Repairing HP9800 Desktop Calculators – Part 7

Tony Duell (ard@p850ug1.demon.co.uk)

Electronic Troubleshooting

In may ways this section is totally unnecessary. With the theory-of-operation that I have already given and basic electronic knowledge, you should be able to trace any fault in an HP9800 machine. But a description of the basic tests will be useful. Of course I can't give a sequence of tests to precisely locate the faulty component. But what I can do is describe how to determine which section of the machine is (probably) faulty and how to further investigate that section.

The methods of troubleshooting the power supply, processor and memory sections are essentially the same for all the HP9800 machines.

This will not be a list of 'stock faults'. I could say, for example, that the reason my HP9820 had no display was that U4 on the **Memory Address** PCB had failed. While this would be true, it would also be useless. As you doubtless realise by now, the same symptoms – a totally blank display – could be caused by the failure of just about any IC on that PCB. Or on any of the other PCBs in the memory or processor sections.

I am, of course, going to assume a basic familiarity with troubleshooting techniques and the testing of components. This information is not specific to the HP9800 series and is covered in many books on faultfinding and repair.

1 Initial tests

At this point the machine should be in the following state. All the plug-in PCBs and the keyboard should be removed, the only parts in the case should be the **Backplane**, the cardcage metalwork, and the (tested) rear panel assembly. The mains switch should be connected to its in-line connector and turned on, Make sure the connections on this switch can't come into contact with anything – including you. Of course the machine should not be connected to the mains at this point.

1.1 Power supply troubleshooting

Fit the **PSU** PCB(s), connect the mains cable and turn it on. If the cooling fan starts running, the mains fuse has not blown, and the next stage is to check the power supply output voltages. There are testpoints for all of these apart from possibly the +5V rail on the power supply PCBs, check the voltage on each of these testpoints with respect to the metal chassis. If there isn't a +5V testpoint, check this on the appropriate pin of an edge connector on the **Backplane**.

If one or more of the power supply voltages is incorrect, first remember that all the other outputs use the +12V output as a voltage reference, so this output must be put right first. Check the unregulated input voltage to the regulator, if this is missing, check the diodes and smoothing capacitor. Check the pass (or chopper) transistor. Measure the voltages on the inputs of the associated 723 IC, and based on the theory of operation of that regulator, diagnose the problem.

If the mains fuse blows when the power supply PCB(s) are fitted, it almost certainly indicates a short-circuit in the power supply section. the most likely problems are a shorted rectifier diode, a shorted smoothing capacitor[1], or a shorted pass transistor. The last will cause the crowbar circuit on that power supply output to operate, and as the pass transistor is assumed to be shorted, this will put a short across the unregulated supply, blowing the mains fuse.

If you have difficulties with the power supply, desolder all the rectifier diodes (on all the PCBs) apart from the pair that feed the +12V line. Check those last 2, the smoothing capacitor, the pass transistor and all the associated components. Get the +12V rail right first of all. Then fit th 2 diodes – after checking them – that feed the -12V regulator, and get that one working. Add the diodes for the remaining linear regulators one set at a time and get each supply working. Finally fit the diodes for the +5V switching regulator (which is the most complicated one) and troubleshoot that.

Once the power supply is working correctly, refit the power supply cover ant its screws (starting with the 3 countersunk ones at the right hand side) in an HP9810 or HP9820, or screw the **PSU** PCB heatsink to the cardcage in an HP9830. This ensures the power supply components have the correct heatsink arrangements.

2 System clock

One digital section can be easily tested independently of all the others and that is the system clock. With the machine turned off, fit the **Clock** PCB into its slot. Switch on and check the **MClk** signal at U5d with a logic analyser or oscilloscope – with nothing else in the backplane it is easy to connect a probe

[1] Actually, the electrolytic capacitors used in these machines are a lot more reliable than some people believe and rarely need replacement

here. You should see an 8MHz square wave, if not, check the master clock components.

