Repairing the HP9866A printer – Part 1

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

In my articles on repairing the HP9800 series of desktop calculators, I covered the printer interface in the HP9830, but I did not cover the HP9866A printer that is normally connected to this interface. In many ways this was reasonable since the HP9866A is a separate instrument, it is not part of the HP9800 series of machines. On the other hand, the HP9866A is required for many uses of the HP9830, and thus it is useful to be able to repair such a device.

In this series of articles, I will remedy the omission and cover troubleshooting and repair of this printer. I shall assume that the reader has already read the HP9800 series of articles and thus, since much of the general repair tips, necessary tools and test gear and safety information are the same for the HP9866 as for the HP9800 machines, this information will not be repeated here.

As in the case of the HP9800 calculators, the official HP service manual is mostly a 'boardswapper guide', although it does contain schematics of the **Logic PSU** and **Motor Driver** PCBs. A full schematic is however available and is almost essential for repairing this printer. The source listing of the control state machine is also available, but this information will be given in a future article in this series.

Overview

1 Basic operation of the HP9866A

Since the operation of the HP9866A is not as well-known as that of the HP9800 machines themselves, this section will give an overview of the printer operation.

The design of the printer's control electronics was dictated by the design of the mechanism, and more particularly the printhead. This is fixed to the printer chassis, it does not move across the paper, and consists of 400 tiny heating elements in a horizontal row across the paper. These elements are not evenly spaced, rather they are in 80 groups of 5, forming the 5 dot columns of the 80 characters that this printer prints on each line. When the heating element is energized, it causes the (specially-coated) paper to darken.

Therefore, at the most basic level, the operation of the printer is as follows :

1. Read in characters from the host and store them in an internal line buffer memory until an end-of-line character is received

2. Read out the buffer store, one character at a time, convert it to the appropriate bit pattern for the next dot line of that character and send it to the print head, where it will be stored, but do not energise the print elements yet

3. When all 80 characters have been processed, turn on the selected elements for a given time

4. Advance the paper one dot line.

5. If not all 7 lines have been printed, go back to step 2 to print the next line of dots.

6. When all 7 lines have been printed, advance the paper to give the inter-line gap

Unfortunately, in practice, it's a little more complicated. To save storage in the printhead circuit, and more importantly to reduce the maximum current that might be drawn from the power supply, only a quarter of the printhead elements are energised at a time. On the first 'pass' through the buffer store, characters 0,4,8,12,...are printed. On the next pass 1,5,9,13,...and so on.

Also if fewer than 80 characters are send by the host before the end-of-line character, the remainder of the buffer will presumably contain random data. Therefore it is necessary to fill the remainder of the buffer with space characters when the end-of-line character is received before attempting to print the line.

Therefore a more complete description of the printer operation is :

1. Read in characters and store them in the buffer until an end-of-line character is received

2. When the end-of-line character is received, fill the remainder of the buffer with spaces

3. Read out every fourth character of the buffer and convert it to the correct bit pattern for the next line of the character, then end that to the printhead

4. When 20 characters have been processed in this way, energise the printhead elements for the standard time

5. If all 80 characters have not been processed, then select the next quarter of the buffer and go back to step 3

6. When the entire dot line has been printed, advance the paper one dot line and go back to step 3 to print the next dot lien of the characters

7. When all 7 dot lines have been printed, advance the paper for the inter-line gap

2 Implementation and layout

Before considering the theory of operation of the various sections of the printer it is helpful to have a basic idea of the design and layout. As with many computer-related devices, the electronics can be split into 2 main sections, namely the **data path** and the **control**.

The **data path** contains the buffer memory register which is implemented using 80-bit shift registers. It also includes the character generator ROM and the printhead storage register, again built from shift registers.

The **control** section is based round a 32-state finite-state machine. Associated with this are are several binary counters which keep track of the number of characters in the buffer store and the number of dot-lines printed.

Physically the electronics consists of 7 PCBs. The 2 main logic boards contain the **data path** and **control** sections. These are plugged into the **logic backplane** PCB along with a further PCB, the **Logic PSU**. Plugged into the side of this backplane is the **Printhead Driver** PCB which contains the printhead storage shift registers. 2 Further PCBs, the **Printhead PSU**, which provides the high voltage needed for the printhead elements, and the **Motor Driver** are plugged into connectors mounted on the main chassis. Also separately mounted are some of the larger power supply components such as the mains transformer.

Mechanically the printer mechanism is very simple. Since the printhead covers the full width of the line and does not move, the only motion is that of the paper. The platen roller is directly coupled to the stepper motor spindle, a pressure roller held against the platen is coupled to the paper loading roller by a toothed belt.

