DRAG'NDROP

RISC OS **Pi** and all RISC OS 5 machines

January 2015 Volume 6 Issue 2 £3.50

Programming

Wimps

Calculator

PDFtoText

RiscLua

Mr Miner

Type-in game

RiscPiC

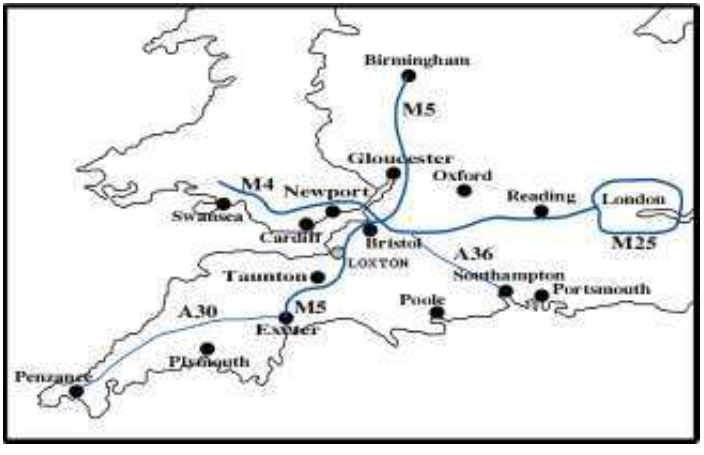
ORPHEUS INTERNET

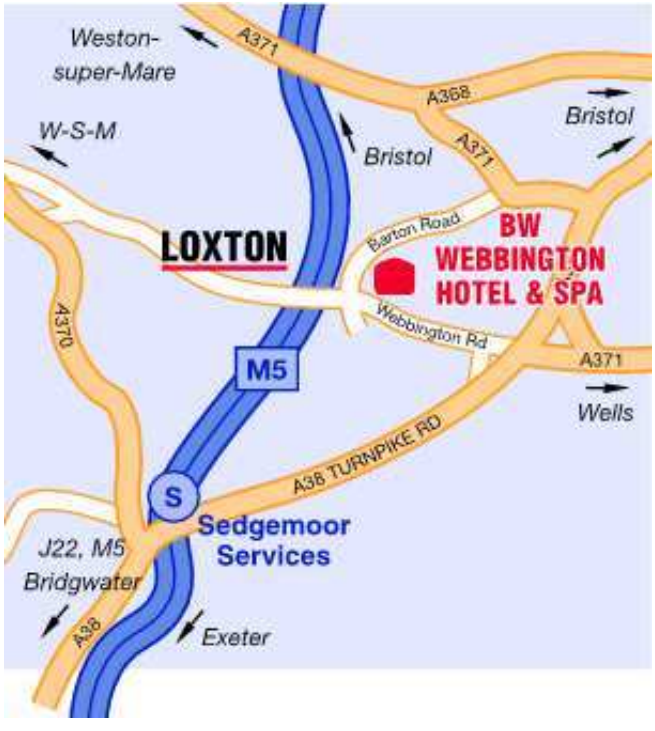
RCamp

RISC OS SouthWest Show 2015

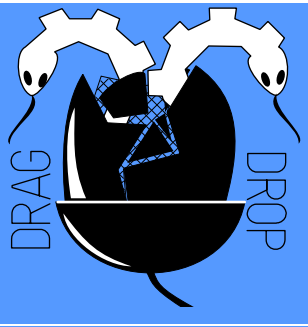
Webbington Hotel, BS26 2HU

10.30 – 4.00 Saturday 21st February





www.riscos-swshow.co.uk



This issue has been blessed with contributions from the following people:

Jon Robinson (RISC OS programming in Basic)
Gavin Wraith (Why not Basic?)
Paul Dunnington (HtBackPack)
Christopher Dewhurst (everything else)

The views expressed in this magazine are not necessarily those of the editor. Alternative views are always welcome and can be expressed by either writing an article or a short editorial.

All articles and advertisements are published in good faith. No materials in this publication are meant to be offesnsive or misleading. If you come across something you believe is either of the above please contact the editor using the details below.

Contact Information

Editor: Christopher Dewhurst
Email: editor@dragdrop.co.uk
www.dragdrop.co.uk

EDITORIAL

Happy new year to all our readers.

Whilst documentation on RISC is very good there needs to be more: the more writers there are on a subject the more likely you are to find something accessible to you.

That's why we have been running two series on Wimps programming by different authors. The first rounds off development of a simple desktop calculator. The second continues the PDF-to-text application.

Hopefully 2015 will see more developments on the software side. We've been flooded with new hardware, all very good, but we're very much in need of web browsers like Netsurf to make more websites available to RISC OS.

Chris.

Christopher Dewhurst

At a glance...

