

Performance Note for the HP 12040B Multiplexer

This note is written in response to numerous requests for performance and application information on the HP12040B multiplexer (MUX). It should help you decide how to apply the MUX in a given situation. We won't provide detailed information on each possible application -- there are too many permutations for that; rather, we will give you a few benchmarks and guidelines that should be useful in the majority of cases.

The first section of this note discusses MUX performance in general terms for the reader who wants a high-level summary of the subject. The remainder of this note is aimed at the technically sophisticated user and treats MUX performance in considerable detail.



General Information

Traditionally, MUX performance is discussed in terms of aggregate throughput. The throughput figure bandied about for the MUX is usually 76800 baud (which is equivalent to 8 channels running at 9600 baud or 4 channels running at 19.2k baud). While we will make use of this approach in this note, please be aware that such specifications can be misleading. For example, the MUX can attain an aggregate transfer rate of 76800 baud for short bursts, but it cannot maintain that rate continuously.

As a rule of thumb, the MUX can meet any two of the following three specifications at one time:

- 1) full number of channels (8 channels at 9600 baud, 4 channels at 19.2k baud, or some suitable combination)
- 2) full transfer rate on each channel used (9600 baud, 19.2k baud, or whatever is appropriate)
- 3) continuous data transfer

Note that for most of these cases the above rule of thumb assumes the most favorable conditions for the data transfers. The MUX throughput is affected by the record size used, the method of record termination, buffering, edit and echo settings, and the use of a device driver; we will discuss the effects of these factors later in this note.

Your real concern in all this is whether an application will work in the way you want. Table 1 shows some of the most common applications and gives some comments on how well they work. Note that the specified conditions stack the deck in favor of success as much as possible; less favorable situations are discussed in the text that follows.

Table 1. MUX Application Summary

I N P U T	Assumptions: end on character, unbuffered read, edit and echo off; not using device driver DD.00. Target: 8 devices at 9600 baud (or equivalent).	
Input device	Remarks	
Terminal (character entry from keyboard)	This works fine with no data loss, primarily because nobody types at 9600 baud. The MUX is at its best as a terminal multiplexer. (This application uses the full number of channels at the full transfer rate, but does not involve continuous transfer of data.)	
Terminal (block mode entry, using terminal handshake)	This works well, with no data loss, as long as the block size is not more than 254 bytes. After each input block is sent, the terminal waits for the host to acknowledge receipt of the data before sending the next block. Since the host must finish reading the block in order to acknowledge it, the input buffers of the MUX are always emptied before the next input, and no data overflow will occur during that next input. Typical applications are data entry programs and screen editors. (Full number of channels, full transfer rate, not continuous transfer.)	
Black box (with handshake)	This application works fine, as long as the block size does not exceed 254 bytes. In such an application, the black box inputs a block of data and waits for some sort of acknowledgment from the host computer before inputting the next block. The host computer may perform a checksum or CRC test to verify acknowledgment. Since the host computer empties the MUX input buffers as it reads the input data, those buffers are left ready for the next input, and no overflow (or data loss) will occur during the next input. (Full number of channels, full transfer rate, not continuous transfer.)	

Black box (no handshake)	<p>This application is a potential troublemaker. It assumes a full number of black boxes sending continuous data at full transfer rates, with no pauses for acknowledgments. As such, it won't work without losing data. To prevent data loss, you must cut back either the transfer rates or the number of black boxes, or you must insert delays between each block of data. For example, you could input to 6 channels continuously at 9600 baud without losing data, as long as the blocks were between 160 and 254 bytes. Alternatively, you could keep 8 channels going at 9600 baud (with block sizes between 1 and 254 bytes) if you inserted an 80-millisecond delay after each block.</p>
--------------------------	---