Data Path Theory of Operation

As mentioned earlier, the control system has 2 modes of operation. In the first, characters are loaded into the buffer store from the host. In the second, the contents of the buffer are printed. The **RotEn** signal determines the current mode. This signal is produced by the dot line counter. When it is low, the printer is in input mode, and when it is high the buffer shift register rotates (hence the name of the signal – ROTate ENable) and the buffer is read out so the contents can be printed. A clear understanding of this signal will make

understanding the rest of the printer much easier.

The 2 operating modes of the data path circuitry will be described separately.

1 Input Mode

The input register consists of 7 D-type flip-flops U226, U17, U11 and U6 on the **Data Path** PCB. Characters are loaded into this register from the host by the **InClk** signal which is derived from an output of the control state machine by U21c. Note that the data lines from the host (**IFD(n)**) are active-low signals, so that the Q outputs of this register are also active low, while the Q/ outputs are active-high.

The outputs of 6 of the flip-flops are gated into 6 80-bit shift registers, since this printer only handles upper-case characters, 6 bits are sufficient. The least significant bit (**InData(0)**) will be described, all others are similar. The output of bit 0 of the input register, U22b, is fed to the AND-OR-Invert gate U15b. Since **RotEn** is low, U20f sets **InEn** high and the data is passed to the input of U14b, which is an 80 bit MOS shift register. The clock for this register is produced by buffering the **SClk** signal from the control state machine with U20a and the transistor Q1.

At the end of the input line, the remainder of the shift register must be filled with space characters. The **ForceSpace/** signal from the state machine goes low, and since **InEn** is high, **Space/** is driven low by U16a. This logically sets bit 5 of the input register and logically clears the remaining bits, thus forcing the register to contain the ASCII code for space. The state machine then produces the appropriate number of **SClk** pulses to fill the remainder of the shift register with space characters

There are 2 circuits that decode the input character and provide inputs to the state machine. When a linefeed character is received, the output of U5 (**LF**) is asserted (low). This indicates the end of the input line and causes the state machine to start the print cycle.

The other circuit asserts the **PrtChar** (printable character) signal. If the character is not a control code (that is, at least one of bits 5 and 6 is asserted, detected by U16d), and it's not DEL (detected by U10) or if it's a linefeed, the output of U16b goes high, indicating that the character is to be processed by the printer, this is fed to an input of the control state machine.

2 Print Mode

In this mode, **RotEn** is high, so **InEn** is set low by U20f. Again considering bit 0 of the data shift register, the output of the shift register U14b is inverted by U7b and fed back to the AND-OR-Invert gate U15b. This gates this signal back to the input of the shift register. Thus the shift register recirculates (rotates) and the data can be repeatedly red out. The output of U7b is further inverted by U7a, providing an active-high character code bit. The 6 character code bits are fed to the low 6 address inputs of the character generator ROM to select the appropriate character pattern, the high 3 address lines select one row of that pattern and come from the row counter, which will be described as part of the control section.

The character generator is U1, and is a 512*8 ROM of the same type as are used for the HP9800 firmware. It is permanently enabled (**CE** and **CS** are both connected to +12V). 5 outputs of this ROM, containing the bit pattern for the current row of the current character are inverted, e.g. by U13c, and fed via the backplane to the **Printhead Driver** PCB. There are 5 identical circuits on this board, corresponding to the 5 dot columns of each character, again only one – column 0 – will be described.

The **PHData(0)** signal coming into the **Printhead Driver** PCB is inverted by U4a and fed to the 20 bit shift register consisting of U9 and U6. The character rows are shifted into this register by the **PHCLk** signal from the control state machine, inverted by U4b. This shift register provides 20 of the printhead data signals, those for the first dot column in each character, **Drv(0), Drv(5)...Drv(95)**. The 5 registers thus provide all100 signals needed to control the selected section of the printhead. These signals are wired to the connectors for the 4 printheads.

Each printhead takes in 25 of the signals and contains circuitry to energise the appropriate elements selected by the **Bank0...Bank 3** signals. Again, the operation of these signals will be described with the control logic.

Control System Theory of Operation

1 Control state machine

As was mentioned earlier, the control system is based around a finite state machine with 32 states. Therefore the state can be represented as a 5 bit number which is stored in the 5 D-type flip-flops U2a, U6b, U6a, U18b and

U2b on the **Control** PCB. The outputs of these flip-flops form the address for the 32 byte ROM U12 which contains the state table.

A description of this state table, which could be considered to be the control firmware of the printer, will be given later, but at present only the hardware will be described.

The next state is of course determined by the logic levels on the D inputs of the flip-flops. The highest 2 state bits are simply determined by 2 of the ROM outputs, while the **CC** (Condition Code) signal selects one of 2 ROM outputs for each of the lower 3 state bits. For example, consider the input to U2a, which holds the lowest state bit.