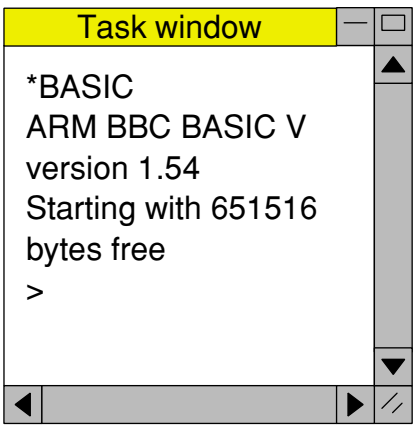
Beginner's Bit

Programs listed in *Drag 'N Drop* are assumed to work on all machines with RISC OS 5 e.g. Raspberry Pi, unless otherwise stated

To get the BBC Basic prompt press F12 and type *BASIC and press Return. Type AUTO for automatic line numbering.

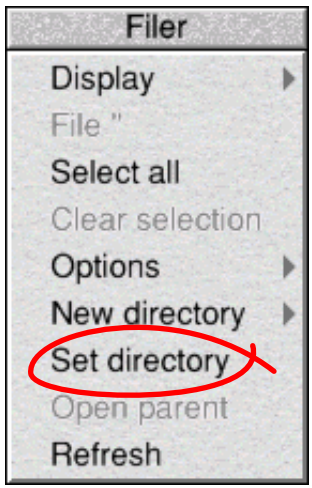
To return to the desktop type *QUIT.

You can also type programs in a task window, hold down Ctrl and press F12. You can't use the cursor editing facility or change MODE, however.



Type SAVE "prog" when you are ready to save your program where "prog" is the program's name.

You can also program and run Basic programs from the desktop. Double-clicking on the filer icon runs it, holding down Shift and double clicking loads it into your text editor.



To set the current directory, click the filer menu and select "Set directory".

Or use the EasyCSD application listed in the Autum 2014 issue.

News and Apps

Awards 2014-15

It's time to vote at the RISC OS awards. There are 12 categories in which to vote for something in the RISC OS world which has stood out in some way over the past year. Closing date is 7th February so get along to www.riscosawards.co.uk (and please nominate *Drag 'N Drop* in the 'Best publication or Offline resource'!)

New email address

The new email address for *Drag 'N Drop* is

dragdrop@dragdrop.co.uk

We thought it would be easier to remember than the various submissions@..., editor@.... The old addresses will work for a while though.

Showtime

The 2015 South West RISC OS show is happening on 21st Feburary at the Webbington Hotel near Weston-Super-Mare in Somerset. Doors open 10.30. More details at www.riscos-swshow.co.uk. See you there!

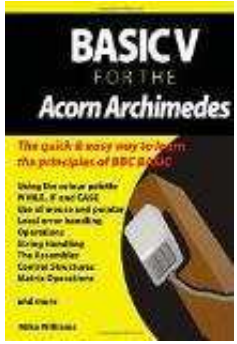
New case for your Pi

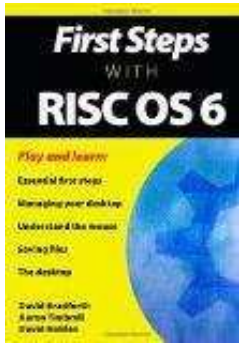
Give your Raspberry Pi the vintage RiscPC look with this stunning new product from Soft Rock Software. The 3D-Printed case is suitable for any model of Pi up to the B+ and was unveiled at the London Show in October 2014. Price is expected to be about £30 from www.softrock.co.uk

Remember it hasn't shipped on a barge carrying 19,100 containers of mass produced tat. *(And will be here long after said tat breaks and goes to landfill – Ed.)* Please support your UK suppliers!

Books

Two new books have appeared on the market, *First Steps with RISC OS 6* and *Basic V for the Archimedes.* Although not specifically for the Pi, both are relevant. The second title is a re-print of a book from the early days of RISC OS and useful for the Basic programmer of 2015. Price £5.00 each plus postage from www.amazon.co.uk.





Forthcoming Book

We're putting together a book, *The Book of Draw Stuff*. It will be of interest to anyone who uses Draw. The first half will be about using the built-in vector application to achieve useful effects you never knew about and the second half will be more technical and based on the recent 'Anatomy of a Drawfile' series in the magazine. More details in the next issue!

Competition Time

Win a fully-fledged Raspberry Pi B+! RISC OS France (ROF) are holding a programming competition. All you have to do is write a demonstration, Wimp application, screensaver etc. program in Basic and email it to temp1267@riscos.fr. See www.riscos.fr for more details. ROF are happy to help freely (in English if necessary!) with their 20 years of experience.

