 name$(A%)=""
 1890 FOR B%=0 TO 7:name$(A%)=name$(A%)+
CHR$((C%?B%)AND127):NEXT
 1900 name$(A%)=RIGHT$(name$(A%),1)+"."+
LEFT$(name$(A%),7)
 1910 C%=C%+8
 1920 NEXT:CLS
 1930 ENDPROC
 1940 :
 1950 DEF PROCreadcsd
 1960 LOCAL Y%:csd$=""
 1970 A%=&6:X%=blk%MOD256:Y%=blk%DIV256
 1980 ?blk%=0:blk%!1=buf%:CALL &FFD1
 1990 dr%=buf%?(1)-48
 2000 t=buf%?(2)
 2010 FOR Y%=3 TO 2+t
 2020 temp$=CHR$(buf%?(Y%))
 2030 csd$=csd$+temp$
 2040 NEXT
 2050 ENDPROC
 2060 :
 2070 DEF PROCdtitle
 2080 LOCAL t,A%,X%,Y%
 2090 tt$=""
 2100 A%=&5:X%=blk%MOD256:Y%=blk%DIV256
 2110 ?blk%=0:blk%!1=buf%:CALL &FFD1
 2120 t=buf%?(0)
 2130 FOR Y%=1 TO t
 2140 temp$=CHR$(buf%?(Y%))
 2150 tt$=tt$+temp$
 2160 NEXT
 2170 ENDPROC
 2180 :
 2190 DEF FNgdata
 2200 info$=""
 2210 REPEAT
 2220 data=BGET#C%
 2230 IF data>&7E THEN data=&20
 2240 IF data>=&20 THEN info$=info$+CHR$
(data)
 2250 UNTIL data<&20
 2260 =info$
 2270 :
 2280 DEF PROCreadad(Z%)
 2290 LOCAL A%,X%,Y%
 2300 $name=name$(Z%)
 2310 ?block=name MOD 256:block?1=name D
IV 256
 2320 A%=5:X%=block MOD 256:Y%=block DIV
 256:CALL &FFDD
 2330 load(Z%)=(block!2)
 2340 ex(Z%)=(block!6):length(Z%)=block!
10
 2350 lock=block!14
 2360 PROClock
 2370 ENDPROC
 2380 :
 2390 DEF PROClock
 2400 lock$(Z%)=""
 2410 IF (lock AND &0)=&0 THEN lock$(Z%)
=""
 2420 IF (lock AND &8)=&8 THEN lock$(Z%)
="L"
 2430 ENDPROC
 2440 :
 2450 DEF PROCchBasic
 2460 IF type%(I%)=8 ENDPROC
 2470 basic=FALSE
 2480 C%=OPENUP(name$(I%))
 2490 data=BGET#C%
 2500 IF data<>&0D CLOSE#C%:ENDPROC
 2510 CLOSE#C%
 2520 basic=TRUE
 2530 IF basic THEN type%(I%)=1:ba%=ba%+
1
 2540 ENDPROC
 2550 :
 2560 DEF PROCchROM
 2570 IF type%(I%)=8 ENDPROC
 2580 IF load(I%)=&8000 OR load(I%)=&FFF
F8000 THEN type%(I%)=2:ro%=ro%+1
 2590 ENDPROC
 2600 :
 2610 DEF PROCcView
 2620 IF type%(I%)=8 ENDPROC
 2630 C%=OPENUP(name$(I%))
 2640 data=BGET#C%
 2650 IF data<>&80 CLOSE#C%:ENDPROC
 2660 data=BGET#C%
 2670 IF data<>&43 CLOSE#C%:ENDPROC
 2680 type%(I%)=3
 2690 CLOSE#C%
 2700 vi%=vi%+1
 2710 ENDPROC
 2720 :
 2730 DEF PROCcText
 2740 IF type%(I%)=8 ENDPROC
 2750 IF (load(I%)=-1 AND ex(I%)=-1) THE
N ENDPROC
 2760 IF ex(I%)<>-1 THEN ENDPROC
 2770 IF load(I%)=0 AND ex(I%)=-1 THEN t
ype%(I%)=6:tx%=tx%+1
 2780 ENDPROC
 2790 :
 2800 DEF PROCcVST
 2810 enddata%=&01:pad%=&00
 2820 LOCAL ptr%,data%
 2830 C%=OPENUP(name$(I%))
 2840 ptr%=EXT#C%
 2850 REPEAT
 2860 ptr%=ptr%-1
 2870 PTR# C%=ptr%
 2880 data%=BGET#C%
 2890 UNTIL(data%<>pad%)OR(ptr%=0)
 2900 CLOSE# C%
 2910 IF (data%<>enddata%)OR(ptr%=0)THEN
 ENDPROC
 2920 IF data%=enddata% THEN type%(I%)=5
:st%=st%+1
 2930 ENDPROC
 2940 :
 2950 DEF PROCcVSHT
 2960 C%=OPENUP(name$(I%))
 2970 FOR J%=1 TO 2
 2980 data=BGET#C%
 2990 NEXT
 3000 IF data=&03 THEN type%(I%)=4:sh%=s
h%+1
 3010 CLOSE# C%
 3020 ENDPROC
 3030 :
 3040 DEF PROCoptions
 3050 LOCAL A,T%:ex=FALSE
 3060 REPEAT
 3070 PROCtitle:PRINT
 3080 PRINTCHR$131;:PROCcen("Change :"):
PRINT
 3090 RESTORE 10020
 3100 READ Y
 3110 FOR T%=1 TO Y
 3120 READA$:PRINTTAB(12)CHR$130;A$
 3130 NEXT
 3140 A=GET:A=A OR &20
 3150 IF A=100 PROCchdrv
 3160 IF A=120 ex=TRUE
 3170 UNTIL ex
 3180 ENDPROC
 3190 :
 3200 DEF PROCchdrv
 3210 PROCfs
 3220 LOCAL A
 3230 PRINT''TAB(6)CHR$131;"Drive No.? "
;
 3240 REPEAT
 3250 A=GET:A=A-48
 3260 UNTIL A>=0 AND A<6
 3270 PRINTTAB(20)A
 3280 A$=STR$(A)
 3290 dr%=A
 3300 OSCLI("DRIVE "+A$)
 3310 new=TRUE
 3320 ENDPROC
 3330 :
 3340 DEF PROCshow
 3350 LOCAL I%
 3360 IF read=FALSE ENDPROC
 3370 IF prt THEN VDU2
 3380 PROCline:PRINT
 3390 PROCcen("WOTAMI? Disc/File Identif
ier")
 3400 PROCline
 3410 IF Date$="" Date$=LEFT$(TIME$,15)
 3420 PRINT"Date        : ";Date$
 3430 PRINT"Disc Format : ";
 3440 PRINT"DFS    ";tk%;" Tracks"
 3450 PRINT"Disc Title  : ";tt$
 3460 PRINT"Drive       : ";dr%
 3470 PROCline
 3480 PRINT'"Filename   Type     Length
 Status"
 3490 PROCline
 3500 PRINT
 3510 FOR I%=0 TO nof%-1
 3520 PRINTname$(I%);TAB(11)type$(type%(
I%)),length(I%);TAB(29)~lock$(I%)
 3530 NEXT
 3540 PRINT'
 3550 PROCline:PRINT
 3560 PROCcen("Number of files : "+STR$(
nof%-di%))
 3570 PRINT'
 3580 dfree%=free%-dlen%
 3590 PROCcen("Space used = "+STR$(dlen%
)+" bytes")
 3600 PRINT:PROCcen("Space free = "+STR$
(dfree%)+" bytes"):PRINT
 3610 PROCline:VDU3:PROCky
 3620 ENDPROC
 3630 :
 3640 DEF PROCcheck
 3650 LOCAL I%
 3660 IF nof%=0 ENDPROC
 3670 PROCtitle
 3680 PRINT'
 3690 PROCcen(CHR$131+"Identifying Files
")
 3700 PRINT'
 3710 PROCcen("Please wait ...")
 3720 FOR I%=0 TO nof%-1
 3730 PROCreadad(I%)
 3740 dlen%=dlen%+length(I%)
 3750 IF type%(I%)=0 PROCchROM
 3760 IF type%(I%)=0 PROCchBasic
 3770 IF type%(I%)=0 PROCcView
 3780 IF type%(I%)=0 PROCcVSHT
 3790 IF type%(I%)=0 PROCcVST
 3800 IF type%(I%)=0 PROCcText
 3810 IF type%(I%)=0 THEN type%(I%)=7:mc
%=mc%+1
 3820 NEXT
 3830 ENDPROC
 3840 :
 3850 DEF PROCprint
 3860 LOCAL A
 3870 IF read=FALSE ENDPROC
 3880 prt=TRUE
 3890 PROCtitle:PRINT
 3900 PROCcen(CHR$134+"Print Options")
 3910 PRINT
 3920 RESTORE 10030
 3930 READ Y
 3940 FOR T=1 TO Y
 3950 READ A$:PRINTTAB(12)CHR$130;A$
 3960 NEXT
 3970 REPEAT
 3980 A=GET:A=A OR &20
 3990 UNTIL A=115 OR A=119 OR A=120
 4000 IF A=120 THEN prt=FALSE:ENDPROC
 4010 IF A=115 PROCshow
 4020 prt=FALSE
 4030 ENDPROC
 4040 :
 4050 DEF PROCstar
 4060 CLS:PRINT'"*":INPUTstar$
 4070 OSCLI(star$):PRINT':PROCky
 4080 ENDPROC
 4090 :
 4100 DEF PROCcen(t$):PRINTTAB(18-(LENt$
)/2)t$:ENDPROC
 4110 :
 4120 DEF PROCtime(t):TIME=0:REPEAT:UNTI
L TIME>t:ENDPROC
 4130 :
 4140 DEF PROCky:PROCcen("Press any key"
):A=GET:ENDPROC
 4150 :
 4160 DEF PROCline:PROCcen(STRING$(36,"_
")):ENDPROC
 4170 :
10000 DATA Basic,ROM,View,VSheet,VStore,
Text,Data,<DIR>
10010 DATA 6,(R)ead Disc,(S)how Files,(P
)rint Files,(O)ptions,(*) Command,(Q)uit
10020 DATA 2,(D)isk Drive,e(X)it to Menu
10030 DATA 2,(S)how,e(X)it
10040 :
10050 VDU3:CLS:REPORT:PRINT;" at line ";
ERL:END
```

A Thesaurus for the Beeb

Mike Williams reviews Keyword, a new product from Swift Software for aspiring wordsmiths.

Product	**Keyword**
Supplier	**Swift Software**
	6 Ennerdale Road,
	Stockport, Cheshire SK1 4NR.
	Tel. 061-477 8405
Price	**£9.95** (40 track 5.25" disc)
	£19.95 (80 track 5.25" DFS disc or 3.5" ADFS disc)
	Prices include VAT and p&p.

At one time word processors were simple to use and straightforward in application. Nowadays the opposite is too often true. However, there are certain features which have proven invaluable. The first of these was some form of spelling checker, now an essential component of almost any word processor that claims even modest respectability.

The latest and growing demand is for a thesaurus. Printed versions have been in existence for a long time, Roget's Thesaurus probably being the best known. This allows you to look up any word in an alphabetical index which directs you to a section containing similar, equivalent or related words and phrases. Such a thesaurus can be a boon to anyone who undertakes much writing.

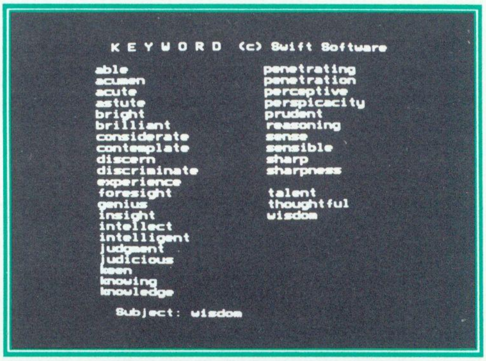
Until now, I have been unaware of any such software for the BBC micro (or even the Archimedes), but Swift Software has galloped to the rescue with Keyword, which is available in versions on disc for all Acorn micros.

The software is supplied in a smart looking pack, but the documentation, though well printed, amounts to no more than two A5 sides of instructions. Pressing Shift-Break boots the software.

Enter a keyword and press Return twice to be presented with a list of possible categories. Use the cursor keys to select the one of your choice and press Return again to be presented with a corresponding list of words. Using the cursor keys, more words can always be selected from the list displayed, for further associations.

As well as searching for synonyms and antonyms, you can also type in, say 'th', and see displayed all words beginning with these two letters. It is also possible to browse through the 10,000 or so words which Keyword claims to provide (the 40 track version has 5000 words).

In use the access times seemed acceptable, but I do have two points of criticism. The first may be considered temporary, in that it relates to the documentation. I did not find this really sufficient at the outset, and I learnt to use the package more by trial and error. Moreover, on my system at least, the main example used in the documentation fails to work as described!



```
       K E Y W O R D   (c) Swift Software

     able                   penetrating
     acumen                 penetration
     acute                  perceptive
     astute                 perspicacity
     bright                 prudent
     brilliant              reasoning
     considerate            sense
     contemplate            sensible
     discern                sharp
     discriminate           sharpness
     experience
     foresight              talent
     genius                 thoughtful
     insight                wisdom
     intellect
     intelligent
     judgment
     judicious
     keen
     knowing
     knowledge

        Subject: wisdom
```

My other disappointment concerns the method of operation. I would have liked to access Keyword while working on a document, in the manner of Computer Concept's Spellmaster. Unfortunately, this appears impossible. Instead, you must save your document, boot Keyword and do your searching. Then reload your word processor and original document. This seems to me to be a significant weakness in an otherwise admirable product. I understand from Swift Software that a ROM-based version is under development which will overcome this problem, and which will be available to existing purchasers for the cost of post and packing as an upgrade. Keyword also requires that its own disc be always available, as only parts of its dictionary are ever loaded into memory at any time.

In principle, this is an excellent product, offering as it does the facilities that many users increasingly are asking for in conjunction with a word processor. However, until the suppliers do produce a version which integrates better with the word processor of your choice, then I for one will still be reaching for my printed edition of Roget's Thesaurus when the need arises. B

A Good Report

Andrew Roland explains how to enhance the error reporting in your programs, by emulating the REPORT$ keyword of the Archimedes' Basic V.

"To err is human, but to make a real mess-up you need a computer." It is one of my philosophies of life that as mistakes are inevitable, you might as well learn to deal with them as quickly and painlessly as possible. Here are two ideas for making those programming errors more bearable.

The first is really just a tip. Do the error handling parts of your programs just print an error message and line number, and stop at that? Why not get the computer to list the line causing the trouble as well - at least while you're still debugging the program? One way of doing this is shown in listing 1.

Listing1

```
  10 REM Listing 1
  20 ON ERROR GOTO 1000
  30 rubbish
  40:
1000 ON ERROR OFF
1010 REPORT:IF ERL PRINT" at line ";ERL
1020 OSCLI "Key0 LIST "+STR$(ERL)+"|F|M"
1030 *FX21
1040 *FX138,0,128
1050 VDU21
1060 END
```

Unfortunately, you can't just type LIST ERL - you have to convert the value of ERL into a string using STR$, as in line 1020, which is used as part of a function key definition. Line 1040 then inserts the function key's code into the keyboard buffer just as if you had actually pressed f0, and the next time an error occurs, the whole line is listed on the screen.

Even better, why not use Paul Pibworth's Basic Line Editor from BEEBUG Vol.8 No.2? If you're like me, you get to hate cursor-copying whole lines just to alter one small mistake. If you change line 1020 of that program to:

```
1020 OSCLI"Key0 CALL &7719|M"+STR$ERL+"|M"
```

and engage the editor with *ED, the offending line will go straight into a word processor style line editor - that's what I call time saving! By the way, CALL &7719 assumes you have not altered the addresses in Paul Pibworth's program - if you have, it may need adjusting.

ERROR MESSAGES

The second idea is concerned with how error messages appear after your program has been completed, and a user encounters a problem like 'Disc full'. How many professional looking programs are spoilt by tatty error messages! For example, have you ever tried (or wanted) to do this in mode 7, hoping to get a red error message?

```
100 PRINT TAB(0,23);CHR$(129);"Error - ";
110 REPORT
```

Of course, it won't do what it's supposed to because REPORT always insists on starting a new line first (try it and see). What's more, if you want double-height letters, you're really stuck, unless you own an Archimedes - then you could use the pseudo-variable REPORT$. This gives you access to the error message as a character string, allowing you to process and manipulate it in any way you choose.

But how can we Model B and Master owners read an error message into a string? The traditional answers to this problem tend to involve lots of machine code and calls into the Basic ROM, which changes with every new version of Basic. The answer I came up with is short and sweet. If you just want to type in the program and dazzle your friends with your technicolor error messages, you can skip the explanation all together.

USING THE PROGRAM

We cannot create a variable called REPORT$ (variables may not start with a keyword) so we will use a function instead - FNreport. This is given high line numbers (from 30000) so that it can be added to your own programs without line number clashes. Later, you will only need lines 30000 onwards, but for the time being enter all of Listing 2, including the demonstration at the beginning. Be careful to save the program before running it - better safe than sorry.

Listing 2

```
   10 REM Program REPORT$ simulation
   20 REM Version 1.00
   30 REM Author  Andrew Rowland
   40 REM BEEBUG  November 1989
   50 REM Program subject to copyright
   60 :
  100 ON ERROR GOTO 180
  110 CLS
  120 DIM buffer 8
  130 PRINT'"REPORT$ DEMO"
  140 PRINT"Press ESCAPE ";
  150 REPEAT UNTIL FALSE
  160 END
  170 :
  180 ON ERROR OFF
  190 mode=?&355
  200 IF mode=7 PROCcentre(CHR$129+CHR$1
41+FNreport+" ",22):PROCcentre(CHR$129+C
HR$141+FNreport+" ",23)
  210 IF mode<7 PROCdouble(0,30,FNreport
)
  220 PRINT
  230 END
  240 :
 1000 DEF PROCcentre(A$,Y)
 1010 LOCAL X:X=(39-LENA$) DIV 2
 1020 PRINTTAB(X,Y)A$;
 1030 ENDPROC
 1040 :
 1050 DEF PROCdouble(x,y,A$)
 1060 LOCAL A%,X%,Y%
 1070 osword=&FFF1
 1080 PRINTTAB(x,y);
 1090 X%=buffer MOD 256
 1100 Y%=buffer DIV 256
 1110 A%=10
 1120 FOR I%=1 TO LEN A$
 1130 ?buffer=ASC(MID$(A$,I%))
 1140 CALL osword
 1150 VDU 23,254,buffer?1,buffer?1,buffe
r?2,buffer?2,buffer?3,buffer?3,buffer?4,
buffer?4
 1160 VDU 23,255,buffer?5,buffer?5,buffe
r?6,buffer?6,buffer?7,buffer?7,buffer?8,
buffer?8
 1170 VDU 255,8,11,254,10
 1180 NEXT
 1190 ENDPROC
 1200 :
30000 DEF FNreport
30010 LOCAL P%,Q%,X%,Xtemp,entry
30020 Q%=&900:store=Q%+&20
30030 FOR pass=0 TO 2 STEP 2
30040 P%=Q%
30050 [OPT pass
30060 .Xtemp EQUB 0 \ make pointer zero
30070 \ enter here via OSWRCH vector
30080 .entry
30090 \ skip any VDU 10's
30100 CMP #10:BEQ out
30110 LDX Xtemp
30120 STA store,X \ and store it
30130 INC Xtemp \ add 1 to pointer
30140 .out RTS
30150 ]NEXT
30160 REM keep old contents of vector
30170 oldv=!&20E
30180 REM divert OSWRCH vector
30190 ?&20E=entry MOD 256
30200 ?&20F=entry DIV 256
30210 REPORT
30220 IF ERL PRINT" at ";ERL:ELSE PRINT
30230 !&20E=oldv:REM restore vector
30240 =$(store+1)
```

Try the program in mode 7, and then in mode 1. When you press Escape, the corresponding error message is read into a string and then printed in double-height letters (regardless of mode). Two useful procedures are used in the demonstration: PROCcentre(A$,Y) (line 1000) which centres a string at line Y; and PROCdouble(x,y,A$) (line 1050) which prints a string in double-height letters at co-ordinates (x,y). This only works in a graphics modes.

When you are satisfied that all is well, delete lines 10 to 1200 and save the procedure as, say, Report. When you start a new program, load Report, type AUTO and start writing. Alternatively, consult the User Guide for instructions on merging it with one of your existing programs.

FNreport can be used exactly like a normal string variable. You can PRINT it, split it with LEFT$() and RIGHT$() and even write lines like:

