68K/OS

Chris Bidmead examines a multi-tasking OS for the 68000 cpu, now available on the Sinclair QL

68K/OS was originally commissioned by Sinclair as the production operating system for the QL, and when the Sinclair machine was launched in January last year it was 68K/os that was shown to the press. At that stage QDOS, which was supplied with the machine to paying customers, was something Sinclair began getting together in-house when it became clear that 68K/OS was not going to leave room for Superbasic. ironically, QDOS also got too big, and the first production machines were sent out with the notorius extra protruding ROM pack.

GST's product is now marketed as the alternative operating system for the QL. It is aimed at the advanced home user who wants to do a lot of machine-code hacking and write personalised utilities in high- level languages when they come on stream later. For the student of computer science, too, GST's 68K/OS will provide a low- cost experimentation system to support course work. It is also designed for the small independent software supplier who needs a cheap software development system.

Five different screen modes are available from the opening menu.


The IOSS interface menu.

Bare boards

GST has an agreement with Sinclair to sell the QL processor board — without case or keyboard — in conjunction with its operating system for OEMs who want to bring low-cost 68000 machines to the market. The operating system is also being evaluated by manufacturers of other 68000-based systems.

Transfer outside the realm of the QL is implicit in the design of the product. Like CP/M, it gathers all its machine-dependent aspects into a single module, leaving the bulk of the OS machine-independent and portable. GST claims to be able to port the system across to any other 68000 device within two to three months.

68K/OS is a single-user multi-tasking OS inspired by the Unix kernel. The main advantage over QDOS, apart from its device independence, is the ease with which windowing software can be written, as the operating system comes complete with an outer wrapping along the lines of Microsoft Windows.

The early evaluation kit came on a pair of 16K ROMs and required some fairly in-depth interior reconstruction to install on the standard QL. Purchasers of the production version of 68K/OS do not have to go through these traumas: conversion is simply a matter of slipping a small pcb into the expansion slot on the left-hand side of the machine. A set of utility programs on Microdrive is also supplied — see box.

The best news is that you do not lose QDOS. A switch on the board protrudes discreetly from beneath the QL's left edge, and allows you to revert to the Sinclair operating system at a touch.

On power-up 68K/OS signs on and displays options for five different screen modes, depending on whether you have a TV or dedicated monitor, and offers a 40- 42- 60- 80- or 85-character wide screen. The choice at this point affects more than the size of the characters, because the screen layout is adjusted accordingly.

On the Hitachi television we used initially we found the 60-column display the easiest to read. The 'Getting Started'' part of the GST manual was biased towards the 80-column display, so the layouts were slightly different, but it was not difficult to see what was going on.

After your selection of the screen mode the system loads Adam from the ROM. Adam is the user interface to the operating system, the equivalent of CP/M's CCP or MS-DOS's Command.Com. Using the operating system's Menu Manager it divides the screen into a number of different display and data areas, the precise layout depending on the capabilties of your monitor or TV. Adam makes use of the five QL function keys to select the basic functions — see box on page 82.

All code that runs under 68K/OS is reentrant and position independent. As its name implies, position-independent code contaias no absolute addresses, and can therefore be executed at any point in memory, which is essential when you are trying to run several independent processes simultaneously. Re-entrant code is code that does not alter itself as it runs, so that the identical routine can simultaneously form part of several processes.

Adam can run itself just like any other program; an illustrative exercise, even before mounting any Microdrives, is to do just that. A second copy of the Adam layout appears on the screen, overlaying the first but leaving the top line of the original Adam screen visible, like a pair of stacked card-index cards. You can repeat the process several times, building up a pile of Adam images.

Like Unix, 68K/OS sets independent default data directories and program directories, and the Adam screen provides a separate window for each. If you have chosen the TV type of display the two directory screens replace each other as you toggle with the F3 button. With 80 or more characters to the screen line there is room for the two screens to appear simultaneously side by side. In this mode the same F3 key swaps the cursor between the two areas.

Mounting

Also Unix-like is the idea of mounting devices. Maths purists will be delighted that under 68K/OS the drives are known as 0 and 1, but this will be yet another source of confusion to QL users who have come to know their drives as 1 and 2. The only time you really need the unit numbers is during the act of mounting.

