Department of Electrical and Computer Engineering

Computer Engineering

Course ECSE-322B

Problem Set 7 Solutions

18 February 2008

1. Two computers are to be connected together via a serial link. There are two possible choices for the protocol to be used for communication; either full RS232 or the use of Start and Stop bits.

(a) Explain the advantages and disadvantages of each protocol

Answer:
The RS232 makes maximum use of the available time on the send and receive lines because the synchronisation is handled on DTR, DSR, etc. However, it needs many electrical circuits, i.e. a large number of wires.
The Stop/Start bit protocol uses fewer lines than RS232 (basically just send and receive and ground) so it is a simpler connection. However, some of the available time on the send line is used up in the protocol so the effective data rate is lower than RS232.

(b) If the user chose a Start/Stop bit protocol with 7 bits of data, no parity and 2 stop bits, what percentage of the transmission time would be taken up in the protocol?

Answer:
1 start plus 7 data plus 2 stop = 10 bits, 7 are useful data so the protocol is using 3 out of 10 bits. Thus 30% of transmission time goes in the protocol.

(c) Given your answers to parts (a) and (b), which protocol would you recommend if the intention was to use the link to move large volumes of data continuously?

Answer:
Probably RS-232 since it is basically more reliable and does not waste time on the send and receive lines with protocol information - hence the throughput is maximised.

(d) Would you make the same recommendations if the link was to be used occasionally so that one computer could access a printer connected to the other computer? Give your reasons for this decision.

Answer:
No - on a link which is moving low volumes of data, the simple start/stop bit protocol would work well enough and is basically simpler to implement from a cabling point of view. However, if you already had a complete RS232 cable, i.e. all 25 lines, then use RS232.

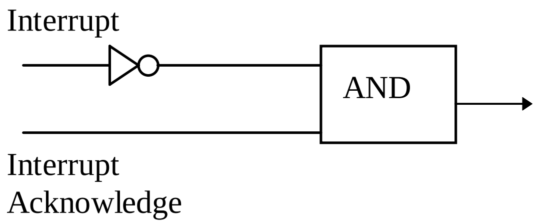
2. What is the basic difference between an interrupt priority established via a polled interrupt system and that created by a vectored interrupt system? Which of the two is likely to be more efficient and why?

Answer:
This is similar to a question on Problem set 1.

- A polled interrupt system allows more flexibility in the sense that the order in which devices are checked (the priority order) is dependent on the sequence specified in a service routine (software).
- In a vectored interrupt system (of the sort described in class) the priority is established electrically because the Interrupt Acknowledge signal is blocked by the first device which has set an interrupt on receiving it. This establishes a spatial priority structure - the nearer (in an electrical sense) a device is to the CPU, the higher its priority. As such, in a vectored interrupt system, the priority is hard to override.
- The vectored interrupt system is likely to be more efficient because all the set up for the interrupt service routine is performed in hardware running at full bus speeds.
- Also, the polled interrupt system runs the danger that it could, itself be interrupted while it is determining who interrupted (i.e. in the ISR) - the hardware structure only allows a higher priority device to break in during the period of time when the system is trying to determine which device interrupted.

3. Design the basic circuit for a daisy chained Interrupt Acknowledge circuit, i.e. the intention is that the Interrupt Acknowledge signal should be blocked by a device which has interrupted and is waiting for service, but should be passed on by any devices which have not interrupted.

Answer:



The assumption here is that when an Interrupt is set, the interrupt line is in state '1' and when the Interrupt Acknowledge is set, its line is in state '1'.

4. A low-speed I/O device provides (on average) one byte of information every second. The device is scanned by the CPU regularly every 50 ms (programmed I/O). Some analysis indicates that each time the CPU scans the device it consumes about 20 microseconds of its time. Calculate the improvement ratio (in terms of CPU time) if we equip the I/O device with interrupt capability. Assume the interrupt service routine and associated overheads amount to 40 microseconds.

Answer:

Calculate the amounts of pertinent CPU times consumed over a period of 1s = 1000ms for both cases.