```
IF INSTR(FNreport,"Escape") THEN...
```

How about that for readability? The most useful thing I have used it for was in a program I was writing for a blind friend with Superior's Speech! The line:

```
OSCLI ("*SAY "+FNreport)
```

spoke out every error message, so he always knew what was going wrong. The only limitation is that it cannot be used in a second processor.

HOW ERROR MESSAGES ARE PRODUCED

When considering how to implement a REPORT$ emulation, my first thought was to look at what the computer does when it comes across your latest typing mistake. Well, it stops what it's doing and makes a note of where the error message is (the message is in fact a string somewhere in memory); that is, it stores the address of the start of the error message in two locations - &FD and &FE. It then continues execution after the last ON ERROR statement, but it doesn't actually print the message on the screen until it meets a REPORT.

So I wrote a program to recover the address from these locations and read the message into a variable, report$, a letter at a time, until it meets a zero, which marks the end of the message (Listing 3).

Listing 3

```
  10 REM Listing 3
  20 ON ERROR GOTO 1000
  30:
1000 address=?&FD+?&FE*256
1010 I%=0:report$=""
1020 REPEAT I%=I%+1
1030 A%=address?I%
1040 IF A%>0 report$=report$+CHR$A%
1050 UNTIL A%=0
1060 PRINT report$
```

And most of the time this works. Messages like 'Syntax error' and 'No such line' appear beautifully. But I soon found that messages like 'No REPEAT' or 'Bad MODE' did odd things. The reason is that Basic does not store keywords as individual letters, but gives each one a number (called a token). All the tokens are numbers above 127, so 128 is the token for AND, 129 is DIV and so on. When you list a program, Basic expands the tokens again to read properly. This means that programs take up less memory and run faster, but when tokens appear in error messages they rather defeat our purpose - they have to be expanded before report$ can be used.

I didn't really fancy typing in the 127 different keywords and their tokens, especially when Basic can do it for you. But how? If I type REPORT, the error message, complete with expanded keywords, only appears on the screen. How can I read them into a string?

The solution lies in the fact that there is an operating system routine which writes letters on the screen, which the manuals call OSWRCH - Operating System WRite CHaracter. Every time we print a letter, whether it's using VDU 65, PRINT CHR$(65) or PRINT "A" (all of which produce the same thing) we use OSWRCH. You can try it yourself by typing:

```
A%=65:CALL &FFEE
```

You should see an 'A' appear, as 65 is the ASCII code for A. &FFEE is OSWRCH's call address, and Basic is kind enough to pass the contents of A% to any routine we call (See also this month's First Course article). REPORT uses OSWRCH too when it prints an error message, sending in turn the ASCII code of each letter to appear on the screen.

VECTORS

Wouldn't it be nice if we could intercept OSWRCH and store the letters instead? Fortunately the operating system provides a convenient means of doing this using a vector. A vector is a pointer to an operating system routine. It consists of two bytes in memory which contain the address of the actual routine we want. It's a bit like finding the location of the last clue in a treasure hunt. It might be a tree at the end of your street, and pinned to it is a note saying that the treasure is hidden in the garden shed - the tree doesn't hide the treasure itself, but has the address of the real thing. If we were to cheat by altering the clue to say the treasure was in the kitchen, anyone following us would go to the wrong place and never find the treasure.

In the computer, the 'treasure' is the operating system routine which displays letters on the screen, the 'tree' is the vector, which for OSWRCH is at &20E/&20F, and the 'kitchen' will be a very short machine code routine we will supply, which stores the letters in a string. Just what we want!

HOW THE PROGRAM WORKS

FNreport starts at line 30000 (listing 2), so we'll begin there. It starts by assembling the short routine I referred to earlier, but note that it doesn't execute this code yet. My program uses page &900, but you can move it to any convenient location: it only makes temporary use of this area and it can be used for other things outside FNreport.

Line 30170 keeps a note of the present contents of the vector - the address of the 'real' OSWRCH - and then lines 30190-30200 alter the vector to point to our replacement routine. The REPORT and PRINT statements in lines 30210-30220 cause repeated calls to be made to our routine - one for every letter that was to appear on the screen, but will now not do so. Each time, the pointer Xtemp is incremented and the letters stored one after another starting at store. Finally, we restore the original contents of the vector, or we won't be able to PRINT again (line 30230).

The procedure exits by returning the string starting at store+1. Why that +1? Well, remember that our original problem was that REPORT always starts a new line at the beginning? It does this by doing the equivalent of a VDU 10,13 - that is, move down a line then go to the beginning of the line. The 10 is skipped by our machine code (line 30100), but the 13 stored, so we need to retrieve the string after the 13.

Just one more point: your REPORT statement must end by starting a new line - PRINT does this automatically unless you end the statement with a semi-colon - so that a 13 can be stored as an end of string marker; hence the ELSE PRINT in line 30220.

So there we have it - now your Beeb is as good as an Archimedes. Well, nearly! And you've mastered a technique which can be used for more than just error messages. Anything which can be printed can be stored in this way, so long as it ends with a Return and is no longer than 254 characters.

My final offering is a program which reads the lines of a program into a string, and speaks them using Superior Software's Speech! (Listing 4 - magazine disc/tape only). However, Speech! doesn't cope very well with all the weird and wonderful things programs can contain - I leave solving that problem to you! For more examples of 'speaking' programs, see the Bingo caller elsewhere in this issue.

Points Arising....Points Arising....Points Arising....Points Arising....

USING ASSEMBLER (Part 4) BEEBUG Vol.7 No.6

The example program given in this issue fails to work correctly on a model B, because the function key definitions in lines 1700 to 1730 are too long. To produce a working program change these as follows:

```
 1700*KEY0V.21|MV.6:I."hhmm "A$:?&72=0:?&7
3=EV.("&"+RI.A$,2)):?&74=EV.("&"+LE.A$,2))
:CA.&906:CA.&900:A%=5:CA.&903|M
 1710*KEY1V.21|MV.6:P."Stopped":CA.&906|M
 1720*KEY2V.21|MV.6:P."Restarted":CA.&906:
CA.&900:A%=5:CA.&903|M
 1730*KEY3V.21|MV.6:P."Zeroed":?&72=&99:?&
73=&99:?&74=&99|M
```

Make sure you enter no spaces other than those clearly shown above.

LEARNING FOREIGN LANGUAGES (BEEBUG Vol.8 No.2)

If the foreign character definitions are to be used on a model B, then changes are needed to make the definition programs work correctly. Change line 110 to read *FX20,6 not *FX20,1. Then add two new lines after this command:

```
112A%=131:P%=((USR&FFF4)AND&FFFF00)DIV256
114IF PAGE<P% THEN PAGE=P%:CHAIN"French"
```

Alternatively, raise the value of PAGE by at least &600 before loading and running a character definition program (see also BEEBUG Vol.7 No.5).

ACES HIGH (BEEBUG Vol.8 No.4)

As a result of some last minute changes, the program as listed is too long to run in a model B without alteration. To achieve this, just add the following lines to the original:

```
 70*K.0*TAPE|MF.I%=0 TO TOP-PA. S.4:I%!
&E00=!(I%+PA.):N.|MPA.=&E00|MRUN
 80IF PA.>&E00 OSCLI"FX 138,0,128":END
```

FONT DESIGNER (BEEBUG Vol.8 Nos.4 & 5)

A function implementing an OSCLI call was inadvertently omitted from the printed listings. The simplest solution is to add these lines to the complete program:

```
6400 DEF PROCos(p0$)
6410 OSCLI(p0$)
6420 ENDPROC
```

Ultra Intelligent Machine

Reviewed by Peter Rochford

Product	**UIM**
Publisher	**The Fourth Dimension**
	P.O.Box 4444, Sheffield.
	Tel. (0742) 700661
Price	**£19.95 (5.25" disc) inc. VAT**
	£21.95 (3.5" disc) inc. VAT

I find it remarkable that at this stage in the history of the Beeb, anyone would ever release another so-called 'mega-game' for the machine. What else can you do that could possibly top the classic Elite ?

However, The 4th Dimension must seem to think that its new release UIM (Ultra Intelligent Machine) has all the right ingredients. Taking two years to write, UIM represents a lot of work and a great deal of confidence that the finished product will capture the imagination of those who use the Beeb for entertainment. UIM can be best described as an underwater Elite. There is no getting away from it, the influences and similarities are all there.

The scenario of the game is set in the future, when the greenhouse effect has caused life on land to become intolerable. The people have taken to the oceans, and in their quest to colonise their new underwater world, have created a replicator robot to work at depths that they cannot. As its name suggests, the robot self-replicates and it is this ability that ultimately leads to disaster, when they start to reproduce in a form that is mutated and able to attack and overrun their human masters. Your mission is to find the Ultra Intelligent Machine (UIM), a device that will wipe out the replicators and save the world.

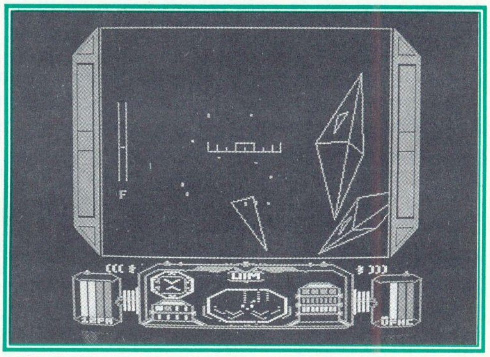
You must pilot your submarine around the great oceans starting from the port Anase and venturing to any one of 256 networks and 65,000 ports. At each port you can buy and sell goods to arm your submarine with a host of add-ons to help you in your quest.

Of course, on your way, you will meet a great deal of resistance to your passage. These enemy craft come in a huge range of shapes and levels of power and weaponry.

As in Elite, you must choose carefully what you attack and when. Your submarine will need the right armaments and the right navigational aids if you are to be successful. All this depends on skilful travelling and trading. Trading goes well beyond mere commodities, and includes stock market speculation, for example.

What more can I say about this game? It IS underwater Elite. But the scale of the game is much vaster than Elite, and should take those who buy it a lot longer to complete. The graphics are very similar, being 3D vector, but if anything they are coarser, although still excellent.

It is difficult to sum up this game. Why should you buy it if you have Elite and have already exhausted your interest in that? Why indeed? My own personal feelings are that it is too much of a copy of Elite to become a success. It has all been done before. Elite was revolutionary when released and took everyone by storm. Now we take games and graphics of that kind much more for granted.



If you still have an interest in Elite then UIM may be for you. Take a look at it before you buy. You will, however, need a Master to run it, or a Model B with at least 16K of sideways RAM.

Archimedes owners may be interested to know that a much enhanced version of the game (price £29.95) will be available shortly for them, and naturally it is claimed that it will make good use of that machine's graphics and sound capabilities.

Amateur Research (2)

John Belcher continues his forthright account of science and the role of the amateur researcher.

THE NON-EXISTENT-SPACE AGE

A little-known feature of this exciting space age is that space, apparently, as such no longer exists. Not even in the dictionary! It was finally laid to rest with its epitaph provided by the headlines in an American newspaper, saying "Don't Bring Back The Aether"!

This *aether* was originally held to be a medium which made possible the propagation of light waves - or in more general modern terms, electromagnetic waves - through space. Today we would recognise such a medium as having the properties of permittivity and permeability. René Descartes, no less, went further and suggested that the aether created *vortices* in space which accounted both for planetary rotation and orbital motion.

The Michelson-Morley experiment, however, soon put an end to all that. Michelson reasoned that if the aether did exist, and if we were travelling through it, then measurement of the velocity of light in two directions should indicate our velocity relative to it. The result of the Michelson-Morley experiment unexpectedly showed no difference in the velocity of light measured in the two directions. "Ergo", the cognoscenti proclaimed, "the aether does not exist!"

What Michelson, et al, had conveniently overlooked, was the possibility of the aether surrounding the planet attaining the same rotational motion as the planet itself, in true Cartesian fashion! Understandably, to suggest at this stage, dear reader, that the planet attains the motion of the surrounding aether, is perhaps asking too much of your gullibility. But you might with advantage bear it in mind.

Subsequent to Michelson, we had the Einstein era, where 'everything was relative'; where houseflies walked around the boundless space of oranges; and space-travel became the secret of eternal youth! Or so re-hash journalism told us. Meanwhile, space is our oyster, so let's make what we can of it while there is still time.

Key in and save program BPROG2, and run it, but don't press anything yet!

THE GRAVITATIONAL CONSTANT

The dimensions of G, the universal gravitational constant, are 1/(density*time^2). Today this is taken to describe the results of the 'big bang', i.e. the 'density of space' - a right misnomer if there ever was one - and 'Hubble Time' since the 'bang' itself.

The question which surely should have been asked is, "What has Hubble Time and the density of space got to do with Newton's Equation?", i.e. Fg=G*M0*M1/R1^2.

Be that as it may, if we combine this equation with Fi=M1*V1^2/R1, we finish up with the equation involving Kepler's Law:

```
        G*M0=4*PI*R1^3/T1^2
    Now let G=1/(d0*t0^2), and rearranging:
    M0/t0^2 = 3*PI*(4*PI/3*R1^3*d0)/T1^2
            = 3*PI*m0/T1^2.
```

Examining this equation carefully, we find on the left-hand side a mass, M0, together with its rotational period, t0; and on the right-hand side an apparent mass, m0, together with its rotational period, T1.

The apparent mass, m0, is a spherical volume of space, of radius equal to R1, and of a density equal to d0. One of Descartes' vortices, in fact, nicely illustrated in Fig-2.1.

When the body is first formed as a coherent mass, T1^2=t0^2, and the density of the body is D0=3*PI*d0, which I take to mean the density of the newly created matter in terms of the potential density of the space from which it is formed (not to worry, it all comes out in the wash).

I'll demonstrate. Let t0=86164.09056, the sidereal rotation period of the Earth in seconds. Then d0=1/(G*t0^2)=2.0184 kg m^{-3}. If we compare this potential-density of our surrounding vortex, with the density of dry air, Dair=1.2928 kg m^{-3}, we find close agreement! The agreement is even closer for the very cold air at -97 degrees C as shown on your monitor (press any key)!

"Big deal!" you will say. How can a medium of this low a density support the mass of the Earth? Very well, compare their relative masses. Let M0=5.9742E24 kg: Let R1=3.84404377E8 m: Quite a bit larger, but by how much?

THE SPACE-MATTER EQUATION

Admittedly, up to this point, what has been said is no more convincing than 'Hubble time' and the 'density of space'. Let M1=7.3483E22 kg, the mass of the Moon.
Then MO/M1=81.300, and m0/M0=80.385. Hence:

M0=SQR(m0*M1)

This equation - notice the geometric mean again - defines a Cartesian vortical space system in terms of its enclosed matter and space. Furthermore, it relates the masses of the primary body, M0, and the secondary body, M1, by way of the potential mass of the vortex, m0. The overall situation can of course be seen by reference to Fig-2.1.

Yes, I know! I've fiddled the figures somewhat. But then we are using a simple model. We assumed the general application of Kepler's Second Law, and assumed the Moon's orbit to be circular.

Kepler's Second Law refers to a point-source Moon. Hence, Let r1=1.7379E6 m, the Moon's radius, then R1'=R1+r1

Let e1=0.055, the eccentricity of the Moon's orbit. The length of its semi-minor axis, b=R1'*SQR(1-e1^2). Thus,

m0=4*PI/3*(R1'*R1'*b)*d0 = 4.8605E26 kg,
and m0/M0=81.358

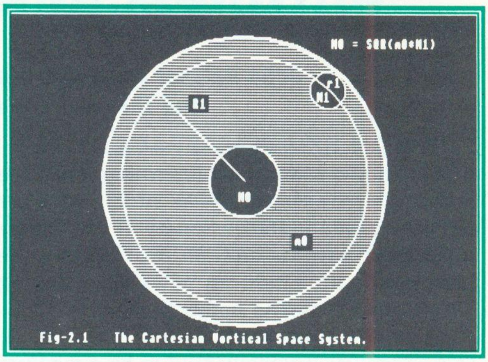


Fig. 2.1

A problem which arises here is the ultimate shape of the vortex. Is it a spheroid? An oblate spheroid? Or even a prolate spheroid? They all give different solutions.

THE MULTI-ELEMENT TO SINGLE-ELEMENT SPACE-SYSTEM TRANSFORM

This Transform opens up a whole new world of physics. What it does do is to replace a many-bodied space-system with its two-bodied equivalent, such as the Earth-Moon system. Despite all the frustration and time that went into its derivation, the Transform is delightfully simple.

1. The equivalent mass of the primary:
   M0=M0
2. The equivalent mass of the secondary system:
   ME=M1+M2+..+MN
3. The equivalent orbital radius of ME:
   RE=(R1*R2*..*RN)^(1/N)

You will, of course, recognise our old friend, unit radius (press any key)!

THE SUN'S ROTATIONAL PERIOD

We on Earth find it difficult to measure the sun's rotational period by way of its photosphere rather than its denser core. So we will attempt to determine the rotational period, ts, of the sun, Msun, by way of the Space-Matter Equation and the Multi-Body Transform.

Let Msun=1.9891E30 kg. Let Mp=Msun/743, the total mass of the secondary system. Then msun=Msun*743.
Let RU=3.242*1.49597892E11 m

dsun = msun/(4*PI/3*RU^3) = 3.093E-3 kg m^{-3}
tsun = SQR(1/(G*dsun))
= 2201213 secs = 25.477 days

This you will see is in good agreement with the published figure of 25.380 days - bearing in mind that we assumed the sun's vortex to be spherical and not spheroidal.

THE ROCHE VORTICAL SPACE-SYSTEM.

René Descartes' concept of vortices envisaged each body being encircled by its own vortex, and the vortices of adjacent bodies 'rolling' around each other to produce orbital motion.

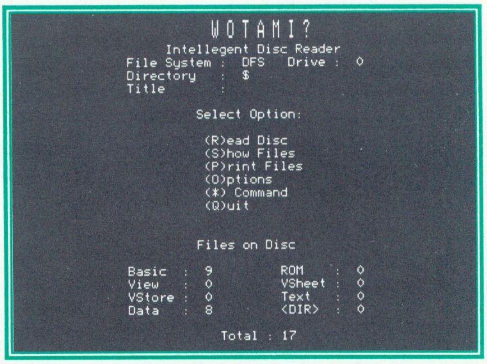
One similarly misunderstood Frenchman was one Edouard Roche who, some 150 years ago researched differential gravitational force, an inverse-cube-law force. This led to the concept of the Roche Limit, whereby a massive body sets up strains in the surrounding medium, resulting in discontinuities - or fractures - in the medium concerned.

Of course, the scientific world made a pig's-ear of the whole concept, giving rise to science fiction concerning the Moon being torn from the Pacific Ocean, and the formation of Saturn's ring system (c/f Cassini's Division and unit radius). The concept has now fallen into disuse.

My own research demonstrates that there are two main aspects of the Roche Limit, one involving non-rotating bodies, and one involving rotating bodies. The latter situation gives rise to the whole range of phenomena intuited by Descartes, and much more besides - gravitational (or Roche) waves, the latitude effect of sunspots, high-altitude jet-streams, and so on.

A major breakthrough has been the corrected Roche Constant, which - bearing in mind the ubiquitous constant PI - is given by:
B = 2*PI/((3*PI)^(1/3))

In the Earth-Moon system, Roche fractures form Cartesian vortices around each body, their radii being in the ratio (Mearth/Mmoon)^(1/3):1 = 4.3321:1, as shown in Fig-2.2. At the point where they touch, a satellite orbiting the Moon has the same period as a satellite orbiting the Earth, i.e. Ts=20.07 days.



