Department of Electrical and Computer Engineering

Computer Engineering

Course ECSE-322B

Problem Set 5 Solutions

8 February 2008

1. Consider the timings involved in the keyboard circuit discussed in class:

   a. There is the clock driving the counter.
   b. There is a clock driving the shift register connected to the serial line.
   c. There is the user.

   We would like to design the clock used for driving the counter such that synchronization problems are avoided. Under the following assumptions:

   - The serial line runs at 9600 bits per second (baud).
   - The keyboard has 64 keys.
   - The user is a good typist and can type 5 characters per second (This translates to about 60 words per minute or about 6 to 7 minutes to type a page).
   - When a key is pressed on the keyboard, the switch is closed for half of the time.

   Compute a minimum frequency for the counter clock in order to obtain reliable operation of the circuit. Which elements do/don't present synchronization problems for the system?

Solution:

Assuming that the serial line runs at 9600 bits per second (baud), then 8 bits takes 0.8333 milliseconds to transmit a character. On this basis, we should not recognize another character for at least 0.833 milliseconds or we will have a synchronization problem in the circuit and we will lose data.

If the user types 5 characters per second, this implies a character input time of 200 milliseconds. Part of this time is closing the key and part is opening it (this depends on spring strengths, etc.). Assuming that the key is actually closing the switch for half of this time, i.e. 100 milliseconds. Thus the serial line speed is unimportant – the user is the slowest item in the system.

If the keyboard has 64 keys (a rather small keyboard but the one in the example) – then there are 8 lines to be scanned. In the worst case, all 8 lines have to be scanned to recognize the key being held down. Thus we need to scan 8 lines in 100 milliseconds or 80 lines per second.

Thus the clock rate should be at least 80 Hz.

2. Given the differences in clock speed in the operation of the keyboard, what problems do you foresee if the clock speed on the counter is increased? Describe them in point form. How would you modify the circuit to overcome them?

Solution:

If the clock speed on the counter is increased, two effects can happen:

- A key being held down will be recognized several times thus leading to several characters being sent for each one typed.
- If the matrix is scanned too quickly, the data in the shift register could be overwritten before it has been transmitted down the line.

This would suggest that there is a "safe" upper limit on the counter clock speed which is related to the shift register clock unless changes are made to the circuitry.

The two problems to solve are:

a. multiple characters being recorded,
b. overwriting data

One solution to this problem would be to inhibit the clock on the counter for a specified period of time once a key has been recognized (in other words, when a character is being loaded). The assumption is that if the clock is not running, then the keyboard will not be scanned since the signals will not be switched onto each row. To do this, the load signal that is generated as a result of a key being identified could be used to block the counter clock from the counter, i.e. invert the load signal and AND it with the clock signal prior to the clock signal reaching the counter. This would, in effect, stop the counter and thus stop the keyboard scanning until the key is released (when the key is released, the load signal should disappear and the clock should start up again.)

3. How would you modify the keyboard circuit to allow for two keys to be depressed simultaneously to define a character, e.g. “Ctrl” + “A”?

Solution:

The simple answer to this is to actually connect the “control” keys directly to the ROM and increase the number of bits in the ROM address so that pressing two keys together addresses a different part of the ROM.

4. In the keyboard circuit discussed in class, the matrix is scanned by using a clock to drive a counter which, in turn, uses a multiplexer circuit to connect a zero voltage to each of the rows of the matrix sequentially. If the keyboard was designed to handle 144 keys, rather than 64 and the time a user holds a particular key down for is about 50 ms, what would be the maximum frequency that can be used to drive the counter if the key is guaranteed to be recognized only once?

Solution:

To handle a 144 key keyboard, the key connections would be structured as a 12 by 12 matrix. Thus the row scanning circuitry would have to scan 12 rows in a complete cycle. If a key is held down for 50 milliseconds we need to arrange the scanning frequency such that the row containing the key is scanned once and only once in 50 milliseconds. Thus we have to guarantee that at least all 12 lines are looked at in 50 milliseconds. In other words, the worst case is that the key is pressed just before we scan the row containing the key - leading to 12 rows to scan. This, then will give us the maximum frequency which can be used.

Hence the time to scan one row is

$$t = \frac{50 \times 10^{-3}}{12}$$

= 4.167 milliseconds

Max frequency = $1 / t = 1 (4.167 \times 10^{-3}) = 239.98$ or 240 Hz

5. In the keyboard described in Question 4, it is desired to eliminate switch bounce, i.e. the possibility that, in closing, a switch makes a circuit, opens and then makes the circuit. Describe how you would modify the circuit for the keyboard to avoid recording multiple identical characters for a single keystroke.

Solution:

The intention is to ensure that only one character is decoded for a single keystroke. The standard solution for switch debouncing is to feed the signal from a switch into a simple S-R latch. This would solve the problem of switch bounce and would guarantee that the pressing of a key is recognized. If it is desired to stop multiple characters being transmitted from a single keystroke, i.e. to control auto-repeat, then circuitry needs to be added before the ROM to remember the current character. This character can then be compared with the next character being sent and, if a match is achieved, the new character should not be passed to the ROM unless a pre-determined time interval has passed.