error control bits, then sends an acknowledgment packet (ACK) back to the host. The hub forwards the token and data packets downstream. All I/O devices receive this sequence of packets, but only the device that recognizes its address in the token packet accepts the data in the packet that follows. After verifying that transmission has been error free, it sends an ACK packet to the hub.

Successive data packets on a full-speed or low-speed pipe carry the numbers 0 and 1, alternately. This simplifies recovery from transmission errors. If a token, data, or acknowledgment packet is lost as a result of a transmission error, the sender resends the entire sequence. By checking the data packet number in the PID field, the receiver can detect and discard duplicate packets. High-speed data packets are sequentially numbered 0, 1, 2, 0, and so on.

Input operations follow a similar procedure. The host sends a token packet of type IN containing the device address. In effect, this packet is a poll asking the device to send any input data it may have. The device responds by sending a data packet followed by an ACK. If it has no data ready, it responds by sending a negative acknowledgment (NAK) instead.

In earlier discussion, we pointed out that a bus that has a mix of full/low-speed links and high-speed links uses the split-traffic mode of operation in order not to delay messages on high-speed links. In such cases, an IN or an OUT packet intended for a full- or low-speed device is preceded by a special control packet that starts the split-traffic mode.

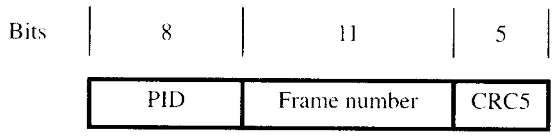
This discussion should give the reader an idea about the data transfer protocols used on the USB. There are many different ways in which such transactions take place and many protocol rules governing the behavior of the devices involved. A detailed description of these protocols can be found in the USB specification document [3].

Isochronous Traffic on USB

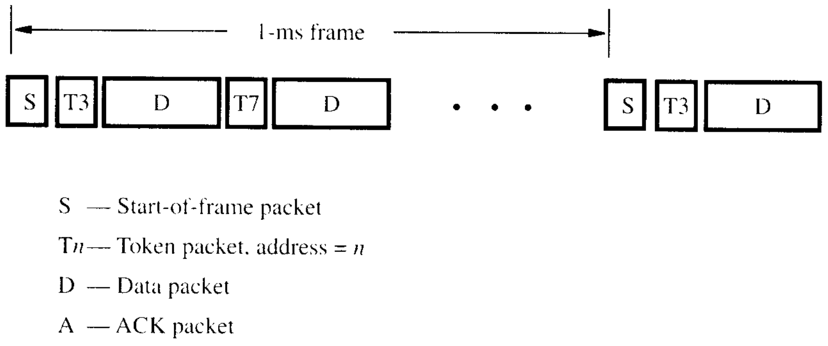
One of the key objectives of the USB is to support the transfer of isochronous data, such as sampled voice, in a simple manner. Devices that generate or receive isochronous data require a time reference to control the sampling process. To provide this reference, transmission over the USB is divided into *frames* of equal length. A frame is 1 ms long for low- and full-speed data. The root hub generates a Start Of Frame control packet (SOF) precisely once every 1 ms to mark the beginning of a new frame.

The arrival of an SOF packet at any device constitutes a regular clock signal that the device can use for its own purposes. To assist devices that may need longer periods of time, the SOF packet carries an 11-bit frame number, as shown in Figure 4.47*a*. Following each SOF packet, the host carries out input and output transfers for isochronous devices. This means that each device will have an opportunity for an input or output transfer once every 1 ms.

The main requirement for isochronous traffic is consistent timing. An occasional error can be tolerated. Hence, there is no need to retransmit packets that are lost or to send acknowledgments. Figure 4.47*b* shows the first two transmissions following SOF. A control packet carrying device address 3 is followed by data for that device. This may be input or output data, depending on whether the control packet is an IN or OUT control packet. There is no acknowledgment packet. The next transmission sequence is for device 7.



(a) SOF Packet



(b) Frame example

Figure 4.47 USB frames.