```
                WOTAMI?
         Intellegent Disc Reader
  File System  :  DFS    Drive :  0
  Directory    :  $
  Title        :

             Select Option:

              (R)ead Disc
              (S)how Files
              (P)rint Files
              (O)ptions
              (*) Command
              (Q)uit

             Files on Disc

  Basic  :  9          ROM    :  0
  View   :  0          VSheet :  0
  VStore :  0          Text   :  0
  Data   :  8          <DIR>  :  0

               Total : 17
```

Fig. 2.2

The Moon and its vortex, is held to the Earth and its vortex, by mutual gravitational attraction. The Earth attempts to swing the Moon's vortex around in a circle in 20.10 days. The Moon does likewise, albeit in the opposite direction as seen from the Earth. The resulting situation is similar to the old problem of a snail climbing out of a well, and slipping back one metre for every four metres it climbs. At the end of some 26 days or so, the Moon's vortex completes its orbit around the Earth. By the way, bear in mind that this is a circular orbit (press any key).

Hold on, you will say, Tmoon=27.32166140 solar days! Of course! My calculation gives the orbital period for a system at rest! But because we in the Earth-Moon system are not at rest, time dilation predicted by Einstein becomes apparent, and Tmoon = beta*T1. Thence, knowing that beta=1.000099430, one can calculate the *absolute velocity* of the Solar System et al to 10 significant figures!

Explanation? We are measuring *translational motion* - which is relativistic, in terms of *rotational motion* - which is not relativistic.

But to get back to more important issues. The time taken for a Roche Wave to travel from the Earth to the Moon and back, taking care not to be reflected at the Roche discontinuity but to cross over where the two vortices touch, is Tmoon=2*R1/Vr, where Vr is its velocity. We find this velocity to be, Vr=325.7m s^{-1}, the velocity of SOUND in that somewhat cold air! Remember?

Now the velocity of sound is defined in terms of Vsound = SQR(elasticity/density). We know the potential-density of the surrounding Space, d0=2.0201 kg m^{-3}. What then is 'X' in the equation Vr = SQR(X/potential-density)?

A possibility I fancy is 'viscosity'. That said, here is a chance for you, the reader, to do your own bit of amateur research. Come up with a 'viscous' hypothesis using Kepler's Law to determine the velocity gradient of the medium, and you're in with a good chance!

RETROSPECT AND PROSPECT

Well, we have looked at the concept of space and vortical space-systems, and have apparently seen a need to change our minds concerning Descartes' contribution to 'non-material' physics. Next month we shall be looking at the concept of force.

```
   10 REM Program BPROG2
   20 REM Version B1.0
   30 REM Author  J.C.Belcher
   40 REM BEEBUG  November 1989
   50 REM Program subject to copyright
   60 :
  100 MODE4:G=6.6732E-11:t0=86164.09056
  110 PRINT''TAB(6)"THE GRAVITATIONAL CO
NSTANT, G"''
  120 d0=1/(G*t0^2):PRINT"The potential-
density of Earth's Space"''SPC7"= ";d0'
  130 tempC=((1.2928/d0)-1)/0.00367:PRIN
T"The  temperature  of  Earth's  Space i
n"''"terms of dry air"''SPC7"= ";tempC;"
 deg-C"':PROCpause
  140 PRINT''TAB(8)"THE CARTESIAN VORTEX
"''TAB(4)"& THE SPACE-MATTER EQUATION"''
  150 M0=5.9742E24:R1=3.84404377E8:m0=4*
PI/3*R1^3*d0:M1=7.3483E22:PRINT"m0     =
 "m0''"M0/M1 = "M0/M1''"m0/M0 = "m0/M0'
  160 r1=1.7379E6:R1a=R1+r1:e1=0.055:b=R
1a*SQR(1-e1^2):m0=4*PI/3*(R1a^2*b)*d0:PR
INT"The value of m0/M0 when a spheroidal
"''"vortex is assumed = "m0/M0:PROCpause
  170 PRINT''TAB(6)"THE SUN'S ROTATIONAL
 PERIOD"''':Msun=1.9891E30:Mp=Msun/743
  180 msun=Msun*743:RU=3.242*1.49597892E
11:dsun=msun/(4*PI/3*RU^3):tsun=SQR(1/(G
*dsun)):PRINT''"dsun = "dsun;" pot-kg m^
-3"''"tsun = "tsun;" secs"'
  190 tsun=tsun/(3600*24):PRINT"      = "
tsun;" days"''"The published value for t
sun"''"      = 25.380 days":PROCpause
  200 PRINT''TAB(12)"THE ROCHE VORTEX"'
  210 B=2*PI/((3*PI)^(1/3)):N=(M0/M1)^(1
/3):Rs=N/(N+1)*R1:Ts=SQR((4*PI^2*Rs^3)/(
G*M0)):Ts=Ts/(24*3600):PRINT'"The orbita
l period of a satellite at the"'"Roche v
ortices = "Ts;" days"'
  220 PRINT'"'The snail climbing the wal
l'!"''''TAB(5)"Increment"TAB(20)"Elapsed
time"':atime=0
  230 FORI%=0 TO 5:dtime=Ts*(1/N^I%):ati
me=atime+dtime:PRINT"N^";I% TAB(5)dtime
TAB(20)atime:NEXT
  240 PRINT''"The simple answer! (Ts*N/(
N-1))"':atime=Ts*N/(N-1):PRINTTAB(6) ati
me;" days":PROCpause
  250 PRINT''TAB(6)"THE ABSOLUTE VELOCIT
Y OF SPACE"'':T1=(B^3+1):PRINT"The theor
etical value of T1"''SPC7"= ";T1;" days"
  260 Tmoon=27.32166140:PRINT'"The measu
red value of T1"''SPC7"= ";Tmoon;" days"
  270 beta=Tmoon/T1:PRINT'"The time dila
tion resulting from this,"''"puts the va
lue of beta"''SPC7"= ";beta''
  280 c=2.99792459E8:v=SQR(c^2-(c^2/beta
^2)):PRINT"From which the ABSOLUTE VELOC
ITY of the"''"Solar System"''SPC7"= ";v/
1000;" km/sec":PROCpause
  290  PRINT''SPC5"THE VELOCITY OF A ROC
HE WAVE"'':Vr=2*R1/(Tmoon*24*3600):PRINT
"The mean velocity of a Roche Wave that"
''"travels to the Moon and back is found
"''" to be - "Vr;"m s^-1"
  300  PRINT''"This is the velocity of S
OUND in a cold"''"environment"'''''':END
  310  :
 1000 DEF PROCpause:PRINT TAB(13,31)"Pre
ss any key":A$=GET$:PRINT TAB(13,31);SPC
13:CLS:ENDPROC
```

Creating a Plus-or-Minus Character

Sebastian Lazareno explains how to add a useful character to the screen or printer repertoire.

A useful symbol which is missing from the Beeb's keyboard and normal character set is '±'. The program listed here, PLUSMIN, creates a machine code routine, saved as *pm*, which allows a user-selected character to be displayed as '±' both on screen (except in mode 7) and on an Epson RX80 printer (and other similar printers). Simply type the program in and save as usual, before running the program to create the code.

The program works by intercepting the Write Character vector, and substituting for the selected character a string of codes which temporarily redefine CHR$129 and control the printer. CHR$129 is redefined with a VDU23 string, printed, and restored to its original definition with another VDU23 string. These characters have no effect on the printer. The printer's HX-20 Graphics Mode is selected, CHR$159 printed, and Graphics Mode deselected. These characters are each preceded with CHR$1 and have no effect on the screen, or if the printer is off.

If the '±' is spooled from Basic (or from Wordwise Plus with an embedded command), the spooled file will contain the control codes and will always print ± when it is *TYPEd. If the '±' is spooled from the Wordwise Plus menu, the original character will be stored, so the spooled file can be edited but *pm* must be active for a correct display.

Once the code to generate the '±' character has been saved to disc, typing **pm* at any time will be sufficient to re-activate it.

```
  10 REM Program PLUSMIN
  20 REM Version B1.1
  30 REM Author  Sebastian Lazareno
  40 REM BEEBUG  November 1989
  50 REM Program subject to copyright
  60 :
 100 cc=129:code=&900
 110 wrchv=&20E:osword=&FFF1
 120 osbyte=&FFF4:osnewl=&FFE7
 130 mes$="  prints "
 140 MODE6
 150 FORpass=0 TO 2 STEP 2
 160 P%=code
 170 [OPT pass
 180 LDA wrchv:LDX wrchv+1
 190 CMP #start MOD256:BNE store
 200 CPX #start DIV256:BEQ message
 210 .store
 220 STA owv:STX owv+1:PHP:SEI
 230 LDA #start MOD256:STA wrchv
 240 LDA #start DIV256:STA wrchv+1
 250 PLP
 260 .message
 270 LDX #key MOD256:LDY #LEN mes$
 280 JSR pmes:JSR pm:JMP osnewl
 290 :
 300 .start
 310 CMP key:BNE wrch
 320 BIT &26A:BMI wrch \ chk vdu q
 330 PHA:TXA:PHA:TYA:PHA
 340 JSR pm
 350 PLA:TAY:PLA:TAX:PLA
 360 RTS
 370 :
 380 .pm
 390 LDX #oldef MOD256
 400 LDY #oldef DIV256
 410 LDA #10:JSR osword
 420 LDX #vnew MOD256:LDY #wrch-vnew
 430 .pmes
 440 LDA code,X:JSR wrch
 450 INX:DEY:BNE pmes
 460 RTS
 470 .key EQUS mes$
 480 .vnew
 490 EQUB 23:EQUB cc:EQUD &187E1818
 500 EQUD &7E0018:EQUB cc
 510 .vold EQUB 23
 520 .oldef
 530 EQUB cc:EQUD -1:EQUD -1
 540 .pcode
 550 EQUD &6D011B01:EQUD &9F010401
 560 EQUD &6D011B01:EQUW 1
 570 .wrch EQUB &4C  \ JMP
 580 .owv EQUW !wrchv
 590 ]
 600 NEXT
 610 REPEAT
 620 PRINT"Select key for ";
 630 CALLpm
 640 A=GET:PRINT"  "CHR$A
 650 UNTIL A>31 AND A<>127
 660 ?key=A:CALLcode
 670 PRINT"Save code ? ";
 680 IF INSTR("Yy",GET$)=0 PRINT"N":END
 690 PRINT"Y"
 700 OSCLI"SA.pm"+FNh(code)+FNh(owv)
 710 END
 720 :
1000 DEF FNh(A%)
1010 =" FF"+RIGHT$("000"+STR$~A%,4)
```

Using Operating System Routines

This month, Mike Williams explores, in simple terms, the value of calling operating system routines from within Basic.

As you may be aware, your BBC computer runs under the control of an *operating system*. Without this you would be reduced to communicating with your computer in the most primitive way, possibly just in binary, and you would find yourself obliged to write your own routines for what may seem the most elementary of functions, such as keyboard input, or saving a program.

Now any operating system is itself no more than a computer program, usually written in machine code and rather more complex than the average user program. Like any other program, the operating system consists of a large number of procedures or routines each performing a specific task. Because of their more general usefulness, many of these routines are documented, and may be accessed by the user, not only in assembler, but also from Basic. What is more, some of these routines provide functions which are simply not available in Basic. Thus learning how to call such operating system routines from within Basic can actually expand the range of tasks which your program can accomplish. Sometimes there is quite simply no other way.

That is what we shall be looking at in this article - how a Basic program can call a machine code routine, together with an investigation of some of the more useful operating system routines which are available. Don't be put off - you don't need to know how to program in machine code or assembler to use these calls.

You may already have come across operating system routines with references to OSBYTE calls, OSRDCH, OSWORD and the like. If you haven't, and these names seem somewhat daunting, bear with me and all will be revealed in due course.

CALLING O.S. ROUTINES FROM BASIC

The first thing which we have to do is to learn the basic principles of calling machine code routines from within Basic, and this applies to any machine code not just operating system calls. There are two keywords in Basic provided for this purpose, CALL and USR. These are quite similar but with one fundamental difference: USR returns a value to the host Basic program, whereas CALL does not, rather like the difference between the use of procedures (PROC) and functions (FN) in Basic.

Before we go any further we need to understand just a little about the 6502 processor, which is the heart of any BBC micro. This processor has a number of registers, specific locations each of which can hold a single byte. The three most important are the A register (accumulator), the X register and the Y register.

When a machine code routine is called, the accumulator, X and Y registers are often used as a means of passing information to the routine. Basic uses the variables A%, X% and Y% for the same purpose with CALL or USR.

If there is a need to send more than three values to a machine code routine this can be achieved using CALL. In this case matters become a lot more complex. An area of memory is reserved for the extra values or *parameters* which must follow a predefined format. However, this is all becoming much too complicated for the needs of this article and so we shall ignore any such complications.

The USR keyword is used like a function call in Basic and returns four values, the contents of the four registers P, Y, X, and A. These four bytes are returned as a single four-byte number. If we require any of these values individually, then the four bytes have to be separated as we shall see later.

The final point to understand at this stage is the manner by which any particular routine is identified when it is to be called. This is done by including with CALL or USR the memory address of the start of the required routine. For example, if a routine starts at address &3000 (in hex), then in Basic we could write:

```
        CALL &3000
```

or alternatively:

```
        P%=USR(&3000)
```

Note how the USR function has to return a value to a variable.

Now a numeric address like &3000 is not very memorable, or descriptive in any way of the routine being called, so most programmers like to use a name instead. Thus:

```
        readch=&3000
        CALL readch
```

This has the advantage that the numeric address only has to be assigned once to the variable *readch*, but it can be called as many times as required.

Most operating system calls are also given names, again as a memory aid, but Basic will not recognise the routine unless the name is correctly associated with the proper numeric address. For example, the OSBYTE call that I referred to much earlier is actually located at address &FFF4. Thus a program will likely have near its start a statement like:

```
        OSBYTE=&FFF4
```

From then on, but only then, we can write:

```
        CALL OSBYTE
```

whenever we need to call that routine.

Of course, not all programmers bother to make this assignment, so you will often see things like:

```
        CALL &FFF4
```

or:

```
        CALL &FFF7
```

If you are not familiar with the hexadecimal address it is very difficult to know what such a program is doing.

USING OPERATING SYSTEM CALLS

I want to start with a simple example, but one I hope which you will find easy to follow, even if it is redundant in many cases. In Basic II and later versions of Basic there is a statement OSCLI. This is a way for a Basic program to execute a star command, but without some of the disadvantages. Let's take a typical example.

Suppose we have a program written in Basic which will save the state of screen at intervals determined by the user. We want the first screen to be saved with the name *Screen1*, the second with the name *Screen2* and so on. If the file names can be given explicitly, i.e. as:

```
        *SAVE Screen1 7C00 8000 7C00 7C00
```

then there is no problem. But suppose we want to embed such a statement in a loop of some kind so that each time round the loop, if some key is pressed a new screen will be saved.

We need to use some variable, say S%, which will initially be set to 1, and be incremented each time a screen display is saved. There is no way that this can be done in a straightforward *SAVE command. The solution is to use OSCLI:

```
        S%=1
        <start of loop>
        ...............
        OSCLI("SAVE Screen"+STR$(S%)+" 7C00
 8000 7C00 7C00"):S%=S%+1
        ...............
        <end of loop>
```

Each time the OSCLI statement is executed, the string enclosed in brackets is treated as a star command. The value of S% determines whether Screen1, Screen2 etc is to be used.

Now you may be wondering what this has to do with machine code routines called from Basic. Well, first of all, OSCLI is simply a Basic keyword which itself calls a machine code routine of the same name with start address &FFF7.

Furthermore, in Basic I, the keyword OSCLI was unavailable in Basic. Thus the only way to perform the task described above was to call this machine code routine from within Basic. Here's how it can be done.

First of all it is necessary to reserve some bytes of memory for the command string, such as:

```
        DIM os 50
```

which reserves 50 bytes starting at address *os* (Basic determines just where this is - it doesn't matter to us). The equivalent code now looks as follows:

```
    DIM os 50
    OSCLI=&FFF7
    S%=1
    <start of loop>
    ...............
    $os="SAVE Screen"+STR$(S%)+" 7C00 80
00 7C00 7C00"
    X%=os MOD 256
    Y%=os DIV 256
    CALL OSCLI
    ...............
    <end of loop>
```

X% holds the low byte of the address of the command string (obtained with MOD), while Y% holds the high byte of the address (obtained with DIV). The OSCLI routine performs a task rather than returning any information, so the CALL statement is entirely adequate (as opposed to using USR).

I present this merely as an example which I hope you have followed alright, as most readers will now be using Basic II or later versions. Nevertheless, it is a useful example for demonstrating the use of operating system calls in Basic, and the only method available in Basic I.

Now let's have a look at another of the other calls which are available to us. In fact all the *FX calls which we can use in Basic are examples of calls to the routine called OSBYTE. In this case, the value in the accumulator (the value assigned to A% before calling the routine) determines which OSBYTE call is used. Naturally, the FX calls are easier to use, but there are many more OSBYTE calls than there are FX equivalents.

Most of the OSBYTE calls which do not have FX equivalents are those which return a value of some kind, something which FX calls cannot do. One useful one is OSBYTE 135. This returns the character at the current cursor position on the screen. There is no other way of achieving this from within Basic.

After the call, the X register holds the ASCII value of the character in the specified position. Because a value has to be returned, we must use USR this time rather than CALL.

In fact, we can parcel the whole lot up in a convenient Basic function to read the character in any specified position (X,Y), thus (assuming *OSBYTE* has been previously defined before the function is called):