Sometimes you find a very poor square wave around 19MHz here. No, you don't have a faster-than-normal HP9800. What you are seeing is U5 oscillating by itself because the crystal has failed or is not connected. Check that the 8MHz crystal is properly plugged in, and if necessary replace it.

Now, again using the logic analyser or oscilloscope, monitor the **Bitclk** and **μClk** signals at the outputs of U13. These are (or at least should be) repetitive waveforms so a normal oscilloscope is adequate for this. You should seen 16 pulses on **Bitclk**, then a pulse on **μClk** then another 16 pulses on **Bitclk** and so on. If not, troubleshoot the bit counter and clock control circuitry.

If you want to do a more thorough test, you can ground the **μ(8)**...**μ(11)** signals in various combinations and check that you get 1 more **Bitclk** pulse between **μClk** pulses than the binary number at the inputs to U14. If not, then U14 is suspect, but this is a very uncommon fault.

3 Obtaining a display

Since the display in these machines is controlled by the system firmware, to get anything at all on the display, the processor, memory system and display logic itself must all be operational. Unfortunately there is no simple way to test these sections one at a time, it is simpler to assemble the complete processor and memory system and try to diagnose any faults in the system as a whole.

With the machine disconnected from the mains, fit the remaining 3 CPU PCBs, the complete memory system (apart from any RAM expansion PCBs) and the display system. For an HP9810 or HP9820, the 'memory system' consists of the memory box and the PCBs in it apart from the RAM expansion PCB in the very top connector. For an HP9830, it's the **Memory Address**,**Memory Data**, **Ext ROM Selector** and **ROM** PCBs, the extension ROM PCB in the rearmost connector and the **RAM** PCB in the connector nearest the **PSU PCB**. The display system consists of the **Display** PCB in all machines and in the case of the HP9830 the **Display Driver** PCB too.

Switch on the machine. If you are very lucky, you will get something on the display. If you are extremely lucky you will get the correct display. But in most cases the display will remain blank.

Check the power supply voltages again. It is entirely possible that a short-circuit on one of the PCBs is causing one of the regulators to shut down. This may be due to a defective decoupling capacitor. In any case, find the short and correct it before carrying on.

Now think back to the fundamental rule of troubleshooting, and consider what the machine should be doing. At one level what it should be doing is 'executing machine instructions, some of which output data to the display'. That means the processor should be running, it should be accessing ROM and RAM, and it should be loading data into various registers (including the I/O register) and driving the display.

For the processor to be executing instructions, it means that it's reading them from memory into the Q register and that it's incrementing the program counter (using the ALU) to fetch the next instruction (most of the time, there will be jumps, of course).

First monitor the **DispStb/** signal at the display. If it is pulsing, then it is likely the processor and memory system is close to working. Monitor the I/O register outputs and if those are toggling too, the problem is most likely to be in the display logic itself. This is rarely the case, but you will feel a right idiot if you spend days working through the processor logic only to find it was a problem with a decoder on the display PCB. I shall have more to say about troubleshooting the display itself later on.

Check **Bitclk** and **μClk** again. If they are no longer running, something must be stopping them. If just **μClk** is stopped on an HP9820 or HP9830, check round the DMA flip-flop in case **DisμClk2** is being asserted. If both clock signals are missing, then check the **Wait/** input to U7a on the **Clock** PCB.

3.1 Is it stuck in a refresh cycle?

If **Wait/** is asserted, it is likely that the machine is stuck in the memory refresh cycle. Examine this logic on the **Memory Address** or **Memory Timing** PCB. In particular, is the refresh cycle running at all (is the memory control counter or state machine running), does the refresh address increment so that the refresh cycle should end, and if it does, why isn't **Wait/** being deasserted?

3.2 Ins the microcode running at all?

Since **μClk** is operating, the outputs of the microcode program counter should be changing. Check the **μAddr** signals at the test connector on the **CPU Control** PCB. At least some should be toggling. If not, see if any of the inputs to the microcode program counter flip-flops are high (these are also available on the same test connector). If they are, suspect a problem with the microcode program counter flip-flop.