As an example, the data packet for device 3 may contain 8 bytes of data. One such packet is sent in each frame, providing a 64-kilobits/s isochronous channel. Such a channel may be used for a voice connection. The transmission of 8 bytes of data requires a 3-byte token packet followed by an 11-byte data packet (including the PID and CRC fields), for a total of 132 bits. A minimum of three more bytes are needed for clock synchronization and to mark the end of a packet sequence. At a speed of 12 megabits/s, this takes about 13 μs. Clearly, there is room in a frame to support several such devices. After serving all isochronous devices on the bus, whatever time is left in a frame is used to service asynchronous devices and exchange control and status information.

Isochronous data are allowed only on full-speed and high-speed links. For high-speed links, the SOF packet is repeated eight times at equal intervals within the 1-ms frame to create eight microframes of 125 μs each.

Electrical Characteristics

The cables used for USB connections consist of four wires. Two are used to carry power, +5 V and Ground. Thus, a hub or an I/O device may be powered directly from the bus, or it may have its own external power connection. The other two wires are used to carry data. Different signaling schemes are used for different speeds of transmission. At low speed, 1s and 0s are transmitted by sending a high voltage state (5 V) on one or the other of the two signal wires. For high-speed links, differential transmission is used.

4.8 CONCLUDING REMARKS

In this chapter, we discussed three basic approaches to I/O transfers. The simplest technique is programmed I/O, in which the processor performs all the necessary control functions under direct control of program instructions. The second approach is based on the use of interrupts; this mechanism makes it possible to interrupt normal execution of programs in order to service higher-priority requests that require more urgent attention. Although all computers have a mechanism for dealing with such situations, the complexity and sophistication of interrupt-handling schemes vary from one computer to another. The third I/O scheme involves direct memory access; the DMA controller transfers data between an I/O device and the main memory without continuous processor intervention. Access to memory is shared between the DMA controller and the processor.

Three popular interconnection standards are described, the PCI, SCSI, and USB. They represent different approaches that meet the needs of various devices and reflect the increasing importance of plug-and-play features that increase user convenience.

PROBLEMS

4.1 The input status bit in an interface circuit is cleared as soon as the input data buffer is read. Why is this important?

4.2 Write a program that displays the contents of 10 bytes of the main memory in hexadecimal format on a video display. Use either the assembler instructions of a processor of your choice or pseudo-instructions. Start at location LOC in the memory, and use two hex characters per byte. The contents of successive bytes should be separated by a space.

4.3 The address bus of a computer has 16 address lines, A_{15-0}. If the address assigned to one device is $7CA4_{16}$ and the address decoder for that device ignores lines A_8 and A_9, what are all the addresses to which this device will respond?

4.4 What is the difference between a subroutine and an interrupt-service routine?

4.5 The discussion in this chapter assumed that interrupts are not acknowledged until the current machine instruction completes execution. Consider the possibility of suspending operation of the processor in the middle of executing an instruction in order to acknowledge an interrupt. Discuss the difficulties that may arise.

4.6 Three devices, A, B, and C, are connected to the bus of a computer. I/O transfers for all three devices use interrupt control. Interrupt nesting for devices A and B is not allowed, but interrupt requests from C may be accepted while either A or B is being serviced. Suggest different ways in which this can be accomplished in each of the following cases:

(*a*) The computer has one interrupt-request line.

(*b*) Two interrupt-request lines, INTR1 and INTR2, are available, with INTR1 having higher priority.

Specify when and how interrupts are enabled and disabled in each case.

4.7 Consider a computer in which several devices are connected to a common interrupt-request line, as in Figure 4.8*a*. Explain how you would arrange for interrupts from device *j* to be accepted before the execution of the interrupt-service routine for device *i* is completed. Comment in particular on the times at which interrupts must be enabled and disabled at various points in the system.

4.8 Consider the daisy chain arrangement in Figure 4.8*a*. Assume that after a device generates an interrupt request, it turns off that request as soon as it receives the interrupt-acknowledge signal. Is it still necessary to disable interrupts in the processor before entering the interrupt-service routine? Why?

4.9 Successive data blocks of N bytes each are to be read from a character-oriented input device, and program PROG is to perform some computation on each block of data. Write a control program, CONTROL, for the 68000, ARM, or Pentium processors that will perform the following functions.

(*a*) Read data block 1.

(*b*) Activate PROG and point it to the location of block 1 in the main memory.

(*c*) Read block 2 using interrupts while PROG is performing computations on block 1.

(*d*) Start PROG on block 2, and meanwhile start reading block 3, and so on.