```
    DEF FNreadch(X,Y)
    LOCAL A%,X%,oldX,oldY
    oldX=POS:oldY=VPOS
    VDU31,X,Y
    A%=135
    X%=USR(OSBYTE)
    X%=(X% AND &FF) DIV &100
    VDU31,oldX,oldY
    =CHR$(X%)
```

Because OSBYTE 135 reads the character in the *current cursor position,* the cursor must first be moved to the (X,Y) position given. For later reference, the initial position of the cursor is saved in (oldX,oldY) first. A% (for the accumulator) is set to the value of this OSBYTE call. After calling the routine, the value returned is assigned to X%. We want the value from the X register, which is the next to lowest byte of the returned value, extracted with the AND and DIV operators (see below). The cursor is then returned to its original position before the routine exits with the required character.

For example, in mode 3 we could write:

```
A$=FNreadch(20,20)
```

which would assign to the variable A$, the character currently in position (20,20) on the screen, leaving the cursor wherever it was before the call was made.

To finish this month, I want to explain in more detail how the four bytes returned by USR may be separated. In fact two steps are required in most cases, firstly using a *mask* to separate the required eight bits (represented as two hexadecimal digits) from the rest of the number, and secondly shifting those bits to form the correct number. The mask required is shown in figure 1 (in practice leading zeros can be omitted). After separation it is necessary to shift the eight bits to the rightmost position, and this can be achieved by dividing (in hex) by &100, &10000, or &1000000 as appropriate. The value of the A register is already in the rightmost position and does not therefore need shifting.

continued on page 58

A Postscript Screen Dump Utility

Following our previous introduction to the PostScript language, Willem van Schaik describes a program to convert a graphics screen dump into a PostScript file for output to a laser printer.

INTRODUCTION

When you have the opportunity to use a PostScript printer, it only takes a short while before you want to use it for high-resolution screen dumps. If you use the printer for desktop publishing, incorporating BBC screen dumps in your publications becomes another desirable feature. Because of the flexibility of PostScript it is quite easy to build in facilities for scaling and inverting images, and other functions for picture manipulation.

Most people who have tried programming with the PostScript "page description language", will have kept to the use of vectorized drawings and fonts. This way of addressing PostScript printers gives a high output quality coupled with maximum flexibility.

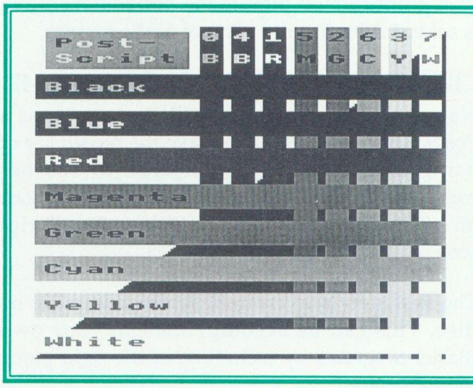
Besides the use of vectors, PostScript also has commands for the use of bitmaps. For readers who do not know the difference: a vectorized description of a picture can be looked at as a collection of move, draw and plot commands, like those in BBC Basic, while a bitmap definition of the same picture is the resulting screen-image which will be displayed on a monitor, but which can also be saved or printed.

A screen dump is essentially a bit image, where all objects consist merely of pixels on the screen (rather than in the form of lines and vectors). Screen dumps can be both saved and printed.

The purpose of the program listed here is to provide a means of converting graphics screen dumps (that is, screen dumps in modes 0, 1, 2, 4 and 5) into equivalent PostScript files. These files can then be output to a PostScript laser printer connected to your Beeb, or transferred to any other system which uses a PostScript printer.

POSTSCRIPT IMAGE COMMAND

In general, screen dumps consist of many pixels (visible points on the screen), where one or more bits are needed to define the colour or grey-scale of each pixel. What varies from one computer to another is the order in which the pixels are processed (by row or by column, for example,) and the starting point (upper left-hand corner with a BBC). The BBC micro also has an intricate way of mixing the bits that define two or four pixels.



A mode 2 demo screen output to a laser printer

In the PostScript language, the image command is used to establish a bit-mapped picture. The image command requires 5 parameters: first the number of lines and the number of pixels on each line, then the number of bits per pixel. With this parameter the number of greys can be changed - when 1 is chosen for this parameter, a black and white picture is created like a BBC mode 0 screen - the maximum is 8 (bits per pixel), allowing 256 possible grey-values.

The fourth parameter is a so-called transformation matrix (see Introducing PostScript, BEEBUG Vol.8 No.4 p18). In this screen dump program, the transformation matrix is used to change the BBC format, where the picture starts with the upper left-hand corner, into the PostScript way where the lower left-hand corner comes first. The last parameter is a PostScript procedure which contains the picture data. If everything is alright, the size in bits will be the number-of-lines x pixels-per-line x bits-per-pixel. I have used what PostScript calls a hexadecimal string for the picture data. This means that for each 8 bits, two bytes are needed (00 to FF).

If nothing is done before the image command is given, the result will be a picture one PostScript unit in size. This is 1/72 inch by 1/72 inch. To get a larger picture, some scaling is needed using the PostScript scale command. To get a well proportioned picture the horizontal scaling is set to 1.25 times the vertical scaling factor.

CREATING A POSTSCRIPT SCREEN DUMP

Not many BBC owners are likely to possess a PostScript printer, but may have the possibility of using such a printer connected to another computer (in the office say). Thus I have chosen an off-line approach for creating a PostScript screen dump.

The first step is to create a screen dump file on disc. This can be accomplished in your own (Basic) programs by including, at the right point, the statement:

```
    *SAVE DUMP 3000+5000 0000 3000
```

when the picture is in mode 0, 1 or 2, or for modes 4 and 5:

```
    *SAVE DUMP 5800+2800 0000 5800
```

To convert this to a PostScript file type in and save the PostScript screen dump program listed at the end of this article. When you run the program it will prompt for the necessary information:

1. The file name of the screen dump (in our example DUMP).

2. The mode of screen: 0, 1, 2, 4 or 5.

3. The file name of the PostScript file (for example P.DUMP) to be created. Take care that there is plenty of room on the disc, as these files are large (as much as 160K in some modes).

4. When the picture is to be incorporated into a desktop publishing system, a so-called encapsulated PostScript file is needed. Normally answer N(o) to this question.

5. You can rotate the picture to a landscape format. For some DTP packages this leads to problems, but normally the answer to this question is Y(es).

6. The width of the resulting picture must be given in millimetres.

7. To save toner in the laser printer, a picture with white lines on a black background can be inverted.

8. A small black border around the picture can be optionally added.

When all of this is done, the PostScript file will be generated on screen, and at the same time spooled to disc, which can take some time. The resulting file can then be transferred to the computer to which the printer is connected. Note: Kermit is one effective way of transferring files between systems. When the destination is an IBM-PC (or compatible), take care that the end-of-record of a BBC spool file is <0A><0D> as the PC uses <0D><0A> as end-of-record. The same file can also be output to any PostScript printer directly connected to your BBC micro.

PROGRAM DETAILS

Looking at the program, most of the procedures are self-explanatory. However, a few tricks were needed to convert from a BBC style of bitmap to a PostScript one. The different ways of pixel sequencing (PostScript being much more straightforward), is solved by the nesting of the loops in lines 270-290 and the calculation in line 300.

FNconvert performs the most important part of the program. In BBC modes 1 and 5, the two bits that define one pixel are spaced 4 bits apart. This is converted in line 7050 to the PostScript format where the 2 most significant bits define the first pixel and the next 2 its neighbour, and so on. In mode 2 things are even more complicated. The first step is to unravel the odd and the even bits. This results in two nibbles defining two pixels. Then the RGB values must be resequenced. The simplest way is to use an array which is initialized in PROCinit. Keep in mind that in PostScript, &0 is black and &F is white. Further, the flashing colours are set equal to the non-flashing ones.

The array p2% in PROCinit is filled with some powers of 2, which results in a large performance improvement.

The last part to explain is the PostScript BoundingBox in lines 5070-5090. With the BoundingBox, a DTP package can find out which part of the page contains the picture it must include. Therefore it is defined in the PostScript default co-ordinate-system: (0,0) is the lower left-hand corner of the page using a unit-size of 1/72 inch. After the %%BoundingBox: comment, the co-ordinates of the lower left-hand corner and of the upper right-hand corner of the picture must be given. To accomplish this some calculations caused by the possibility of scaling and rotation are needed.

WHAT'S NEXT?

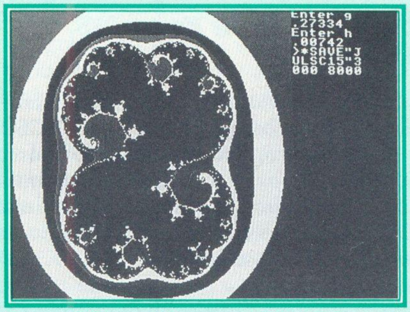
In a following article I will give another method to address a PostScript printer. When all screen drawing commands in a BBC Basic program are replaced by procedure calls that generate the PostScript equivalents, we can really make use of the high resolution provided by laser-printers.

```
   10 REM Program PostScript screendump
   20 REM Version B 1.6
   30 REM Author  Willem van Schaik
   40 REM BEEBUG  November 1989
   50 REM Program subject to copyright
   60 :
  100 ON ERROR CLOSE#0:OSCLI("SPOOL"):RE
PORT:PRINT" at line ";ERL:END
  110 MODE 7
  120 PROCinit
  130 PROCinput
  140 PROCparam(mode%)
  150 base%=&2C00
  160 OSCLI("LOAD "+filename$+" "+STR$~(
base%))
  170 OSCLI("SPOOL "+postfile$)
  180 PROCprolog
  190 PRINT "300 420 translate"
  200 IF rot% THEN PRINT "90 rotate"
  210 IF bor% THEN PROCborder
  220 PRINT ;wix%;" neg ";wiy%;" neg tra
nslate"
  230 PRINT ;2*wix%;" ";2*wiy%;" scale"
  240 PRINT ;pix%;" 256 ";bpp%;" [";pix%
;" 0 0 -256 0 256] {<"
  250 :
  260 p%=0
  270 FOR line%=0 TO 31
  280 FOR row%=0 TO 7
  290 FOR col%=0 TO pix%*bpp%-1 STEP 8
  300 scrbyte%=?(base%+pix%*bpp%*line%+r
ow%+col%)
  310 psbyte%=FNconvert(mode%,scrbyte%)
  320 IF inv% THEN psbyte%=255-psbyte%
  330 IF psbyte%>&F THEN PRINT ;~psbyte%
; ELSE PRINT ;0;~psbyte%;
  340 p%=p%+2 : IF p%=80 THEN PRINT:p%=0
  350 NEXT col%
  360 NEXT row%
  370 NEXT line%
  380 :
  390 PRINT">} image"
  400 IF NOT eps% THEN PRINT"showpage"
  410 OSCLI("SPOOL")
  420 END
  430 :
 1000 DEF PROCinit
 1010 LOCAL i%
 1020 DIM p2%(7)
 1030 FOR i%=0 TO 7
 1040 p2%(i%)=2^i%
 1050 NEXT i%
 1060 DIM mo2col%(15)
 1070 FOR i%=0 TO 15
 1080 READ mo2col%(i%)
 1090 NEXT i%
 1100 DATA 0,4,8,14,1,7,11,15,0,4,8,14,1
,7,11,15
 1110 ENDPROC
 1120 :
 2000 DEF PROCinput
 2010 PRINT STRING$(4,"-");" BEEBUG Post
Script Screen Dump ";STRING$(4,"-")
 2020 REPEAT
 2030 INPUT'"Screen dump filename: " fil
ename$
 2040 sc%=OPENIN(filename$)
 2050 UNTIL sc%<>0
 2060 CLOSE# sc%
 2070 REPEAT
 2080 INPUT "Mode of screen dump : " mod
e%
 2090 UNTIL mode%>=0 AND mode%<>3 AND mo
de%<=5
 2100 REPEAT
 2110 INPUT "Postscript file name: " pos
tfile$
 2120 ps%=OPENIN(postfile$)
 2130 CLOSE# ps%
 2140 IF ps%<>0 THEN INPUT "Replace file
  <Y/N>: " yn$ : rpl%=FNyn(yn$) ELSE rpl
%=FALSE
 2150 UNTIL ps%=0 OR rpl%
 2160 INPUT "Encapsulated  <Y/N>: " yn$
: eps%=FNyn(yn$)
 2170 INPUT "Rotate dump    <Y/N>: " yn$
: rot%=FNyn(yn$)
 2180 INPUT "Width dump in mm.  : " widt
h
 2190 INPUT "Dump inversed <Y/N>: " yn$
: inv%=FNyn(yn$)
 2200 INPUT "Add border     <Y/N>: " yn$
: bor%=FNyn(yn$)
 2210 PRINT'STRING$(40,"-")
 2220 wix%=width*36/25.4
 2230 wiy%=width*36/31.75
 2240 ENDPROC
 2250 :
 3000 DEF FNyn(answer$)
 3010 IF INSTR(" Yy",answer$)>1 THEN =TR
UE ELSE =FALSE
 3020 :
 4000 DEF PROCparam(mode%)
 4010 LOCAL i%,dummy%
 4020 FOR i%=0 TO mode%
 4030 READ dummy%,pix%,bpp%
 4040 NEXT i%
 4050 REM mode,horiz.pixels,bits/pixel
 4060 DATA 0,640,1
 4070 DATA 1,320,2
 4080 DATA 2,160,4
 4090 DATA 3,0,0
 4100 DATA 4,320,1
 4110 DATA 5,160,2
 4120 ENDPROC
 4130 :
 5000 DEF PROCprolog
 5010 PRINT "%!PS-Adobe-1.0"
 5020 PRINT "%%DocumentFonts:"
 5030 PRINT "%%Title: ";postfile$
 5040 PRINT "%%Creator: Acorn BBC Screen
Dump"
 5050 PRINT "%%Pages: ";
 5060 IF NOT eps% THEN PRINT "1" ELSE PR
INT "0"
 5070 PRINT "%%BoundingBox: ";
 5080 IF rot% THEN PRINT ;300+wiy%+1;" "
; 420-wix%-1;" ";300-wiy%-1;" ";420+wix%
+1
 5090 IF NOT rot% THEN PRINT ;300-wix%-1
;" "; 420-wiy%-1;" ";300+wix%+1;" ";420+
wiy%+1
 5100 PRINT "%%EndComments"
 5110 PRINT "%%EndProlog"
 5120 ENDPROC
 5130 :
 6000 DEF PROCborder
 6010 PRINT "0 setgray"
 6020 PRINT "newpath "
 6030 PRINT ;wix%+1;" neg ";wiy%+1;" neg
moveto"
 6040 PRINT ;wix%+1;" ";wiy%+1;" neg lineto"
 6050 PRINT ;wix%+1;" ";wiy%+1;" lineto"
 6060 PRINT ;wix%+1;" neg ";wiy%+1;"
lineto"
 6070 PRINT "closepath fill"
 6080 ENDPROC
 6090 :
 7000 DEF FNconvert(mo%,sb%)
 7010 LOCAL i%
 7020 IF mo%=0 OR mo%=4 THEN =sb%
 7030 pb%=0
 7040 FOR i%=0 TO 3
 7050 IF mo%<>2 THEN pb%=pb% + (sb%ANDp2
%(i%))*p2%(i%) + (sb%AND(p2%(i%)*&10))DI
Vp2%(3-i%)
 7060 IF mo%=2 THEN pb%=pb% + (sb%ANDp2%
(2*i%))DIVp2%(i%) + (sb%ANDp2%(2*i%+1))*
p2%(3-i%)
 7070 NEXT i%
 7080 IF mo%<>2 THEN =pb% ELSE =mo2col%(
pb%DIV&10)*&10 + mo2col%(pb%MOD&10)
 7090 :
```

The Best of BEEBUG

Basic Booster ROM

Super Squeeze	A program compressor.
Partial Renumber	A very useful utility which renumbers a selected block of lines.
Textload and Textsave	Save and load a Basic program as text.
Program Lister	List any program direct from a file.
Resequencer	Rearrange the lines in a Basic program - line numbering is automatically adjusted.
Smart Renumber	Renumber a program so that procedures start at a particular line number.



```
Enter g
.27334
Enter h
.00742
>*SAVE"J
ULSC15"3
000 8000
```

Applications II Disc

Share Investor
A program which assists decision making when buying and selling stocks and shares.

Label Processor
Design, save and print labels at any size on an Epson compatible printer.

3D Landscapes
Create computer-generated three dimensional landscapes with this program.

Monthly Desk Diary
A month-to-view calendar which can be used on-screen or printed out.

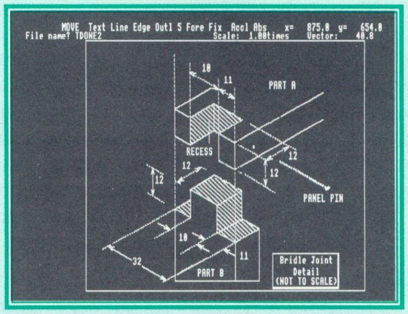
Real Time Clock
A real time digital alarm clock displayed for all BBC micros.

Running Four Temperatures
A program for calibrating and plotting up to four temperatures.

Crossword Editor
Design, edit and solve crosswords with this program.

Julia Sets
Fascinating displays of Julia sets, the extensions of the Mandelbrot set.

Foreign Language Tester
Define foreign characters and test your knowledge of foreign languages.



