Repairing HP9800 Desktop Calculators – Part 4

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

Display Theory of Operation

The HP9810, HP9820 and HP9830 obvious have very different display systems. The HP9810 has 3 rows of 15 digits, the HP9820 has a 16 character dot-matrix display and the HP9830 has a 32 character dot-matrix display. However, in all 3 cases, the display is controlled by the 16 bits of the I/O register. The system firmware loads a value into the this register and causes the I/O control system to assert **DispStb/**. The contents of the I/O register select a particular display element and also determine what is displayed in that element.

In the case of the HP9810, an element is one of the 45 digits., and the segments are individually controlled by bits in the I/O register. In the HP9820 it's one of the 80 columns that make up the 16 characters, the LEDs in that column are again individually controllable. The HP9830 display has a hardware character generator, the contents of the I/O register select a pair of characters 16 positions apart in the display and determine the characters to be displayed in those 2 positions.

The 3 display systems will now be described individually.

1 HP9810 display

There are in fact 2 versions of the HP9810 **Display PCB**. The earlier one is fitted with 45 separate 7-segment display devices, the later one has nine 5-digit display modules, similar to those used in the 'classic series' handheld calculators. The PCBs are electrically very similar, the following description strictly applies to the later, more common, version, but it will be of use in repairing the earlier one too.

The bits of the I/O register are used in the following way :

DO(0)...DO(3) : Select a particular digit in a display row (binary encoded)

DO(4)...DO(6) : Select one of the 3 rows (1-of-n encoded)

DO(70), SO(0)...SO(3), CO(0)...CO(2) : These 8 bits give the segment pattern for the selected digit.

The display PCB has a separate anode driver circuit for each display row and a common cathode driver circuit for all 3 rows.

There are 24 essentially identical anode driver circuits, I shall describe the one for the 'a' segment of the X register display, the others work in the same way.

The **CO(1)** bit from the I/O register is buffered by U7c, giving **Segment(a)**. **DO(4)**, which is low to select the X register display row, is inverted by U15a, giving the **XSel** signal. Therefore, the output of U11b is low when the 'a' segment of the X register is to be turned on. This then saturates the associated PNP driver transistor, applying 5V to the appropriate display anode pin through a 25.5Ω resistor.

The cathode driver is based around a pair of decoder ICs U12 and U13. **DO(0)...DO(3)** are buffered, e.g. by U4a, and fed to the inputs of these decoders. U12 is enabled by U10a when **Digit(3)** is low, while U13 is enabled by U10b and U13f when **Digit(3)** is high. The selected output of the selected decoder goes low, and pulls the appropriate cathode pin of the display low via an PNP emitter follower such as U6d.

To enable either decoder (and thus any of the cathode lines), **DispEn/** must be low. This comes from the display protection circuit based around the monostable U4. For **DispEn/** to be low, **DispStb** from the **I/O Interface** PCB must be low, but U4 must not have timed out.**DispStb** is set low by the **I/O Interface** to display a particular digit but if, due to some fault, it is stuck low, U4 will time out, disable the cathode drivers and protect the displays.

2 HP9820 display

The HP9820 display appears to the machine as an 80-column, 7-row matrix of LEDs. The bits of the I/O register control it in the following way :

DO(0)...DO(6) : The pattern of LEDs in the selected column

DO7,SO(0)...SO(3), CO(0), CO(1) : Select a particular column. This is not a simple binary code, instead **SO(3)**, **CO(0)** and **CO(1)** select a group of 10 columns and **DO(7)** and **SO(0)...SO(2)** select a particular column within that group.

An 80-column, 7-row LED matrix would appear to need 80 cathode (column) drivers and 7 anode (row) drivers, for a total of 87 circuits. However, electrically, the display is wired as 20-column 28-row matrix requiring 20 cathode drivers and 28 anode drivers. The anode drivers are further simplified by considering them as 4 groups of 7.

A particular group of anode drivers is selected by the decoder U6. For any of the outputs 0...3 to be asserted, both the C input (**DispStb/**) and the D input (output of the monostable U5) must be low. This is essentially the same display protection circuit as I described for the HP9810 display. U6, therefore, decodes the **CO(0)** and **CO(1)** signals, enabling a particular **RowSel** signal via one of 4 PNP transistors.

