www.sean.co.uk
UK freelance writer Sean McManus

Downgrading

Sean McManus explains how to convert programs between the CPC6128 and CPC464

As the public domain thrives, many programs are being developed on the 6128 and being distributed virtually freely. (Remember, this was written in 1993!) Some 464 users are being denied access to some of this well of new software, due to its incompatability with earlier Amstrads. Although many of these programs will only run on a 6128 without adjustment, the updates are often relatively easy to carry out.

Exceptions to the rule are obvious. If the program requires a disc drive and you don't have one, no amount of amendment short of obtaining one is going to work the magic. A few spells can convert some programs to tape usage though, usually those that do not require multi-loading nor constant data access.

The CAT command on the 6128 displays the disc's directory. On the 464 this same command will keep playing a tape until [ESC] is pressed, probably halting the program's execution. It's a good idea, then, to remove

Quick lookup:

ON BREAK CONT
PLOT, PLOTR

any CAT commands lurking in a program you wish to use on a tape based computer. The disc commands preceded by a bar (|A, |B, |TAPE, |DISC, |USER, |REN, |ERA) will all need to be removed and this will occasionally bring about adverse affects. If the routines that use these commands are optional features in the program (e. g. pressing a certain key for a directory of datafiles), don't use them or better still seal them off by beginning their subroutine with a RETURN command.

Disc files also have a three character extension, which identifies their datatype. For example ". BIN" flags binary files and ". BAS" indicates basic programs. If omitted, a disc machine will supply the relevant extension in default. On a tape machine, the extension is simply part of the filename and so "DATA. BIN" and "DATA" are two completely separate files, which would nevertheless behave the same on disc machines. If the program is inconsistent in its inclusion of extensions, chaos will follow. The best idea is to remove extensions where possible, or to try to supply them yourself where a filename is repeated with different extensions. Tape filenames are substantially longer than disc ones, so disc users may find that they have to modify 464 code to allow for this.

Disc errors are flagged by the system variable DERR. Any commands featuring it should be deleted. If this causes problems try replacing DERR with ERR. If the routine is ever used, the program ought to be going astray already. It will, however, complicate debugging and lead to misleading error messages in the event of a crash.

Another hardware difference is the extra 64k which the 6128 has. Unless your machine has an expansion ROM, these programs will not work on the 464 at all without either a ROM or complete rewriting (in the unlikely event that they could fit if they were compressed). Commands worth scanning for are OUT &7F00, which are used to page in extra memory. Any reference to the bank manager program in the instructions, or use of the RSXs SCREENSWAP, SCREENCOPY, BANKOPEN, BANKREAD, BANKWRITE or BANKFIND are giveaways that the software expects more legroom than 64k.

Updates to Basic

Version 1.1 of Basic as featured on the later machines differs from the original in several ways. As far as sound is concerned, the volume levels run from 0 to 15 on the 6128, whereas the 464's volumes run from 0-7 and start at silent again for 8. To rectify this, the fourth number in any sound command (if it is present) should be halved and rounded sensibly on the 464. 464 volumes need to be doubled for 6128 listeners.

464 Basic was also particular about where you placed your DATA commands: Anywhere bar the end of the line was forbidden. Any DATA commands in the middle or start of lines on the 6128 should therefore be moved to the end of the same line. It's just a case of juggling the order of commands. For example, on the 6128 the following is valid:

```
10 DATA 1,2,3,4:CLS:PRINT "Hello retro gaming freaks!"
```

But a 464 version would have to have the DATA command (from DATA up until the next colon) shifted to the end. All the other commands are kept in the same order, as this conversion shows:

```
10 CLS:PRINT "Hello retro gaming freaks!":DATA 1,2,3,4
```

RSX's, new basic commands written in machine code, also behave differently between the two machines. As mentioned in section one, you should never place a bar character in a REM statement, because it sometimes makes the program behave erratically. Passing parameters differs between the two machines. On the 464, the @ character had to be placed before string variables, a formality dropped for the later Basic versions. The later Basic could also cope with data being passed directly to a command, instead of via variables. The solution in the first case is to always place an @ before string variable names in RSXs and secondly to create dummy variables where necessary.

```
|MESSAGE,"Text"
```

should therefore become:

```
dummy$="Text":|MESSAGE,@dummy$
```

Since variables cannot be command words, 6128 users might have problems with 464 programs which include variables such as FRAME which have since become keywords. The best way to get around this is to just add a letter consistently to the end of each occurance of such a variable, so that it becomes FRAMEx for example. This will be valid.

Some machine code programs are not compatable across the range and this applies as much to early 464 games as to later 6128 creations. Converting these requires quite a deep knowledge of machine code and is beyond this discussion's limits. If a Basic program includes machine code which uses specific routines, it could likewise fail, but such listings are in the minority.

Basic 1.1 boasts some new commands, previously unreleased on the 464. These can, on the whole, be circumvented using short routines as we shall now see. Some of the old favourites have also been slightly modified for Basic 1.1. The rest of this chapter details the changes made to Basic commands. There is a command index at the top of the page.

CLEAR INPUT

This command exists on the later machines to empty the keyboard buffer, so that any keys pressed since last checking it (e.g. with an INKEY$ command) are forgotten. It can be replaced with CALL &BB03 on the 464. It is particularly useful before any input that follows a delay of any sort, to ensure that keys tapped inbetween do not clog up the next data input. Some applications lend themselves more readily to allowing type-ahead, such as some text adventures.