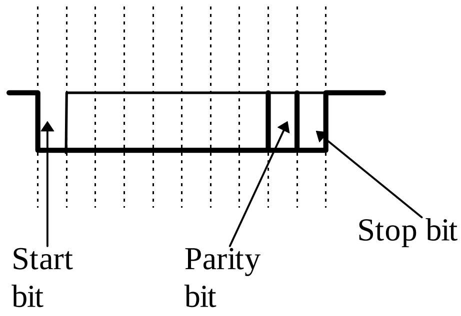
- In the programmed I/O mode: the CPU consumes time for 1000/50 = 20 scans at 20 microseconds per scan -- that is a total of 20*20 = 400 microseconds.
- If we operate in interrupt mode: the CPU spends only 40 microseconds.

This represents an improvement of (400-40)/400 = 90%.

5. An asynchronous serial communication controller (SCC) is programmed for a character length of seven bits, odd parity, and one stop bit. The transmission rate is 1200b/s.

(a) Illustrate the overall character format (including start and stop bits).

Answer:
(a) bit time interval is 1000 (ms/s)/1200 (b/s) = 0.83 ms/b.



(b) Up to how many characters can be transmitted per second?

Answer:
Each character is 10 bits long. Hence, we can transmit up to a maximum of:
1200 (b/s)/10 (b/char) = 120 characters/s.

(c) The system is designed such that the SCC causes an interrupt whenever ready to accept or deliver another character. At what rate will the CPU be interrupted if we communicate in full duplex mode?

Answer:
The composite character rate in full duplex mode can be up to 2*120 = 240 characters/s. So, the CPU could be interrupted up to that many times per second.

(d) Our system exchanges with some other system two 64-character-long blocks per second (on the average). The servicing of an SCC interrupt takes a total of about 83 microseconds. Determine (on a percent basis) what fraction of CPU time is consumed for serial I/O.

Answer:
The CPU is interrupted 2*64 = 128 times per second. That means spending 128*0.083 = 10.6 milliseconds -- that is about 1% of its time -- for serial communication.

6. How many characters per second can be transmitted over a 1200 baud line in each of the following modes? (Assume a character code of 8 bits)

(a) Synchronous serial transmission

Answer:
1200/8 = 150 characters per second

(b) Asynchronous serial transmission with two stop bits

Answer:
1200/11 = 109 characters per second

(c) Asynchronous serial transmission with one stop bit.

Answer:
1200/10 = 120 characters per second

What is the difference between synchronous and asynchronous serial transfer of information?

Answer:
An asynchronous transmission of data uses the start and stop bit convention to signal the beginning and end of a character. Each character consists of 3 parts, the start bit, the character bits and the stop bits. A character is detected at the receiver from a knowledge of the transmission rules. The stop bit(s) allow the equipment to resynchronize.

Synchronous transmission does not use start-stop bits to frame a character. The equipment at either end of the line has internal clocks which are set to the frequency at which the bits are being transmitted. The clocks at either end must remain synchronized at all times. Thus the receiving end would probably use the signal transitions in the received data to synchronize the clock. To maintain synchronism, a synchronous transmission must send a continuous message. A message consists of a group of bits that form a block of data. The entire block is transmitted with special control bits at the beginning and the end in order to frame the entire block into one unit of information.. To initialise synchronization, the transmitter sends a SYN character (10010110) several times. The receiver picks up the character and looks for SYN. If it doesn't match it, it throws away the first bit received and adds another at the back until it recognizes the SYN string. At this point the system is synchronized and the block of data can be moved. The last string is an ETX (End of text) - 00000011 - and, when the receiver gets this string, it knows that the block has been completed. The last field in the transmission is a Block check character which is used for error checking. Often this is an LRC - longitudinal redundancy check - which is an 8-bit parity over all the characters of the message in the frame. It is the accumulation of the exclusive-OR of all transmitted characters.

7. A TV monitor has a vertical scanning rate of 70 Hz. The vertical and horizontal blanking intervals are 0.5 milliseconds and 3 microseconds, respectively. The dot clock is set at 68.5 Mhz and the horizontal scanning rate is 55.7 Khz.