NetRadio

Listen to and record broadcasts online on your Raspberry Pi with NetRadio available from www.bapfish.co.uk/netradio.htm. If you are going to be a regular user please observe that it is Giftware and send the author a donation to keep up his good work!

New computer

At the London show 2014, R-Comp unveiled the latest model in their line of 'pure' (i.e. using Arm chips) RISC OS computers. Called the iMX6 and yet to be available to order online, the computer boasts direct connectivity for SSD (Solid State Drives), 2Gb RAM, 1Gb Ethernet and faster computing than ever before. Price is expected to be around £500-£600.

TBX 0.7.3

This is a C++ library to help with the development of C++ applications complete with user guide. Download from sites.google.com/site/alansriscosstuff/tbx

MSPdata / MLAdata

These apps give information on Members of Scottish and Northern Ireland Parliaments for a given postcode. www.kevsoft.co.uk/news/category/ptools/mspdata/ and www.kevsoft.co.uk/news/category/ptools/mladata/.

OpenVector suite patch

A patch has been released to solve instability problems with the OpenVector, DrawPlus, OpenGridPro series. Download from www.users.on.net/~belles/software/openvector.

Review

Product: QuizMaster/QuizMaker
Price: £20.00
Supplier: Archiesoft (via !Store or eBay.co.uk)

QuizMaster is a multiple-choice trivia program, now compatible with the Raspberry Pi, with its own utility (QuizMaker) to create quizzes of your own. After installing itself on the iconbar a window opens onto which you can drag a questions file, either the supplied or one you have created yourself using the accompanying QuizMaker app.

Clicking on the Start button puts the computer into full screen mode with the question at the top, a picture in the middle and possible answers at the bottom.

The mouse isn't used; instead you press the key for your chosen answer. At the end of the contest you are given your score before being returned to the desktop.

With the sister program, QuizMaker, you can type up your own questions and answers, drag in Drawfiles, JPEG files and sound files to accompany the trivia. The results can be saved out for future editing or loaded into QuizMaster.

The RISC OS system is so easy to use that building quizzes with pictures using built-in Draw etc. is a pleasure.

Manuals for QuizMaker and QuizMaster are supplied in HTML format and are helpful and comprehensive.

I found the software to be not quite polished. For example the 'About this program' window is out of date and QuizMaster seems to unceremonously dump a screenshot of your score in the root directory of your hard disc.

The manual mentions that more quizzes planned and you might wait for the improved version to be released before parting with your cash.

Writing a simple app Part 4

Last time we got as far as getting the application to respond to calculator buttons by making a sound corresponding to the icon number (button) clicked.

There are 16 icons numbered 0 to 17, with icons 0 to 9 being the digit keys of the calculator, icon 10 the decimal point, icon 11 the equals sign, icons 12 to 15 the operator buttons (add, subtract, multiply, divide) and 16 the clear button.

Icon 17 the calculator display field. It's different from the others because we need to know how to change the text in the icon.

For example pressing a digit key needs to add the digit to the display. How do we achieve this?

First of all type in the listing, which is complete listing of the whole application for convenience.

PROCupdate (lines 440-510) is the code which we need to update the text in the icon. We use the parameter *block%* to tell the Wimp which icon is to be updated and with what.

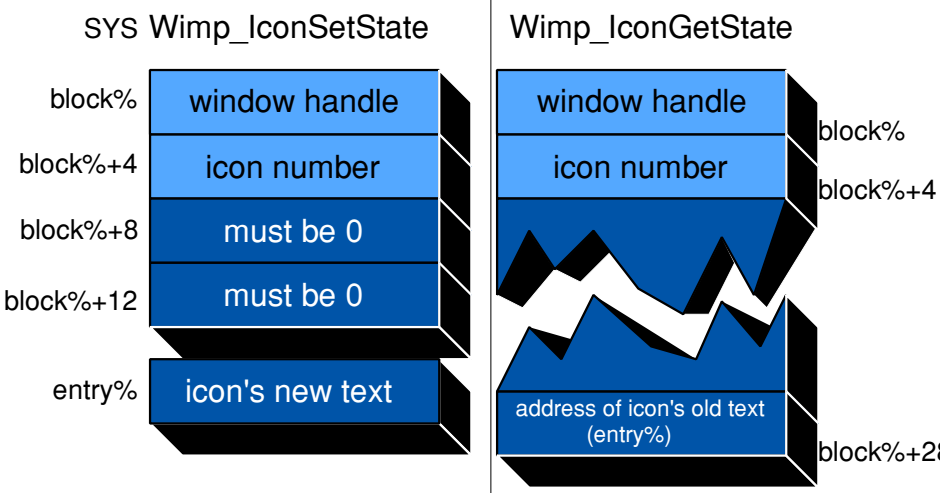
The first word of *block%* is the window handle, which is *main%*, the calculator wondow. The second word contains the number of the icon to be updated. The third and fourth words must contain zero. (They can be other numbers for other effects like changing the icon's border but that's beyond the scope of this article.) See Figure 1.