```
      MOVE  Text Line Edge Outl 5 Fore Fix  Accl Abs    x=   875.0  y=   654.0
File name? TDONE2                                Scale:  1.00times   Vector:    40.8
```

ASTAAD

Enhanced ASTAAD CAD program with

- full mouse and joystick control,
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

Applications I Disc

- Business Graphics
- Video Cataloguer
- World by Night and Day
- Phone Book
- Page Designer
- Personalised Letter-Heads
- Mapping the British Isles
- Selective Breeding
- Appointments Diary
- The Earth from Space
- Personalised Address Book

General Utilities Disc

- Printer Buffer *
- ROM Controller
- Sprite Editor/Animator
- Multi-Character Printer Driver for View
- Mode 7 Screen Editor
- Multi-Column Printing
- Epson Character Definer
- BEEBUG MiniWimp †
- ROM Filing System Generator

*** Master series only. † Requires sideways RAM.**

✂ -

Please rush me my Best of BEEBUG disc at the members price of £5.75 (non-members price £15)
(ASTAAD disc - members price of £9.95 , non-members price £19.95)

Applications II *Code 1411A (80 track DFS)*	☐	Applications II *Code 1412A (3.5" ADFS)*	☐
Basic Booster ROM *Code 1403A*	☐	Basic Booster Disc *Code 1402A*	☐
ASTAAD Code 1407A *(80 track DFS)*	☐	ASTAAD Code 1408A *(3.5" ADFS)*	☐
General Utilities Disc *Code 1405A (80 track DFS)*	☐	General Utilities Disc *Code 1413A (3.5" ADFS)*	☐
Applications I Disc *Code 1404A (80 track DFS)*	☐	Applications I Disc *Code 1409A (3.5" ADFS)*	☐

Name ______________________________

Address ____________________________ Price £________

____________________________ Postage £ 0.60

Memership No________________________ Total £________

I enclose a cheque for £____________ OR please debit my Access, Visa or Connect account, Card No_________/_________/_______/________ Expiry_____/____ Signed______________________

Send to: BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Telephone (0727) 40303.

The CLOSE#0 Bug in Master DFS

Derek Gibbons explains the CLOSE#0 bug present in the Master DFS, and presents a solution.

Reference is often made to the bug in the Master's DFS (version 2.24). This bug results in data files (i.e. files opened for writing using OPENOUT or OPENUP) being closed by CLOSE#0 at their original size instead of at a size which reflects the amount of data just written.

For new files, this is not too bad as they are closed at the default size of &4000 bytes, so at least all the data written will, in most situations, actually be present in the file. However, it means that when the file is subsequently opened for input, any attempt to read data on an 'UNTIL EOF#X' basis will result in garbage being read once the true end of data has been passed.

On the other hand, if an existing file is re-used, it will be closed by CLOSE#0 at its original size even if more data is written than was originally present. This means that the tail-end of such new data will be irretrievably lost.

The problem is illustrated by the program in Listing 1 below where line 40 ensures that the data file does not already exist; lines 80 and 140 show the extent of the data before the file is closed; and lines 100 & 160 show the size of the file after it has been closed. As the listing stands, the file is &4000 bytes long on each occasion, despite the fact that only &12 and &19 bytes are written respectively.

But now change line 90 to read CLOSE#X. On both occasions the file is now &12 bytes long and the tail-end of the longer string is lost, and an EOF error is generated if an attempt is made to read the longer string. Now change line 150 to also read CLOSE#X and see what should really happen - on each occasion the file is closed with a size equivalent to the data written. Note that more bytes are used than the number of characters in a string because of the header bytes associated with each string (see User Guide).

It must be emphasised that this problem only arises with the use of CLOSE#0; if CLOSE#X is used, as above, the file is closed correctly. Unfortunately, even commercial programs often use CLOSE#0 for simplicity, and may also use CLOSE#0 at the start of a program to ensure that no other files are open.

Listing 1

```
   10 REM Prog1
   20 REM D.GIBBONS November 1989
   30 :
   40 *REMOVE TEMP
   50 X=OPENOUT"TEMP"
   60 S$="This is a string"
   70 PRINT#X,S$
   80 PRINT X;" contains - ";S$;" - ";~E
XT#X;" bytes long"
   90 CLOSE#0
  100 *I.TEMP
  110 X=OPENOUT"TEMP"
  120 S$="This is a longer string"
  130 PRINT#X,S$
  140 PRINT X;" contains - ";S$;" - ";~E
XT#X;" bytes long"
  150 CLOSE#0
  160 *I.TEMP
  170 END
```

Fortunately, as CLOSE#X does work correctly, it is possible to simulate a correct CLOSE#0 by closing, or at least attempting to close, in turn all the five files permitted by the DFS. These will have 'handles' in the range 17 to 21, but any attempt to close a file which is not open will cause a Channel error (error no. 222) so it is necessary to trap this error and then ignore it.

In a program designed for universal use, it would be preferable to check also that DFS, rather than ADFS, is indeed in use (ADFS uses different file handles and, in any case, the CLOSE#0 error does not occur), and even that the program is running on a Master rather than say a Model B. It is assumed here, however, that any program changes are being made because the program is being run on a Master under the DFS. The program in Listing 2 shows an answer to the problem. This is not very elegant, and is certainly not 'structured'. It would be nice to have a PROCcloseall or FNcloseall, but BBC Basic inconveniently forgets virtually all pointers when an error occurs, and this then precludes the use of a procedure or function (even a GOSUB) as well as REPEAT-UNTIL or FOR-NEXT loops. Hence, this would appear to be at least one situation where the ubiquitous GOTO comes into its own!

Listing 2

```
   10 REM Prog2
   20 REM D.GIBBONS November 1989
   30 :
   40 *REMOVE TEMP
   50 X=OPENOUT"TEMP"
   60 S$="This is a string"
   70 PRINT#X,S$
   80 PRINT X;" contains - ";S$;" - ";~E
XT#X;" bytes long"
   90 add%=100:GOTO 200
  100 *I.TEMP
  110 X=OPENOUT"TEMP"
  120 S$="This is a longer string"
  130 PRINT#X,S$
  140 PRINT X;" contains - ";S$;" - ";~E
XT#X;" bytes long"
  150 add%=160:GOTO 200
  160 *I.TEMP
  170 END
  180 :
  190 REM CLOSE#0 SIMULATION
  200 N=22:*FX119
  210 N=N-1:IF N=16 THEN GOTO add%
  220 ON ERROR GOTO 250
  230 CLOSE#N
  240 GOTO 210
  250 ON ERROR OFF
  260 IF ERR=222 THEN GOTO210 ELSE PRINT
"Error ";ERR;" at ";ERL
  270 END
```

Lines 90 and 150 now contain the jumps to the CLOSE#0 routine, with add% being used as a return address pointer and set, in each case, to contain the line number of the next line (if the routine is used only once in a program, add% can be dispensed with and an explicit jump back made at line 210). Lines from 200 onwards contain the actual CLOSE#0 simulation, which is simple enough to be more or less self-explanatory. The most puzzling aspect may be the FX119 command in line 200 which closes SPOOL and EXEC files. This is required because CLOSE#0 means literally close ALL open files, including any EXEC or SPOOL files which may be open, and this is indeed what happens with a 'good' DFS.

This brings to light another aspect of CLOSE#0 which occurs even with a good DFS. When a !BOOT file or any other EXEC file is used to chain a program, the !BOOT file is treated as a data file and opened for input of the command lines in sequence. The file remains open until the Basic program has completed, control has returned to the !BOOT file, and any subsequent commands in the file have been executed.

Try the !BOOT file shown here with the original program 1, and then again after changing lines 90 and 150 as above to CLOSE#X and re-saving the program. Incidentally, this means that only four actual data files can then be opened, rather than the 5 permitted by the DFS, because the !BOOT file is actually the first one. This is another reason why some programs begin with CLOSE#0.

```
*|| !BOOT DEMO
*||
CHAIN"Prog1"
PRINT"returned to BOOT file"
```

In the CLOSE#0 simulation, if all files are closed simply by sequential use of CLOSE#N, when control is returned to the !BOOT file its file handle will have been removed, but the OS does not know that the file is no longer in use for EXEC purposes. It therefore attempts to read any subsequent commands from an unopened file and a Channel error again occurs. The *FX119 command corrects this situation making things no worse than with a 'good' DFS.

512 Forum

by Robin Burton

As promised, this Forum includes a list of packages known to run with Problem Solver's assistance. This is at the end of the Forum for convenience, but first to more 'traditional' Forum matters.

The first snippet is that I understand Acorn no longer charge for the DOS Plus 2.1 upgrade. Send your original 1.2 discs to Acorn Computers Ltd., Customer Services, Fulbourn Rd., Cherry Hinton, Cambridge CB1 4JN.

BATCH FILES AGAIN

It's a couple of months since we looked at the 'PATH' command, so I thought it was time I kept my promise to cover batch files again. We also looked at the 127 byte buffer used to hold the command tail, the parameters entered when a program is first loaded. This time we'll be looking at precisely the other end of proceedings.

In particular, I've had several letters requesting information on 'ERRORLEVEL', a point missed entirely or poorly explained in so many DOS guides (could it be that the authors aren't sure either - surely not!). At the same time we'll investigate a little more how DOS programs work.

IF ERRORLEVEL

Most users are aware of this batch file construct, and how to include the relevant line in a batch file, but equally, many don't know how to use it. This is because good explanations of what 'ERRORLEVEL' means and how it works are few and far between. In fact it's quite simple, if the information is gathered together and given with an explanation, instead of being scattered throughout reference books with an assumption that you can make the connections yourself, as is too common.

The main point to appreciate is how programs terminate in DOS. Just like other DOS facilities and functions, if performed legally, this involves calling an interrupt. There are other, 'dirtier' methods of terminating, and there is a range of termination functions, but for illustration we'll confine ourselves to one legal technique. We'll consider the sort of program that loads, carries out one or two specific operations and then disappears, leaving its memory free for further use.

Most of the programs that operate like this are utilities, and are most frequently 'COM' files, (or 'CMD', but more in a moment about these). However, the principle applies equally to 'EXE' files, whether these are linked from object modules originally written in assembler, or compiled from high level languages like 'C', Basic, Pascal or whatever. Any program which, on final termination, can report overall success or failure to DOS is therefore included. What this topic doesn't include are user errors handled entirely within applications.

Let's take for example a program which reads a file, performs a specific operation (or several) and then terminates. Examples of this type of program are the 512 'FIND.EXE' and 'SORT.EXE' utilities included on two recent BEEBUG discs (Vol.8 Nos.4 & 5) - get the back-issues out if you missed them. These programs take a filename and one or more optional parameters when they are called from the command line, but then execute without further user input until completion. Of course, if you call the program manually and the parameters are invalid, you are informed at once and you simply re-enter them correctly.

On the other hand, if the program were to be called by commands contained in a batch file you wouldn't be given the opportunity to re-enter the parameters correctly. In any case failure might have nothing to do with entered parameters, it might be a file error, the wrong disc in a drive and so on.

More importantly, if left to itself, a batch file will continue to execute subsequent instructions, probably quite pointlessly and possibly even dangerously if an earlier process has failed. Of course, you can always sit and watch the screen, issuing a hasty Ctrl-C when you spot a problem, but as usual there's a better way.

PROGRAM TERMINATION

To fully understand how this better method (i.e. ERRORLEVEL) works, we need to look at DOS program termination.

If a program is written correctly, that is according to accepted DOS standards, on termination it should set a condition which can be detected by the program or batch file that called it. Primarily this condition (a numeric value) indicates whether the program was completely successful or not, but by varying the value it can also be used to identify the seriousness and type of any error.

Using machine code in our example, here's how it's done. By the way, this applies only to DOS. Remember that in DOS Plus we have both CP/M (CMD) and MS/PC-DOS (COM) programs. While CP/M has a similar facility, it doesn't work with 'ERRORLEVEL'. CP/M return codes can only be obtained by a calling program, not by batch files, and they also differ in that they are two bytes, not one as in DOS.

We'll ignore 'terminate and stay resident' functions such as INT 21h function 31h or the now redundant INT 27h, as used by 'pop-ups' and programs like RAM discs. In the main, programs don't stay resident if they encounter errors (although a return code is also passed with INT 21h f31h).

For a permanent exit, programs can terminate using one of two interrupt 21h functions, 0 or 4Ch. INT 21h incidentally, is also commonly referred to as the general function despatcher.

For those not too interested in the internals of DOS or 80186 programming, INT 21h (which simply means operating system call number &21) can be regarded as a general purpose OS call combining most of the facilities provided in the BBC micro by OSBYTE, OSWORD and the filing system calls. Registers (and sometimes parameter blocks) are set up and the call is made with a number to identify it.

Other interrupts exist for more specialised purposes, but INT 21h provides most of the facilities users require. It is used for console line or character input/output, disc (directory, sector, file and record) operations, program and memory management as well as for miscellaneous tasks like reading or setting the system's clock or date (it even includes functions for intercepting and re-directing the interrupt vectors!).

In order to end execution, a program can make a call to one of three other interrupts, INT 20h, 27h or 21h using either function 0 or function 4Ch.

INT 21h function 0 is of no use if we want to return a code (oddly enough called the return code) to indicate success or failure, because it doesn't provide this capability. Interrupt 20h is also a program termination call, but its use is not recommended either. In fact these two (and INT 27h) are all left-overs from DOS version 1, and are retained purely for compatibility with early software. For all current purposes these calls are obsolete and shouldn't be used.

This leaves INT 21h function 4Ch, which has been the recommended method of final program termination since DOS version 2 appeared, and it still is. Here's a short section of source code which will show how it's called in a program. Several other 'bits' would be needed for assembly, but this extract is enough to illustrate the point we're concerned with.

```
; Constant declarations

G_F_D       equ 021h    ; INT 21h
TERMINATE   equ 04Ch    ; with return code

; Program code
.....                   ; This is the main
.....                   ; body of the program
.....

exit:                   ; Common exit point
mov  al, [Return_Code]B ; The final result
mov  ah, TERMINATE      ; Set up terminate...
int  G_F_D              ; and do it!

; Variable declarations

Return_Code db 0        ; Default zero
```

The constant declarations at the top simply assign hex values 21 and 4C to the two names given. Between these and the label 'exit:' the main functions of the program would be coded, including suitable displays to advise about progress during execution.

In this example, I've assumed that whatever happens in the main code, the program always jumps to 'exit:' to terminate. The variable declared as 'Return_Code' at the bottom of the source has a default value of zero, which means successful execution. This is pre-set when the code is assembled, but if during execution the program detected an error the value would be changed.

When execution arrives at 'exit:' the value now in 'Return_Code' is moved into byte register AL, which is where INT 21h function 4Ch expects the return code to be placed. The next line moves the value 4Ch into byte register AH to identify the interrupt function required, and the last line, 'int G_F_D', calls interrupt 21h, which actually ends the program.

Within this call all file buffers are flushed (i.e. pending output records are written to disc), all files are closed, and the program's memory is freed for re-use. A few other things happen, which needn't concern us here, except to say that control then passes back to the original caller. This might be another program, a batch file or you, by means of the DOS command line.

If the caller was a program, the return code can be obtained by means of INT 21h function 4Dh, which will pass the return code back to the calling program in register AL. The program can then decide what action to take depending on the value returned.

If the caller was a batch file the return code is made available via the pseudo-variable 'ERRORLEVEL'. Again, by a suitable line in the batch file, subsequent execution can be controlled depending on the value returned (see below).

If you call a program manually from the command line, the return code should be irrelevant, because the program should have advised you of success or otherwise by a simple confirmation or an error message.

Interestingly, however, you can still enter an 'IF ERRORLEVEL' statement manually and it will work normally (with, of course, the exception of jumping to a label - I hope I needn't explain that!). The format of the command, either manually or in a batch file is:

```
    IF ERRORLEVEL <value> <command>
```

where <value> is a number between 0 and 255 (although higher values can be entered), and <command> is any DOS command except one which is itself conditioned by another 'IF'.

Note that there's no '=' sign. If you include one, you'll get an error. The most useful <command> of course is 'GOTO label', which allows you to change the execution route. A label can be any string ending with a colon, but only the first 8 characters are significant. For example:

```
    LABEL1:
    LABEL2:
```

and so on would be acceptable, but:

```
    LABELNUMBER1:
    LABELNUMBER2:
```

would be treated as duplicate labels.

IMPORTANT POINTS

The reason that this pseudo-variable is called 'ERRORLEVEL', rather than 'ERRORCODE' or something else is most important. You must remember that 'IF ERRORLEVEL' performs a 'greater than' test, not an equality test.
In other words:

```
    IF ERRORLEVEL 4 <command>
```

actually means, "If the return code is equal to OR GREATER than 4, execute the command".

This means that when you test for each of several results, you MUST test for the higher values first. Put another way, a return code of 255 (the standard 'general failure' code) always produces a 'TRUE' result no matter what value you test it with up to 255.

A side-effect of this is that if <value> in the statement is zero, i.e.:

```
    IF ERRORLEVEL 0 <command>
```

<command> is ALWAYS executed. A value of zero in this statement is therefore utterly meaningless, since it always gives 'TRUE' for any value, including zero. The easiest way to remember this is that you can't directly test for success using 'ERRORLEVEL', only for failure.

PROBLEM SOLVER APPLICATIONS LIST

This list is compiled from information supplied by Shibumi Soft and 512 Forum readers. Where version numbers were available they are provided, but where omitted users should exercise care.

Title	Publisher
688 Attack Sub	Electric Arts
Adventure Writer	P.D.
Alley Cat	Syn Soft
Ancient Art of War	
As Easy As 3.0	Trius
Autoroute	NextBase
Brief	Underware Inc
Bushido	
Cashbook	Freeway Ltd.
Charlie Chaplin US Gold	
Chiwriter	Horstman
Cyrus Chess	
Dancad 3D	
Dark Side	
DBase III+	Ashton Tate (1.2)
Deluxe Paint 2	
Digger	Windmill Soft
Digita Diary	
Dream Hiuse	Computer Easy
Driller	
Droege	P.D.
Easy Boot	
Elite	Firebird
Falcon	Spectrum
Fast Graph	
First Publisher	
Flight Simulator 2.13	Microsoft
Flight Simulator 3	Microsoft
Flodraw	Dabs S.W.
Flowcharting	Patton & Patton
Fontasy	Prosoft
Formtool	Bloc Developments
Formwork	Analytex International
Framework II	Ashton Tate
Freefile	S.W.
Freelance	Graphic Communications
French Teacher 1/2	Micro Tutor P.
Frogger	
FSD	IBM
Galaxy 2.3	Ominiverse
Game of Life	Scientific G.
German Teacher 1/2	Micro Tutor P.
Graphing Assists	IBM
Grime	S.W.
Homebase	Brown Bag Software
How's Your Heart	
Impact	
Infidel	Infocom
Infiltrator	
ISS Calendar Plus 2.2	
Jet	
Leisure Suit Larry	
Life Forms	S.W.
Masterfile PC	
Mandelbrot Generator	P.D.
Microsoft Chart Microsoft	
Mindreader 2.0	Brown Bag Software
Mini Office Pers.	
Mix C Compiler	Analytical Engineering
Mix C Editor/Trace	Analytical Engineering
Newsmaster 2	
Newword	Newstar
Osbit	
Paperbase Deluxe	
Payroll	Micro-Aid

Title	Publisher
Professional File	
Professional Plan	
Pango	Sheng-Cheung L.
PC Draw	Micrografix
PC Calc	Buttonware
PC Man	Orion Software
PC Outline	
PC Storyboard	IBM
PC Tools Deluxe 4.11	Central P. Soft
PC Tutor	
PC File+	P.D.
PC File 3	P.D.
Pipedream	Colton
Pitstop	Epyx
Planning Assistant	IBM
Powermenu	S.W.
Printshop	Pixellite
Print Master	
Process Engineering Package	
Professional Write	
Prolog2	I.B.M.
Prospero Pascal	
PSI Trader	
Quadralien	Logotron
Reflex	Borland
Savoir	Intelligent Software
Script	IBM
Sorceror	Infoçom
Space Commanders	Columbia
Spell	Microsoft
Starglider	Firebird
Starquake	Mandarin
Strip Poker	Artwork
Strip Poker	Electric Arts
Symphony 1.2	Lotus
Tas+ Database	Megatech
Teed-Off 3.0	S.W.
Tennis	
Test Drive	Mastertronic
Topcopy Plus	Innova Soft
Turbo Basic	Borland
Turbo C 1.5	Borland
Turbo Calc	Borland
Turbo Debug	Borland
Turbo Pascal 3.0	Borland
Turbo Pascal 4.0	Borland
Turbo Prolog	Borland
VP Graphics	Paperback Soft
War	Broderbund
Wizz Ball	Ocean
Where in the World is Carmen Sandiego	
Word	Microsoft
Wordperfect 4.2	Wordperfect Corp.
World Class Golf	
World Class Leaderboard	Access/US Gold
World Tour Golf	Electric Arts
Yes Chancellor	

P.D. = Public domain
S.W. = Shareware

A BEEBUG Graph Plotter (continued from page 15)