(a) What is the resolution of this monitor? (i.e. how many dots horizontally by how many vertically)

Answer:
70 Hz corresponds to a frame time of 1/70 = 14.286 milliseconds. The vertical blanking period is 0.5 milliseconds so 14.286 - 0.5 is display time = 13.786 milliseconds. If the horizontal frequency is 55.7 Khz, then the number of lines which can be drawn (i.e. the vertical resolution) is 55.7*13.786 = 767.88 or 768 dots of vertical resolution.
The time for a horizontal line is 1/55700 = 17.95 microseconds. The horizontal synch takes 3 microseconds. Thus, the time available for displaying visible dots is 17.95 - 3 = 14.95

microseconds. If the dot clock is 68.5 Mhz, the number of dots horizontally = 14.95 * 68.5 = 1024
Thus the screen resolution is 1024 by 768.

(b) What fraction of the frame time is actually used for displaying visible dots?

Frame time = 14.286 milliseconds. A horizontal line has a displayable time of 14.95 microseconds and there are 768 lines. Thus the time used for displaying visible dots is 768 * 14.95 microseconds = 11.48 milliseconds.
Thus the fraction of frame time is (time for visible dots/ frame time) = 11.48/14.286 = 80.36%

8. A TV monitor has a 60Hz frame refresh rate and a horizontal scanning frequency of 33720 Hz.

(a) Up to how many scan lines could it possibly provide per frame?

Answer:
Such a monitor could provide up to a maximum of 33720 (lines/sec) / 60 (frames/sec) = 562 lines per frame.

(b) Suppose we use this monitor in an application requiring a resolution of 512 by 512. How many scan lines correspond to the vertical blanking interval? What is the duration of this interval?

Answer:
The vertical blanking interval corresponds to 562 - 512 = 50 scan lines. One horizontal scan takes 10^6 / 33720 = 30 microseconds. Hence, the vertical blanking interval is 50 * 30 = 1500 microseconds, or 1.5 milliseconds.

(c) How long is the usable portion of a scan line (in terms of time) given that we waste only 4 microseconds for horizontal blanking?

Answer:
The length of the usable portion of a scan line is 30 - 4 = 26 microseconds

(d) What would the dot clock frequency be?

Answer:
During the usable portion of a scan line we must be able to display 512 dots. To do this we need a clock frequency of 512 /26 Mhz -- that is, about 20 Mhz.

9. Consider an alphanumeric CRT display employing a 7 x 9 character font and a 9 x 12 character block. The display format is 80 x 24. The vertical and horizontal scanning rates are 60 and 18000 Hz, respectively.

(a) How many usable scan lines do we need?

Answer:
We need 24 x 12 = 288 usable scan lines (since there are 24 character lines and 12 dot rows per character block). Actually this number could be reduced by 12 - 9 = 3 lines given that the top and bottom rows do not require spacing on both sides.

(b) Usually, the horizontal blanking time is chosen to be an integral number of periods of the character clock. Here we assume it corresponds to 20 periods of the character clock. Determine the length of the usable portion of a scan line (in terms of time).

Answer:
The total length of a scan line is 1/18000 = 55.55 microseconds. Since there are 80 characters per character row, there is a total of 80 + 20 = 100 character clock periods per scan line. So, the usable portion of a scan line is 55.55 x 0.8 = 44.44 microseconds.

(c) What is the frequency of the character clock?

Answer:
Since there are 100 character clock periods in a time interval of 55.55 microseconds (that is, during a scan line) the frequency of the character clock is 100/55.55 = 1.8 Mhz.

(d) What is the frequency of the dot clock?

Answer:
Multiplying the number of columns per character block with the character clock frequency we find 9 x 1.8 = 16.3 Mhz for the frequency of the dot clock.

(e) The vertical blanking interval is itself chosen to be an integral number of scan line periods. Verify that this is indeed true. What would you vary slightly if it was not?

Answer:
The total number of scan lines per frame is (1/60)(1/18000) = 18000 (lines/sec) /60 (frames/sec) = 300 (lines/frame). Since 288 of them are usable, the vertical blanking interval corresponds to 12 scan lines exactly. If this was not the case we would vary slightly the horizontal scanning rate.