# QUINKEY FINGERS

Peter Voke lays a hand on the new Microwriter add-on for the Beeb

A SURPRISING new hardware add-on for the BBC micro has appeared. The Quinkey is a one-handed keyboard produced by Microwriter that provides a novel way of accessing the computer. It costs a little under £50, including VAT.

The Microwriter, from which the Quinkey is derived, was invented by a film director, and is reputed to have been used to write the film *Zulu Dawn*. That may not strike everyone as much of a recommendation, but the Microwriter has survived, and flourishes. Basically a portable text-recording device, in the Quinkey it has found a new and potentially revolutionary application as a plug-in keyboard for the Beeb. Versions are likely to follow for the Electron, the Spectrum, and the Commodore 64.

What is special about the Quinkey? First, in place of the conventional typewriter keyboard there's a specially shaped block about the size of a paperback book, with just six keys on it. Five of these six keys lie naturally under the fingers and thumb of the right hand, while the sixth is reached by a slight downward movement of the thumb. On starting to use the Quinkey the naturalness of the position of one's hand compared to a conventional keyboard is immediately apparent – and one can sit back in an easy chair with the Quinkey on one knee, rather than having to hunch over the keyboard in the usual hacker's style. Microwriter has been sensible enough to give the Quinkey a long lead.

So, one up for the Quinkey: no more backache. The next test is whether it does what it is supposed to do. The letters of the alphabet are entered by pressing combinations of fingers on the right hand, as a pianist plays chords, rather than one key at a time. This might sound more complicated than typing, but you quickly discover that the claims of Microwriter are no exaggeration: the finger combinations have been chosen to seem so natural that it takes only a few minutes – literally – to remember all the shapes for a to z. I didn't believe this claim until I tried it myself, but only needed to go back to the manual once to check a few letters a second time.

To cope with the BBC control codes (things like CTRL-U and CTRL-N) the Microwriter scheme has been extended in a simple way, and one presses a special 'shape' with one's fingers (COM-X) followed by the letter concerned. Yet another special shape (COM-H) allows the Quinkey to mimic the red function keys on the BBC micro, and a few non-standard characters such as # and >. It might seem impossible to remember all the finger combinations involved, but once the basic 26 letters and five common punctuation marks are learned, the rest comes fairly naturally.

Microwriter's Quinkey, a 'complete alternative keyboard' for the BBC micro at £49.95

Microwriter claims that people take to the Quinkey-style keyboard much quicker than to a qwerty typewriter layout. A typing speed of 25 words per minute is reached faster, though it is difficult to get beyond 40 or 45 words per minute because of the one-handed nature of the action. This is likely to bother only the superfast professional typist, who can get up to 90 words per minute on a qwerty keyboard, and is not really a limitation for the micro user. Having used a Quinkey for a couple of hours, I am convinced Microwriter is correct.

The Quinkey arrives from Vector Marketing in a good solid polystyrene pack, with a separate interface pack. The latter contains a user's guide that is exceptionally clear and complete, with a useful technical section at the back. (One or two small points: the use of the thumb key alone for the space character is not dealt with properly along with the other main letters, and numeric-X actually produces an asterisk, not a letter X.)

In addition you get a small adapter, which plugs into the back of the microcomputer, allowing four Quinkeys to access the computer simultaneously. I had only one Quinkey and so cannot comment on the practicality of up to four people using the computer simultaneously.

To use the Quinkey you obviously have to program the Beeb to accept its input, and Microwriter provides two programs on tape to do this. One, called *Prog*, is for ordinary use in programming, or in programs designed to use the Quinkey input, and the other called *WP*, is for use with wordprocessors. The Quinkey works with either *Wordwise* or *View*.

In addition, the tape has a tutor program which some people may like to use, although it isn't really needed to learn to use the Quinkey.

When the *Prog* is run, it relocates some machine code of length &300 bytes to just above PAGE, and shifts PAGE up by &300. You now have a situation rather like that produced by an extra ROM that uses private workspace. The three pages you have lost are occupied in this case by the software that reads the Quinkey keyboard.

What happens is that the 100Hz interrupt from the system VIA is intercepted through IRQ2V and redirected to the Quinkey software. This technical mumbo-jumbo boils down to some-

**PRICES**

**Basic:** Interface, keyboard and two software packs, £49.95 (each keyboard £29.95)

**Education:** Interface, four keyboards and multi-user software, £148.80

thing very simple: the BBC micro checks the analogue input a hundred times a second to see if you are doing anything with the Quinkey keyboard. If you are, it updates its 'current key press' and then, when you release the keys completely, puts the character into the input buffer.

One thing to note: the character is input when you release the keys, not before. This is slightly disconcerting at first, but makes sense, since it means that the keys you want pressed don't all have to all go down or spring up together. If they did, using a Quinkey or Microwriter would be impossible.

Taking a look at the machine code that runs the Quinkey, it turns out to be fairly simple but efficient. No workspace is used outside the 3/4k of Quinkey private workspace. You can adjust the action of the Quinkey in a number of ways, simply by knowing a little about what is going on in those three pages. So, for the benefit of present and future Quinkey owners, here are some hints.

