WORKING NOTE AC108

Alan Creak
13 December 1996

SUBVERTING BIOS FOR THE COMPAQ ELITE

This is a record of information about the Compaq LTE Elite machine acquired while trying to find out how to catch the interrupts for every keypress, unsullied by the imposition of a rollover limit and automatic key repeat signals. I'm recording it not because it constitutes finished work or a major (or even minor) achievement, but to ensure that I don't lose it.

I want to be able to detect every key operation on the computer keyboard in order to investigate some novel uses of keyboards as interfaces for rehabilitation purposes[1], but I can't do that on my Compaq Elite LTE machine, because the keyboard signals are mangled on the way. Part of this is for the MS-DOS BIOS, but they seem to do some other things too. For example, it's impossible to turn off the automatic repeat.

In an effort to find out how to do better (and assuming, not necessarily with justification, that it would be possible to do so), I approached Compaq for advice. This note is a sort of record of what I found. I have tried to preserve the potentially useful material while weeding out the chaff, but don't guarantee to have been successful. Much of what follows is more or less directly copied from electronic mail conversations with Andrew Jury[2].

STATEMENT OF THE PROBLEM.

I want to write a device driver for the keyboard of my LTE Elite 4/40CX machine. To do so, I have to bypass the built-in BIOS, which I think is a ROM chip, because the built-in rollover and automatic character repeat is unacceptable for my purposes. I therefore want to know the technical specifications of the keyboard itself – in particular, what signals it produces, and how it communicates with the processor. That will at least start me off – but no one has been able to tell me how to find the information.

THE PROBLEM IN MORE DETAIL.

I want to get as much information as I can about what is actually happening at the keyboard. The immediate problem which led to my present difficulties has to do with people with disabilities who want to use keyboards. Many people have hands which are spastic or poorly controlled for one reason or another, so have difficulty in using a keyboard in a conventional way. Others have good control, but can only move very slowly. And so on. Quite often, they can get reasonably close to what they want to type, but have difficulty in doing so precisely – perhaps they can't help pressing several keys at once, or the slowness is a hindrance.

I am working on the principle that if people can't use a keyboard in the way that the system defines, then we should change how the keyboard works to fit the people. If I can find out just what the people are doing with the keyboard, I have a very good chance of working out what they were trying to do, or providing them with additional facilities. Specifically –

- if people inadvertently press many keys, then by reading the many keys I can work out where their hands are;

- if people type very slowly but accurately, then by allowing them to press several adjacent keys simultaneously (such as S+D, S+E+D, S+W+E+D, etc.) I can in effect provide a much enlarged keyboard, so that they can choose from many more alternatives at each key operation.

For BOTH of these, I want to be able to find out when the keys go down, and when they come up. INT 9 gives me that – but only up to the built-in rollover limit, which seems to be 2. Also, I'm stuck with the automatic repeat, which just adds to the confusion for people who might only be able to react very slowly anyway, so I want to switch that off. If I want automatic repeat, I can do it for myself.

The main difficulty is with people who can't easily press only one key at once. Instead, they press several keys more or less simultaneously, and what I want to do is collect ALL the make and break interrupts for all the keys they press so that I can work out what they meant. Because of the rollover, I can only see two pressed keys at a time, where I might want a much larger number. (There's no fixed maximum – I just want to collect all the keyboard interrupts that I can.)

I think that one of the reasons for the rollover limit is that once you get several keys depressed you start getting wrong results because of alternative paths through the keyboard circuitry. That doesn't bother me, provided that the results are consistent, because all I have to do is associate the complete keyboard pattern with the intended meaning, but if I can't see the complete pattern I'm lost.

Therefore : I would like to find out just what's happening at the keyboard, when it happens, with no decorations added by the BIOS. Ideally, I'd like to know just which keys go up and down when, but I know that's not always straightforward with some keyboard designs. That doesn't matter much, though, provided that the signature of keyboard outputs produced is good enough to identify what happened, even if it's wrong in detail.

I think, therefore, that I have to get round the BIOS and build my own keyboard device driver. Whence my difficulties.

(In fact, that's only the beginning. If I could catch all the keyboard events, I could do some other interesting things too, but the description above is how I got into the mess in the first place.)

FIRST STEP, AND A POSSIBLE SOLUTION.

In response to that statement, Andrew sent information about both the Elite and Deskpro machines. The Elite documentation, an excerpt from the Elite Technical Reference Guide[3], was to give the basic theory of operation, while the corresponding Deskpro excerpt[4] gave more information about the 8042 keyboard controller. By inspecting both of these carefully, I was able to work out how I might be able to solve my problem, but as I had combined information from the two manuals I was not sure whether my conclusions were sound.

1 The keyboard has a 24-byte scan code buffer. (Elite, p7-2.)

Excellent – what I want to do is possible in principle. I don't seem to have been able to beat the 2-key rollover so far, but perhaps that's because I've stuck with normal mode operation.