As an aside at this point, if all the **μAddr** signals are toggling *apart from* **μAddr(1)**, it is very likely the microcode is stuck in the I/O loop at 0512, 1207. We will confirm this in a moment.

3.3 Tracing the microcode

If you have a logic analyser, one very useful technique is to trace the microcode. Connect 8 of the logic analyser channels to the **μAddr** signals on the **CPU Control** PCB test connector and if possible clock the analyser (at least for the simpler tests) from the rising edge of **μClk**.

Using the microcode source listing, see if the sequence of states is possible. If not, then the fault is most likely to be in the microcode program counter or microcode ROM areas of the machine. If the sequence of states is possible, then see if the machine ever executes the instruction fetch sequence starting at 0202. If not, then the microcode must be stuck in a loop somewhere. Determine what should get it out of that loop, and find out why it isn't. One common loop is, as I mentioned earlier, the I/O instruction loop, and I will say more about that later on.

Another possible loop is the address indirection loop at 0003, 0602. If, due to a fault, all addresses are read with their MSB set, the microcode will loop here as it repeatedly indirects on them. This generally indicates a fault with the memory or **Memory Data** PCBs. Check the logic that loads the MSB of the T register.

If you do not have a logic analyser, all it not lost. It is possible to do a simpler test with a home-made device. It won't give you as much information, in particular, it won't tell you the exact microcode sequence. But it will let you see if various microinstructions are being executed.

You need an 8 bit comparator chip such as a 74LS688. Power it from the machine's +5V supply and connect one set of inputs – say P(n) – to **μAddr** signals on the **CPU Control** PCB test connector. Connect the other inputs of the comparator (Q(n)) to a suitable switch (an 8 position DIP switch, for example) with pull-up resistors. Ground the enable input on the comparator and monitor the output with some device that will detect a narrow pulse such as a logic probe. By setting the switches it is possible to see if particular microinstructions are being executed,

For example, see if instructions are being set by setting the Q inputs of the comparator to 1612 (11101010_2). Or see if the processor ever enters the I/O loop by setting 0512 (01011010_2) and if it leaves it by setting 0513 (01011011_2). Of course if you are using a switch to ground with pull-up resistors, a closed switch represents a logic 0.

In this way it is generally possible to get a good idea as to what the processor is doing with quite simple equipment.

3.4 Is the processor attempting to read instructions from memory?

The first part of the instruction fetch at microcode locations 0202 and 1603 increments the P register (machine code program counter and also loads the increments version in the M register (memory address register). If those microinstructions are being executed, the clock input to U10 on the **Data Path** PCB should be active. **Pout** should be toggling as the P register is read into the ALU. If you have a logic analyser, or some other way to check the relative timing of signals you could check that the data is getting to the ALU via the data selector U3, and that the ALU output is being transfered back to the P register. For these tests, where you are looking at the processor behaviour during a microinstruction, the analyser should be clocked by **MClk**

In any case the binary carry flag, U12b, on the **Data Path** PCB should be toggling. Without a logic analyser you can't tell if this is happening at the right time, but if it's 'stuck' then that area needs investigation.

Since the M register should be loaded by these microinstructions, its clock (on the Clk1 inputs of the shift register ICs) should be active. If not then there is a problem with the clock control circuitry on the **Memory Address** PCB. Now look at the **M** outputs of the M register. As the new address is shifted into the register, all of them should be changing. If they are stuck after a certainly point, then there is a problem with the M register. But take care. If, for example **M(7)** and all lower bits are stuck low, you may think the problem has to be in the shift register IC that produces **M(7)**...**M(4)**. However, if the next *higher* shift register chip is malfunctioning, the timing of the serial input to the suspected one may be incorrect and this could be the cause of the fault[2]. So check carefully before replacing any ICs.