We also have to know where in memory the Wimp stores the text of the icon. This is done by calling Wimp_GetIconState in the initialisaton procedure (PROCinit) in lines 980-1030.

Word one of *block%* contains the window handle and the second word the icon number. Wimp_GetIconState returns the address of the text in the eighth word (at *block%+28*).

Line 1030 makes a copy of this

Figure 1. Parameters for retrieving and altering an icon's text.



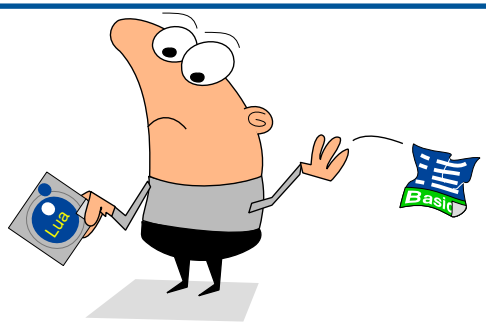
Why Not Basic?

My first computer was an Acorn Atom and my first efforts at programming were in Atom Basic. RISC OS users are likely to have used BBC Basic for their first attempts at programming. So nostalgia makes Basic popular.

However, there are some programming tasks for which Basic is not suited and if Basic is the only language you've tried you may not realize how your horizons have been limited.

Thanks to its editor, Rick Murray, I wrote an article *Why Lua is not Basic* in volume 22 of Frobnicate (a magazine now sadly defunct). In it I looked at the following problem:

Write a program that takes as input the name of a file of receipts, and the kind of expense, which prints out a list of the employees who incurred expenses of that kind, together with the total of each's receipts.



Each receipt has the format:

RECEIPT {employee_name, amount, type_of_expense }

I gave two solutions: one in BASIC which had 68 lines and the other in Lua – only 15 lines.

The 'McGuffin' of the Lua solution is that RECEIPT can be the name of a function which adds its argument as an item into a table.

The receipts file can simply be executed as a piece of Lua code (using the **dofile** function) and no parsing is needed.

Lua was originally designed as a data-entry language so you might say that the test is not really fair, because Basic was definitely *not* designed as a data-entry language.

I am not trying to do BASIC down – merely to point out that it is a question of the right horse for the right course.

Here are some more problems you might like to consider in Basic:

Write a program to read in a textfile and output its lines in lexicographic order.

Here is a Lua solution:

```
»#! lua
»local a = { } -- empty table
»for line in io.lines (arg[1])
do  a[1 + #a] = line end --
for
»table.sort (a)
»for i = 1, #a do print (a[i])
end
```

Here **arg[1]** denotes the first argument on the commandline, which we presume to be the pathname of the textfile. The # operator gives the length of an array or string.

Easy Scroll

This is a simple multi-tasking program which makes scrolling windows easy.

Instead of moving the mouse pointer to the scroll bar arrows and clicking you can hover near the edges of the window and scrolling automatically takes place.

See the program description for how it works. To quit the program, right click on the Raspberry icon to bring up the Application tasks window and scroll up (by hovering just below the title bar!) and click Menu over EZScroll > Task 'EZScroll' > Quit.

There are many improvements you could make. For starters, make it into a proper Wimp application with an iconbar icon (refer to the Wimp tutorials in this issue).

Include a menu with at least a Quit option, and maybe the margin and step speed could be made into menu options.

Also the program doesn't currently check if the window

actually has scroll bars so it's entirely possible to scroll windows with no scroll bars but in fact have large work areas. Can you find the hidden red box in the "About the Operating System" window?