O U T P U T	<p>Assumptions: unbuffered write; not using device driver DD.00; ENQ/ACK handshaking in effect. Target: 8 devices at 9600 baud (or equivalent).</p>	
Output device	Remarks	
Terminal, printer, or black box.	<p>This application works well. There is no data loss on output, because the interface driver waits until an output buffer is available before sending the next block. Because of limitations inherent in the MUX subsystem, this application is limited to a maximum of 6 channels of continuous output at 9600 baud, or 8 channels at an effective continuous rate of about 9300 baud, depending on the block size used.</p> <p>(This lower effective rate generally is not a problem with terminals and printers, since it is difficult to see the difference between 9300 and 9600 baud in most such applications. In addition, while data transfer may occur at 9600 baud, some terminals and printers do not display the information that fast. Also, many terminals and printers have some sort of handshake scheme to ensure data integrity and protect against data overflow, and such device limitations prevent the device from achieving continuous transfer at 9600 baud.)</p>	

In the following paragraphs we will discuss the factors affecting MUX throughput in more detail.

Input Considerations

Processing power for the MUX is provided by the Z-80 processor unit on the MUX card. (We will refer to that processor as the ZPU, to distinguish it from the host CPU.) The ZPU handles transfers across both the frontplane and the backplane of the MUX. Given the right conditions, it can provide the full transfer rate on either the frontplane or the backplane, but it runs into limitations when dealing with both at the same time (as in the black box - no handshake case).

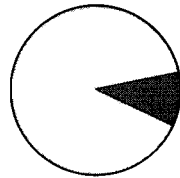
Frontplane inputs are handled as interrupts, and as such have a high priority in competing for the ZPU's processing power. Table 2 shows how much time it takes the ZPU to handle each character that comes in across the frontplane; the time varies according to the ending and editing options in use. Assuming the most favorable conditions (end on character, edit and echo off, no buffer switching), it takes approximately 76% of the ZPU's time to handle the input from 8 channels running continuously at 9600 baud.

Table 2. ZPU Interrupt Service Routine Times

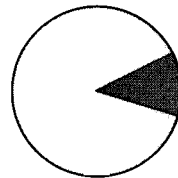
98 us	Input (end on character, edit and echo off)
135 us	Input (end on character, edit and echo on)
125 us	Input (end on count, edit and echo off)
156 us	Input (end on count, edit and echo on)
273 us	Input (end on character, edit and echo off, buffer switch)
335 us	Input (end on character, edit and echo on, buffer switch)
297 us	Input (end on count, edit and echo off, buffer switch)
332 us	Input (end on count, edit and echo on, buffer switch)
75 us	Output (ENQ/ACK off)
90 us	Output (ENQ/ACK on)
16 us	Timer
41 us	Backplane interrupt (NMI)

Backplane transfers between the MUX card and the host computer are background processes; they take place in the time left over after the frontplane interrupts (and any other interrupt-driven processes) have completed. Figure 1 shows the amount of time available for backplane operations under some typical input conditions. If the MUX is receiving continuous inputs at the full rate (8 * 9600 baud), the ZPU simply does not have enough processing time left over to transmit all of the input data across the backplane to the host. (In addition to the ZPU time taken to prepare the data for shipment across the backplane, the DMA transfer process consumes ZPU processing time. Thus, while DMA transfers are in progress, the ZPU's effective performance drops to about 34% of normal, on the average.) The net result is that the input buffers overflow and incoming data are lost.

Z80 CPU UTILIZATION
 READ OPERATION (EDIT/ECHO OFF)

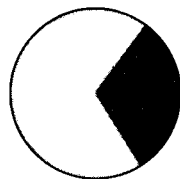


90%/10%

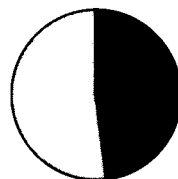


88%/12%

1 PORT

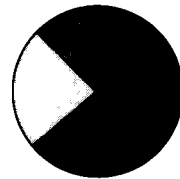


62%/38%

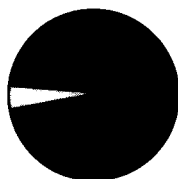


52%/48%

4 PORTS



24%/76%




4%/96%

8 PORTS

END ON CHARACTER

END ON COUNT

 % OF CPU
 AVAILABLE FOR
 BACKPLANE


 % OF CPU
 USED BY
 FRONTPLANE

Figure 1. Z-80 CPU Utilization on Input

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

The above discussion was based on the most favorable input conditions. The next most favorable case (end on count, edit and echo disabled, no buffer switch) uses 96% of the ZPU's time for handling the characters coming in over the frontplane (assuming continuous input on 8 channels at 9600 baud). Still more time is required for input if you enable edit and echo (the feature that echoes characters on your terminal screen and lets you backspace and re-type characters), as indicated in Table 2. And the 254-byte capacity of the MUX input buffers means that block sizes larger than 254 bytes will force a buffer switch, taking even more time.