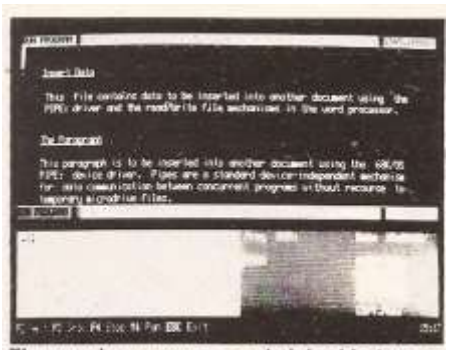
The full physical device names are MDO: and MD1:, to allow for a variety of non-Microdrive devices, but individual cartridges also have logical names. Mount a cartridge called Mydata on MD1: and the directory will henceforth be known as MD:Mydata. When you have finished with Mydata you dismount it to tell the system it is no longer available. Adam provides options to perform this mounting and dismounting explicitly, but normally they are never used. The operations take place automatically as a by-product of setting the program or data default.

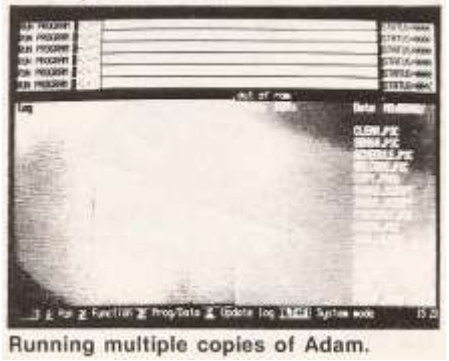
The mounting concept makes it easy to extend the operating system to include new devices — such as floppies, hard discs, etc. — as they come along, but is rather cumbersome when all you are dealing with is a pair of Microdrives. Once program and data cartridge have been mounted, Adam is no longer available by default. You can still get to it by giving its full path name of
<logical device>:<filename>
in this case
ROM: ADAM

Alternatively you can change the default program directory to ROM:, access Adam, then reset the default directory to what it was before, giving its logical name. This happens quickly because Adam keeps the full directory of each mounted logical device. You can, of course, reset the directory by giving the physical name, but this will evoke an implied mount, re-reading the directory from the tape.
Adam has a third screen area, which is used to log session activity. Here the names of the various programs used during the session are allowed to accumulate, together with a note of what became of them.
At this stage we noticed that the TV set was incapable of displaying the row of tiny dots in the left-hand corner that are supposed to give an indication of processor activity. We didn't miss that very much until we came to run the time and data utility, which replaces these dots with a rather more useful digital clock. No doubt the television could have been adjusted to bring this into view, but on the Hitachi the internal control was hard to get at, and we were in no mood to fool with high voltages. As luck would have it, at this point a very handsome black monitor specially designed for the QL arrived from Microvitec. Now we were able to switch to 85 characters per line, and see a full screen of crisp, easily readable characters.
There is one crucial difference between 68K/OS and a single-tasking system like CP/M. All the components of CP/M are synchronous. 68K/OS, on the other hand, also supports asynchronous processes that take place in their own time, almost as if they were running on separate processors. One practical implication of this to the user is the way 68K/OS appears to be able to read and write to the Microdrives without pausing in its service of an application program.
Aside from these asynchronous processes, the interface between the operating system and applications running under it divides into four categories of functions, each category having its own entry point. Specific functions within the categories are differentiated in the conventional way by loading a different function number into the DO register.
One entry point, called the SP vector, is reserved for direct assaults on the hardware-dependent system primitives, notably the graphics routines. Software is provided to draw points, lines, blocks and conic sections, but these functions are strictly specific to the QL and so should be avoided by applications that aspire to be portable.
Another entry point, the input/output subsystem or lOSS, creates a connection to device drivers, which are either standard to the operating system or supplied by the user. Device drivers can be interrupt driven, polled or run as asynchronous programs, but the interface with the lOSS ensures that whatever the level of complexity of the device driver, it simply appears as a subroutine to the rest of the operating system.

Limitations

Unfortunately hardware limitations mean that it still has not been possible to make the resident driver for the two RS-232 devices work entirely sensibly. You still have to put up with not being able to set separate baud rates for send and receive, so Prestel-type comms requires expensive additional hardware. If you want different baud rates on each line you have to confine communication to transmitting, and give up any idea of receiving; there can be no soft handshaking, for example. These hardware shortcomings are, frankly, tatty but QL users should console themselves with the thought that they could spend over twice as much on a Tandy 1000 and get no RS-232 at all.
A third entry point, the DFM vector, takes care of all the display-file manager calls. Display files are sets of linked lists retained in RAM that maintain the contents of windows, allowing them to scroll or be overlaid without loss of data.