EZScroll listing

```
   10 REM EZScroll
   20 REM (c) Drag N Drop 2015
   30 :
   40 DIM block% &100
   50 step%=4:REM controls scr
olling speed
   60 margin%=16:REM how close
 the pointer has to get to the
 window edge
   70 count%=0:delay%=20:REM l
ength of the delay before star
ting to scroll
   80 flag%=FALSE:REM Scroll f
lag
   90 SYS "Wimp_Initialise",50
0,&4B534154,"EZScroll"
  100 ON ERROR PROCerror
  110 quit%=FALSE
  120 REPEAT
  130 SYS "Wimp_Poll",0,block%
 TO reason%
  140 CASE reason% OF
  150 WHEN 0 : PROCscroll
  160 WHEN 17,18 : quit%=TRUE
  170 ENDCASE
  180 UNTIL quit%=TRUE
  190 SYS "Wimp_CloseDown"
  200 END
  210 :
  220 DEF PROCscroll
  230 SYS "Wimp_GetPointerInfo
",,block%
  240 xptr%=!block%
  250 yptr%=block%!4
  260 IF block%!16<-1 OR block
%!12<0 THEN ENDPROC
  270 !block%=block%!12
  280 SYS "Wimp_GetWindowState
",,block%
  290 xscroll%=block%!20
  300 yscroll%=block%!24
  310 IF xptr%<block%!4+margin
% THEN xscroll%-=step%*4:flag%
=TRUE
  320 IF xptr%>block%!12-margi
n% THEN xscroll%+=step%*4:flag
%=TRUE
  330 IF yptr%<block%!8+margin
% THEN yscroll%-=step%*4:flag%
=TRUE
  340 IF yptr%>block%!16-margi
n% THEN yscroll%+=step%*4:flag
%=TRUE
  350 IF flag% THEN
  360  count%+=1
  370  IF count%>delay% THEN
  380   block%!20=xscroll%
  390   block%!24=yscroll%
  400   SYS "Wimp_SendMessage"
,2,block%,!block%
  410  ENDIF
  420  ELSE
```

BASIC Programming Tutorial 5 – Displaying Messages

In the last instalment, we used WinEd to create the main program window, and got the program to display the window when a file was dragged onto its icon.

In this instalment, we are going to:

- learn how to discriminate between PDF and non-PDF files
- develop a messaging routine, which will make it easier to see what's going on inside the program
- get the program to detect a click on its OK button
- get the Back icon working properly.

Dig out your copy of !PDFText. This instalment's listing should be saved as *!PDFText.Stage5* and the last line of the !Run file updated to

```
Run <Obey$Dir>.Stage5
```

Refer back to the Summer 2014 issue of *Drag 'N Drop* if you are unsure of what other files you need. You can save yourself some typing time with the *Stage5* file since a lot of it is the same as *Stage4* with the lines being changed or added described below.

Reading the File Type

Every time the program executes the main SYS "Wimp_Poll" loop, the Block is loaded with a set of values, which describe the event that has just happened (if any).

When a file is dragged onto the program's icon bar icon, and PROC FileArrived is triggered, RISC OS will have left the file type of the file at *Block%!40*.

When the incoming file is a PDF, Block%!40 will have been set to &ADF (in hex) – the file type for PDF.

We can now refine FileArrived, to open the main window ONLY when the incoming file is a PDF.

```
  570 DEF PROCFileArrived
  590 LOCAL FileType% :FileType%=Block%!40
  610 IF FileType%=&ADF THEN
  620 Block%!0=SaveHandle%
  630 PROCOpenWindow
  640 ELSE
  650 VDU7
  660 ENDIF
  680 ENDPROC
```

If you type this in correctly, the main program window will open, if a PDF file is dragged onto its icon. If the file is not a PDF, it will beep instead.

```
enu in first 28 bytes.
 1520
 1530 $(IconMenu%)="PDF Text" :REM> Name at t
op of menu window
 1540 IconMenu%?12=7 :REM> Title foreground =
black
 1550 IconMenu%?13=2 :REM> Title background =
 grey
 1560 IconMenu%?14=7 :REM> Work area foregrou
nd = grey
 1570 IconMenu%?15=0 :REM> Work area backgrou
nd = white
 1580 IconMenu%!16=60 :REM> Menu width
 1590 IconMenu%!20=44 :REM> Menu height
 1600 IconMenu%!24=0 :REM> Gap between items
 1620 REM> 24 bytes used to define each item
on the menu.
 1630
 1640 MenuSpace%=MenuSpace%+28 :REM> Data for
 "Info"
 1650 !MenuSpace%=0 :REM> Menu flags
 1660 MenuSpace%!4=InfoHandle% :REM> Sub menu
 pointer
 1670 MenuSpace%!8=&07000021 :REM> Icon flags
 1680 $(MenuSpace%+12)="Info"
 1690
 1700 MenuSpace%=MenuSpace%+24 :REM> Data for
 "Quit"
 1710 !MenuSpace%=&80 :REM> Bit seven of menu
 flags set
 1720 MenuSpace%!4=-1 :REM> Sub menu pointer
 1730 MenuSpace%!8=&07000021 :REM> Icon flags
 1740 $(MenuSpace%+12)="Quit"
 1760 ENDPROC
 1780 DEF PROCIconBarClick
 1800 CASE ButtonClicked% OF
 1810 WHEN 2: PROCShowIconMenu(IconMenu%, !Bl
ock%-64, 96+2*44)
 1820 ENDCASE
 1840 ENDPROC
 1860 DEF PROCShowIconMenu(Menu%, x%, y%)
 1880 SYS "Wimp_CreateMenu",, Menu%, x%, y%
 1900 ENDPROC
 1920 DEF PROCCloseIconClicked
 1940 REM> Called from main loop when EventCo
de%=3
 1960 SYS"Wimp_CloseWindow",,Block%
 1970 ENDPROC
```

will be back next month

Have you written a program for your RISC OS Pi?