2 Select mode "allows any or all keys to be reassigned to make only, make/break, or Typematic operation". (Elite, p7-3.)

Also excellent – I want all keys to give make and break codes. But what do I do to reassign the keys ?

3 If I want select mode, I have to "select this mode via the F0h keyboard command". (Elite, p7-18.)

All right, but how ? The Elite material seems to give no further enlightenment. I don't think that MS-DOS makes any provision for output to the keyboard. But the L-model material is more helpful :

4 The 8042 interface ports are identified. (L-model, p8-8.)

The 8042 is memory-mapped at two addresses, 60h and 64h. Each of them has input and output variants, so there are four ports in all. To write F0h to the keyboard, I have to write F0h to port 60h. (Bottom of page 8-10.) But are these addresses the same for the Elite ?

5 To change the mode, I send F0h followed by an option byte to the keyboard. (L-model, page 8-49.)

So far as I can tell from the mode descriptions, I want mode 3. (Compare L-model p8-41 with Elite p7-18.) So I send F0h, then 3. But is it still 3 with the Elite, where it seems that there are only two modes ?

RESPONSE TO THE SUGGESTED SOLUTION.

Andrew's response to my comments follows; I haven't changed the spelling. It seems that I have plumbed the available knowledge of the Compaq corporation to the depths, and still the answer isn't there. Andrew's message begins with his comment, and continues with an excerpt from some manual – probably the Elite Technical Reference Guide, but I don't know – which continues to the ACKNOWLEDGMENT paragraph.

You asked several questions in your last mail. As you have identified the keyboard probably needs to be in the 'select' mode of operation to achieve the seperate make/break action desired. How to do this is not clear in the Elite TRG. It would appear that your need to write 'F0' to port 64h, to prepare for the option byte, and then send 03h to the same port to set the required mode. What is not clear is whether or not you have to use an INT 16h BIOS service routine to get this to work. The INT 16h table is given below:

Later -

I read through the reference material I had at home and the 'select' function definitely is not enabled using any ROM BIOS service routine. The book makes it look (as do the Compaq TRGs) that all you have to do is preceed the make-key scan code with 'F0'. But, I am certain it is not as easy as that. What about the keyboard processors typematic function?!

"Keyboard Functions

The BIOS generally controls all interactions with the keyboard. However, the interrupts and memory locations used for the keyboard make it very easy to change the keyboard functions. Table 2-14 lists the keyboard interrupts and memory locations used. Table 2-15 lists the memory locations used by the keyboard functions.

Table 2-14. Keyboard Interrupts

Interrupt	Type	Location	Bytes	Function
INT 09h	HW	0000:0024	4	IRQ1, Keyboard Interrupt
INT 16h	SW	0000:0058	4	Keyboard I/O determined by AH: 00h = Get key 01h = Check for key status 02h = Read shift status 03h = Set repeat key rate and delay 05h = Place scan code/character in type ahead buffer 10h = Get enhanced key from type ahead buffer 11h = Check for enhanced key in buffer 12h = Get enhanced key status F2h = Determine attached keyboard
INT 1Bh	SW	0000:0066	4	CTRL-BREAK service
INT 74h	HW	0000:01D0	4	IRQ12, Auxiliary input

Table 2-15. Other Memory Locations Affected By Keyboard Functions

Address	Bytes	Contents
0040:0015	1	Previous scan code
0040:0016	1	Key click loudness
0040:0017	2	Keyboard bit status
0040:0019	1	Accumulator for ALT key input
0040:001A	2	Keyboard buffer pointer head
0040:001C	2	Keyboard buffer pointer tail
0040:001E	32	Keyboard type-ahead buffer (16 entries)
0040:0071	1	Break Bit (bit <7>)
0040:0080	2	Keyboard buffer begin
0040:0082	2	Keyboard buffer end
0040:0096	1	Enhanced shift status
0040:0097	1	Keyboard LED flags

SYS REQ Key

The SYS REQ key is a special key. It is not encoded, nor is anything placed in the keyboard queue when it is pressed.

Pressing the SYS REQ key invokes INT 15h with AH = 85h, AL = 00h (SYS REQ Make code). Releasing the SYS REQ key invokes INT 15h with AH = 85h, AL = 01h (SYS REQ Break code).

The SYS REQ key does not interact with any other key and is not repeating. An application must trap INT 15h in order to make use of the SYS REQ key.

RAM location 0040:0018 stores the SYS REQ key status. If bit <2> in the status byte at 0040:0018 is set, the SYS REQ key is currently held down. The bit is cleared when the SYS REQ key is released.

Keyboard Indicators

The BIOS normally controls the state of the keyboard LED indicators. It automatically changes the state of the LED indicators to reflect the current status of CAPS LOCK, NUM LOCK, and SCROLL LOCK keyboard functions.