If **CC** sis low, then U1d sets **CC/** high. Therefore U1c is inhibited and its output is forced high. U1a inverts **SMD(2)** and passes it to U1b, which inverts it again., So in this case, the input to U2a is **SMD(2)**. If **CC** is high, then U1a is inhibited, while U1c inverts and passes on **SMD(7)**, which is then re-inverted by U1b. So in this case the input to U2a is **SMD(7)**.

So, depending on the state of **CC**, the new state is given by the SMD bits as follows ;

CC=0 : 6,4,0,1,2

CC=1 : 6,4,3,5,7

There is, of course, no reason why the ROM can't be programmed so the 2 possible next states have the same value, meaning that the state of the **CC** signal is irrelevant. Needless to say, such unconditional transitions are quite common in the state machine.

The **CC** signal is thus the control input to the state machine. It comes from the 16 input multiplexer U14, which selects the various signals that need to be tested. The first 8 inputs of this multiplexer as used in print mode, the last 8 in input mode, since the D select input is driven by **RotEn/**. The other 3 select inputs are driven by 3 of the ROM outputs – thus the next state is partially determined by the signal to be tested.

The 16 output decoder U13 is driven by the lower 4 state flip-flops. It provided the output signals from the state machine to the rest of the logic. Thus each output signal is produced in different stares (since the highest state flip-flop is not used by this decoder). However, since **SCLk**, the clock signal to the buffer shift register, is required in 4 states, 2 of the outputs of the decoder are logically ORed by U20d.

2 Master Clock and Reset

The clock for the control state machine comes initially from the RC oscillator formed from U8a, U8b and U8c on the **Control** PCB. This provides a clock signal of approximately 3.5MHz which clocks the 2 cross-coupled JK flip-flops U7a and U7b. These cycle through the stares 00, 10, 11, 01, 00...and thus produce a pair of signals of a quarter the frequency of **MCLk** in phase quadrature. The output of U7b is used to clock the state machine, that of U7a enables the decoder U13 when it is low. This has 2 consequences. The first is that the outputs of U13 pulse every cycle of the clock divider even if the state machine remains in the same state. The second is that the outputs of U13 re disabled when the state machine is changing state, thus presenting glitches on these outputs.

If the **Wait/** pin on the test connector is pulled low, U20c will cause the clock divider to halt in state with OutEn/ low, that is with U13 enabled. This facility does not seem to be particularly useful.

In order to start the system off in a known state, the **Init/** signal is pulsed low by the **Logic PSU** at power-on. This is buffered by U27c and U27f on the **Control** PCB and clears all the state flip-flops, and other parts of the logic. This also asserts the **Rst** signal via U16c, which clears the interface logic. This logic can also be cleared by **IFInit**, provided the printer is in input mode – the inverted form of **IFInit** is ANDed with **RotEn/** by U16a.

3 Interface Handshake

In order to describe the interface handshake, the particular example of an H(830 host will be used.

The trailing edge of the **LatchEn** signal on the HP9830 **I/O Backplane** sets the flip-flop U3a. The Q/ output of this flip-flop goes low, causing the printer to appear busy via U1b. The Q output asserts **PrtStb/** via U2c.

This strobe signal is fed to the **Control** PCB in the HP9866 where is it called **IFStb/**. It is received by U23c, and provided the printer has paper and that the manual feed button is not pressed (that is, **PaperOut/** and **Feed/** are both high), the output of U22b goes low. This is an input to the state machine input multiplexer, U14.

The state machine now begins the character read sequence. When it asserts **InCLk/**, the data is latched into the input register. The trailing edge of this signal sets U22a on the **Data Path** PCB, which asserts **IFFLg/** via U20c.

Back on the HP9830 **I/O Backplane**, this flag signal, now called **PFLg/** clears U3a and thus removes the strobe. It also causes the busy signal to be maintained via U1b.

When the HP9866 has processed the incoming character, the state machine asserts the **CLrFLg** signal. This resets U22a on the **Data Path** PCB and de-asserts **IFFLg/**. The printer is now ready for another character.

4 Control Counters

It is necessary for the control system to keep a count of the number of characters stored or printed and the current dot line. In order to simply the control system, this information is not stored in the state machine, instead separate counters are used. There are 3 such counters, one counts the characters transfered to or from the buffer shift register; the second counts the rows of dots being printed and the last is a simple 4 state counter used to count the shifts when the shift register is moved on by 4 characters during each section of the print cycle. These counters will be described individually.

4.1 Character Counter

the character counter is an 80-state counter consisting of the divide-by-10 counter U4 and the divide-by-8 counter U10 on the **Control** PCB., It therefore counts from 0 to 79 in BCD.