Can you write an article to describe it?

Get £15 to appear in Drag 'N Drop!

dragdrop@dragdrop.co.uk

Mr. Miner

The object of this game is to collect all of the jewels, dig out all of the earth, and kill the poisonous mushrooms.

Rocks always fall if they are unsupported and if they land on a gem or another rock they will roll off left or right if possible.

Rocks must be dropped onto mushrooms to squash them, you cannot push rocks into mushrooms.

You must journey through five caves before you can leave the system.

Use Z and X for left and right and P and L for up and down. To quit just click on the window's close icon.

The game uses Mode 10 sprites with special OS routines to display them correctly on the desktop. Paint doesn't allow you to create Mode 10 sprites directly. Instead type in and run Listing 1 which creates a set of blanks which you double click to open in Paint and fill in by referring to the grids on page 28-30.

Create an application directory called *!MrMiner* and save the completed sprites inside as *!MrMiner.msprites*.

Then type in Listing 2, an Obey file, which should be saved as *!MrMiner.!Run*.

Finally type in Listing 3 the Wimp application and game and save it as *!MrMiner.Miner*.

● First published in *Electron User* September 1987. Enhanced version for RISC OS published in *Drag 'N Drop* 2015.

Listing 1

```
   10 REM Mr Miner
   20 REM Generate blanks for
editing in Paint
   30 REM (c) Drag N Drop 2015
   40 DIM sprites% &1400
   50 number%=10
   60 ptr%=12
   70 !sprites%=number%
   80 sprites%!4=&10
   90 w%=12:d%=16:size%=w%*d%*
2
  100 FOR shape%=224 TO 224+nu
mber%-1
  110 ptr%!sprites%=44+size%:R
EM offset to next sprite
  120 $(sprites%+ptr%+4)=STR$s
hape%+STRING$(12-LENSTR$shape%
,CHR$0)
  130 ptr%!(sprites%+16)=w% DI
V2 -1
  140 ptr%!(sprites%+20)=d%-1
  150 ptr%!(sprites%+24)=0
  160 ptr%!(sprites%+28)=31
  170 ptr%!(sprites%+32)=&2C
  180 ptr%!(sprites%+36)=&2C
  190 ptr%!(sprites%+40)=10:RE
```

```
4160 DATA FAFFAFAABABAAAAAAE
4170 DATA EFFAEEACEEEAECEEAE
4180 DATA AAAFFAAFFFFAEAAEAE
4190 DATA ECAFBHAAGCFAEAEEAE
4200 DATA AFFFEFAFCFFAEAEAAE
4210 DATA FFAADAAFFDDACAEEEE
4220 DATA CAEFBFAAAAAABAAAAE
4230 DATA EFHFEFFAABAFEEEEAE
4240 DATA AAFFDFCAEEEADAAEAE
4250 DATA EAFAFAFAEHEFFEEEAE
4260 DATA CAFAAEFAEEEACEAAAE
4270 DATA CACAEHFAAFAAAEABFE
4280 DATA BAFAHFFFAFFFAEAFFA
4290 DATA EFFAFFCFAAAFAEAFCH
4300 DATA CACAFFFFFFFFAEACFF
4310 DATA FFFAFDFAEAACAEAFDF
4320 :
4330 REM Level 5
4340 DATA CEEEEFBFBEEBAFBEEC
4350 DATA FEBCAFHFHECEAFAECB
4360 DATA HEBEACADAEECADAEEB
4370 DATA FFBEAAAAAAFECACBEB
4380 DATA CEBEFFEFEFFAAAEEAA
4390 DATA FEBEFAAAAAAACCFBEB
4400 DATA FACAFAEFCFEACADAEE
4410 DATA FFACFFFAECAEACABAF
4420 DATA FCCFFAFEAAFFBEEEEE
4430 DATA FFFFAFEAEFFCAEBFFE
4440 DATA EFFAEEACFFBACCABFF
4450 DATA AAAEFAEEFCAECABADD
4460 DATA HCFFACEAFACBABEAAA
4470 DATA FCEFBAEFCACEEEBEBF
4480 DATA EFEAECAFAACACCEAEF
4490 DATA EAEEFAEEBFABAAEACF
4500 DATA CCAEBACACABEBFFAHF
4510 DATA EAACEAEBAEEAEFAECC
4520 DATA EFAACAFCAEDBAACEFC
4530 DATA AFFAEAFCACCEAEBEEH
4540 DATA CHFACAFABAAEEEHEEC
4550 DATA AFFACFEAEFFAFBADEC
4560 DATA FFAAEAAAHFFFAEEACC
4570 DATA FAFCDCFFFFFFAFHHHH
4580 DATA FFFFCFFFFDIACECCCG
```