```
 1820  ENDPROC
 1830  DEFPROCsolve(X,Y,Z)
 1840  I=X-Y-Y+Z:IFABS(I)<1E-6THEN1880
 1850  G=(Z-X)/I-z-z:Y=2*Y/I-z*(G+z):X=G
/2:Z=X*X-Y:IFZ<-1E-4c=0:ENDPROC
 1860  IFZ<1E-4c=1:Z=0ELSEc=2
 1870  v=s*(-X+SQR(Z)):V=-G*s-v:ENDPROC
 1880  IFY<>X:c=1:v=s*(z-Y/(Y-X)):V=1E9:
ELSEc=0:IFY:ELSEl=-1:IFF=2MOVEa*f,W:DRAW
A*f,W:ELSEMOVEW,B*g:DRAWW,b*g
 1890  ENDPROC
 1900  DEFPROCdraw(X,Y,x,y)
 1910  IFABS(X-x)+ABS(Y-y)<&FFMOVEX,Y:DR
AWx,y:ELSEc=c-1
 1920  ENDPROC
 2000  DEFPROCharder
 2010  a$=FNswap($I%,M$,"X")
 2020  PROCconvert:PROCtype(1)
 2030  t=INSTR("RX",CHR$?I%):PROCrange
 2040  W=u-w:C=2050:IFt=2:$K%=$J%:$J%=$(
J%+INSTR($J%,"="))
 2050  N=4:IFW*w>=U*w:C=D:ENDPROC
 2060  FORW=W+w TOU STEPw:X=W:aa=W
 2070  IFR%=0ANDt=1aa=RADW
 2080  Z=EVAL$J%
 2090  IFt=1I=Z*f*COSaa:G=Z*g*SINaa:ELSE
I=f*EVAL$K%:G=g*Z
 2100  IFN=5:IFABS(I-x)+ABS(G-y)>300N=4
 2110  PLOTN,I,G:x=I:y=G:N=5
 2120  NEXT:ENDPROC
 2130  DEFPROCrange
 2140  PRINTTAB(10)M$"1="SPC(6)M$"2=";
 2150  IFF%PRINTSPC(6)"dK=";
 2160  PRINTTAB(13,1);:PROCinput(5)
 2170  PROCconvert:u=EVALa$
 2180  PRINTTAB(22,1);:PROCinput(5)
 2190  PROCconvert:U=EVALa$
 2200  IFF%PRINTTAB(31,1);:PROCinput(8):
PROCconvert:w=VALa$:ELSEw=(U-u)/200:IFt<
>2w=SGNw/(1+14*R%)
 2210  VDU13:ENDPROC
```

Writing a Compiler (Part 1)

David Spencer shows how you can write a simple compiler.

The idea of a compiler is easy - a utility that will take a program in some language and convert it to another language. Normally, the source program will be in a high level language such as Basic, C or Pascal, and the compiler will output a machine code program that is equivalent to this. However, in practice compilers tend to be thought of as black boxes which other programmers write, and I'm sure that not many people would be happy if faced with the task of writing one. The main reason for this is that if you adopt the 'try it and see' approach for compiler writing, you are likely to spend a very long time producing something which only partly works - a lesson learnt by many early compiler writers. Instead, compiler writing is a prime example of a case where a mathematical approach to program design is needed. These days, programmers can, with the help of various software tools, write bug-free compilers for any language in a very short time.

What we shall do in this series of Workshops is to explain some of the theory of compilers, and use this knowledge to write a simple compiler which will take a basic numeric expression and generate a 6502 machine code program to evaluate it. This may seem a long way from a compiler for a real language, but the principles are the same, and it must be remembered that expression evaluation is a major part of many languages. To make our compiler as easy to understand as possible, it will be written entirely in Basic.

ANATOMY OF A COMPILER

We will start with a look at the traditional parts that make up a compiler, as shown in figure 1. Each part is referred to as a *phase* because conceptually the source program can be thought of as passing through each section, with a final compiled version emerging at the end.

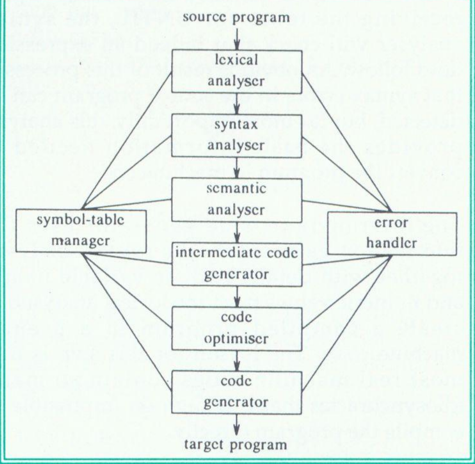


Figure 1. The phases of a compiler

The function of each phase will now be described briefly.

The lexical analyser takes the program being compiled character-by-character, and converts it into a series of more manageable units called *tokens*. For example, suppose you were hand-compiling a Basic program and you come across a PRINT statement. When working out how to compile the statement it is more important to know that the keyword PRINT has been found than the letter P followed by R etc.

Therefore, the lexical analyser would replace the letters P-R-I-N-T by a single number indicating the keyword PRINT. This is very similar to the tokenisation performed by the Basic interpreter when a program is typed in. Similarly, if you come across a number in the program then initially it is only necessary to know that it is a number - not its actual value. Therefore the lexical analyser will replace the number by the token that represents a number, and store the actual value somewhere for later. A similar argument applies to variable names.

The syntax analyser looks at the stream of tokens coming from the lexical analyser and attempts to match them against the various constructs that make up the language. For example, an UNTIL statement in Basic must be followed by an expression and then a statement terminator (end of line or colon). Upon receiving the token for UNTIL, the syntax analyser will check that indeed an expression does follow. An obvious result of this process is that syntax errors in the source program can be detected, but far more importantly, this analysis provides the basic information needed to convert the program to machine code.

The intermediate code generator uses the information gathered by the syntax analyser, together with details such as variable names and numeric values from the lexical analyser, to create a compiled program in a pseudo machine code. The reason for this step is that most real machine codes contain so many idiosynchracies that it is nigh on impossible to compile the program directly.

The optimisation phase examines the intermediate code and attempts to optimise it, either by shortening it or speeding it up. Finally, the code generator takes the optimised intermediate code and converts it to the appropriate machine code for the target computer.

Two areas which apply to the whole compiler are the symbol manager which keeps track of the numeric values and variable names put aside by the lexical analyser, and the error handler which deals with any errors detected, and tidies up as much as possible.

In practice, the distinction between phases is not as concrete as figure 1 suggests. For example, the lexical analyser will nearly always be a subroutine which is called by the syntax checker when an input symbol is needed. The optimisation phase may be missed out completely, and it is very unlikely that the intermediate code will be a palatable language - more likely it will be some efficient internal representation.

DOWN TO WORK

Having explained some of the fundamentals of compilers, we can start to write ours. As we proceed new techniques and methods will be introduced as necessary. We will not follow the structure of figure 1 exactly, but rather simplify it. The optimisation phase will be dropped completely, and the intermediate code generator and the code generator proper will be merged into a single step.

As a starting point we will develop the lexical analyser for our compiler. As outlined above, this will be in the form of a routine called by the syntax analyser. The routine has to return at least one value, namely the next input token, and hence will be written as a Basic function. The input symbols we have to recognise are the numeric operators +,-,* and /, the unary negation operator -, any valid integer, and the (and) symbols for changing precedence. It should be noted that from the point of view of the syntax analyser, one number is much the same as another - all that is important is that it is a number, and not say a '+' operator. Therefore, the lexical analyser will return a single token to represent any number. However, the compiler will need to know the actual value of the number at the code generation stage, so the lexical analyser must make a note of the value. This will be done by storing the value in a global variable.

There are many complex algorithms and techniques for the semi-automatic generation of lexical analysers. For example, UNIX-based computers have a utility called 'Lex' which takes the definitions of the keywords, and other tokens making up the language, using so-called *regular expressions* as a shorthand notation, and generates the lexical analyser in the form of a C function. However, for our simple lexical analyser we can get away with a cut and thrust method. Before doing this, we need to make a couple of decisions.

Firstly, what effect does a *white space* character have on the lexical analyser. (*White space* is the term used to describe any characters which are essentially formatting controls, such as space itself and the tab character). Our lexical analyser will totally ignore white space, except in the case of a number, where a white space character will mark the end of the number. Secondly, what do we do with minus and unary negation. Both operators are traditionally represented by the '-' symbol, but are in fact totally different operators. We can resolve this problem by using the juxtaposition of the '-'. If the '-' follows a number then it must be a subtraction operator, while if it comes after another operator, or at the start of an expression, then it represents the negation operator.

The final decision is what values should be returned by the lexical analyser function for each input token. The chosen values are shown in table 1.

TOKEN		VALUE
'+'		43
'-'	(subtraction)	45
'*'		42
'/'		47
'-'	(negation)	95
'('		40
')'		41
number		1

Table 1. Lexical analyser return codes

The values for the arithmetic operators are simply the ASCII codes of the characters representing them, while the value for negation is the ASCII code for an underscore character. The value used to represent a number is chosen arbitrarily, the only criterion being that it is different to the values used for any other tokens. A special token with the value 0 is used to indicate the end of the input string.

Listing 1 shows the function implementing the lexical analyser. It reads characters from the global variable *input$* which contains the rest of the input line (in other words starting with the next character to be looked at), and returns the value of the token found. *input$* has to be global in BBC Basic to allow it to be changed by the function. The other global variable used is *minusflag,* which is initially set to FALSE to indicate that a '-' will represent negation, and is then changed as necessary by the function. In the case of a number, its value is stored in the variable *value.* An extra piece of information attached to a token like this is called an attribute of the token. It should be easy to see how the function in listing 1 works.

Listing 1

```
 1000 DEF FNlex
 1010 REPEAT
 1020 IF LEFT$(input$,1)=" " THEN input$
=MID$(input$,2)
 1030 UNTIL LEFT$(input$,1)<>" "
 1040 IFinput$="" THEN =0
 1050 A$=LEFT$(input$,1):IF NOT minusfla
g AND A$="-" THEN token=ASC"_":input$=MI
D$(input$,2):=token
 1060 IF INSTR("+-*/",A$) THEN token=ASC
 input$:input$=MID$(input$,2):minusflag=
FALSE:=token
 1070 IF INSTR("()",A$) THEN token=ASC i
nput$:input$=MID$(input$,2):=token
 1080 IF INSTR("0123456789",A$)=0 THEN P
RINT "Mistake":END
 1090 value=VAL(input$)
 1100 REPEAT
 1110 IF INSTR("0123456789",LEFT$(input$
,1)) THEN input$=MID$(input$,2)
 1120 UNTIL INSTR("0123456789",LEFT$(inp
ut$,1))=0 OR input$=""
 1130 minusflag=TRUE
 1140 =1
```

One important feature implemented by listing 1, but not mentioned yet, is error handling. If the next character read from *input$* doesn't match one of the operator symbols, and isn't a digit, then a *lexical error* has been detected. This indicates that a meaningless input expression

Continued on page 58

BEEBUG Bingo

Al Harwood describes a system for printing bingo cards, and calling the numbers, literally if you also have Superior's Speech system installed.

There you are on that Sunday afternoon, just finished the washing up from the Sunday roast, nothing much on the tele, what to do? Fear no more, here is the very answer in the form of BEEBUG Bingo, no not the type held around the corner every Saturday night, but a fully computerised version.

Following are two listings which can either be used separately, together or with Superior Software's Speech system.

The first listing is a bingo card printer, it prints pages of bingo cards, each page contains ten cards. The numbers on each card are randomly allocated by the program. On running, you are asked how many pages you want to print, at this point set up your printer and enter the number of pages you require. Once printed, the cards should be cut out and kept.

The second program is a bingo number caller. Once run, just press any key for it to start calling numbers. As each number is chosen it is displayed and is also marked off on the master board, which covers most of the screen. So by looking at the master board you can tell which numbers have been previously called.

Pressing the space bar at any time will suspend play until the space bar is pressed again. Pressing Escape will abandon the current game and start a new one, while pressing Shift-Escape will exit from the program altogether.

This program can also be used in conjunction with Superior Software's Speech system. Just *RUN the main Speech program first. Our bingo program will recognise it, and as each number is chosen it will literally be called out.

Beebug Bingo can be played by any number of people, but each player should have the same number of cards. As the numbers called out match those on your bingo cards cross them off and the first person to cross all their numbers off wins.

Note: Lines 1120 and 1150 in the print program switch underline mode on and off, and assume an Epson compatible printer.

```
   10 REM Program BINGO CARD PRINTER
   20 REM Version B1.0
   30 REM Author  Al Harwood
   40 REM BEEBUG  November 1989
   50 REM Program subject to copyright
   60 :
  100 MODE3:ON ERROR GOTO 150
  110 DIM C(8,2),N(89)
  120 PROCprinter
  130 END
  140 :
  150 MODE7
  160 IF ERR<>17 REPORT:PRINT" at line "
;ERL
  170 END
  180 :
 1000 DEFPROCcard
 1010 FOR R=0 TO 2:FOR C=0 TO 8:C(C,R)=0
:NEXT,
 1020 FOR N=0 TO 89:N(N)=1:NEXT
 1030 FOR R=0 TO 2:FOR C=1 TO 5
 1040 C0=RND(9)-1:IF C(C0,R)=-1 GOTO1040
 1050 C(C0,R)=-1:NEXT,
 1060 FOR C=0 TO 8:FOR R=0 TO 2
 1070 IF C=0ANDC(0,R)=-1 REPEAT N=RND(9)
:UNTIL N(N-1):N(N-1)=0:GOTO1100
 1080 IF C=8ANDC(8,R)=-1 REPEAT N=RND(11
)+79:UNTIL N(N-1):N(N-1)=0:GOTO1100
 1090 IF C(C,R)=-1 REPEATN=RND(10)-1+C*1
0:UNTIL N(N-1):N(N-1)=0
 1100 IF C(C,R)=-1 C(C,R)=N
 1110 NEXT,
 1120 VDU6,1,27,1,45,1,1,21
 1130 PRINT SPC46;:FOR R=0 TO 2:PRINT'"|
";:FOR C=0 TO 8:IF C(C,R)=0 PRINT SPC4"|
"; ELSE PRINT SPC(3-LEN(STR$(C(C,R))));C
(C,R);SPC1"|";
 1140 NEXT,:PRINT'"|"SPC16"BEEBUG BINGO"
SPC16"|"
```

```
 1150 VDU6,1,27,1,45,1,0,21
 1160 ENDPROC
 1170 :
 1180 DEF PROCpage
 1190 VDU2,21
 1200 PRINT'"BEEBUG BINGO CARD PRINTER,
By Al Harwood"''
 1210 FOR A%=0 TO 9:PROCcard:PRINT:NEXT:
PRINT'
 1220 VDU6,3
 1230 ENDPROC
 1240 :
 1250 DEF PROCprinter
 1260 VDU19,1,2;0;
 1270 CLS:PRINTTAB(28,3)"BEEBUG BINGO CA
RD PRINTER"TAB(34,5)"By Al Harwood"TAB(0
,10)"* Each page consists of ten Beebug
bingo cards"
 1280 INPUTTAB(0,12)"Enter number of pag
es to be printed: "P
 1290 PRINTTAB(0,14)"Are you sure (y/n)?
";:REPEAT A$=GET$:UNTIL INSTR("YyNn",A$)
:IF INSTR("Nn",A$) GOTO1270
 1300 PRINTTAB(0,16)"Pages still to prin
t: ";
 1310 FOR pages=P TO 1 STEP -1:PRINT pag
es;CHR$8;:PROCpage:NEXT
 1320 PRINTTAB(0,18)"Bye."'':ENDPROC
```