[2]If this sounds like the voice of experience, you are right.

3.5 Is the memory controller running?

The microinstruction at location 1612 reads the next machine instruction from memory. It therefore asserts **R(6)**, so check this signal is going low. This, in turn should assert **MemEn**, which should be checked.

If so then the memory control counter or state machine should be running (again, check it). A logic analyser will let you check the exact sequence of the control signals (and this is almost essential to trace faults within the state-machine based memory controller in the HP9830), but even a logic probe will allow you to see if the machine is trying to access memory

It is extremely likely that the processor will be trying to write to the RAM too. Check if **R(3)** is ever going low, if so, then check that there is some evidence of RAM writing, such as the the write enable input to the RAMs being asserted.

If you have a logic analyser, you can check the timing of all the memory control signals against the theory-of-operation of this section. This is necessary if simple tests don't find the fault.

3.6 is the memory being addressed?

Since the M register is being loaded and the memory controller is running, there should be activity on the memory address lines. Check this. In particular check the level shifters to the RAM address inputs.

Check the address decoding circuitry to find out which memory device(s) should be being accessed. In particular, make sure a ROM is being accessed, and if memory write operations are taking place that RAM is being accessed too. Again, check the level shifters for these signals, they can cause problems. Check the logic that provides the A(8) and A(8)/ enable inputs to the ROMs too.

If all this appears to be working, the selected ROM should be outputting something. Check it is. Of course it is unlikely you will know just what the data should be, but if any data line appears 'stuck', then it needs further investigation. If RAM is being accessed, check that the RAM output level shifters are working correctly.

Of course the final stage of a memory read operation is to load the data into the T register. Check there is activity on the data inputs of these shift register chips and that the mode and clock inputs are causing it to parallel-load. You should also, of course see activity on all the outputs of the T register both as it is loaded and as it recirculates to read the data into the processor. Only a logic analyser can distinguish between these, but a logic probe should at least show activity on all the T register outputs.

3.7 is the instruction being read and decoded?

The next microinstruction, at location 0616, transfers the machine instruction from the T register to the Q register. Check that this microinstruction is being executed. Then check that the T register is recirculating as it is read. The mode input to these shift register ICs should be low and the clk1 inputs should be toggling at this time. With a logic analyser you can verify this exactly, with a logic probe you should see activity on both these signals (the mode input would be high during the read cycle and go low for the recirculation, so you should see it changing state with a logic probe).

Of course during recirculation the data from the LSB of the T register is fed back to the serial input. This is something that is very difficult to check without a logic analyser, since data from the processor will be being loaded into the T register before RAM write operations. But if nothing is happening at the serial input of the T register, there is certainly a fault.

Check that the appropriate gate in the ALU input data selector is being enabled (Again, a logic analyser will also check this this is happening at the right time). The data should be passing through the ALU to the input of the Q register. A logic probe on its own will tell you almost nothing here (the ALU output is changing, sure, but then it would be changing in most other microinstructions too).

Now, the Q register clock input, from U11c on the **CPU Control** PCB,should be active to load the data into the Q register, so check this next. And as with the other shift registers, check that all the outputs of the Q register are toggling as the instruction is loaded it.

At some point a memory reference instruction will be executed, and after calculating the address the microcode will end up at location 0602. There is little point in checking the output of U4 on the **CPU Control** PCB with a logic probe, since it should be changing as the Q register is loaded, but if it's stuck, it is probably worth checking this gate. It's likely any problems in the microcode branch condition logic would have been found by now , but it may be worth checking this too.

At location 0602 there is the multi-way QμJmp operation, Check that **QμJmp/** is being taken low, if not, debug round U15b on the **Data Path** PCB. The actual multi-way jump, controlled by U15 on the **CPU Control** PCB is very difficult to trace without a logic analyser, and while U15 must be doing something for the microcode to get this far, a fault could affect QμJmp operations only, although this is fairly unlikely.

