Dordt University

Engineering 304, Embedded Systems, Spring 2023

PS#7, Problem on Interrupts

3.19 **Critical Regions**. It is of paramount importance to be able to disable interrupts to protect critical regions in our code. Consider, for example, the common technique for changing a bit on an output port while leaving the other bits unchanged. We can read the port (even though it is an output port) into an accumulator. Then we can force the selected bit (e.g., bit 5) to 1 with an OR instruction (leaving the other bits unchanged). Finally, we can write the result back out to the port.

a.) Assuming that there is an interrupt service routine which changes bit 3 on this same port when it is called, why should interrupts be disabled for the above operation which is intended to change bit five? To answer this, describe a scenario which leads to a malfunction.

b.) How often is such an error likely to recur if bit 3 is changed once per millisecond, with the three instructions taking 1 µs each, and if bit 5 is changed once every 10 ms or so? Assume that the two events are not synchronized to each other, but that at least one of them occurs in response to an unsynchronized, external event.

c.) Is there any benefit to be had in an instruction which sets (or clears or toggles) selected bits of a port *directly*? Explain. (Not all microcontrollers have such an instruction, but some do, including both the Motorola 68HC11 and the Intel 8096. The AVR microcontrollers used on Arduino boards have these bit-set and bit-reset instructions for some registers, but not for main memory.)

Problems 3.19 is derived from a nearly identical problem in   
Peatman, John B, *Design With Microcontrollers*, McGraw Hill, 1988.