Note that CNTRL must maintain correct buffer pointers, keep track of the character count, and correctly transfer control to PROG, whether PROG takes more or less time than block input.

4.10 A computer is required to accept characters from 20 video terminals. The main memory area to be used for storing data for each terminal is pointed to by a pointer PNTRn, where $n = 1$ through 20. Input data must be collected from the terminals while another program PROG is being executed. This may be accomplished in one of two ways:

(*a*) Every T seconds, program PROG calls a polling subroutine POLL. This subroutine checks the status of each of the 20 terminals in sequence and transfers any input characters to the memory. Then it returns to PROG.

(*b*) Whenever a character is ready in any of the interface buffers of the terminals, an interrupt request is generated. This causes the interrupt routine INTERRUPT to be executed. After polling the status registers, INTERRUPT transfers the input character and then returns to PROG.

Write the routines POLL and INTERRUPT using either pseudocode or the assembler language of the processor of your choice. Let the maximum character rate for any terminal be c characters per second, with an average rate equal to rc, where $r \leq 1$. In method (*a*), what is the maximum value of T for which it is still possible to guarantee that no input characters will be lost? What is the equivalent value for method (*b*)? Estimate, on the average, the percentage of time spent in servicing the terminals for methods (*a*) and (*b*), for $c = 100$ characters per second and $r = 0.01, 0.1, 0.5$, and 1. Assume that POLL takes 800 ns to poll all 20 devices and that an interrupt from a device requires 200 ns to process.

4.11 Consider an I/O device that uses the vectored-interrupt capability of the 68000 processor.

(*a*) Describe the sequence of steps that take place when the processor receives an interrupt request, and give the number of bus transfers required during each of these steps. Do not give details of bus signals or the microprogram.

(*b*) When an interrupt request is received, the processor completes execution of the current instruction before accepting the interrupt. Examine the instruction table in Appendix C, and estimate the maximum possible number of memory transfers that can take place during that period.

(*c*) Estimate the number of bus transfers that can occur from the instant a device requests an interrupt until the first instruction of the interrupt-service routine is fetched for execution.

4.12 A logic circuit is needed to implement the priority network shown in Figure 4.8*b*. The network handles three interrupt request lines. When a request is received on line INTR*i*, the network generates an acknowledgment on line INTA*i*. If more than one request is received, only the highest-priority request is acknowledged, where the ordering of priorities is

$$\text{priority of INTR1} > \text{priority of INTR2} > \text{priority of INTR3}$$

(*a*) Give a truth table for each of the outputs INTA1, INTA2, and INTA3.

(*b*) Give a logic circuit for implementing this priority network.

(*c*) Can your design be easily extended for more interrupt-request lines?

(*d*) By adding inputs DECIDE and RESET, modify your design such that INTA*i* is set to 1 when a pulse is received on the input DECIDE and is reset to 0 when a pulse is received on the input RESET.

4.13 Interrupts and bus arbitration require means for selecting one of several requests based on their priority. Design a circuit that implements a rotating-priority scheme for four input lines, REQ1 through REQ4. Initially, REQ1 has the highest and REQ4 the lowest priority. After some line receives service, it becomes the lowest priority line, and the next line receives highest priority. For example, after REQ2 has been serviced, the priority order, starting with the highest, becomes REQ3, REQ4, REQ1, REQ2. Your circuit should generate four output grant signals, GR1 through GR4, one for each input request line. One of these outputs should be asserted when a pulse is received on a line called DECIDE.

4.14 The 68000 processor has a set of three lines called IPL2–0 that are used to signal interrupt requests. The 3-bit binary number on these lines is interpreted by the processor as representing the highest-priority device requesting an interrupt. Design a priority encoder circuit that accepts interrupt requests from as many as seven devices and generates a 3-bit code representing the request with the highest priority.

4.15 (This problem is suitable for use as a laboratory experiment.) Given a video terminal connected to the computer in your laboratory, complete the following two assignments.