Note that if the Q register is always 0, either because the T register is not being correctly loaded from memory during the instruction fetch, so that it ends up containing 0, or because the T register is not correctly transferred to the Q register due to data path problems, then the processor will repeatedly execute ADA instructions. That is, the microcode will execute the routine at 0000, 0005, etc after every fetch. If this appears to be occurring, check to see if the T and Q registers are being loaded correctly

3.8 Are instructions being executed correctly?

This, essentially, means 'Is the data path – the ALU and registers – working correctly?'. Some faults in this area will already have been detected, for example if the ALU output was stuck high or low, the M register could not be loaded correctly.

Without a logic analyser, again all you can really do is test that things are doing something, For example, when the microinstruction at 1213 is executed, as it should be during the instruction decode routine, the accumulator selection flip-flop, U2 on the **CPU Control** PCB is set from bit 11 of the Q register. It is very likely, therefore, that this flip-flop should be changing state as different machine instructions are executed. A logic probe will tell you that, a logic analyser will allow you to check it's actually doing the right thing.

With a logic analyser you can trigger on a particular machine instruction in the Q register and check that the right accumulator control signals are produced during the microinstructions for that machine instruction. Without one, you can only see if the accumulators are doing *something*. Actually, this is often enough, most common faults cause a signal to be totally absent.

3.9 Are any I/O operations occurring, and are they terminating

If the microcode is running normally. see if it ever gets to location 0217 which is the start of the I/O routine. If not, then either there's a problem with the instruction decode routines (possibly in the microcode conditional branch logic) or genuinely no I/O instructions are being read by the processor. This may be due to a (very rare, fortunately) problem with the system ROMs.

Assuming the I/O routine is being entered, check the output of the I/O Operation flip-flip, U1, on the **Clock** PCB. **I/OOp** should be asserted at the start of I/O operations and deasserted at the end.

If it is never deasserted, the microcode will sit in the I/O loop already mentioned for ever. If the microcode is stuck in this loop but **I/OOp** is not asserted, check the microcode conditional branch logic, in particular **Cond(15)**.

If the microcode is still stuck in the I.O loop with **I/OOp** asserted, check the outputs of the I/O Cmd Decoder, U9 on the **Clock** PCB. Hopefully it will be stuck in one of the data transfer operations with one of outputs 10, 12 or 14 of U9 pulsing. If so, check if U4 is counting down, and if it is, whether **I/OBr** is ever being asserted. If U4 is not behaving correctly, then check it, and its inputs. If U4 is working properly, or if the I/O control system is stuck performing a non-data-transfer operation (which should complete in a couple of cycles), investigate the I/O End logic at the inputs to the I/O Cmd Register.

If the I/O microcode is apparently running normally, that is, if the I/O loop is being entered and exited normally, check if any output instruction are being executed. This can be easily done by seeing if output 12 of U9 on the **Clock** PCB is ever being asserted. Again, if not, there may be a firmware problem. If it is, see if the I/O register on the **I/O Interface** PCB is receiving data from U13b on that PCB, and if that data is being shifted along the register. A logic probe applied to all 16 outputs in turn (these are accessible on the edge connector for the **I/O Backplane** should show them all toggling.

finally, if you still have no display, investigate the Display Strobe flip flop consisting of U2c and U7a on the **I/O Interface** PCB.

By now you should have something on the display. The next task is to find any faults in the display logic itself.

4 Obtaining the right display

Once something appears in the display, the processor and memory sections must be basically working. However, the display could still be incorrect. You may find it useful to read the next section and get the keyboard working so that you can enter data into the display for some tests.

There are several cases to consider :

4.1 The display is blank, even though the processor is accessing it

If you have a logic analyser, capture the contents of the I/O register when **DispStb** is active, and check that the machine is outputting something that should result in a display. If not, then this would suggest a RAM or firmware ROM problem.