Back to top

COPYCHR$

One of the most useful additions, this command will return the ascii value of the text character where the cursor is on screen. If you are using user defined symbols for graphics, this command is brilliant for collision detection. This short routine, which can be relocated in memory if necessary by changing the value of addr in line 20, will emulate COPYCHR$.

```
10 DATA dd,7e,00,cd,b4,bb,f5,cd,60,bb,32,00,00,f1,c3,b4,bb
20 addr=40000:FOR g=0 to 16:READ a$:POKE addr+g,VAL("&"+a$):NEXT
```

This program will place the returned ascii value into memory location zero. It is used in the following manner:

```
a=COPYCHR$(#5)
```

after running the above routine, becomes

```
CALL 40000,5:a=PEEK(0)
```

Back to top

DEC$

This command formats a number in the way specified for use in strings. It can be replaced by STR$ and when display is required, by PRINT USING. The routine containing DEC$ could be separated from any routine that displays it on screen, so it could take some scouring of the code to find where the PRINT USING should go. This example demonstrates:

```
PRINT DEC$(11^5,"######,. ##") [6128]
PRINT USING("######,. ##");STR$(11^5) [464]
```

Back to top

DEFSTR

DEFINT and DEFREAL allow a variable to be defined as an integer or a real variable without the need to use the % or ! marker throughout the program after the variable name. DEFSTR is a newer cousin which allows string variables to drop the dollar sign. This makes a="a string" perfectly legitimate, if a little confusing for most programmers of the old school. This command has to be removed and the variables it names need to be noted, so that they can have their string flags (the dollar sign) returned to them throughout the program.

Back to top

DRAW, DRAWR, PLOT, PLOTR

All the plotting commands have an extra parameter tagged on the end in Basic 1.1, which sets the graphics plot mode. Thus DRAW, DRAWR, PLOT and PLOTR can all take the parameters x, y, pen colour and (6128 only) graphics plot mode. The graphics plot mode controls how the background at each point influences the colour plotted on screen. Usually, it doesn't influence it at all. As this book will later explore, though, it is possible to make the background modify the colour plotted using certain logic functions. Control codes are used to emulate this effect on the 464: a combination of CHR$(23) and the CHR$ of the parameter supplied as shown here:

```
DRAW 320,400,1,4
```

becomes, on the 464

```
PRINT CHR$(23)CHR$(4):DRAW 320,400,1
```

You can find out how to use this feature in 'A character building experience'

Back to top

FILL

The best way to handle this on the 464 is to remove the command. Its effect is to fill the shape surrounding the graphics cursor and enclosed by solid lines with ink of the colour specified. There is a program on the disc to emulate this, which is used by replacing FILL x with CALL 40000,x. The title screen on the disc, incidentally, has the background letters filled in on the 6128 and hollow on the 464. The way it does this is by setting the machine to jump to just after the fill routine ON ERROR, where it continues as if nothing had happened. The error concerned would be caused by FILL bringing about a Syntax Error on a 464.

Screenshot of my CPC464 FILL program

Back to top

FRAME

Computer monitors, like television screens, draw images by a single beam moving down the screen in a zig zag pattern. FRAME will wait until the beam is at its start position and is used to make animation smoother, by eliminating the flicker encountered when a design is written to the screen half way through a scan. In fact, graphics are so slow in Basic that the difference is

virtually never noticeable. Nevertheless, this command can be simulated by CALL &BD19

Back to top

GRAPHICS PAPER

This will set the graphics paper colour, i.e. the colour that the screen clears to when a CLG command is issued. The following short routine, which can be located anywhere in memory to fit around other routines, will emulate this. After it has been run, the graphics pen can be changed to colour i by issuing a CALL 40000,i (changing the 40000 for whatever address you relocate it to where necessary).

```
10 DATA dd,7e,00,c3,e4,bb
20 addr=40000
30 FOR g=0 TO 5:READ a$:POKE addr+g,VAL("&"+a$):NEXT
```

Back to top

GRAPHICS PEN

To change the colour of the graphics pen without plotting a point, the routine used for GRAPHICS PAPER above is amended. The e4 in the data line is changed to de. If you need to use both routines at the same time, remember to change the address value of one of them so that the second one does not replace the first one in memory.

Back to top

MOVE

Move is a graphics command that can, on the 6128, take two new parameters. One is the graphics plot mode and the other is the colour the pen is to be set

to. More than a little illogical ("move cursor to point x,y and don't plot a point in this colour") this is emulated using a combination of the plot mode patch for DRAW and the GRAPHICS PEN emulator. After the usual x and y co-ordinates, the next parameter sets the graphics pen and the final parameter is the plot mode.

Back to top

ON BREAK CONT

This command is easily replicated by creating a dummy routine to ON BREAK GOSUB and making its sole function to RETURN whence it came. If this command is found in type-ins, it could lock you out, so ensure that you save the program first.

```
10 ON BREAK GOSUB 9091
20 REM The rest of the program is here.
30 REM This demonstrates how to emulate ON BREAK CONT
9090 END: REM END=Always a good idea before the subroutines
9091 RETURN
```

Other ways to disable the ESCAPE key are discussed in The Protection Racket section on protecting programs.

Back to top

PLOT, PLOTR

See DRAW.

Most PD programs can be converted to work in one form or another across the range. Previously incompatability might have driven you to aimlessly kick your machine. Now you can try for the conversion.

Credits

Visit www.sean.co.uk for free chapters from Sean's coding books (including Mission Python, Scratch Programming in Easy Steps and Coder Academy) and more!