Each **RowSel** signal applies 5V power to the emitters of a further 7 PNP transistors. These are individually controlled by the low 7 bits of the I/O register via inverters such as U13f. When a particular bit of the I/O register is set, the output of the inverter goes low, saturating the transistor. The collectors of these 28 PNP transistors feed the display matrix anode connections via 16.5Ω resistors.

The cathode driver circuit consists of a pair of 1-of-10 decoder ICs, U11 and U8. When **SO(3)** is low, the outputs U7c and U7d are forced high, so none of U8's outputs are selected. U7a and U7b are enabled, so 4 bits of the I/O register are passed to U11. Similarly when **SO(3)** is high, the outputs of U7a and U7b are force high, disabling U11, while U8 decodes the same 4 bits from the I/O register. The 20 outputs of these decoders are buffered by PNP emitter followers such as U1d, and connected to the cathode lines of the display matrix.

3 HP9830 display

The HP9830 display system is considerably more complicated because the character patters are generated in hardware, rather than by the machine's firmware. The 16 bits of the I/O register are used in the following way :

DO(0)...DO(3) : Select a pair of display locations, 16 characters apart, that is 0 and 16, 1 and 17...15 and 31.

DO(4)...DO(7), SO(0), SO(1) : The character code to be displayed in the second selected location.

SO(2), SO(3), CO(0)...CO(3) : The character code to be displayed in the first selected location.

The display system consists of 2 PCBs. The **Display** PCB contains the LED displays themselves and the anode/cathode drivers. The **Display Driver** PCB contains the character generator circuits. I shall begin with the latter.

To start a display scan, **DispStb/** pulses low. This sets monostable U5a, and since **Char(0)** is low, U5b is triggers, asserting **DispEn/** (**Char(0)** is simply

DO(0), buffered by U18b on the **Display** PCB). **DispStb/** also clears the scan counter U12, which is used to select individual columns in the a particular display character. This is decoded by U7, the outputs of which select a column of the character generator ROM U1.

The asymmetric clock oscillator built from the monostables U6a and U6b now runs, clocking U12. Thus the counter scans through the columns of the character. The AND-OR-invert data selectors such as U14b are controlled by the clock signal. When **Clk** is high, the character code from the first selected position is gated to the inputs of the character generator U1. The outputs of this ROM are fed into the latch U9 and become the **LRow** signals to the the **Display** PCB. When **Clk** goes low. that pattern is latched in U9, and the character code for the second position is applied to character generator inputs. The AND gates such as U11c are enabled, so the character pattern appears on the **Row** lines Due to the asymmetry of the clock, the actual output consists of a short period while U9 is updated, then a much longer period where the **Row** and **LRow** signals are stable, giving the column patterns for the current column of the 2 characters. The column counter U12 is then incremented, there is the short update period again, followed by a stable period when the next column of the 2 characters is output. And the end of the character scan, output 6 of U7 goes low, disabling the clock generator until U12 is reset by the next **DispStb** signal.

A similar trick to the HP9820 display is used to reduced the number of driver circuits required. Rather than a 7-row, 160-column matrix, the display is wired as a 28-row, 40 column one. The inputs to the display driver circuitry on the **Display** PCB therefore consist of :

3 bits, DO(0)...DO(2) to select 1 of 8 sets of 4 characters

DO(3) to select a particular pair of character in that set, and to gate the Row and LRow lines to the appropriate anode drivers

3 bits, Col(0)...Col(2) to select a particular column within the characters.

2 sets of 7 signals (Row and LRow) giving the patterns in the 2 selected columns

DispEn/ : An overall enable signal.

The **DO(3)** bit is buffered by U18d and the inverted/buffered by the 4 gates in U22 providing the anode driver enable signals. The **LeftEn** signals are high when the characters to be displayed are in the first 8 positions of each half

of the display, the **RightEn** signals are high when they are in the second 8 positions. These signals enable NAND gates in the row driver circuitry. For example, U19d gates **LRow(0)** to the base of U1a when **LeftEnB** is high, thus controlling the anode line **DispRowA(1)**.

The low 3 bits of the I/O register are buffered (for example by U18b) and become the **Char** lines. The **Col** lines are decoded by U35, which mimics the operation of U7 on the **Display Driver** PCB. Together these signals control 5 further decoders, each selecting 1 of 8 display columns. U36, for example, is enabled whenever the leftmost column of a character is being displayed, the **Char** signals determine which display location is used. The outputs of these decoders are buffered and connected to the display cathode lines. The outputs of U36 enable the leftmost columns of the display characters, those of U37 the second-from-left and so on up to U40 which selects the rightmost columns of the characters.