```
   10 REM Program BINGO CALLER
   20 REM VERSION B1.1
   30 REM AUTHOR  Al Harwood
   40 REM BEEBUG  November 1989
   50 REM Program subject to copyright
   60 :
  100 *FX229,1
  110 MODE7:ON ERROR GOTO 190
  120 VDU23,1,0;0;0;0;
  130 DIM N(89),cli 255
  140 REPEAT
  150 PROCscreen
  160 PROCplay
  170 UNTIL INKEY-1
  180 :
  190 MODE7
  200 REPORT:PRINT" at line ";ERL
  210 END
  220 :
 1000 DEF PROCscreen
 1010 CLS
 1020 PRINT'CHR$141CHR$130"BEEBUG BINGO"
SPC11"by Al Harwood"'CHR$141CHR$130"BEEB
UG BINGO"SPC11"by Al Harwood"'CHR$145STR
ING$(38,"£");
 1030 FOR A=0 TO 8:PRINT:FOR B=1 TO 10:P
RINTCHR$131SPC(2-LENSTR$(A*10+B));A*10+B
;SPC1;:NEXT,:PRINTCHR$145STRING$(38,"p")
 1040 ENDPROC
 1050 :
 1060 DEF PROCplay
 1070 FOR N=0 TO 89:N(N)=-1:NEXT
 1080 PRINTCHR$134"Press any key to star
t game. Pressing"'CHR$134"Escape will re
start, (use for a false"'CHR$134"start o
r when someone has won).";
 1090 IF GET PRINTTAB(0,22)SPC119;
 1100 REPEAT
 1110 REPEATN=RND(90):UNTIL N(N-1):N(N-1
)=0
 1120 IF LEN(STR$(N))=1 H1=1 ELSE H1=0
 1130 IF N MOD10=0 V1=-1 ELSE V1=0
 1140 H=N MOD10:IF H=0 H=10
 1150 H=(H-1)*4:V=(N DIV10+V1)*2+4
 1160 PRINTTAB(H,V)CHR$129TAB(17+H1,23)C
HR$141CHR$134;N;TAB(17+H1,24)CHR$141CHR$
134;N;
 1170 PROCsay(N)
 1180 key=INKEY(300):IF key=27 F=1:GOTO1
220
 1190 IF key=32 REPEAT:key=GET:UNTIL key
=32
 1200 F=1:FOR N=0 TO 89:IF N(N) F=0
 1210 NEXT
 1220 UNTIL F
 1230 ENDPROC
 1240 :
 1250 DEF PROCsay(N)
 1260 IF?&6A53<>ASC"s"OR?&6A54<>ASC"a"OR
?&6A55<>ASC"y"ENDPROC
 1270 D0=N DIV10:D1=N MOD10
 1280 IF D0=0S$=STR$N:GOTO1320
 1290 IF D0=1 RESTORE1340:FOR A=0 TO D1:
READS$:NEXT:GOTO1320
 1300 RESTORE1350:FOR A=2 TO D0:READS$:N
EXT
 1310 IF D1 S$=S$+STR$D1
 1320 $cli="*say"+S$:X%=cli MOD256:Y%=cl
i DIV256:CALL&FFF7
 1330 ENDPROC
 1340 DATA ten,eleven,twelve,thirteen,fo
urteen,fifteen,sixteen,sevnteen,ateen,ni
neteen
 1350 DATA twenty,thirty,fourty,fifty,si
xty,sevnty,aty,ninety
```

Special Offer must end December 31st 1989
Upgrade to Wordwise Plus 2
for only £25.95 incl. VAT & UK postage.

Until the end of December 1989, Wordwise Plus owners can upgrade to Wordwise Plus 2 (WW+2) for only £25.95 fully inclusive. (Normally £38.95)

WW+2 is not just another Wordwise add-on. It is a complete word-processing system which offers all the features of Wordwise Plus, and much more besides.

Most Wordwise support packages have the disadvantage that they occupy an extra ROM socket, and are often very slow. **But WW+2 replaces your existing word-processor chip.** And whereas Wordaid (Watford Electronics) takes 45 minutes to sort 200 addresses, WW+2 takes 2 seconds!

New facilities include;

28 day money-back guarantee

★ Supplied on a single 32K chip, compatible with all BBC micros (not A3000)
★ Upgrades from Wordwise also available
★ Menu-driven file access (works with DFS and ADFS)
★ Mail-merge and label printing capabilities
★ Runs all WW+ segment programs, but faster. (Programs may even be tokenised for extra speed and memory savings)
★ Fixes "String bug" in Wordwise Plus.
★ Very fast and versatile data sorting
★ Compatible with existing text files, segment programs and Spell Master
★ Over 30 new keywords in programming language including eg MID$, LEFT$
★ Over 12 new CTRL keys, such as;

CTRL/H	Help option lists all CTRL keys
CTRL/X	Search and replace (works in the segments too)
CTRL/G	Fast and convenient string search
CTRL/P	Mark the current paragraph
CTRL/Z	Mark the word at the cursor
CTRL/L	Delete the current line
CTRL/T	Select the text area
CTRL/fn	Select any segment

The WW+2 package

As an upgrade from Wordwise Plus, you get the new WW+2 ROM, two discs and 112 page manual.

Our 28 day money back guarantee applies to all Wordwise Plus 2 purchases.

The following extracts from reviews and letters concerning WW+2 are all on file and open to inspection.

"So good that I would recommend all WW+ owners to upgrade". "I have to compliment you on the standard of this upgrade". "Delighted with all of the extra commands."

"A no-quibble money back guarantee. What have you got to lose by checking it out?" (Micronet)

You can order WW+2 by telephone or in writing. Official orders welcome. The two utility discs are normally 80 track format. So when you order, please state whether 40 track discs are needed.

Prices (including VAT)

Upgrade from Wordwise Plus	£24.95
Upgrade from Wordwise	£47.15
Complete WW+2 package	£56.35
Spell Master	£51.00
16K RAM module	£16.50

Please allow £1 postage (UK)

IFEL, 36 Upland Drive, Derriford, Plymouth PL6 6BD (0752) 847286

POSTBAG

PLUGGING THE GAP IN TELETEXT MODE

I found your articles on using teletext mode (First Course, Vol.7 No.10 to Vol.8 No.4) very helpful. However, you are wrong to say that you cannot have double height text in a frame without a hole appearing in the border. I had this difficulty at one time, but the solution I discovered is very simple - put the CHR$141 outside the frame. You can draw the frame before or after printing any message, but be careful not to overwrite any necessary control characters.

John Waddell

It is good to have a solution to this problem at long last. It just shows that even now you can still learn something new about the BBC micro.

FAIR SHARES FOR ALL

One thing I cannot understand with BEEBUG is how very clever additions to published programs are merely printed in *Postbag* (for example, *Brightening up the Landscape* and *Indexing Watford Double Length Catalogues* in Vol.8 No.4) while others are published under *Hints & Tips* for which the contributor is then paid £5. While this is of little consequence, I feel it is unfair.

Might I suggest that *Postbag* be restricted to questions requiring replies or general correspondence, and additions to programs and useful hints be located where they should be, and at least give some token reward, however minimal.

D.P.Dyer

Mr.Dyer has highlighted an anomaly which has arisen quite accidentally, and which, until now, had been overlooked. I propose, in future, to pay £5 for all hints and letters published, with £15 for any outstanding contribution on either page. So let's be hearing more from you, views, comments or information as you wish.

Normally, technical information on the BBC micro and third party software and hardware appears under the Hints & Tips banner; general comment on the Acorn world, BEEBUG magazine, and articles and programs published in the magazine, is put under the Postbag heading; while any errors that come to light in published programs are detailed in Points Arising. This is the system to which we have been working for many years now.

READERS REQUESTS

I have been a subscriber since issue number one, but I must confess that I have recently been wondering if the magazine is of any further use to me. If I could learn just one thing from each issue, it would be worthwhile.

Here are some suggestions for articles which would make the magazine more valuable to me:

1. Sideways RAM - more information on its use and relevance to Basic.
2. Keeping large arrays in Sideways RAM.
3. Information on the relative merits of other high level languages, Lisp, Pascal, Forth or whatever.
4. OSBYTE calls - a complete mystery.
5. Saving space on a model B.
6. A simple explanation of how the Beeb works.
7. An article on what is involved in upgrading from a BBC micro to an Archimedes.

D.A.G.King

Specific requests from readers for future articles or programs are always most useful. In many cases we can respond positively (see this month's First Course article on OSBYTE and similar calls). If other readers have particular topics they would like to see covered (or maybe support or disagree with those suggested by Mr.Dyer), or would like to offer articles on these subjects for publication, then why not write to us?

HELPING ONESELF

I have just typed in the Self-help utility from Vol.8 No.4. I find it excellent. It makes full use of sideways RAM which I use, and I have combined it with some of the previous sideways RAM utilities, some of which I believe were also written by the same author. Now I do not have most of my sideways RAM unused, and I do not have to stop what I am doing and load another disc just to look up a telephone number for example. A star command will do it all for me.

I was thinking of stopping my subscription to BEEBUG, as my Beeb is old and I thought forgotten now that the Master, Archimedes and RISC have taken over. If you can continue to support us BBC users with such good software, then I will most certainly continue to support your magazine. Please pass on my thanks to Mr.Hill for his imaginative and useful program.

Very few of the programs and articles we publish are specific to the Master series, and BEEBUG will most certainly continue to support all users with the best that we can find and commission for the magazine as long as there is demand for this.

HINTS and tips

A miscellany of hints compiled by Mike Williams.

DIRECTING PRINT ON THE MASTER

Andrew Rowland

The Master 128 allows you to change the direction in which characters are printed on the screen. Normally, text runs from left to right (the X direction), and when it spills over the edge of the screen, it moves down to the next line (the Y direction). However, if you want to label an axis of a graph with the text running down the screen, it is not necessary to mess around printing a character at a time or to define a small text window: you can set the computer to print downwards automatically. If you want to use your word processor for a language which writes from right to left, this too can be achieved (on screen, anyway).

All you do is enter VDU 23,16,n| where n is selected as appropriate from table 1. For example, VDU 23,16,2| will move the home position (the place PRINT TAB(0,0); takes the cursor) to the top right of the screen and the text is written from right to left. VDU 23,16,8| displays text down the screen Chinese-style. Of course, it will not actually twist each letter - each one is still displayed vertically. Add 1 to n if you want scroll protect (like the *CONFIGURE NoScroll option).

Direction			
X	Y	n	Home
right	down	0	top left
left	down	2	top right
right	up	4	bottom left
left	up	6	bottom right
down	right	8	top left
down	left	10	top right
up	right	12	bottom left
up	left	14	bottom right

Table 1. The setting remains in force when you change mode, but is reset by pressing Break.

ZERO PAGE WORKSPACE

Andrew Benham

Machine code routines often need some zero page workspace. Contrary to popular belief, locations &70 to &8F are NOT reserved for the user. Locations &00 to &8F are allocated to the current language: Basic allows the user &70 to & 8F, but other languages may not (e.g. View uses all locations from &70 to &8F, and can crash if these are tampered with). If zero page locations are needed, the original contents should be restored afterwards (the exceptions being &A8 to &AF which are allocated as star command workspace - for both ROMs and library routines stored on the current filing system).

PAGING WITHOUT PAGE NUMBERS IN WW+

Jack Fish

Several segment programs are available which enable multiple-copy printing of documents in Wordwise Plus. When it is a single page one, it is often a nuisance to have "Page 1" printed at the foot. To enable page printing without page numbers, the following commands are needed:

```
    <green> EP <white>
    <green> DF <green> CE <white> <space>
<space> <Return>
```

To mark the spaces use the default hard space character '|' (Shift-\ next to the cursor keys). Finish the document with:

```
    <green> BP
```

and nothing further at all.

GHOSTING LETTERS

Andrew Rowland

There is a quick way to 'ghost' out characters in two colour modes, which can be useful in pull-down menus to indicate that an item is not currently available. Simply use ?&D2=&AA:?&D3=&AA and then print normally. This causes alternate columns of each character to be masked out, leaving dotty, but legible text. To return to normal printing, use ?&D2=0:?&D3=0. You can also get reverse video with ?&D3=&FF, all this without redefining characters and no loss of speed! If Tube compatibility is required, use OSWORD 6 to write to I/O memory.

DOUBLE BLUR

P.Mudham

A quick way to achieve 'double height' text in mode 1 on a BBC micro is to use the following line:

```
    VDU 240,23,0,0,255,0,0,0,0,0,0
```

This re-programs the 6845 display registers to produce the double-height (if blurred) effect.

has been found, and the function picks this out and reports the error as a 'Mistake'.

THE SYNTAX ANALYSER

Having designed and implemented the lexical analyser, we can turn our attention to the syntax analyser. As mentioned above, the purpose of this phase is to check that a sequence of input tokens forms a valid element of the language in question, and determine which element.

As an example, a FOR statement in Basic (without the optional STEP) has the following syntax:

```
        FOR <var> = <expr> TO <expr>
```

The symbols FOR, =, and TO specify literal tokens which must appear in the input string. These are called *terminals*. On the other hand, <var> and <expr> represent a variable name and an expression respectively. Exactly what series of input characters constitutes a variable name or an expression will be defined elsewhere in the compiler. These symbols are called *non-terminals*. The syntax analyser knows that for a FOR statement to exist it must be made up exactly as above. For example, the FOR must be followed by a variable name which must be followed by an equals sign, and so on. If the input string doesn't exactly match the template, then the input doesn't represent a FOR statement and the syntax analyser must check further to see if it represents another structure within the language.

That is all there is room for this time. In next month's Workshop we will introduce the concept of Grammars - a very powerful method for defining exactly the structure of a language, such as what constitutes a FOR statement or an expression. We will use this information to build a working program.

FINDING OUT MORE

There is no way that a short series of articles can explain the theory of compiler writing in any depth. If you want to know more, then the standard text on compilers is a book called *Compilers - Principles, Techniques and Tools,* by Aho, Sethi and Ullman (Addison Wesley £19.95). This is the second edition of a book colloquially known as the dragon book because of its cover illustration. The book is fairly heavy going, but doesn't assume any prior knowledge of the subject, although an understanding of C is helpful in order to follow the examples.

First Course - Using Operating System Routines (continued from page 36)

USR	XX	XX	XX	XX
A	00	00	00	FF
X	00	00	FF	00
Y	00	FF	00	00
P	FF	00	00	00

Figure 1. Masks for extracting bytes from value returned by USR.

For reference, the contents of the four registers can be extracted as follows:

```
A%=USR(address) AND &FF
X%=(USR(address) AND &FF00) DIV &100
Y%=(USR(address) AND &FF0000 DIV &10000
P%=(USR(address) AND &FF000000 DIV &1000000
```

where *address* is the address of the routine being called. If you still feel somewhat unsure about this process, then just treat the lines above as a set of formulae which will give the right result when needed.

That's all I have space for this month. I suggest you peruse further information on OSBYTE and other machine code calls given in your manual, and experiment to see what else you can achieve. Next month I will investigate more of the OS routines available to the Basic programmer.

The idea for this First Course article came from a BEEBUG reader. If you have any topics which you would like to see covered in this series then please write in and let me know.

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 15th of each month.

Delta/Card Index ROM £30, Delta Reference Guide £8, Delta Gamma ROM £20, Delta Mailshot £8, Delta Reporter £8, Delta Inter Link £8, (all with manuals) Edword Superpack £20, 'Double View' BBC/B 2x ROM £18, Mini Office II 80T + Dabs mini driver + Dabs Guide £15, Multi Font NLQ £8. All prices include P&P. Tel. 01-399 2865.

BBC B 1.2 issue 7 with 40/80T DS Cumana drive, GXR ROM, BBC Basic editor ROM, Solidisk 32 sideways/shadow board, Dumpout3 ROM, Wordwise ROM, 50+ discs and box, books, mags, Mini Office II etc. £300. Tel. (0602) 215126.

BBC Master 128 and dual 800k 40/80T DD in plinth £500. Acorn teletext adaptor with ATS, free with computer and disc drives. Microvitec 1541 hi/res colour monitor, all leads £200. Manuals; Reference guides I & II, The New Advanced User Guide, View, View a Dabhand Guide, also magazines (BEEBUG, Micro User, Acorn User). Software; Elite, Barbarian, Strykers Run, Ingrid's Back, Modem Master, Dumpmaster, Commstar II, Watford Pro Printer Driver. All boxed and in excellent condition. Tel. 01- 992 0087 after 6pm.

Printwise, Spellcheck III, Wordease, Discmaster, Quickcalc, Forth Language (cassette), Printmaster (Epson), Toolkit, Tape to Disc ROM, few 2764 EpROMs (new), Goulds switched mode power supply, manuals with software. Tel. (0254) 706127 after 11am.

BBC issue 7 with 1770 DFS/ADFS + view £295, 80T DS disc drive + PSU (400k) £90, Aries B32 Shadow RAM board £55, Aries B12 + Adaptor + RAM (6264) £28, Acorn Z80 2nd processor + manuals £150, Acorn cassette recorder £12, Star SG10 Printer + Star Printmaster ROM £160. Tel. (0403) 784976.

Archimedes 310 base system + RISC OS, only 9 months old £650. Also Epson RX80FT+ printer £110 o.n.o. Tel. (0452) 500528.

BBC+ computer with Aries B32 and B12, OS 1.2, Wordwise +, Spellcheck II, Sciways, Wysiwyg +, Artist, Toolkit, Graphics, Printmaster, Commstar, Zromm, View and Viewsheet (all ROMs). Paintbox II, 3D Graphics, Disc User, Betabase (all discs). Printer Taxan/Kaga KP-810 (2 spare ribbons), Digimouse, Demon modem, Microvitec Cub colour monitor 1431, two Cumana disc drives with PSU, Voltmace joysticks (dual), three storey computer desk, games on tapes and discs, Books, magazines, BEEBUG back issues and all manuals for software. More than £2000 of equipment in very good condition, will accept £600 o.n.o. Tel. 061-794 2456.

WANTED: Bolt-on tractor feed for Epson FX80, any reasonable price paid. Tel. (0494) 447088.

BBC model B software, many games (on tape) mostly £2 or less each. Tel. (0945) 85 565 after 6pm.

BBC Master 128 + 512 co-processor. 40/80 T twin disc drive (mains powered) Zenith hi/res mono monitor + plinth. ISO Pascal ROM + ROM pack + BBC and PC software £650. Tel. (0252) 621930.

Unwanted/duplicate gifts; 1st Word+ for Archimedes £50, AMX Paint Pot for BBC £5, both unused complete, in original packing. Tel. (0742) 670866.

Epson FX80 printer, boxed with manual, plus BBC software and documentation £225. Tel. 01-444 0521.

AMX StopPress (DTP) boxed, brand new (without mouse) £22. Tel. (0223) 321128.

Hewlett Packard Deskjet (laser quality) printer, extra font module and 4 new ink cartridges (compatible with BBC, Archimedes or PC reviewed BEEBUG 7.3) £475. Pace Linnet modem £85, MS-DOS sotware on 3.5" disc all suitable for Archimedes running PC emulation, Sage Retrieve Database £60, Ashton Tate Byline Desk Top Publisher £75, Smartcom III comms package £65, New Advanced User Guide for BBC £10, all as new and in makers boxes and include insured delivery anywhere in UK. Tel. (0536) 20 00 94.

Master 128 immaculate condition with assorted software £300 o.n.o. Tel. 041-638 4328 after 4.30pm.

WANTED: "Red Arrows" and "Replicab" on 40T disc for BBC master. Tel. (0673) 60892.

Epson MX80 F/T II/III printer. Defective paper-feed, otherwise ok. £50 o.n.o. Tel. 01-866 2030.

Interword V1.02 £27, Morley Teletext adaptor complete with manuals ROMs and seperate PSU £67, Acorn 65C102 Turbo second processor complete with Advanced Basic by Tubelink £82, (or exchange for 512 board) Acorn Z80 second processor complete and as new £82 (or exchange for 512 board), Advanced Disc User Guide £6. Tel. 051-647 5367.

80186 board with Watford co-processor adapter and Dabs Press book/disc £130, BEEBUG mags vols 1-7 complete with binders £40, Master ROM (BEEBUG) £16, Morley Programmer plus 6 x 27128 epROMs and utility disc £20, Electron, Plus 1, Plus 3, Philips green screen monitor, View, Viewsheet £200. Tel. (0236) 723615.

Solidisk WD1770/8271 disc interface for BBC B (8271 not supplied) for use with beebs already fitted with 8271 DFS Solidisk DFS and ADFS ROMs supplied also £15 o.n.o. Tel. 091-581 9989 eves.

BBC Master 128, Cumana CS 400S 80T DS DD, Archimedes RGB colour monitor, Panasonic KX-P1081 printer, all purchased from BEEBUG and less than 1 year old, includes manuals and leads £700 o.n.o. Tel. (0202) 518361.

Epson RX80 F/T+ printer in good condition £100. Tel. (0420) 83555.

Video Digitiser (Watford) for BBC B or Master with manual and ROM, BNC lead to scart plug included £75. Tel. 031 339 6979.

Technomatic double sided dual DD 40/80T switchable. The unit is in very good condition £135. Tel. 02407 5670.

WANTED: Sinclair Microvision, preferably working, plus any data. Tel. (0865) 59066.

Master 128 Morley AA board, Reference manuals, 8 cartridges, Viewstore, Viewspell, Teletext adaptor, mouse, joystick plus other books and software total value £1100 will accept £600 or will split. Tel. (04243) 4500.

512 co-processor fitted in Universal 2nd Proc unit, DOS V2.1, mouse, User Guide, Dabs 'Master 512 User Guide' & disc, Dabs Shareware collection 1&2 (11 discs) £150. Acorn Teletext adaptor, User Guide, ATS ROM, collection of Telesoftware Teletext integrated software including Improved ATS ROM image £60 or £200 the lot. P&P extra. Tel. (0209) 843294 (0800-1700 hrs), (0209) 842870 eves/weekends.

Master 512 excellent condition c/w Dabs Shareware collection £400, Viewstore/Cartridge£30, Toolkit/Cartridge £10, Master ref manuals 1&2 £15, Dabs View Guide with disc £12, REVS (4 track) £7, Vols 1-7 BEEBUG with index in binders - offers? prefer collect, carriage extra. Tel. (0689) 57245.

INVOICING & ACCOUNTS

New: V3 of The Account Book now available. Comprehensive small business accounts to trial balance. VAT approved. Absolutely the easiest program to use, with neat final books and hundreds of reports. No entry limits. "The Account Book gets first prize for both price and performance"- comparison in Micro User-July "89. A true user -friendly program. It succeeds admirably "-Beebug -Oct '88. And that was Version 2, V3 has many new features. **£27.95.**

New: V2 The Invoice Program. The magazines havn't had a chance to review this yet, but our customers have, and they love it. Database, Invoices (unpaid and paid), Statements (individual and automatic), Stock presets, Debtor lists, Linking with The Account Book and loads more **£27.95.**

Special Offer: £49.95 if purchased together.

Apricote Studios
2 Purls Bridge Farm
Manea
Cambs
PE15 OND

ACCESS VISA

Tel: 035 478 432 for information, help or to order.

Archimedes 310 entry with Acorn backplane, RISC OS and PR manuals £700, half priced software, Zarch, Pacmania, Corruption, Terramex and Jansons Drawing board, BBC B+ 128 plus printer/cassette leads, Delta 3B twin joysticks and various tapes including Elite £300, Acorn 6502 second processor £70, Opus 40/80T double sided DD £70, Pace Nightingale modem + Commstar ROM £60, ROMs; Acorn ADFS £15, Viewstore £25, B+ GXR £10, BCPL package £20, disc software, Elite £8, Revs £8, Acheton £8, Red Arrows £5, Castle Quest £5, Clares Betabase £5, or £550 for the lot. Technomat 10 MEG hard disc, includes power supply plugs in 1MHz bus £300, all o.n.o. and in excellent condition with original packaging and manuals. Tel. 061-483 2983 eves.

Master 512, DOS+ V2.1, 5.25" & 3.5" double sided 80T disc drives linked as a dual drive, Master reference manuals part 1&2, Dabs Press M512 Shareware collection, MOS+, Conversion Kit, Sidewriter, various other books, some with programs disc, disc box, 5.25" discs, £570. BEEBUG C + Standalone Generator £30, Panasonic KXP-1081 printer incl lead £85. Tel. (0924) 826483 after 5pm.

Master 512 with DOS+ V2.1 5.25" & 3.5" drives double console, Acorn med/high res colour monitor, Shareware collection, GEM etc, Morley AA board, BEEBUG Master ROM Smart cartridge, AMX mouse with StopPress & Extra Extra DTP, Revs 4 track, Aviator, Typing Tutor etc. £725. Tel. (0980) 610303 eves/weekends.

Archimedes 310 colour system plus 1st Word+ £800, Logistix spreadsheet/database £55 and Artisan Art 2 discs £30, Logotron Logo £35, NEC P2200 24pin printer £195 o.n.o. Tel. 01-341 2187.

BBC B OS 1.2 Watford DFS, Watford twin DS 40T D/D Watford solderless ROM board incl. 2 off 6264 RAM, Aries B20 Shadow RAM, Philips 12" green monitor £275 will split. Tel. 01- 524 4239.

Viglen dual D/S 400k 40T D/D £45 o.n.o. Viewstore database in original box £25 o.n.o. Tel. 01-642 6270.

ACP "Advanced Disc Toolkit" and "Advanced ROM Manager" & Watford "ROMspell", Three original ROMs and manuals £25 or £10 each, also BBC Teletext adaptor with ATROM, as new £55. Tel. (0943) 607425.

BBC B issue 7 operating system 1.2 without DFS but with disc and tape software £150 o.n.o. Tel. (0734) 883872.

WANTED: View ROM for BBC B and Printwise disc 40/80T. Tel. (0329) 280507.

Books, 40 for BBC up to BBC 128+ all mint. Send SAE for list to; Mr C Harvey, April Cottage, Callow Hill, Rock, Worcs, DY14 9XL.

Interbase still boxed £30 o.n.o. or swap for Interword or anything useful. Tel. (035 87) 229.

Master Compact, colour system, plus A+B Computing 100 programs. Mini Office II, Typing Tutor, Micro Maths plus joystick £475 o.n.o. AMX Super Art Mk III, mouse for Master, 128 boxed unused £40, Philips TP200 anti glare green screen monitor boxed unused £70. Tel. 01-805 8791 after 6pm.

BBC +128K 1770 DFS2 23, ATPL board, Autobeebaid ROM manager, Viewstore, Wordwise+, TDROM, approx thirty discs (ROMs, games, utilities etc),mouse, joysticks, Cumana 40/80T drive, many manuals, Dabhand Guides, 75+ magazines £300 o.n.o. Tel. (0424) 445096 after 6pm.

BBC B issue 3, Basic 2, Watford DFS, ATPL board, Watford 32k shadow RAM, Wordwise plus, Spellmaster, Spellcheck, Help, Printmaster, Sleuth, Microware 80T DS DD 800k disc drive, Microvietc med res colour monitor 14" metal case, Star DP510 printer, plinth, swivel base, printer stand, covers. Checked, in first class working order with all manuals. Upgrading. Bargain at £250 the lot. Tel. (0772) 717017.

Digimouse, Paintbox, Illustrator + 64k printer buffer ROM £45, Knitware Designer £5, Master 128 version of Mini Office II £10. Not a penny more Adventure including novel, Indoor Sports, Play it again Sam II, Last Ninja; games discs £20. Tel. (0326) 562540 after 6pm.

BBC Master 128 £300, Cumana 40/80T DS DD drive £75, Morley Teletext adaptor boxed as new (unused) £75, Viewstore ROM and manual boxed as new £15, Stop Press £10, Elite, XOR, World War 1, Star Wars, Spycat, Spitfire 40, Revs & Revs 4 Tracks, Grand Prix construction set, Superior collection vol 2, Cheat it again Joe 1 £4 each all 10 for £35, Advanced User Guide, Advanced Disc User Guide, BBC Micro book - BASIC, Sound, + Graphics, Advanced programming techniques for the BBC £4 each, entire system £500. Tel. 01-228 0930.

Master 512 Second Processor (with all discs + manual + Nid Valley mouse + software tape sensible offers invited, AMX Super Art Master disc version (no mouse) £15. Vine Micros Master REPLAY board £10. Advanced disc Investigator BBC/Master version £6. Exmon II Master version £12. BEEBUG "C" ROMs + disc + book £25. BEEBUG tapes complete vols. 4 to 7 offers?? Also various books, please telephone for more details. Tel. 01-494 1365 office hours only.

WANTED: Watford MKII EPROM Programmer or TRS80 EPROM programmer and or T004 cassette software as published in "Everyday Electronics" June 1983. Tel. (0526) 21539.

Acorn User issues 1-61, 23 binders + A&B Computing 1984/5. Offers?? Tel. 01-263 0510.

Archimedes 310 with colour monitor hardly used £550 o.n.o. Also Pipedream software, cost £99 used only twice will sell at £50. Tel. (0378) 72570 after 6.30pm, or 01-553 8835 day.

Printer Ribbons suit Epson FT & RX 80 also Panasonic KX-P3131. Ribbons are new, £2.50 each, can post. Tel. (0420) 83555.

BBC issue 7 with Acorn DFS, ATPL ROM board and sideways RAM, View, Viewstore, Viewsheet, Toolkit, Hyperdriver, Spellcheck and many other ROMs, Cumana 40/80 double drive with PSU, Solidisk eprom programmer, Megamouse, Voltmace joystick, many discs and books. Would consider splitting. Price includes carriage £580. Tel. (0403) 55400.

Doubleview £25. WANTED: (for Master): ROM board 3 + Replay add on + Master to B conversion. Tel. 01-698 3772.

Master 512 in Viglen case and detached keyboard, with Z80 co-pro. Separate 40/80 twin disc plinth with own PSU. Mono monitor, joysticks, leads. Full set manuals, books, original software, complete set BEEBUG magazines, extras, all in excellent condition £650 o.n.o. (owner upgrading). Tel. 01-997 1218.

Electron plus 1, 40/80 disc drive (BBC compatible), View, Lisp, Tape to disc ROM, manuals, games, joystick all £225 Tel. (0788) 73606 after 4:30pm.

BBC Master 128 £350, 512 board with DOS+ 2.1 £100, Master reference books 1&2 £5 each, 2 Master ROM cartridges £5 each, Proword wordproc ROM disc and manual £5, Twin 5.25" DS 40/80 disc drive with PSU £100, 1 x 5.25" 80T DD Beeb powered £40, 1 x 3.5" DD caged bare drive can replace 5.25" 80T or for use with a PC as 720K £70, Philips colour monitor CM8833 as new £180, mono monitor 9" green screen comp. video £18, 12" green screen monitor composite video, new tube £10, 7" white screen well worn but works £8, all suitable for Beeb. ZX81 and two 16k RAM packs any offer?? ZX printer and seven rolls of paper £10, Microdrive £10, Interface £10, 40 Microdrive cartridges 75p each, Tasword 2 wordprocessor £3. Tel. (0525) 210551.

Master reference manauals 1&2 £12, Acorn 6502 second processor (with original packing Hi-Basic & DNFS ROMs and user guide) £65, Tube link Advanced Basic ROM with supporting disc and manual £15, Acorn GXR ROM (BBC B) complete with manual £8, Wordwise plus complete with manuals £20, Masterfile (Master ADFS version) £8, BEEBUG Master ROM (boxed with manuals £10, BEEBUG Toolkit ROM £3, Care Quad cartridge £8, Care Dual cartridge £4. Tel. (0384) 373928.

WANTED: Advanced Disc User Guide. Tel. (04574) 5263.

Christian Computer Games and Aids to Bible Study from The Evangel Trust (a charity registered in England). Bible based games for all ages and Bible study packages using the text of AV, TEV and NIV together with databanks and factsheets on background material. Good range of BBC material for DFS or ADFS. Send SAE for brochures (A5 size) to The Evangel Trust, PO Box 224, Kingston Upon Thames, KT1 2NX.

Watford Solderless 12 ROM-RAM board RB1201 with battery back up. 1988, unused, with instructions. List £39. Offered at £20. Tel. (0530) 32619.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES		BEEBUG & RISC USER
£14.50	1 year (10 issues) UK, BFPO, Ch.I	£23.00
£20.00	Rest of Europe & Eire	£33.00
£25.00	Middle East	£40.00
£27.00	Americas & Africa	£44.00
£29.00	Elsewhere	£48.00

BACK ISSUE PRICES (per issue)

Volume	Magazine	Tape	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	-	-
3	£0.70	£1.50	£3.50	-
4	£0.90	£2.00	£4.00	-
5	£1.20	£2.50	£4.50	£4.50
6	£1.30	£3.00	£4.75	£4.75
7	£1.30	£3.50	£4.75	£4.75

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. Note that there is no VAT in magazines.

Destination	First Item	Second Item
UK, BFPO + Ch.I	60p	30p
Europe + Eire	£1	50p
Elsewhere	£2	£1

POST AND PACKING

Please add the cost of p&p as shown opposite.

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Assistant: Glynn Clements
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

BEEBUG Ltd (c) 1989

ISSN - 0263 - 7561

Printed by Newnorth-Burt Ltd (0234) 41111

Magazine Disc/Cassette

NOVEMBER 1989 DISC/CASSETTE CONTENTS

CLOWNS GAME - a bonus item for Christmas!

INSTANT PUBLISHING - a program which converts teletext files into plain text for use with word-processors and DTP.

BASIC ROM IMAGE CREATOR - a utility which allows you to create ROM images of Basic programs.

A BEEBUG GRAPH PLOTTER (Part 2) - the complete programs plotting graphs of relations, families of curves, graphs defined parametrically or by using polar co-ordinates.

DISC FILE IDENTIFIER - a useful utility for identifying and analysing the contents of your DFS discs.

AMATEUR RESEARCH (Part 2) - explore the world of the aether and vertices with this month's program.

A GOOD REPORT - all the programs from the magazine plus a further program which will 'speak' error messages if Speech! is installed.

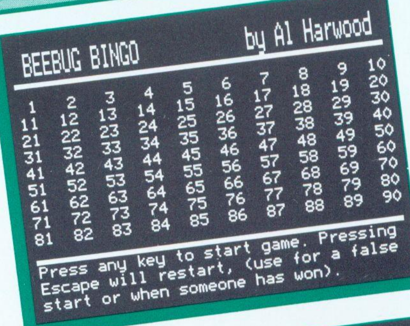
A POSTSCRIPT SCREEN DUMP UTILITY - converts a graphics screen dump into a PostScript file for output on a laser printer.

CREATING A PLUS-OR-MINUS CHARACTER - a utility which creates this useful mathematics sign for display or printing.

THE CLOSE#0 BUG IN MASTER DFS - a program which overcomes this well known bug in the Master.

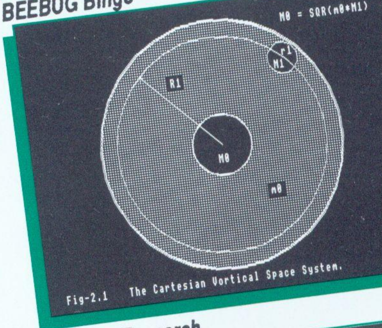
BEEBUG Bingo - an electronic version of this popular game.

MAGSCAN - bibliography for this issue (Vol.8 No.6).

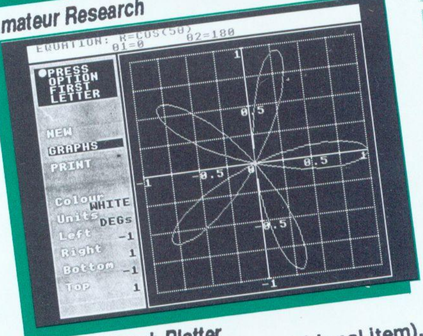


```
BEEBUG BINGO                        by Al Harwood

 1   2   3   4   5   6   7   8   9  10
11  12  13  14  15  16  17  18  19  20
21  22  23  24  25  26  27  28  29  30
31  32  33  34  35  36  37  38  39  40
41  42  43  44  45  46  47  48  49  50
51  52  53  54  55  56  57  58  59  60
61  62  63  64  65  66  67  68  69  70
71  72  73  74  75  76  77  78  79  80
81  82  83  84  85  86  87  88  89  90

Press any key to start game. Pressing
Escape will restart, (use for a false
start or when someone has won).
```

BEEBUG Bingo



Amateur Research



```
EQUATION: R=COS(5θ)
 θ1=0          θ2=180
PRESS
OPTION
FIRST
LETTER
NEW
GRAPHS
PRINT
Colour WHITE
Units  DEGs
Left     -1
Right     1
Bottom   -1
Top       1
```

BEEBUG Graph Plotter

All this for £3. 50 (cassette), £4.75 (5" & 3.5" disc) + 60p p&p (30p for each additional item).
Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

	UK ONLY			OVERSEAS		
SUBSCRIPTION RATES	**5" Disc**	**3.5" Disc**	**Cassette**	**5" Disc**	**3.5" Disc**	**Cassette**
6 months (5 issues)	£25.50	£25.50	£17.00	£30.00	£30.00	£20.00
12 months (10 issues)	£50.00	£50.00	£33.00	£56.00	£56.00	£39.00

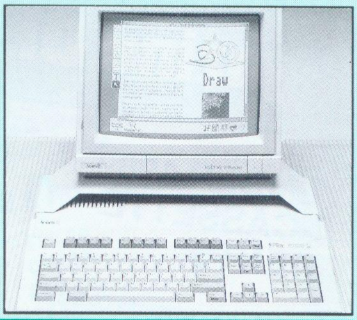
Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:
BEEBUG, 117 Hatfield Road, St.Albans, Herts AL1 4JS.

BEEBUG - THE ARCHIMEDES SPECIALIST

The NEW BBC Micro - The A3000

Ideal for home, education and business use, this powerful computer includes the multi-tasking Operating System, RISCOS. BBC Basic V and a BBC B emulator are supplied as standard. An optional PC emulator disc allows over 90% of DOS software to be used including Lotus, dBase III, MS Word and Wordperfect.



FREE ON-SITE MAINTENANCE

All new Archimedes and A3000 computers purchased through Beebug include one years On-Site Maintenance.
Beebug is the only Acorn dealer offering this service.

0% Finance

Under this scheme you may purchase an Archimedes A3000, 300 or 400 Series computer (base, mono or colour) with an initial deposit and 9 payments at monthly intervals. There is no charge to you for this service.

Some examples of the repayments are:

Machine	Deposit	9 Payments
A3000	71.35	75.00
A3000 Colour	99.70	96.00
310	106.85	103.00
310 Colour	126.20	125.00
310M	112.85	110.00
310M Colour	132.20	132.00
410/1	145.85	137.00
410/1 Colour	165.20	159.00
420/1	198.85	195.00
420/1 Colour	218.20	217.00
440/1	290.85	287.00
440/1 Colour	310.20	309.00

If you wish to take advantage of this scheme please telephone us and we will send you further details. Finance over 12/24/36 months is also available.

THESE PRICES INCLUDE VAT

SEE BELOW FOR SPECIAL OFFERS

ACORN A3000 SERIES		
0255G	A3000 Entry System	649.00
0256G	A3000 Colour System	838.00
ARCHIMEDES 300 SERIES		
0193G	310 Entry System	899.00
0195G	310 Colour System	1088.00
ARCHIMEDES 400 SERIES		
0260G	410/1 Entry System	1199.00
0261G	410/1 Colour system	1388.00
0262G	420/1 Entry System	1699.00
0264G	420/1 Colour System	1888.00
0275G	440/1 Entry System	2499.00
0276G	440/1 Colour System	2688.00

SEE SPECIAL OFFERS BELOW

These prices exclude VAT

SPECIAL OFFER ON COLOUR SYSTEMS

All colour systems are now supplied with the superior Phillips CM8833 monitor featuring Stereo sound and video input. If you prefer the Acorn monitor please specify when ordering.

SPECIAL DISCOUNTS FOR TEACHERS

We are currently able to offer significant discounts to teachers and Acorn Shareholders on computers. Please phone for further information.

RISC USER & BEEBUG MEMBERS BUYING AN ARCHIMEDES SYSTEM WILL RECEIVE FREE:

A3000 -On Site Maintenance, and items to the value of £46 (entry system) or £56 (colour system).
Members using the 0% finance offer receive On-Site Maintenance only.

310 - PC Emulator, Printer Lead, Ten 3.5" discs & Lockable disc box.

410/1 - On Site Maintenance, and items to the value of £105 (Entry system) or £120 (Colour system).
Members using the 0% finance offer receive free On-Site Maintenance only.

420/1- On Site Maintenance, and items to the value of £166 (Entry system) or £177 (Colour system).
Members using the 0% finance offer receive On-Site Maintenance and 10 x 3.5" discs.

440/1 - On Site Maintenance and items to the value of £259 (Entry system) or £269 (Colour system).
Members using 0% finance receive free On Site Maintenance, 10 discs in lockable Box & Interdictor

Please phone or write to receive a copy of our new FREE 40 page retail catalogue.

117 Hatfield Road, St Albans Herts AL1 4JS Tel: (0727) 40303 Fax: (0727) 60263