HP 82240B Infrared Printer
Technical Interfacing Guide

HP 82240B Infrared Printer Technical Interfacing Guide

CONTENTS

Introduction

The HP 82240B Infrared Printer is a convenient 24-character portable battery-powered thermal printer designed specifically for use with certain HP calculators. Several inquires have been received regarding the use of the printer with other, specially designed equipment. This document, together with the printer owner's manual, should provide sufficient information to design and implement infrared interface circuitry that can transmit data to the printer.

While every effort has been made to insure the accuracy of the information contained herein, Hewlett-Packard Company shall bear no responsibility for its correctness, nor for any use to which it may be put.

HP 82240B Infrared Printer Manual

The HP 82240B printer manual contains information on many topics such as:

Batteries	What batteries to use and how to install them.
Control Switches	Explains the control switches on the printer.
Paper	What type of paper to use and how to load it into the printer.
Printer Positioning	How the printer should be positioned relative to the transmitter, assuming that the transmitter's IR beam has the same signal strength and pattern as a Hewlett-Packard calculator.
AC Adapter	What type of AC adapter to use; electrical specs.
Incorrectly Received Characters	A sample printout showing the character printed when the incoming byte had uncorrectable errors.
Buffer Overflow Indication	A sample printout showing the character printed when the printer's 200 byte buffer is overflowed. This is the ONLY way that this character can be printed – no character code transmitted to the printer will cause it to print this pattern.
Character Sets	The Roman8 and ISO 8859/1 character sets are shown as actually printed by the printer.
Operating Information	Battery life optimization. How to tell when the batteries are getting low.

Environmental Limits	Temperature and humidity limits for storage and for operation of the printer.

Service and Regulatory Information

Hewlett-Packard Journal, October 1987

The October 1987 issue of the Hewlett-Packard Journal contains an article titled, "An Infrared Link for Low-Cost Calculators and Printers," which gives additional information such as the reasoning behind design decisions and how the IR is decoded in the printer.

HARDWARE

Infrared Transmitter Circuitry

The transmitting element is an infrared (IR) light-emitting diode. The optimum optical wavelength for the printer is 940 nanometers (nm), but wavelengths as short as 850 nm and as long as 1000 nm can be used with only minor reductions in the distance over which data can be sent to the printer. In most applications, it is desirable to minimize the transmitter directionality; that is, the need to carefully aim the transmitting device at the printer. For this reason, an IR LED with a broad radiation pattern is preferred. One specific example of a device that satisfies these requirements is the SE303A manufactured by NEC. The peak wavelength for this device is 940 nm, the output power is typically 6.5 mW at a drive current of 50 mA, and the total angle between the half-power points of the radiation pattern is slightly over 50 degrees.

The typical drive circuit for this type of IR LED would consist of a voltage source, current-limiting resistor, the LED, and a bipolar transistor used as a saturated switch connected in series (see page xx). A circuit like this is used in the calculators with the voltage source consisting of three batteries, so that the voltage may vary from about 4.5 V down to 3.3 V. The value of the resistor is set at 18 Ω. With component and voltage variations, the current through the LED may be as low as 70 mA or as high as 180 mA. Under these conditions and with these components, the distance and directionality of the infrared link will be very similar to that obtained with calculators.

Timing Definition

The basic unit of transmission is a burst of light pulses. The burst consists of at least six and no more than eight pulses. The frequency of the pulses within the burst is 32768 Hz. Any value from about 31 kHz to 34 kHz is acceptable with minimal reduction in distance. The duty cycle should be close to 50%; duty cycles in the range of 45% to 55% are acceptable. In the nominal case, light will be emitted for a period of 15.26 μs and then no light will be emitted for the same period. This cycle repeats six, seven, or eight times to complete the pulse burst. Partial pulses should not be transmitted. This might occur, for example, if a simple 32768 Hz oscillator were gated asynchronously to drive the LED. The first and/or last pulses of the burst might be partially cut off

by the gating signal. This will cause a substantial reduction in the distance over which the interface will operate.