The word processor sandwiched between two copies of Adam.

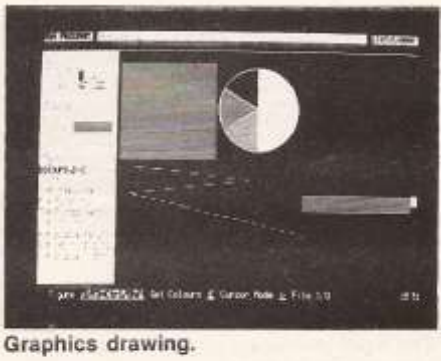

Running multiple copies of Adam.

The snag with windowing under Super-Basic is that if one window overlaps another the contents of the earlier window are destroyed. This can only be avoided by writing additional code that takes responsibility for checking for any damage done by overlapping windows and repairing it as necessary. Separate tasks therefore have to know what the others are up to. Display files, on the other hand, result in windows that can coexist, and even occupy the same physical location, without being mutually destructive. And you can actually scroll text through the windows without losing any of it.
The final entry point, the OS vector, is used to access the remainder of the hardware-independent functions like memory management, heap allocation and multi- process handling. Of these the most novel is the menu manager, a collection of functions designed to make it easy for applications programs to provide a consistent user interface. It draws heavily on the window functions to create two kinds of special window: the menu window and the list-selection window.
The menu window writes a set of indestructible prompts and reads data keyed into input fields, which it also provides. Variable messages can be displayed, to provide context-sensitive help, for instance. The second kind of window displays a scrollable list of items, from which the user can make selections for transfer to the menu window.
Functions like memory management are, of course, crucial to an operating system. But you might be forgiven for asking whether the more exotic functions of windowing and menu management. should not rather be supplied as library routines, to be appended to application programs as needed. Why have them permanently taking up precious operating-system space? The ability to delete and rename files is conventionally a resident part of the operating system, yet CP/M and MS-DOS users coming to 68K/OS will be surprised to find they have to be loaded from the Microdrives as transient programs.
On the QL there may be some argument for this inversion. Because scrollable text- window routines are already part of the operating system and reside in ROM, an application package like a word processor does not have to pull all that code into RAM every time you load it. The advantage of an operating system that takes care of things like windowing becomes clear when you power up WP.Prog, the pre-release copy of the word processor GST plans to release to run under 68K/Os. It loads far faster than the Psion equivalent, as it only contains about 16K of code — though it is only fair to point out that the GST offering has fewer functions than Psion's Quill. This is partly because the tricky business of displaying and scrolling text through a window, which forms a large part of any word processor, is taken care of by the operating system. When it comes, the full word processor is destined to occupy a 16K EPROM for mounting on one of the spare slots on the 68K/OS plug-in board.