So far we have talked about the throughput limitations of the MUX card itself. Operating system considerations in the host computer also affect the throughput; these considerations include drivers and system buffering. Any extra time taken by the operating system in these areas decreases the effective throughput of the MUX.

All devices connected to the MUX use interface driver IDM00. Use of this driver alone is appropriate for dumb terminals and black boxes, and allows full use of the 254-byte buffers on the MUX card. If you add device driver DD.00 in series with IDM00 (say, to take advantage of the features of a smart terminal), the block size is limited to 78 bytes. This means that any data transfers longer than 78 bytes must be broken down into smaller blocks, with a separate call to IDM00 for each block. Thus, you can incur the overhead of repeated accesses to IDM00, in addition to the overhead of using DD.00, if you make large-block-size transfers using DD.00. For example, a 240-byte read using DD.00 causes one call to DD.00 and four calls to IDM00. The same read made without using DD.00 would cause only one call to IDM00.

System buffering (through SAM) takes extra time because it requires two data transfers (card-to-SAM and SAM-to-program), as well as system overhead for allocating the SAM buffer. Unbuffered input requires only one transfer (card-to-program), and does not incur any buffer allocation overhead.

As you may have guessed, it is very difficult to predict the exact throughput of a given configuration, since the throughput is affected by factors that can't be known with certainty. These factors may be controlled by the card (such as the timing of backplane processing and DMA transfers), or they may be controlled in the host CPU (CPU load and, in the case of buffered input, SAM size and SAM fragmentation). However, we expect that the figures in the next few tables will provide some useful indications of the throughput that you can expect.

Up to this point we have discussed several factors that have an effect (mostly negative) on input transfers. The situation is not as negative as it seems, however, because of the nature of the applications that use the MUX. In the most common case, the MUX is used in character mode with users entering data from a keyboard. In such a situation the frontplane is not kept fully loaded, and the MUX will handle up to 8 users at 9600 baud (or 4 users at 19.2k baud, or some equivalent configuration).

In another common class of applications, each device sends a block of data to the host computer, waits for an acknowledgment, and then sends another block. Almost all program development terminals and many kinds of black boxes send data in this way. Table 3 gives the results of empirical testing in this environment.

Table 3. Block Inputs Using Acknowledgment

Test conditions: simultaneous input on the specified number of channels at 9600 baud, edit and echo off, unbuffered, no device driver, using 1 stop bit and no parity, on an A700 processor.					
Block ends on:	Character		Count		
Block size (bytes)	1-254	255-508	1-10	11-254	255-508
Number of support-able inputs	8	7*	8	7*	6*
<p>* The number of devices can be increased to 8 by using a lower baud rate or by not doing simultaneous transfers on all 8 ports.</p> <p>Although measurements were not made for block sizes larger than 508 bytes, it is estimated that the MUX will support simultaneous inputs from 5 or 6 channels for block sizes between 509 and 2048 bytes.</p>					



In a third class of applications, black boxes send data blocks continuously without waiting for host acknowledgment after each block. At high baud rates, and at certain block sizes, input overflow and consequent loss of data can occur. This data loss can be avoided by reducing the baud rate, reducing the number of input channels, or inserting delays after each block of data. Tables 4, 5, and 6 show the results of continuous data transfer tests made at 9600 baud. The numbers in the tables show the inter-block delays required to avoid overflow for given block sizes and numbers of channels.