All communications to the keyboard occur through ports 60h and 64h of the keyboard controller.

To change the keyboard LED state, use the IN and OUT instructions of the processor to:

1. Read port 64h to determine the input/output status, making sure the input buffer is empty.
2. Write the disable keyboard (ADh) command to port 64h to disable the keyboard interface. Read the scan code from port 60h.
3. Wait until the controller input buffer is empty. Output EDh to the keyboard assembly using port 60h. Wait until an ACK (the first of two ACK bytes) is received from port 60h.
4. Write the LED data byte when the keyboard controller input buffer is empty. Wait until the second ACK byte is received.
5. When the controller buffer is empty, write the enable keyboard (AEh) command to the keyboard controller to reenable the keyboard interface.

Enhanced Keyboard

A RAM variable at 0040:0096 is used in conjunction with the Enhanced Keyboard for state information.

The format of RAM location 0040:0096 (byte) is defined below:

BIT FUNCTION

```
----------------
7       Read ID command in progress
6       Last character received was ID byte
5       If Enhanced Keyboard installed, force NUM LOCK
4       Enhanced Keyboard installed
3       Right ALT key down
2       Right CTRL key down
1       Last code was E0h
0       Last code was E1h
```

A RAM variable at 0040:0097 reflects the state of the keyboard LED indicators. The LED indicators are controlled by the keyboard BIOS through the use of commands issued to the keyboard controller.

The information in 0040:0097 is compared with the mode bits in 0040:0017 to determine whether the LED indicators are up-to-date.

The format of RAM location 0040:0097 (byte) is defined below:

```
BIT     FUNCTION
----------------
7       Reserved
6       1 = 8042 command in progress
5       Reserved
4       1 = ACK reply received
3       Reserved
2       1 = CAPS LOCK LED ON
1       1 = NUM LOCK LED ON
0       1 = SCROLL LOCK LED ON
```

The status of the LED indicators are checked:

o Each time a keyboard hardware interrupt occurs
o When the Get Key (INT 16h, AH = 00h or AH = 10h) function is invoked
o When the Check For Key Available (INT 16h, AH = 01h or AH = 11h) function is invoked

The ability to adjust the volume of the key click is a BIOS feature unique to Compaq personal computers. Two RAM locations are associated with the key click:

Table 2-16. Memory Locations Used For Key Click Functions

Address	Bytes	Contents
0040:0015	1	Previous scan code
0040:0016	1	Key click loudness (0..127)

Miscellaneous BIOS Keyboard Information

Immediately after placing a key in the keyboard queue, INT 15h is called with AH = 91h, AL = 02h. (See "Device Wait" and "Device Post" under INT 15h functions.)

Keys and key combinations that do not cause something to be placed in the keyboard queue (such as simply pressing and releasing the CAPS LOCK key) do not cause a Device Post. Pause (CTRL + NUM LOCK) does not perform either a Device Wait or a Device Post.

Decimal keyboard codes can be entered by holding down the ALT key, entering the number on the numeric keypad, then releasing the ALT key. This feature works regardless of the state of the NUM LOCK key. For example, to enter the PI character, hold down the ALT key, type 227 on the numeric keypad, then release the ALT key.

The Get Key function (INT 16h, AH = 00h or AH = 10h) executes a Device Wait (INT 15h, AH = 90h, AL = 02h), if a key code is not currently available in the keyboard queue.

The following key combinations do not place scan codes in the keyboard type-ahead buffer:

o Increase key click loudness (CTRL + ALT + Numeric Keypad "+")
o Decrease key click loudness (CTRL + ALT + Numeric Keypad "-")

To indicate receive time-out errors, parity errors, and overrun errors, the keyboard controller places a scan code of FFh in its output buffer. The system beeps once when it receives the FFh from the keyboard.

To indicate transmit time-out errors, the controller places a scan code of FEh in its output buffer.

Interrupts remain enabled and execution is suspended if CTRL + NUM LOCK is input."

ACKNOWLEDGMENT.

Andrew Jury[2], of Compaq's European system support organisation, was more helpful than the rest of Compaq put together. The main contributions of the rest of Compaq were to make it very unclear whether or not it would be useful for me to buy a Technical Reference Guide for my computer, and then to make it extraordinarily difficult to buy it. (I haven't, yet.)

REFERENCES.

1 : G.A. Creak : *Multiple keying for faster communication*, unpublished Working Note AC91, (September, 1994).

2 : Electronic mail exchanges with Andrew Jury (Systems support, Compaq Europe), July – October, 1996.

3 : Technical Reference Guide for the Compaq LTE Elite family of personal computers (Compaq, 1994).

4 : Technical Reference Guide for the Compaq Deskpro 486/33L personal computer (Compaq, ?)