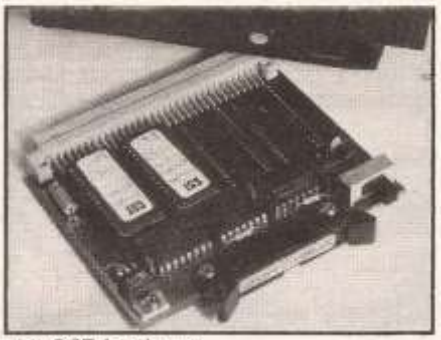
Out of hand

The windowing can get out of hand, as we discovered when trying to save text to

a full microcassette from inside the GST word processor. An error window opens towards the top of the screen, inviting you to make more disc space. Thanks to multitasking you can do this by returning to Adam, expanding the Adam screen until you can see all the data directory, and then evoking Delete.Prog.
This utility also opens a window of its own to ask you to confirm the delete, and rather unnecessarily takes half the screen to do it. When this window closes again you expect to see the lower half of the screen restored. But in fact half the word processor screen popped up here. More curiously still, if you try to repeat the save and have not deleted enough files, the Disk Full warning pops up underneath the top half of the Adam screen, so you do not see it until you hop across to shrink the remains of the Adam screen.
The word processor is still in its prerelease stage, so it is not fair to judge it. But it does seem to indicate that the operating-system window manager — presumably now a mature part of the product — rather curiously allows an application package to think it is showing you a window, while another process of a lower priority is blocking the view.
This serves as a reminder that, as it stands, the operating system is not designed for fools and is not foolproof. I found, for example, that it is possible to corrupt the directory relatively easily. Log a cartridge called, say, Progsa on to md1:
and load a program from it into memory. Swap the cartridge in md1: for a second cartridge — let's call it Progsb — and try setting it as the default program device. The operating system will tell you that you cannot do this, as the directory — of Progsa — is already in use, signposting the activity of the process we have loaded. You might be inclined, by using Alt-Fl, to kill the process that is causing the trouble. Don't! If you do, the system will write the directory back to drive mdl:, assuming that Progsa is still there. As it isn't Progsb will receive the Progsa ditectory, lose its own directory and consequently all hope of finding its own files again.
Attention to the details of true multitasking makes the addition of 68K/OS a great improvement on the standard QL. But it leaves you without any applications software, or even high-level languages to develop your own. An assembler is supplied as an extra, but one of the attractions of the 68000 generation of chips is supposed to be that we can say goodbye to assembler writing forever. Even regarded as an assembler system, there is one very important utility missing — a debugger. Like the high-level languages, this is promised for the future. But until it arrives the system will require a lot of patience from its users, significantly, G5T has been developing what utilities there are under the Motorola Exormacs system and not under 68K/OS.
What we missed most in the new environment was SuperBasic, one of the best things about the QL as bought in the shops. GST hopes to offer a Basic of its own soon, and Fortran and Pascal are rumoured to be on the way.
Swapping programs and data with QDOS is also not a solution. The two operating systems use quite different formatting, and cannot read each other's cartridges.



Graphics drawing.



The GST hardware.

Conclusions

•68K/OS is an ambitious, fully fledged multi-tasking operating system for the QL and other 68000-based machines. It comes with excellent documentation on an easy- to-fit expansion board that allows you to revert to QDOS at the touch of a switch.
• The dearth of languages makes 68K/OS something of an anomaly — a development system for what is supposed to be a high-level language chip, with the only language being ASM.
•68K/OS costs £99.95, which includes a brief Users' Manual. Purchasers of the system are going to want to go into it all a lot more deeply, and won't mind forking out £4.95 for the highly detailed and well- written System Programmers' Manual.
•Unlike CP/M, 68K/OS does not include the assembler as a standard operating-system utility, and it will cost you another £39.95.

Utilities on Microdrive

Format.prg – A program to format a new microdrive cartridge or wipe an existing cartridge.

Copy.prog – Copies program and data files one by one between cartridges.
Time.prog – Sets system date and time.
Dump.prog – Dumps a screen image to an Epson FX-80.
Edit.prog – An easy to use screen editor with a help screen; a real joy comared with Quill, and a lot faster.
Draw.prog – Lets you create drawings on the screen using a menu and two sets of hair-line cursors used for positining rectangles, circles, ellipses, squares, lines, pixels. Text can be added and areas can be filled with various colours. Your keystrokes can be saved to a file which can be called up later. Not a serious utility – it is intended as a demonstration of how application programs can tap into the OS facilities.
Slides.prog – Allows you to display serially a set of pictures created by Draw.prog. Fun for demos.
Print.prog – a program to print out a file. This is where you begint o mi9ss the simplicity of CP/M with its built in Type command/
Rename.prog – Renames files. Here again, you need a separate program to do what other operating systems take charge f themselves.
IOSSMenu.prog – Lets you experiment with the input/output sub-system and also serves as a demonstration of the menu-handling abilities of the system.
Fount.prog – An under-documented teser for use with Edit and some pre-defined founts. There are no details about how to create your own founts.

Function-key assignments

Fl-Selects the Run mode. The program directory can be scanned using the cursor keys, and a program selected by pressing the Escape key when the cursor is against the correct directory entry. This copies the name into the command-line window. Alternatively the name of the program to run can be typed in the normal way.
F2 — Selects the function mode. This is used for setting up the default program and data drives or devices, or explicitly mounting or dismounting them.
F3 — Swaps the directory selection between program and data.
F4 — Updates the log and directory screens something which is not done automatically at the termination of each program.
Alt-F1 — takes you straight out of whatever you are doing and into the System mode. From here you can skip about between the various jobs currently running, expand or contract the window area on a particular display, suspend that job, restart it, kill it, or bring a completely new task to life. You can also perform a warm boot into Adam.