Table 4. Continuous Input, Blocks Ending on Character

Test conditions: simultaneous input on the specified number of channels at 9600 baud, unbuffered, edit and echo off, no device driver, using 1 stop bit and no parity, on an A700 processor.								
Number of simultaneously active ports	1	2	3	4	5	6	7	8
Delay (in milliseconds) required to prevent overflow between blocks of:								
1-10 bytes	10	20	30	40	50	60	70	80
20 bytes	0	0	20	30	40	50	60	80
40 bytes	0	0	0	20	30	50	60	70
60 bytes	0	0	0	0	20	40	60	70
80 bytes	0	0	0	0	10	30	50	70
120 bytes	0	0	0	0	0	20	40	60
160 bytes	0	0	0	0	0	0	30	60
200 bytes	0	0	0	0	0	0	20	60
250 bytes	0	0	0	0	0	0	20	70
260 bytes	20	30	40	50	60	70	80	*
300 bytes	0	0	0	0	0	60	70	*
360 bytes	0	0	0	0	0	40	60	*
500 bytes	0	0	0	0	0	0	30	*
* Overflow can occur during simultaneous transfers at 9600 baud even with a delay between blocks.								
A buffer switch occurs at the beginning of each input block. An additional buffer switch takes place if the block size exceeds 254 bytes. These buffer switches take extra processing time; the extra time is reflected in the delays shown in this table.								

Table 5. Continuous Input, Blocks Ending on Count

Test conditions: simultaneous input on the specified number of channels at 9600 baud, unbuffered, edit and echo off, no device driver, using 1 stop bit and no parity, on an A700 processor.								
Number of simultaneously active ports	1	2	3	4	5	6	7	8
Delay (in milliseconds) required to prevent overflow between blocks of:								
10 bytes	10	10	20	30	40	50	60	*
20 bytes	0	0	20	30	40	50	60	*
40 bytes	0	0	0	20	30	50	60	*
60 bytes	0	0	0	10	30	40	60	*
80 bytes	0	0	0	0	20	40	60	*
120 bytes	0	0	0	0	5	30	60	*
160 bytes	0	0	0	0	0	30	40	*
200 bytes	0	0	0	0	0	20	*	*
250 bytes	0	0	0	0	0	0	*	*
260 bytes	20	30	40	50	60	70	*	*
300 bytes	0	0	0	30	40	60	*	*
360 bytes	0	0	0	0	15	50	*	*
500 bytes	0	0	0	0	0	0	*	*
* Overflow can occur during simultaneous transfers at 9600 baud even with a delay between blocks.								
A buffer switch occurs at the beginning of each input block. An additional buffer switch takes place if the block size exceeds 254 bytes. These buffer switches take extra processing time; the extra time is reflected in the delays shown in this table.								

Table 6. Continuous Input, Blocks Terminated by Host Computer

Test conditions: simultaneous input on the specified number of channels at 9600 baud, unbuffered, edit and echo off, no device driver, using 1 stop bit and no parity, on an A700 processor.								
Number of simultaneously active ports	1	2	3	4	5	6	7	8
Delay (in milliseconds) required to prevent overflow between blocks of:								
1-10 bytes	0	0	0	0	0	10	*	*
20 bytes	0	0	0	0	0	10	*	*
40 bytes	0	0	0	0	0	10	*	*
60 bytes	0	0	0	0	0	20	*	*
80 bytes	0	0	0	0	0	20	*	*
120 bytes	0	0	0	0	0	30	*	*
160 bytes	0	0	0	0	0	40	*	*
200 bytes	0	0	0	0	0	50	*	*
250 bytes	0	0	0	0	0	60	*	*
260 bytes	0	0	0	0	0	*	*	*
300 bytes	0	0	0	0	0	*	*	*
360 bytes	0	0	0	0	0	*	*	*
500 bytes	0	0	0	0	0	*	*	*
* Overflow can occur during simultaneous transfers at 9600 baud even with a delay between blocks.								

All of our testing was done at 9600 baud. If you are transferring data at 19.2k baud, you can, as a first approximation, simply divide the numbers of ports in the above tables by 2 and use the same delay figures.