Keyboard Theory of Operation

The HP9800 keyboards are controlled by circuitry on the **Keyboard Encoder** PCB mounted on the underside of the keyboard itself. The HP9810 and HP9820 keyboards do not use conventional mechanical keyswitches; they use a rather unusual device based on PCB tracks forming a pair of pulse transformers, which will be described shortly. This device was not really suitable for machines where 2 keys may be pressed together so the HP930, which has a shift key, uses normal keyswitches.

Since all 3 keyboards are different, I shall cover them in separate sections, but first...

1 HP9810/HP9820 'keyswitch'

Under each key of the HP9810 or HP9820 keyboard is a pulse transformer formed from spiral copper tracks on the PCB. The transformer has 2 windings, each consisting of a pair of spirals, one on each side of the PCB, connected by vias. When a current pulse is sent through one winding (the primary), a voltage is induced in the other winding (the secondary).

On the bottom of the plastic key plunger there is a metal disk. When the key is pressed, this disk is brought close to the transformer windings on the PCB. It absorbs energy from the primary winding and therefore reduces the induced voltage in the secondary windings.

The transformers are connected in pairs by the PCB tracks. The primary -

windings of each pair are connected in series, as are the secondaries, but in one pair, one of the windings is reversed. Therefore, when a current pulse is sent through the primary pair, the induced voltages in the secondary pair oppose each other and the resultant voltage is close to zero. When one of the keys is pressed, the transformer under that key produces almost no output, so the resulting voltage from the pair consists of that produced by the other key's transformer. The polarity of the output voltage pulse depends on which key of the pair is pressed. Of course if both keys in the pair are pressed simultaneously, there is, again no output voltage, but as I mentioned above, this design of keyboard is not suitable for applications which require several keys to be pressed at the same time.

The primary pairs are arranged in an electrical matrix with a diode in series with each pair to prevent sneak paths. All the secondary winding pairs are connected in series to form the 'Sense Loop'. The **Keyboard Encoder** PCB mounted under the keyboard itself scans the matrix by sending a current pulse through each pair of primary windings in turn, and looks for an induced voltage in the sense loop. Which key is pressed is determined by which matrix location causes a voltage to be induced, and the polarity of this voltage.

2 HP9810 keyboard

The keyboard is scanned by circuitry on the **Keyboard Encoder** PCB. A master clock signal is provided by the free-running oscillator U6a and U6f. When no key is pressed, **KeyIn/** is high, so U4c is enabled and the clock is fed to the scan counter. This is a 7 bit ripple counter made from the D-type flip-flops in U10, U11, U13 and U14a.

The master clock signal is looped through 2 pins on the cable edge connector and applied (as **ClkIn**) to the monostable U8a. The output of this turns on the transistor Q5, providing a current pulse to drive the matrix of transformers. This current is fed through a loop of wire so that it can be monitored by a clip-on current probe – but since I do not own such an instrument, I can't say what the pulse should look like.

Bits 1 ad 2 of the counter are decoded by the NAND gates in U1. Each gate has an associated PNP row driver transistor Q1...Q4, which feeds the current pulse from Q5 into one of the matrix rows. The keyboard column lines are connected to the decoder U9, which grounds one of them based on the top 4 bits of the scan counter. Thus the keyboard matrix locations are selected in turn, and 2 current pulses sent through each pair of transformer primaries.

The sense loop is transformer-coupled to the the comparator device U12. Due to the fact that the ends of the secondary winding of the coupling transformer are in antiphase, the 2 sections of U12 respond to opposite-polarity pulses in the sense loop. These sections are enabled by the LSB of the scan counter and its inverse. When the output of U12 goes high, the scan counter contents identifies the key uniquely, the LSB distinguishing between the 2 keys at the same matrix location.

The output of U12 triggers the monostable U8b (enabled by **RowDrv/**) causing **Key** to go high and **Key/** to go low. The latter reduces the threshold on the comparator U12, providing hysteresis. **Key/** is looped to **KeyIn** on the cable connector so that when a key is pressed, U4c is inhibited, halting the scan counter at that key's location. Thus the monostable U8b is continually retriggered, since the pressed key location is fed with current pulses on each keyboard master clock cycle.