If valid display data is being sent to the display, check the display protection monostable circuit. In an HP9810, check the cathode driver decoders (which are common to all 3 rows of the display). In an HP9820 check the row group selection circuitry.

In an HP9830, check that the display scanning oscillator, U6, on the **Display Driver** PCB is running, and that the scan counter (U12) is incrementing. Check that the outputs of the character generator ROM are active. Also check the column select decoder, U35, and the display enable logic on the **Display** PCB itself.

4.2 The display shows the wrong characters or meaningless patterns

Since the displays on the HP9810 or HP9820 are firmware-driven, this would suggest a problem with the firmware ROMs or system RAM on these machines.

In an HP9830, this could also be due to problems with the character generator ROM and its input multiplexer on the **Display Driver** PCB.

4.3 A complete (electrical) row or column is blank

It is important to remember that on the HP9820 and HP9830 machines, the electrical matrix of LEDs is not the same as the physical 7×80 or 7×160 matrix of LEDs.

If all the LEDs in a particular electrical row or column are blank, then of course attention should be turned to the driver circuit for that row or column and the logic circuitry that feeds that driver. If several rows or columns are blank, look for a common cause. For example, an HP9810 where the 'a', 'b', 'f' and 'g' segments of the 'Y' register display are all dark could mean that U3 on the **Display** PCB has completely failed.

4.4 One LED never lights

If one segment of an HP9810 display or one dot of an HP9820 or HP9830 display is always dark and the display otherwise operates normally, it would suggest that that particular LED has failed.

The LEDs can be tested using an analogue multimeter on the '$\Omega \times 1$' range with the probes applied to the pins of the display device directly[3], the machine

[3] With most analogue multimeters, the black probe is *positive* in the resistance mode

being turned off and the **Display** PCB removed.

The remedy is obvious: replace the defective display device. The problem is obtaining one. The 7-segment LED modules used in the later version of the HP9810 **Display** PCB are similar to those used in 'classic series' handheld calculators, and the latter may be a source of spares. But the other display devices are very difficult to find.

5 Repairing the keyboard

The next stage is to assemble and connect up the keyboard. Fit the **Keyboard Encoder** PCB to the underside of the **Keyboard** with 2 of its screws (you will have to remove it again later) and plug in the interconnection PCB and, on an HP9810 or HP9830, the cable. Connect the keyboard cable to the appropriate connector on the **Backplane** and turn on the machine. Try the keys. If you are lucky, they will all operate correctly, if not...

5.1 The keyboard seems dead

If no keys have any effect, then monitor **KeyDn/** with a logic probe. This signal should pulse low when a key is pressed. On an HP9810, it should stay low while the key is down, on the other machines it may well be set high again by the interrupt service routine setting a flip-flop on the **Keyboard Encoder** PCB.

If **KeyDn** is active, then trace the processor microcode. The microcode interrupt service routine starts at location 1212. If this is being executed correctly, it would suggest that either there's a firmware problem or that the keyboard is sending invalid keycodes. Check the keyboard output gates and that the keycode is being correctly transferred to the I/O register by the logic around U6a on the **I/O Interface** PCB.

If **KeyDn/** is not active, the problem is in the keyboard. Check that the master clock oscillator is running, and that the scan counter is incrementing. Check the column driver, and in an HP9810 or HP9820, the row driver. See if there is any activity on the **Key** signal when a key is pressed. If not, on an HP9830, check the row multiplexer, U3, on the **Keyboard Encoder** PCB. On an HP9180 or HP91820, check the key sense comparator and its associated components.

If **Key** is active, check that it is stopping the scan counter, and check the logic that generates **KeyDn/**

5.2 Only some keys work

If the defective keys all lie in the same row or column of the keyboard matrix, the most likely problem is the driver circuit for that row or column. This should be relatively easy to repair.

If the 2 keys at the same matrix location on an HP9810 or HP920 are defective and all others work, check the diode on the **Keyboard** PCB associated with that location.