The data is encoded in bit times that are subdivided into two equal half-bit times. A bit time is defined as precisely 28 cycles of an exact 32768 Hz reference, about 854.5 µs. Time intervals are measured from the leading edge of the bursts. There are three kinds of bits:

- A ONE bit is defined as a burst at the beginning of the first half-bit time followed by a half-bit time with no burst.

- A ZERO bit is defined as no burst in the first half-bit time followed by a burst at the beginning of the second half-bit time.

- A START bit is defined as bursts at the beginning of three consecutive half-bit times. Note that this sequence will never occur within a stream of data bits.

Because the printer decodes incoming bursts by measuring the time interval between successive bursts, the accuracy of the burst timing is critical to the proper operation of the printer. Leading-edge jitter of the bursts should be controlled very carefully for maximum distance without errors. Within a frame, the time from the leading edge of any burst to the leading edge of a burst seven half-bit times later should be no more than the nominal value – that is, 98 cycles of an exact 32768 Hz reference, about 2990.72 µs. This time should be no less than one reference cycle less than nominal, about 2960.21 µs. This is a maximum total variation of about one percent. If a single 32768 Hz clock is used both to generate the pulses within a burst and to derive the timing of the sending of the bursts, then the 32768 Hz should not vary more then +/-50 Hz to insure that the printer will be able to reliably decode the received data.

Character Encoding and Data Flow

A complete frame contains one byte, or character, of data. A frame consists of a start bit (three half-bit times) followed by four error-correction bits and eight data bits (24 half-bit times) followed by a minimum three half-bit time delay before the start bit of the next frame. Frames are asynchronous with respect to each other, but there may be no additional delays within a frame. The total time to send a frame is then a minimum of 30 half-bit times, or about 12.8 ms. This corresponds to a maximum data rate of about 78 characters or bytes per second.

Error correction bits are encoded in the same way as data bits. Each error correction bit serves as an even parity bit for a different subset of the data bits. The following table indicates the bits (x) in each even parity group:

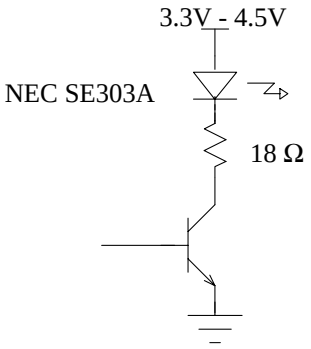
Error Correction Bits				Data Bits							
11	10	9	8	7	6	5	4	3	2	1	0
X					X	X	X	X			
	X			X	X	X			X	X	
		X		X	X		X		X		X
			X	X				X		X	X

For example, the first error correction bit (bit 11) would be sent as a ONE if the number of ONE's in bits 6, 5, 4, and 3 was odd, making the total number of ONE's in this group even. Note that the bits are sent most significant first. A complete example is in order. Suppose it is desired to send the ASCII character "A." The decimal character code for the letter A is 65. This corresponds to the binary sequence 01000001. The error correction bits would be 1101. The entire frame would be sent as a start bit followed by the twelve-bit sequence 110101000001 followed by the inter-frame delay.