(*a*) Write an I/O routine A that prints letters in alphabetical order. It prints two lines as follows, and then stops:

```
ABC ... YZ
ABC ... YZ
```

(*b*) Write an I/O routine B that prints the numeric characters 0 through 9 in increasing order three times. Its output should have the following format:

```
012 ... 9012 ... 9012 ... 9
```

Use program A as the main program and program B as an interrupt-service routine whose execution is initiated by entering any character on the keyboard. Execution of program B can also be interrupted by entering another character on the keyboard. When program B is completed, execution of the most recently interrupted program should be resumed at the point of interruption. Program B should start a new line as appropriate so that the printed output may appear as follows:

```
ABC
012 ... 901
012 ... 9012 ... 9012 ... 9
2... 9012 ... 9
DE ... YZ
```

To start a new line, the program needs to send two characters: CR ($0D_{16}$) and LF ($0A_{16}$). Show how you can use the processor priority to either enable or inhibit interrupt nesting.

4.16 (This problem is suitable for use as a laboratory experiment.) In Problem 4.15, when the printing of a sequence is interrupted and later resumed, the sequence continues at the beginning of a new line. It is desired to add cursor movement control functions such that when printing of a sequence is resumed, the characters are printed on a new line, at the same character position where they would have been had the interruption not occurred. Thus, the printed output would appear as follows:

```
ABC
012 ... 901
012 ... 9012 ... 9012 ... 9
           2 ... 9012 ... 9
   DE ... YZ
```

Rearrange the software you prepared in Problem 4.15 so that a third controller routine, C, is entered when interruption occurs. This routine calls program B to print the number sequence. Then, before returning to the interrupted program, the routine issues cursor movement commands as appropriate.

4.17 Consider the breakpoint scheme described in Section 4.2.5. A software-interrupt instruction replaces a program instruction where the breakpoint is inserted. Before it returns to the original program, the debugging software puts the original program instruction back in its place, thus removing the breakpoint. Explain how the debugger can put the original program instruction in its place, execute it, then install the breakpoint again before any other program instruction is executed.

4.18 The software interrupt instruction, SWI, of the ARM can be used by a program to call the operating system to request some service. The service being requested is specified in the low-order 8 bits of the instruction. Each of the services provided by the operating system is performed by a separate subroutine, and the starting addresses of these routines are stored in a table.

(*a*) Give one or more instructions that the operating system can use to copy the low-order 8 bits of the SWI instruction into a register.

(*b*) Give one or more instructions to call the appropriate service routine.

4.19 The interrupt-request line, which uses the open-collector scheme, carries a signal that is the logical OR of the requests from all the devices connected to it. In a different application, it is required to generate a signal that indicates that all devices connected to the bus are ready. Explain how you can use the open-collector scheme for this purpose.

4.20 In some computers, the processor responds only to the leading edge of the interrupt-request signal on one of its interrupt-request lines. What happens if two independent devices are connected to this line?

4.21 In the arrangement in Figure 4.20, a device becomes the bus master only when it receives a low-to-high transition on its bus grant input. Assume that device 1 requests the bus and receives a grant. While it is still using the bus, device 3 asserts its BR output. Draw a timing diagram showing how device 3 becomes the bus master after device 1 releases the bus.

4.22 Assume that in the bus arbitration arrangement in Figure 4.20, the processor keeps asserting BG1 as long as $\overline{\text{BR}}$ is asserted. When device *i* is requesting the bus, it becomes the bus master only when it receives a low-to-high transition on its BG*i* input.

(*a*) Assume that devices are allowed to assert the BR signal at any time. Give a sequence of events to show that the system can enter a deadlock situation, in which one or more devices are requesting the bus, the bus is free, and no device can become the bus master.

(*b*) Suggest a rule for the devices to observe in order to prevent this deadlock situation from occurring.

4.23 Consider the daisy-chain arrangement shown in Figure P4.1, in which the bus-request signal is fed back directly as the bus grant. Assume that device 3 requests the bus and

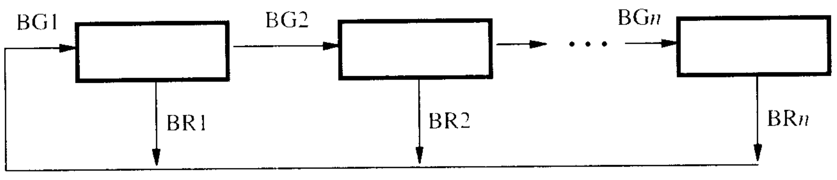


Figure P4.1 A decentralized bus assignment scheme.

begins using it. When device 3 is finished, it deactivates BR3. Assume that the delay from BGi to BG$(i + 1)$ in any device is d. Show that a spurious bus grant pulse will travel downstream from device 3 (spurious because it is not a response to any request). Estimate the width of this pulse.