Two signals increment this counter – they are logically ORed by U16b. During the character input operation, the **PHOut/** signal (which does not enable the printhead elements in this mode) is gated via U15d to produce **IncCC**. During the print cycle, the **PHClk** signal from the state machine, which loads the bit pattern into the printhead shift registers increments this counter. It is important to realise here that the count is not an 'index' of the characters in the data shift register. During the print cycle, the bit pattern of every fourth character is transfered to the printhead shift register, but the counter is only increments once for each character transfered.

The outputs of this counter are decoded by NAND gates to produce condition inputs for the control state machine. The output of U16d, **Newbank/** is asserted when the character count is divisible by 20, that is, when a quarter of the characters have been transferred to the printhead shift register, and thus this section of the dot line should be printed. The **Char0/** signal is asserted by U9a when the count is zero, that is, when the count has wrapped around and

all characters have been processed. Finally, U15c produces **Fristback/** when the first quarter of the dot line is being printed.

Two other circuits are associated with this counter. The SR flip-flop produced by cross-coupling U15b and U22c is cleared when a dot line is printed and set when the character counter is incremented. It thus indicates whether any characters have been loaded into the buffer shift register. The output of this flip-flop, **Empty/**, is combined with **Char0** by U11b. Its output, **Start/** therefore indicates whether the fact that the character counter is 0 indicates. that the buffer is full or not

The other related circuit is located on the **Printhead Driver** PCB. Here, the NOR gates in U1 and U2 decode the top 2 bits of the character counter, and when **PHEn/** is asserted (the source of this signal will be described later), the appropriate bank signal is turned on, energising the printhead elements. Since all 4 of these circuits operate in a similar manner, only one will be described.

For example, for the second quarter of the line, **PHSelA/** is 1 and **PHSelB** is 0. Therefore, when **PhEN/** goes low, the output of U1b goes high, the outputs of U2a, U1a and U2b all remain low. The output of U1b going high turns on the PNP transistor U8d via U3a and thus applies the printhead power supply voltage to the Bank2 signal.

4.2 Row Counter

This is a 10 state counter consisting of U12 on the **Data path PCB**. It is incremented by the **RowClk** output from the control state machine and keeps a count of the number of dot rows printed. When this counter is in state 0, **RotEn** is set low by U19a, selecting the character input mode. At the start of the print cycle, this counter is incremented, asserting **RotEn** and selecting the print cycle mode. The output of U19b goes low during the 7th row of dots; this is the last row and thus the state machine tests this at the end of each printed row to check if the complete character line has been printed.

4.3 Shift Counter

In order to count the shifts when moving on 4 characters during the print cycle, a simple 4 state counter on the **Control** PCB is used. It consists of the interconnected D-type flip-flops U17a and U17b. During the print cycle, the **ForceSpace/** signal (which has no other use in this cycle) increments this counter via U20a. It counts in the sequence 00, 01, 11, 10, 00...but the exact sequence is not important for the operation of the printer. What is important

is that the output of U11a, **LnFlg/** is asserted every 4 cycles. This signal is a condition input to the state machine.

5 Timing Monostables

The printhead takes a significant time to change the state of the thermal paper, similarly the stepper motor takes a significant time to move the paper up by one dot row. Three monostable multivibrators are used to determine these times.

In order to print the current dot pattern (already loaded into the shift registers on the **Printhead Driver** PCB, the state machine asserts **PHOut/**. Since **RotEn** is high, the monostable U24 is triggers via U20b, thus asserting **PhEN/** for about 6ms. This signal energises the printhead elements via the decode circuit on the **Printhead Driver** PCB. Monostable U26a is also triggered, it has a time of around 7.16ms. This output of this monostable is gated via U25 and synchronised by U18a to provide a condition input to the state machine which then waits until the printhead has 'burned' the pattern onto the paper before continuing.

Similarly, to move the paper up one dot row, the sate machine produces the **AutoFeed/** signal. This triggers the monostable U26b, with a period of 8.2ms. This is similarly fed via U25 to the state machine input to cause the state machine to wait for the motor to finish advancing the paper before attempting to print another row of dots.

6 Paper Sensor

The presence of paper in the printer is detected by a changeover microswitch under the chassis. This microswitch is debounced by the SR flip-flop U28a and U28b on the **Control** PCB to produce the **PaperOut** signal and its inverse. This signal has 3 effects : It inhibits the input strobe signal via NAND gate U22b, thus preventing the printer form accepting characters from the host; It turns on transistor Q1 and thus causes the 'Load' indicator on the front of the instrument to illuminate; and finally it assets the **IFPout** signal via U20b on the **Data Path** PCB to indicate to the host that the paper has run out, however this signal is not used by the HP9830.

The next article in this series will cover the state machine firmware and the remaining electronic sections