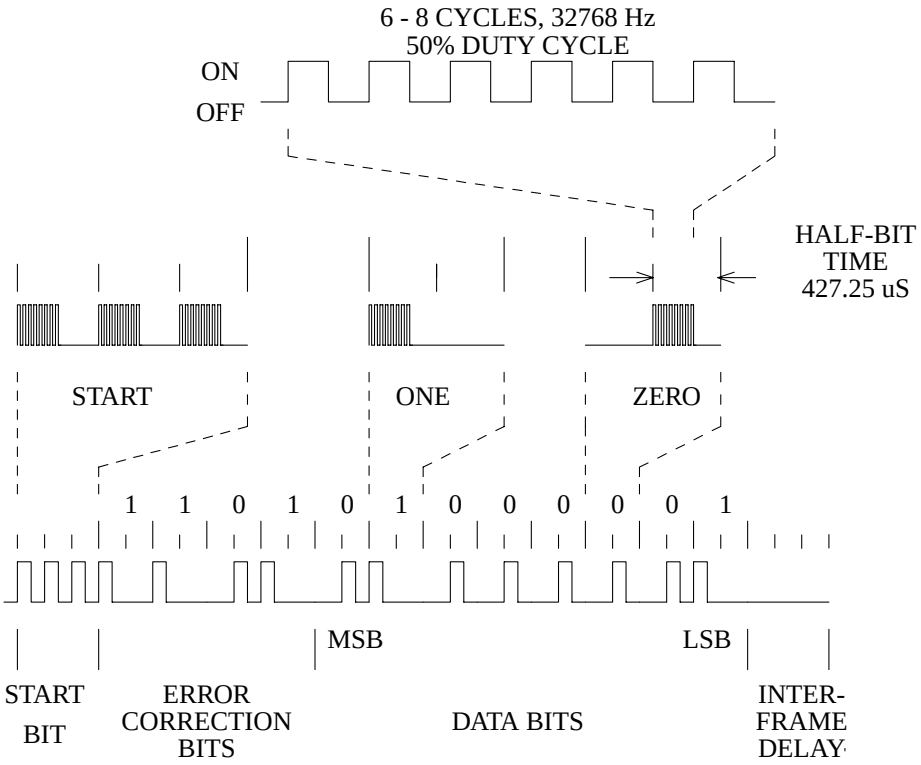
Character codes for the printer can be found in the printer owner's manual. Control codes and escape sequences to control the printer are explained in section 4, "PRINTER FIRMWARE," of this document. Please note that the time to print a line may be substantially slower than the data rate of the interface.

The printer will switch to a low-power mode after about 10 minutes of inactivity. While in the low-power mode, the printer will ignore any infrared data sent to it. Pressing the paper advance button will reactivate the printer without losing any buffered data or any printer mode settings (described below). When the AC adapter is used, the printer will not switch to the low-power mode and will remain active.

TYPICAL TRANSMITTER CIRCUIT



TIMING AND ENCODING



SOFTWARE

Higher Level Concerns

How Often to Set Printer Modes

As described below, the 82240B printer has several modes such as character set selection and single-wide vs. double-wide printing. These modes are all set to their default state every time that the printer is turned off, and the transmitter will have no way to know that this has happened. (The timeout after 10 minutes with no AC adapter will not affect these modes if the paper advance key is used to wake the printer up again.) If you are depending on some of these modes being active, you can resend the escape sequences at appropriate times, such a before each separate print operation (where the printer could have been turned off since the last time something was printed). If a print operation generates several lines of print, setting the modes once at the beginning should be sufficient.

Two Types of Linefeeds

The printer recognizes two control codes as linefeeds. One is the normal ASCII linefeed (character code 10), which causes previously sent data bytes to print and leaves the printhead on the left side of the paper. The second code recognized as a linefeed is character code 4, which also causes previously sent data bytes to print but leaves the printhead at the right side of the paper (at the end of the line just printed).

The reason for the second type of linefeed was to try to consistently start printing on the same dot column position on the paper; on some printers, this starting print position tends to jitter back and forth by one column. This position is determined by signals from the print mechanism, which are not very consistent on some mechanisms when the printhead is started up from the left. These signals appeared to be more consistent when the printhead was started up from the right side of the paper. This is especially important for graphics printing, so the HP calculators that print graphics always use the second linefeed for graphics.

There is a fairly high probability that if the user switches from one type of linefeed to the other, the two lines where the switch was made will start on different dot columns. One simple solution is to insert a blank line (where the dot column registration won't matter). Another solution is to always use the second type of linefeed, although this will obscure the last couple of characters on the line just printed if it was a full line.

All data sent to be printed must be terminated with one of the two linefeeds. Data sent without a linefeed will be stored in the print buffer but will not print until a linefeed is received, even if the user presses the printer's paper advance key. This stored data will be lost if the printer is turned off.