KeyIn, which is simply the **Key** output from U8b after being looped through the cable connector asserts the **KeyDn/** interrupt line via U16a. It also enables the output gates (e.g. U7a) so the scan counter contents can be read into the processor I/O register. When the key is released, U8b times out, removing the interrupt signal and allowing the keyboard scan to resume. NAND gate U15 detects when the 'Stop' key is pressed and provides a reset signal to peripheral devices.

The output of the HP9810 keyboard is, of course, the 7 bit contents of the scan counter. The keyboard matrix is wired so that all the 'programmable' functions produce keycodes with the MSB clear, whereas keycodes with the MSB set correspond to system control keys that cannot be stored in a user program. Thus 6-bit wide RAM is sufficient for storing user programs.

The LEDs on the HP9810 keyboard are very simple. The top 8 bits of the processor I/O register are latched in U2 and U3 by **LEDStb** from the **I/O Interface** PCB. The LEDs are connected (with suitable series resistors) between the outputs of these latches and the +5V rail. Six of the 10 LEDs are independently controlled, the remaing 4 form 2 pairs (Run and Program; Float and Fix), each controlled by 1 bit of the latch. In either such pair one or other of the LEDs is always on.

3 HP9820 keyboard

The HP9820 **Keyboard Encoder** PCB is similar to that in the HP9810 machine. A master clock is formed by U8a and U8f. The output of this, **ClkOut**

is looped through the cable connector whereupon it becomes **ClkIn**, and when no key is pressed, it clocks the 7 bit scan counter via U5c. The scan counter is, as before, a ripple counter made from the 7 D-type flip-flops in U11, U12, U7 and U15a.

ClkIn also triggers the monostable U10a, providing **RowDrv** to a PNP transistor. As in the HP9810, bits 1 nd 2 of the scan counter are decoded by the 4 NAND gates in U4, each of which drives one of the keyboard row driver transistors, feeding the current pulse onto a keyboard row line. Since the HP9820 keyboard has more column lines than that of the HP9810, the column selection decoder requires 2 devices, U2 and U3.

The sense comparator circuitry based round U14 again follows the same pattern as that in the HP9810. When the pressed key is detected, the output of U14 triggers the monostable U10b, which as before reduces the threshold on U14. The outputs of U10b are looped through the cable connector to become **KeyIn** and **KeyIn/.** The latter halts the scan counter by inhibiting U5c. **KeyIn** asserts the **KeyDn** interrupt via U13d. The processor can set U15b using **LEDStb** to remove this interrupt signal and allow other interrupt to be detected, U15 is automatically cleared again when the key is released, ready for the next keypress.

U5b enables the data output buffers when a key is pressed and when the **CO** peripheral address outputs from the I/O register contain 1100_2. The outputs buffers, such as U9a, feed the scan counter contents to the I/O register data inputs. As in th case of the HP9180, the NAND gate U16 detects the 'Stop' key being pressed and asserts the peripheral reset signal.

4 HP9830 keyboard

The HP9830 uses a conventional matrix of mechanical keyswitches, so the **Keyboard Encoder** PCB is somewhat different to those in the other machines.

A free-running master clock is formed by U5f and U5a and buffered by U5c. The clock is divided by 2 by the first section of counter U2, and thus the outputs of U6c and U6d form a non-overlapping 2-phase clock. When no key is pressed, U10a is clear, so the clock at the output of U6d is gated to the 7-bit scan counter formed from U1 and the rest of U2..

The lower 4 bits of this counter are decoded by U4 and U17, which therefore ground each of the **KC** keyboard column lines in turn. The higher 3 bits of the scan counter select one of the **KR** row lines using multiplexer U3. Pressing a key connects a row line to a column line; when the location of the pressed key

is addressed by the scan counter, the selected output of the column decoder grounds the active input of U3 through the closed keyswitch, and so the **Key** signal goes high and **Key/** goes low. **Key/** triggers monostable U10a, enabled by the clock at the output of U6c, stopping the scan counter. **Key** triggers the debouncing monostable network U10b and U11 and the output of U13c goes high.

This asserts the **KeyDn** interrupt signal via U9c. U14b can be set by the processor using **LEDStb** to disable the U9c and allow other interrupts to be checked for, in the same was as in the HP9820.