Output Considerations

A couple of factors make the treatment of output simpler than that of input. First, output is faster than input. The time it takes the ZPU to output a character over the frontplane is 75 us, significantly less than the 98 us of the best input case. Second, there is never any data loss on output, because the driver does not send a block of data to the MUX card unless a buffer is available on the card.

Operating system considerations have the biggest effect on MUX output transfer rates. The use of device driver DD.00 has the same effect on output as it has on input: it limits transfers to 78 bytes. This results in additional driver accesses, with consequent lower transfer rates, for block sizes larger than 78 bytes. Similarly, buffering (through SAM) on output has the same effect as buffering on input: extra data transfers and reduced throughput.

On the level of the MUX card itself, the use of an ENQ/ACK handshake to prevent output data from overflowing a target device also cuts the effective throughput of the MUX. (While this is actually a limitation imposed by the device and not by the MUX, the net result is still reduced throughput.) We feel that ENQ/ACK should generally be used, as terminals may not be able to accept data continuously at 9600 baud.

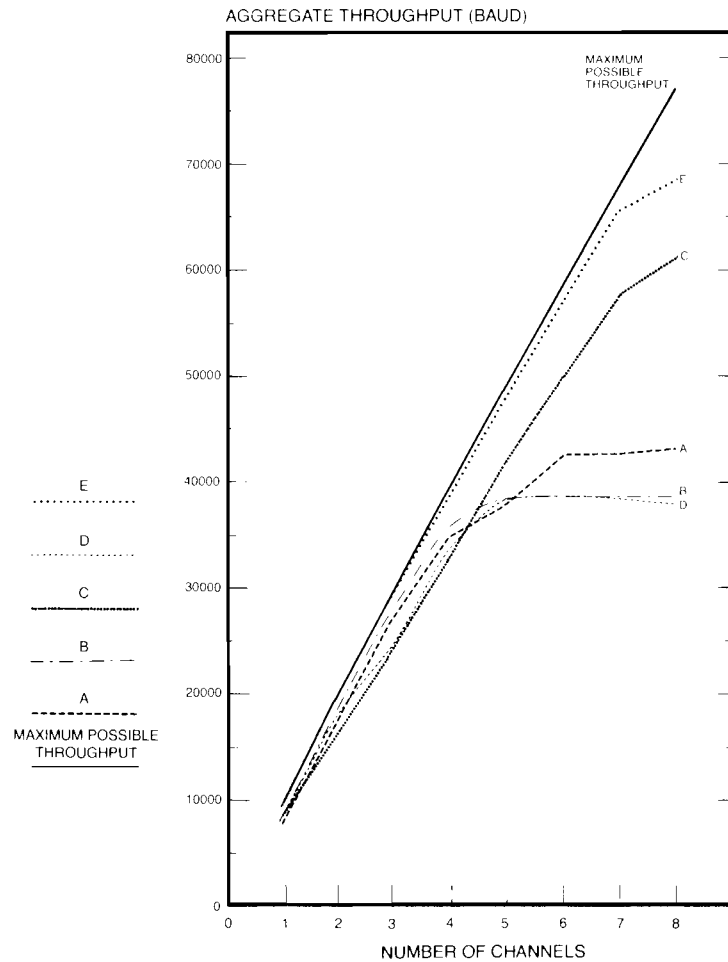
The best-case combination for output transfers by the MUX uses block sizes of 254 bytes and does not use either system buffering or device driver DD.00. This combination is able to attain continuous transfer at 9600 baud on 6 channels, or transfer on 8 channels at an average rate of 9300 baud. The graphs in Figure 2 indicate the relative performance of different combinations of drivers and buffering.

Host CPU Utilization

If your computer is to do more than just I/O processing, it must be fast enough to handle all of the input and output transfers and still have time to do other things. The graphs in Figure 3 show how much time an A600 processor has left over after making output transfers through one MUX under the indicated conditions. Input transfers normally place similar loads on the processor, and combinations of input and output will simply be the sums of the input and output activities. CPU utilization on an A700 processor is approximately the same as that on an A600, since the I/O handling on both processors is the same. (The A700's general performance advantage results from the number crunching power of its hardwired floating point processor, and so it doesn't affect I/O capacity.) An A900 will have significantly more time left over after performing the I/O tasks, as indicated by the graphs.