Byte-Level Transmission Timing

This section explains how to time the transmissions at the byte level. For the sake of brevity, the transmitting device will be referred to as the "transmitter" and the HP 82240B printer will be called simply the "printer."

The reason that timing is crucial is that the printer's infrared (IR) interface is a one-way link. The printer has circuitry to receive IR but there is no circuit allowing IR to be transmitted back. As a result, the transmitter has no direct means to tell when the printer can accept more information. It is the transmitter's responsibility to be sure that the printer's 200-byte buffer never overflows. Data can be transmitted at the maximum rate up to the point that the printer's buffer is full; after that point, take into account the rate at which the printer empties its buffer. This rate varies from printer to printer and also with battery voltage, so if the transmitter must work with any printer the transmitter must assume the worst (slowest) case. If the transmitter makes full use of the printer's buffer, the printer will still print at full speed for the first buffer's worth of lines but will eventually pause between lines as the transmitter usually will be transmitting more slowly than the printer can actually print.

Time to Print a Line

The printhead always traverses the whole line, even for a single linefeed character so full lines and very short lines all take the same amount of time to print. The maximum time to print a line (the slowest printer with mostly discharged batteries) is given below:

With AC Adapter	1.2 seconds
No AC Adapter	1.8 seconds

Remember that a line doesn't start printing until a linefeed is sent, and even then no bytes will be removed from the buffer until the motor moves the printhead to the starting position for print. This can be a substantial time if the printhead was stopped on the righthand edge of the paper (due to the previous linefeed being the 04 character) because printing starts on the left. It's safest to wait for the whole line to print rather than trying to estimate how many bytes of a line have printed part way through.

Plan for 1.2 second line times if the AC adapter will always be used. If it is uncertain whether the adapter will be used or only batteries will be used, plan for 1.8 second line times.

Byte Counting

Keep track of the total number of bytes in the printer's buffer to avoid overflowing it. Once the buffer is full, additional data must wait for a line to print before being sent. Because it is necessary to know how much buffer space is freed when the line prints, also keep track of the number of bytes in each line sent to the printer. The examples below give some ways to do this.

Two General Approaches to Line Timing

If virtually the same thing will be printed each time, such as a standard information form or template with only the values changed but not the overall size, it may be best to embed any required delays into the code that generates the form.

Another more flexible approach is to let higher-level routines generate the data to be printed with no delays but send the data through a lower-level routine that waits when necessary.

Examples of Timing Methods

Simple Text Timing

The simplest technique for text is to treat every line as if it was 25 bytes long (24 data bytes plus a linefeed of either type). This will often be the case if the name/description of something is printed on the left side of the paper and its value (a number, for example) on the right side. Break longer lines into 24 byte lines, each followed by a linefeed. Because every line is treated as being the same length (some could be shorter), it is only necessary to keep track of the number of lines sent. Before sending each line, check whether the printer's buffer is full (it can hold eight 25 byte lines) and wait if necessary.

One timing approach is to store the time that the last line was sent out and how many lines were in the printer's buffer then. Before each new line is sent out, check the stored time against the current time and subtract from the line counter the number of lines that have printed in that time span. If no lines have printed and the line counter equals eight, wait until a line has printed and decrement the line counter. Then send out the new line, set the stored time to the current time, and increment the line counter.

Another approach is to use a timer or timer interrupt to decrement a counter once every line time (1.8 or 1.2 sec), stopping when the counter decrements to zero. Before sending each line, check that the counter is less than eight (waiting if necessary) and then increment the counter after sending the line.

More Efficient Text Timing

If many of the text lines are substantially less than 24 bytes long, make better use of the printer's 200-byte buffer by keeping track of the actual number of bytes in each line. One way to do this is to use a FIFO where each FIFO entry is the number of bytes in a line. Because the maximum number of bytes in a line is 200, each FIFO entry can be a single byte. When enough time has passed so that the oldest line represented in the FIFO has been printed, it will be clear exactly how much space has been freed in the printer buffer. Also be sure that the total number of bytes does not exceed 200.