To read the keycode, the **CO** peripheral select outputs of the I/O register are set to 1100_2. This causes the output of U15a to go high. Thus (since the output of U13c is high), the output buffers such as U8b are enabled by U15b, feeding the scan counter to the inputs of the I/O register. The shift key grounds the input of U16f, thus causing its output to go high, and this is transfered to bit 7 of the I/O register via U7a.

The **STP** peripheral reset signal is asserted by U9d when the key is pressed in row 7 (U12b output is high) and row 1 (U16e output is high). This is, of course. the 'Stop' key.

The 'Rewind' key is directly connected to the rewind flip-flop on the **Tape Controller** PCB, it is not processed by the **Keyboard Encoder.**

The power-on indicator on the front of the HP9830 is simply a filament lamp powered from the 5V rail with a 22Ω series resistor. Anybody who understands the rest of the machine will have no problems with this.

Printer theory of Operation

All HP9800 calculators have a hard copy facility. The HP9810 and HP9820 have an optional internal thermal printer. The HP9830 has a built-in interface for an HP9866 printer which may be stacked on top of the machine. The HP9866 itself is outside our scope, but the internal printer and the interface will be described.

1 HP9810/HP9820 thermal printer

The internal printer of this machines consists of a mechanical system to feed the thermal paper past the printhead elements and 2 PCBs of electronics. The **Printer Interface** PCB is mounted on the right hand side of the mechanism. The **Printer Driver** PCB is located under the mechanism and is plugged into an edge connector on the **Printer Interface** PCB. The printhead is connected

to a flexible printed circuit ('flexiprint') that is an integral part of the **Printer Driver** PCB, while the motor and switches connect to an edge connector at the rear of this PCB.

The printhead has 80 elements and prints a horizontal row of dots for up to 16 characters across the paper. The paper is then advanced by one dot line and the next row of dots for the line of characters is printed. This is repeated until the whole characters have been printed

1.1 Printer Mechanism

The printer is mechanically quite simple. The platen is a rubber roller at the front of the chassis, with a second shaft beneath it carrying 2 belts that run along the bottom of the paper well and round idler rollers towards the rear. These 2 shafts are geared together by the gears on the right hand side of the machine. When the platen revolves, it feeds the paper through the printer. A new roll of paper is loaded by the belts which feed the free end of the paper towards the platen.

The mechanism is driven by a stepper motor at rear of the unit. This drives a pinion on the lower front shaft via a toothed drive belt. This pinion is free to rotate on the shaft, but it meshes with a gear on the platen shaft. Thus the motor drives the platen and advances the paper.

The printhead is pressed against the platen by spring, thus keeping the resistive heater elements in contact with the thermal paper. When the 'Feed' key is pressed, the printhead is lifted off the paper by a linkage in the printer to reduce wear and to ease paper loading. The printhead operates a microswitch at the front of the unit when this occurs, this microswitch causes the electronic section to energise the motor.

The paper-out detector is simply a set of contacts that are held apart by the paper. When the paper runs out, a circuit is completed through these contacts.

1.2 Printer electronics

The electronic part of the printer consists of 2 main parts – to drive the printhead elements and to drive the motor. There is also some control circuitry common to both sections.

We shall begin with the printhead driver. The I/O register in the processor is extended by a further 10 bit shift register consisting of U4 and U2 on the **Printer Interface** PCB, which are serially loaded with the **I/O In** signal (which is, of course, the LSB of the I/O register inverted by U20d on the

I/O Interface PCB). This forms a 26 bit shift register which is loaded by the processor using 2 I/O instructions.

The 80 printhead elements are electrically connected as a 20*4 matrix, with diodes on the **Printer Driver** PCB to prevent sneak paths. The 4 rows of this matrix correspond to 4 quarters of line of dots that the printer is currently printing, the 20 columns corresponds to the 20 dots within that quarter. The 20 column lines are driven from 20 bits of the shift register. The top 10 bits of the I/O register in the processor are inverted and buffered, e.g. by U8a and Q11. The 10 bits from the extra shift register are buffered, for example by U3a and Q10 (U3a is not an inverter, since the serial input to this extra shift register is inverted with respect to the I/O register). Thus the 20 **Col** signals can individually controlled.