"224" Bricks

21	21	21	21	7	252	21	21	21	21	7	252
21	21	21	21	7	252	21	21	21	21	7	252
21	21	21	21	252	252	21	21	252	21	7	252
252	252	252	252	252	252	252	252	252	252	252	252
21	21	7	252	21	21	21	21	7	252	21	21
21	21	7	252	21	21	21	21	7	252	21	21
21	252	7	252	21	21	21	21	7	252	21	21
252	252	252	252	252	252	252	252	252	252	252	252
252	21	21	21	252	21	21	21	252	252	252	252
21	21	21	21	7	252	21	21	21	21	7	252
21	21	21	21	7	252	21	21	21	21	7	252
252	252	252	252	252	252	252	252	252	252	252	252
21	21	21	7	252	252	21	21	21	7	252	21
21	21	21	7	252	21	21	21	21	7	252	21
21	21	252	7	252	21	21	21	21	7	252	21
252	252	252	252	252	252	252	252	252	252	252	252

"225" Boulder

0	0	0	0	39	39	39	39	0	0	0	0
0	0	0	39	69	69	69	39	39	0	0	0
0	0	39	69	69	69	252	69	39	39	0	0
0	0	39	69	252	69	69	69	69	39	0	0
0	39	69	69	252	69	69	69	69	69	39	0
0	39	69	70	69	69	252	69	69	69	39	0
38	68	68	70	70	69	69	69	252	69	69	39
38	68	38	68	68	69	69	69	69	69	69	39
38	68	38	68	68	68	69	69	252	69	69	39
38	68	38	38	68	68	69	69	3	69	69	39
0	38	38	38	68	68	68	69	69	69	69	0
0	38	38	38	3	68	68	3	69	3	39	0
0	0	38	38	38	68	68	68	69	69	0	0
0	0	38	38	69	69	3	68	3	39	0	0
0	0	0	38	38	3	69	69	39	0	0	0
0	0	0	0	37	37	37	37	0	0	0	0

"226" Diamond

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	251	251	0	0	0	0	0
0	0	0	0	251	15	251	251	0	0	0	0
0	0	0	251	251	15	15	251	251	0	0	0
0	0	251	15	251	15	47	251	47	251	0	0
0	0	251	15	15	251	251	47	15	227	0	0
0	251	15	251	251	47	47	251	251	47	227	0
251	15	251	15	47	251	47	251	47	227	47	227
251	15	251	47	251	47	251	47	227	15	227	15
0	251	47	251	251	15	47	227	227	47	227	0
0	0	251	47	47	227	227	47	47	227	0	0
0	0	251	47	227	47	47	227	15	227	0	0
0	0	0	227	227	15	47	227	227	0	0	0
0	0	0	0	227	227	227	227	0	0	0	0
0	0	0	0	0	227	227	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

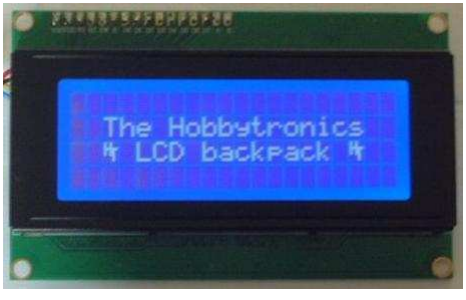
"227" Mushroom

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	21	21	21	21	0	0	0	0
0	0	0	16	255	21	21	255	21	0	0	0
0	16	16	16	21	21	21	21	21	21	21	0
0	16	255	16	21	255	21	21	21	255	21	0
16	16	16	16	16	21	21	255	21	21	21	21
16	16	16	255	16	16	16	21	21	255	21	21
16	255	16	16	16	255	16	16	21	21	255	21
0	16	16	16	0	84	87	0	21	21	21	0
0	0	0	0	0	84	87	0	0	0	0	0
0	0	0	0	0	84	87	0	0	0	0	0
0	0	0	0	0	84	87	0	0	0	0	0
0	0	0	0	0	84	87	0	0	0	0	0
0	0	0	0	0	84	87	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

LCD Backpack Part 3

In the October 2014 issue of *Drag 'N Drop* I presented an improved version of the HtBackpack module to drive the Hitachi LCD display so it could be used with either the Pi's serial port or the I2C connector.

In this article I'll give an you a idea of how the code for PiBaud came about.