If a "line" of text to be printed may contain linefeeds, the timing routine should check for them as the printer will print each data sequence ended with a linefeed as a separate line of print. To get the timing right, consider how many actual lines of print will be generated and how many bytes will be on each line.

Timing Lines of Graphics

A line of graphics consists of an escape character followed by a byte containing the number (max = 166) of graphics data bytes, followed by that number of data bytes, followed by a linefeed. Unlike text, one line of graphics may fill most of the printer's 200-byte buffer but still print in only one line time. If lines of text and lines of graphics are mixed, keep track of this difference.

For graphics lines of more than 140 dot columns, the time to transmit the bytes to the printer (12.82 msec/byte) is greater than the time required to print the graphics (1.8 sec). Printing speed when printing several such successive lines of graphics can be enhanced by beginning to send the next line of graphics when the first line has actually started printing, but remember that printing won't start until the printhead reaches the proper starting position on the left side of the paper – which h may take a while when using the 04 type linefeed. To guard against buffer overflow, it's best to leave a safety margin of empty buffer space.

When using the simple text timing technique outlined above (treat all lines as 25 bytes), a special code will be needed to handle graphics. One way is to allow at most one line of text and one line of graphics in the buffer at the same time.

When using the FIFO approach described previously ("More Efficient Text Timing"), graphics lines will fit in very well; the only difference is that they will have more bytes in a line.

Timing More Complicated Print

The printer is capable of printing text and graphics – both single-wide and double-wide – potentially all mixed together on the same line. When printing double-wide, each dot column is printed twice so only half as much can be printed on one line. The escape sequences needed to switch printer modes must be included in the byte count for each line even though they do not print. If it is certain that a line of print handed to the transmitter will always fit on one line, use the FIFO approach described previously where a simple byte counter can give the total number of bytes for each line.

PRINTER FIRMWARE

This section is for those who wish to know more about the internal workings of the printer. One case in which this knowledge is important is if a "line of print" may sometimes overflow onto a second line on the printer, which can require a more complicated algorithm to actually count the number of dot columns of print each byte generates, mimicking the printer's internal decision about when to overflow to a new line. This would be necessary in order to determine how many bytes will actually go on the current line and how many will overflow to the next line. In order to get the

right dot column count for each byte it may be necessary to decode some escape sequences, such as single-wide, double-wide, and graphics.

Control Codes and Escape Sequences

The printer recognizes only three control characters, which are given below. All other control characters are ignored. Because the print mechanism advances the paper every time the printhead travels from right to left, only linefeeds are needed and the carriage return is ignored.

4	Alternate linefeed. Print the line and leave the printhead at the right. Be sure to use this for graphics, and at least for one line before the graphics.
10	ASCII linefeed. Print the line and leave the printhead at the left.
27	Escape. Signals the beginning of one of the escape sequences listed below.

The escape sequences recognized by the printer are listed below. The effects of the "mode" escape sequences do not end at the end of the line but remain in effect until something is done to change the mode.

27 255	Reset the printer. This sets all printer modes back to defaults and repositions the printhead to the left side of the paper (printing a blank line in the process). Use it with care as the printer will not see characters transmitted during the reset process.
27 254	Start self-test. This is intended primarily for printer testing, so the self-test repeats continually until the printer is turned off.
27 253	Start printing double-wide. In this mode all dot columns are printed twice regardless of whether text or graphics is being printed. Consequently only half as much data will fit on a line. This mode is turned off by three things: the single-wide escape sequence, the reset escape sequence, or turning off the printer.
27 252	Start printing single-wide (normal width). This is the default mode.
27 251	Start underlining. In this mode the bottom dot is printed on every dot column (drawing a line across the paper) regardless of whether text or graphics is being printed. This mode is turned off by three things: the stop underlining escape sequence, the reset escape sequence, or turning off the printer.
27 250	Stop underlining. This is the default mode.
27 249	Start printing in the ISO 8859 Latin 1 character set. This escape sequence is recognized only by HP 82240B printers, not by the older HP 82240A printers (which ignore it). This mode is turned off by three things: the Roman8 escape sequence, the reset escape sequence, or turning off the printer.