The row selection circuit is on the **Printer Driver** PCB. Decoder U4 decodes 2 of the bits of the processor I/O register and enables one of the driver circuits such as Q15 and Q1. This grounds one of the **HDCom** signals. The decoder is enabled by a pair of signals **PrtHdEn/** and **PrtPulse/** from the **Printer Interface** PCB. Both must be low to enable any of the elements in the printhead. When this occurs, one row of the electrical matrix can be selected and the elements in that row turned on by the bits in the shift register.

PrtHdEn/ is asserted by U7a on the **Printer Interface** PCB when the **PrtEn** signal is asserted by the **I/O interface** PCB and the feed button is not pressed. If the paper has not run out, **PrtClkEn/** will be low, enabling monostable U10. This monostable is triggered by **PrtHdEn/**, inverted by U7c. U10 provides the **PrtPulse/** enable signal.

The motor is advanced one step by the **MotorClk** signal (we will see how this actually drives the motor in a moment), which can be produced either by the processor during printing, or by the printer electronics for manual feed operations. During normal printing, **Feed** is deasserted. The output f U9c on the **Printer Interface** PCB thus goes low when the output of U9a is high, that is when the last quarter of the line is selected. When this occurs, the motor is advanced by the trailing edge of **PrtPulse/**, gated via U9b.

For manual feeding. the microswitch in the printer mechanism applies 24V to **FeedSw**. Thus turns on Q24, asserting **Feed**. U7a is inhibited, so the printhead cannot be driven. U9d's output is forced low, enabling U10 (even if the paper has run out, so that a new roll of paper can be loaded). U7d is enabled, gating the output of the astable multivibrator Q21, Q22 into U7c and thus repeatedly triggering U10. Thus the motor circuit is repeatedly clocked and the paper is advanced.

The same signals that trigger U10 also trigger the monostable U5b which has a much longer time constant. This provided an enable signal, **MotorEn/**, to the motor drivers and reduces power dissipation by turning off the motor windings when the printer is not being used.

The motor driver circuit is on the **Printer Driver** PCB. It consists of the 2 JK flip-flops U3a and U3b. The outputs of U3a and U3b which are in phase quadrature, are buffered and connected to the stepper motor windings. If **MotorEn/** is high then the 'lower' transistor of each buffer is certain to be cut off (for example U2c will ground the base of the lower transistor of the **MA** buffer under these conditions), thus ensuring that one end of each motor winding is floating. Therefore, as mentioned earlier, the motor windings then carry no current.

The **PrtFlag** signal to the **I/O Interface** PCB is produced by U7b on the **Printer Interface** PCB. It acts as a 'ready' signal indicating the printer is ready of the next operation. Normally U5a has timed out, so **PrtFlag** is produced by the trailing edge of **PrtPulse/.** But when the last part of the pine is printed and the motor steps the paper on one line, U5a is triggered. This delays the **PrtFlag** signal until the trailing edge of U5a's output, when it times out. This gives the mechanism time to move the paper before the next line of dots is printed.

When the paper runs out, the contacts in the mechanism apply 24V to the **PaperOut** signal. This turns Q23 on, forcing the output of U9d high if the manual feed key is not pressed. The various monostables are now inhibited, which disables the printhead elements, prevents the motor from turning, and prevents **PrtFlag** from occurring. The last is used to indicate to the processor that there is no paper in the printer. As mentioned earlier, the manual feed switch will enable U9d even when **PaperOut** is asserted, enabling the motor to be used to load the new roll of paper.

2 HP9830 printer interface

The printer interface located on the HP9830 **I/O Backplane** PCB is similar in operation to an external interface plugged into one of the I/O connectors, and is quite simple

Select code 15 is decoded from the **CO** outputs by U1a. Its output goes low when the printer interface is selected. When **CEO/** is also asserted, the output of U2b goes high, clocking 7 bits of data from the I/O register into U4 and U5. The outputs of these latches are fed to the printer connector.

The output of U2b is inverted by U2d and sets the flip-flop U3a, thus asserting **PrtStb** via U2c. **PrtFlg** from the printer clears U3a when the printer is ready for further data.

The **SI(O)** input to the I/O register is pulled low by U1b when the device is selected (that is U2a's output is high because that of U1a is low) and the printer is ready for more data (U3a is cleared).

The printer signals are routed through an edge connector on the top of the I/O backplane to the 19 pin MIL-C-26482 connector on the rear of the machine.