If half the keys on an HP9810 or HP920 fail to work, it is likely they all have keycodes with the LSB in the same state, and that one half the key sense comparator, or its associated components, has failed. The main difficulty here is obtaining a replacement comparator IC.

One curious fault is when a random set of keys on an HP9810 or HP9820 fail, all with the same state of the LSB, indicating a problem with one half of the comparator, but other keys using that half work correctly. This is normally due to the fact that the signals from the various keys are not all identical in voltage so if the comparator circuit is malfunctioning it may detect some and not others. The cure is to replace the comparator IC, but since this is difficult to obtain, a temporary fix is to alter the values of the resistors associated with its inputs to increase the sensitivity slightly.

Sometimes this fault occurs on both halves of the comparator at the same time. Then a seemingly random set of keys fails, quite often one key at a particular matrix location works, the other doesn't. The remedy is as I have just described.

An individual key not working on an HP9830 keyboard generally indicates a problem with that particular keyswitch. This can be easily verified removing the **Keyboard Encoder** PCB again and connecting an ohmmeter across the connections to the suspect switch on the **Keyboard** PCB. If it doesn't operate correctly, remove its keycap and the surrounding keycaps, desolder the switch form the PCB and unclip it from the keyboard chassis.

The HP9830 keyswitches were actually a standard part from Cherry at the time the machine was made, but they haven't been produced for many years and are not easy to obtain now. You may have some success in spraying propan-2-ol into the switch around the plunger, or you may be able to crack apart the 2 sections of the body, clean the contacts and glue the switch together again using a suitable solvent.

One tip if you only have a few defective keyswitches on an HP9830 is to exchange them with little-used keys, for example the numeric keypad (which is wired in parallel with the corresponding keys on the main keyboard and is thus not essential to using the machine).

The Shift keys on the HP9830 are not part of the main keyboard matrix. If they fail to work, check the switches themselves and U16f, U7a on the **Keyboard Encoder** PCB. The rewind switch can only be tested once the tape system is reassembled and working, but it is, of course, possible to check that its contacts close at this stage.

5.3 Some keys have the wrong effect

This may, of course, be due to a problem with the firmware routines that handle those keys. But it can also be due to a hardware fault in the keyboard itself.

For each of the faulty keys determine its keycode and the keycode it appears to be acting as using the schematic of the **Keyboard** PCB.

If the 2 keycodes for all the faulty keys differ in the same bit, it is likely that the fault is is the output gate that transfers that bit from the scan counter to the I/O register, or a fault with the I/O register itself.

5.4 The HP9810 keyboard LEDs

The LEDs on the HP9810 keyboard are vary easy to troubleshoot. The LEDs themselves can be tested with an analogue ohmmeter connected to the contacts on the keyboard cable connector (unplugged from the **Keyboard Encoder** PCB, as described above for testing the display LEDs.

If an individual LED doesn't light when it should do, and the LED itself is working, then the most likely problem is the latch that drives it or the inverter that the input of that latch.

If the LEDs never change, check if **LEDStb** is ever active, if not, investigate the logic around U10 on the **I/O Interface** PCB.

6 Fitting the I/O backplane and expansion RAM PCBs

Now that the machine is basically working, fit the **I/O backplane** and test it again. If it fails to work, on an HP9810 or HP9820 the problem can only be a short-circuit between tracks on this PCB. On an HP9830, it is possible for the printer interface logic to corrupt the I/O bus, but this is very rare. It should not be difficult to find the fault if, say, **DO(2)** is being forced low due to a short-circuit in U4.

Next fit the **Expansion RAM** PCB if you have one. Test the machine again and check that the RAM is recognised. A fault here could be due to a defective RAM IC on the **Expansion RAM** PCB, or to a fault in the address decoder logic on the **Memory Address** PCB.

On an HP9830, refit the **ROM backplane**. A failure of one of the buffer ICs on this PCB could stop the machine from working by corrupting the **Dout(n)** memory data bus. But this is normally a trouble-free section.