Test conditions: continuous transfer on the specified number of channels, with ENQ/ACK, using 512-byte blocks, on an A600 processor

- A: using device driver DD.00, unbuffered
- B: using DD.00, buffered
- C: not using DD.00, unbuffered
- D: not using DD.00, buffered
- E: same as C but no ENQ/ACK

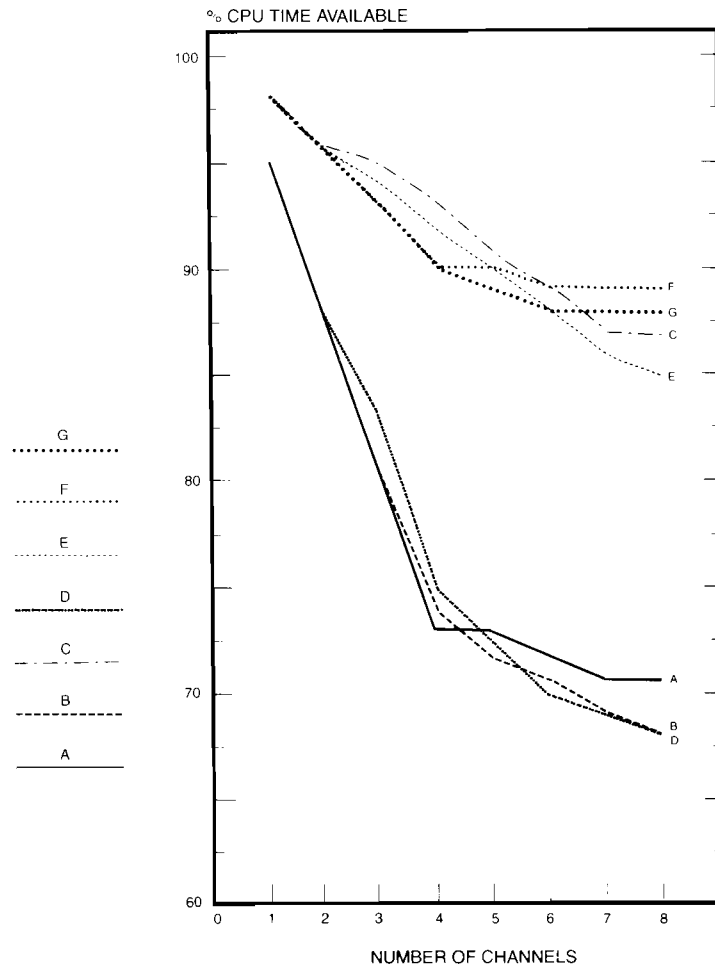


The 512-byte block size was selected to show performance in a typical situation. Optimum performance is obtained when the block size is a multiple of 254. Therefore, these figures should be taken as indicators of relative performance under different conditions, rather than as indicators of the maximum output performance of the MUX.

Figure 2. MUX Output Performance

Test conditions: continuous transfer on the specified number of channels, with ENQ/ACK, using 512-byte blocks, on an A600 processor

- A: using device driver DD.00, unbuffered
- B: using DD.00, buffered
- C: not using DD.00, unbuffered
- D: not using DD.00, buffered
- E: same as C but no ENQ/ACK
- F: same as A but on A900
- G: same as B but on A900



The 512-byte block size was selected to show performance in a typical situation. Optimum performance is obtained when the block size is a multiple of 254. Therefore, these figures should be taken as indicators of relative performance under different conditions, rather than as indicators of maximum performance.

Figure 3. Host CPU Utilization



HEWLETT
PACKARD

PART NO. 12040-90021
Printed in U.S.A.
September 1983

HEWLETT-PACKARD COMPANY
Roseville Network Division
8000 Foothills Boulevard
Roseville, CA 95678