I must offer my thanks to Bruce Smith for giving permission to use the 'division and remainder' routines for PiBaud. The code for these are based on the routines in the book *Raspberry Pi Assembly Language for Beginners: Hands On Guide* by Bruce Smith (reviewed in the Winter 2013 edition of *Drag 'N Drop*).

The registers have been changed to get the results where I needed them to be. *PiBaud* at line 12020, is by far the largest routine in this module.

As mentioned last time I couldn't get two of the baud rates to work at all so it was necessary to write a routine that would calculate the baud rate divisors and also the baud rate from the divisors.

We start at the word before the actual code which holds the value 18,750,000.

```
12000EQUD 18750000
```

We can't move this value into register with a MOV instruction and so it is stored in memory where we can load it with an LDR instruction.

Similarly, we cannot use CMP R1,#-1 but we can use the CoMpare Negative (CMN) instruction. This compares the negated value of 1 with R1.

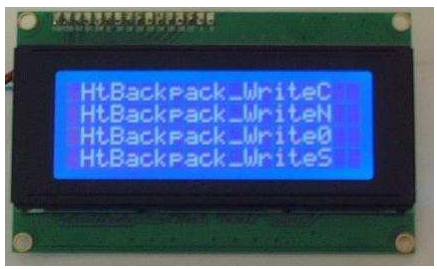
As we are using *PiSerial* subroutine to do our dirty work for us and it uses R1 to write the baud rate, or read the baud rate if R1 = -1, so I we use the same so we don't muddy the waters.

```
12030CMN R1,#1  ; Is it -1?
12040BEQ read_baud
```

If R1 = -1 we jump to *read_baud* which we'll discuss later.

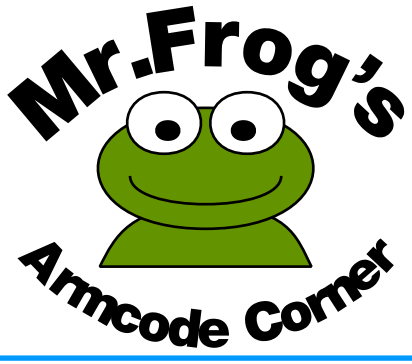
The next two instructions test for division by zero in case someone tries to set 0 baud.

```
12050CMP R1,#0  ; Is it zero?
12060BEQ zero_division
```



There is another check in the read baud rate routine just in case someone has managed to set the divisors to zero.

If you try to divide some number by zero by continually subtracting the zero from the



Greetings, Frogfans. I've just finished my lunch of BLT sandwiches. Which is funny because BLT is also an Armcode code instruction. Just how *do* you pronounce your machine code? Is it "Be eL Tee", or textspeak sounding like "Bullet", or even the full "Branch if Less Than"?

The species of *homo sapiens* who program computers aren't really the talkative type so the answer is that there is no right answer. For beginners, though, it's best to speak each letter, unless it's an obvious English word like AND or ADD.

Last time we wrote a simple program to print "Hello Pond!" on the screen and learned that you must reserve a block of memory to assemble the machine code into, the square brackets start and stop the assembly code, and to actually invoke the code you have to CALL it.

We also added some code to print the message multiple times:

```
  85.loop
 101ADD R1,R1,#1
 102CMP R1,#10
```

which didn't work because we never put the instruction in to go back (or 'branch' as we say in the trade) to the printing routine:

```
103 BNE loop
```

Line 101 ADDs one to Reg 1, line 102 CoMPares Reg 1 with the value 10 and BNE ("Be eN Ee") tells the computer to Branch if Not Equal – *Branch* if Reg 1 is *Not Equal* to 10.

There are other branch instructions and they all begin, unsurprisingly, with B! What would happen if you put BLT in line 103 instead of BNE? Try it and see if there is any difference.

Now, line 110 MOVes the contents of Reg 14 to Reg 15. These registers are also called the Program Counter (PC) and Link Register (LR). MOV PC,R14 can also be written as

```
110 MOV PC,LR
```

We covered the PC last time but what about this LR? Every time machine code is CALLed (from Basic or perhaps another machine code program elsewhere in memory) the computer records the calling address in the LR. So when your mega fantastic machine code routine has finished, a MOV PC,LR restores the address to the PC and the computer knows where to carry on from where it left off.

Now, traditional assembly code has a jump instruction (JSR) and corresponding RTS (return from subroutine) allowing subroutines to be called from other subroutines and so on. You can have as many of these as the computer's *stack* allows.

In Armcode, if it's only one subroutine we can use the special Branch-with-Link (BL).

```
BL subl
...
.subl
...
MOV PC,R14
```

But you wouldn't be able to call another subroutine from *sub1*. Why not? Another handicap of Armcode? Have a think for next time. Hint: read up on *stacks* and you might realise there is a way round this problem.