4.24 Shortly after device 3 in Problem 4.23 releases the bus, devices 1 and 5 request the bus simultaneously. Show that they can both receive a bus grant.

4.25 Consider the bus arbitration scheme shown in Figure 4.20. Assume that a local signal called BUSREQ in the device interface circuit is equal to 1 whenever the device needs to use the bus. Design the part of the interface circuit that has BUSREQ, BGi, and $\overline{\text{BBSY}}$ as inputs and that generates $\overline{\text{BR}}$, BG$(i + 1)$, and $\overline{\text{BBSY}}$ as outputs.

4.26 Consider the arbitration circuit shown in Figure 4.22. Assume that the priority code for a device is stored in a register in the interface circuit. Design a circuit to implement this arbitration scheme. Arbitration begins when $\overline{\text{Start-Arbitration}}$ is asserted. A little later, the arbitration circuit should activate an output called Winner if it wins the arbitration cycle.

4.27 How would the timing diagram in Figure 4.26 be affected if the distance between the processor and the I/O device is increased? How can this increased distance be accommodated in the case of Figure 4.24?

4.28 An industrial plant uses several limit sensors for monitoring temperature, pressure, and other factors. The output of each sensor consists of an ON/OFF switch, and eight such sensors need to be connected to the bus of a small computer. Design an appropriate interface so that the state of all eight switches can be read simultaneously as a single byte at address FE10_{16}. Assume the bus is synchronous and that it uses the timing sequence of Figure 4.24.

4.29 Design an appropriate interface for connecting a seven-segment display as an output device on a synchronous bus. (See Figure A.37 in Appendix A for a description of a seven-segment display.)

4.30 Add an interrupt capability to the interface in Figure 4.29. Show how you can introduce an interrupt-enable bit, which can be set or cleared by the processor as bit 6 of the status register of the interface. The interface should assert an interrupt request line, $\overline{\text{INTR}}$, when interrupts are enabled and input data are available to be read by the processor.

4.31 The bus of a processor uses the multiple-cycle scheme described in Section 4.5.1. The speed of a memory unit is such that a read operation follows the timing diagram shown in Figure 4.25. Design an interface circuit to connect this memory unit to the bus.

4.32 Consider a write operation on a bus that uses the multiple-cycle scheme described in Section 4.5.1. Assume that the processor can send both address and data in the first clock cycle of a bus transaction. But the memory requires two clock cycles after that to store the data.

(*a*) Can the bus be used for other transactions during that period?

(*b*) Can we do away with the memory's response in this case? (*Hint:* Examine carefully the case in which the processor attempts another write operation to the same memory module while that module is still busy completing a previous request. Explain how this situation can be handled.)

4.33 Figures 4.24 to 4.26 provide three different approaches to bus design. What happens in each case if the addressed device does not respond due to a malfunction? What problems would this cause and what remedies are possible?

4.34 In the timing diagram in Figure 4.25, the processor maintains the address on the bus until it receives a response from the device. Is this necessary? What additions are needed on the device side if the processor sends an address for one cycle only?

4.35 Consider a synchronous bus that operates according to the timing diagram in Figure 4.24. The address transmitted by the processor appears on the bus after 4 ns. The propagation delay on the bus wires between the processor and different devices connected varies from 1 to 5 ns, address decoding takes 6 ns, and the addressed device takes between 5 and 10 ns to place the requested data on the bus. The input buffer needs 3 ns of setup time. What is the maximum clock speed at which this bus can operate?

4.36 The time required for a complete bus transfer in the case of Figure 4.26 varies depending on the delays involved. Consider a bus having the same parameters as in Problem 4.35. What is the minimum and maximum bus cycle time?

REFERENCES

1. *PCI Local Bus Specifications*, available at www.pcisig.com/developers.
2. *SCSI-3 Architecture Model (SAM)*, ANSI Standard X3.270, 1996. This and other SCSI documents are available on the web at www.ansi.org.
3. *Universal Serial Bus Specification*, available at www.usb.org/developers.