27 248 Start printing in the Roman8 character set (the only character set on the older HP 82240A printers). This escape sequence is recognized only by HP 82240B printers, not by the older HP 82240A printers (which ignore it). This is the default mode for HP 82240B printers.

27 x dd..d Print "x" dot columns of graphics where "x" is a byte containing the number of graphics data bytes that will follow and "d" represents a data byte. "x" can be any value from 1 to 166 and must be followed by exactly that many data bytes. If too many data bytes are sent, the excess ones will be printed as text; if too few data bytes are sent, the first part of the next thing printed will be interpreted as graphics data bytes until a total of "x" data bytes have been received. The only way to get the printer out of this state (aside from properly finishing the graphics sequence) is to turn it off.

The data bytes each contain the dot pattern for a single eight-dot high column of graphics. The most significant bit of each byte represents the bottom dot.

Print Width and Character Representation

The printer can print a line that is 166 dot columns wide by 8 dots high. Each text character is printed on a five-wide by eight-high dot matrix with a blank eight-dot high column on either side. For characters without descenders, the bottom row of the 5x8 matrix is blank. The total character cell size is 7 dots wide by 8 dots high. 24 single-wide (12 double-wide) characters are printed on a line by skipping the leading and trailing blank columns.

IR Receive

The IR receive section of the printer firmware decodes the incoming IR and places the resulting bytes in the printer's buffer. See the October 1987 issue of the Hewlett-Packard Journal, page 19, for more information about how this is done. When either type of linefeed is received, the IR receive code increments the linefeed counter, which is the signal to the print code to start printing. If the printer's buffer is overflowed, the IR receive code inserts a special overflow byte in the buffer as soon as space is available and throws away all incoming bytes until a linefeed is received. The intent is to restart the printing cleanly at the beginning of a line.

Printing

When the print code sees a non-zero linefeed counter, it starts the motor and waits for the printhead to get to the dot column position where printing begins. The print code reads bytes from the print buffer, decoding escape sequences and storing mode settings, and printing the data represented in the buffer.

A count is kept of the total number of dot columns of print generated on the current line; as each new byte is read from the buffer, it is checked to see whether it will fit on the line or must overflow to the next line. A graphics data byte generates one dot column of print, while a text byte normally

generates seven dot columns of print unless it is the first or last byte on a line, in which case it only generates six dot columns. (The leading blank column on the first text character of a line and the trailing blank column on the last text character of a line are skipped as described in "Print Width and Character Representation".) For double-wide print, the number of dot columns is doubled. Unlike a text byte, a double-wide graphics byte may split across the end of a line, where the first dot column prints at the end of the current line and the second dot column prints at the beginning of the next line.

When a graphics escape sequence is seen, the byte count is saved and used to determine when the sequence is finished. All bytes are received as data during a graphics sequence, including bytes that would normally be seen as linefeeds or escape sequences, so there is no way to get the printer out of a graphics sequence except by turning it off and on or by completing the sequence properly. If users of the transmitter can interrupt the transmission of a graphics sequence, they will have problems with the next thing they try to print unless enough null bytes are sent to finish the sequence. It's better to allow interruptions only between graphics sequences, not during one.

If the users can interrupt a text transmission, other problems may surface, as any text sent to the printer without a terminating linefeed will not print at the time but will be printed before the next line of data that is sent to the printer. If the user is allowed to interrupt text transmissions, send a linefeed when an interruption occurs.

When a linefeed is read from the buffer, the linefeed counter is decremented and the printhead is moved to the appropriate side of the paper. If the linefeed counter is now zero, the printhead is stopped; otherwise, it goes on to print the next line.