First, the characters output by the Quinkey are held in lookup tables, 32 at a time. If *Prog* is loaded at &1900 (set PAGE=&1900 and LOAD 'PROG'), the data tables for all the characters lie between &1C4C and &1D0A. Locations &1C4C to &1C6B contain the ordinary lower-case letters plus the five common punctuation marks produced by the unshifted Quinkey. The Quinkey 'command' codes are in memory from &1C6C to &1C8B (though it is noticeable that the second 16 bytes are all zero, meaning that key presses involving both the thumb keys produce nothing). From &1C8C to &1CAB the capitals are stored, from &1CAC to &1CCB the Quinkey 'numeric shift' characters, and from &1CCC to &1CEB the 'special shift' characters introduced to cope with the BBC function keys and non-standard characters. The BBC control codes (ASCII 01 to 26) follow from &1CEC to &1D0B.

Of course, the characters are not in alphabetical order in these blocks. The position of the characters in the tables determines which character appears on the screen for a particular combination of keys pressed. The five fingers of the right hand each trigger one bit of a five-bit number. For instance, the Quinkey finger shape for the letter 'j' is thumb, ring finger and little finger. The five bit number is therefore 10011, or 19. The Quinkey produces a 'j' in response to this because a 'j' is at position 19 in the table beginning at &1C4C. The little finger alone is the letter 'u', and 'u' is the first letter in the table, as we expect. The whole table from &1C4C contains:

usgoqnbevt,a-.m hkjczyxilrwd'fp

in the form of ASCII codes.

This should allow you to reprogram the Quinkey without difficulty. For instance, to swap the Quinkey finger presses for 'u' and 's', you just swap their positions in the table, putting the ASCII code for 's' (&73) into &1C4C, and the code for 'u' (&75) into &1C4D. It makes more sense to do something useful, like putting the SHIFT+function key codes into unused spaces in the tables: in my personalised version of *Prog* I have put &92, &93, &95 and &96 into locations &1C7C, &1C7D, &1C7F and &1C83 respectively. The Quinkey then produces the teletext colour characters for green, yellow, magenta and cyan when two thumb keys are pressed with one other key.

**Four Quinkeys can access the Beeb simultaneously**

One major way in which the Quinkey differs from the ordinary keyboard is in its handling of autorepeat. Because a character is put into the buffer only when the finger keys are released, no autorepeat takes place when the keys are held down. (If you are not sure what 'autorepeat' means compare this with the action of the BBC micro when you put your finger on a key and keep it there.)

A simulation of autorepeat has been put into the Quinkey, however, which works like this: if, after entering a character, all the keys are pressed including both thumb keys, the character last entered starts to autorepeat at about the normal default rate used on the main keyboard (5/100 second). This rate is not affected by *FX12 but can be altered by changing a byte in the Quinkey driver machine code at &1AAC. This is normally 5 (the second byte of LDA #5) but can be changed to 2 or 3 for a faster autorepeat.

The only keys that need to autorepeat on a regular basis are the four cursor keys, Copy, and perhaps Delete. I feel it would be nice if these keypresses, all of which are Quinkey 'command' shapes, could autorepeat just by holding them down for more than one-third of a second or so, at a controllable rate. Since the actual handling of the Quinkey input is done entirely by software, it is likely that improvements and enhancements like these (if such they are) will become available in due course.

One keyboard facility the Quinkey cannot mimic is the INKEY(-N) statement in Basic, or the equivalent OSBYTE call in machine code. These access the hardware connected to the keyboard rather than the input buffer, and cannot be intercepted through RDCHV: possibly they could be intercepted through BYTEV.

This is not a great problem so long as the Quinkey is considered a text input device rather than a games controller halfway between a keyboard and a joystick. However, I'm sufficiently enthusiastic about the Quinkey after a week of use to think that Microwriter can afford to set its sights as high as possible. It is conceivable, with advances in Very Large Scale Integration continuing, and tiny flat-screen TVs starting to make their appearance, that portable computers in ten years will be no larger than the Quinkey itself – and that is all you will need. If this prediction turns out to be true, the six-button keyboard is the interface of the future, and could be the death of qwerty sooner than anyone expects. Before that happens, we are likely to see Quinkeys linked to the computer by infra-red rather than wires, like the IBM PC keyboard.

Because the Quinkey has not been programmed to intercept INKEY with a negative argument, one popular feature of the BBC micro does not work from it. This is the use of the SHIFT key in scrolling listings in paged mode. Paged mode can still be switched on and off, since CTRL-N and CTRL-O are both characters the Quinkey can output. But once you are in paged mode, you have to press the SHIFT key on the keyboard to keep a listing going.

For a programmer (Basic or assembly language) this is a real disadvantage. The problem that the Quinkey is up against is that it really has to provide all the functions of the keyboard, without exception, if it is going to compete. It comes so close that it's a pity Microwriter didn't make sure of the paging facility at least.

You will probably expect me to finish by saying this article was typed using a Quinkey. It wasn't. I'm still a lot faster on the keyboard, and possibly always will be. Old habits are hard to kill. But I'm going to persevere with the Quinkey, if only to avoid developing the dreaded Hacker's Shoulder in my old age. And to be ready for the hand-held supermicros of the 1990s, of course.