Matt Kruger

Chapter 3 problems 3,1, 3.3, 3.6, 3.8.

HW #12

3.1 [E] The input status bit in an interface circuit is cleared as soon as the input data register is read. Why is this important?

The clearing of the input status bit (used to track the state of a peripheral device, typically indicating whether there is new input data at the data register) signals that the data has been used/acknowledged.

The automatic clearing is crucial because the lack of a bit signals that the system is READY FOR NEW DATA.

if the bits were not cleared, the system may interpret stale data as fresh, or be stuck in a perpetual interrupt service routine - if using interrupts. Even polling systems may be cooked.

ultimately, its vital for correct clearing of control signals for an I/O system.

Matt Krueger

Chapter 3 problems 3,1, 3.3, 3.6, 3.8.

3.3 [E] What is the difference between a subroutine and an interrupt-service routine?

A subroutine is explicitly called by the program while an interrupt service routine is triggered by hardware.

Additionally, subroutines are synchronous and execution is determined by the program. On the other hand, an interrupt is asynchronous and execution being requested by hardware by IRQ.

The location of the ISR is determined by the interrupt vector table, which maps incoming interrupt signal from I/O device to its respective ISR.

Matt Krueger

Chapter 3 problems 3,1, 3.3, 3.6, 3.8.

3.6 [E] In Figure 3.9, the interrupt-enable bit in the PS is set last in the START section of the Main program. Why? Does the order matter for earlier operations in START? Why or why not?

There should be <u>NO INTERRUPTIONS</u> when the program is being set up. Enabling global interupts (Ex 3.9 or SEI for AVR used in Embedded Systems course) ensures that the system is fully ready to service an interrupt.

IF set first (or not last) and an interupt occurs:

system may recive interupt before handlers were configured :(or registers may not be initalized. :(

There are likely other pitfalls that are directly avoidable if setting interupts after all configuration

SEI or <u>IE</u> in the textbook avoids this critical error.

GOOD: Initialization

IE wait

1. config a
2. config b ← interrupt during config b Ignored!
3. config c
4. config d
5. IE

BAD: IE NOT Last

Initialization

1. IE
2. config a
3. config b ← interrupt during config b Interrupt serviced!
4. config c
5. config d

{ → ISR
⋮
← Something may be cooked!

BAD.

Matt Krueger

Chapter 3 problems 3,1, 3.3, 3.6, 3.8.

3.8 [E] A user program could check for a zero divisor immediately preceding each division operation, and then take appropriate action without invoking the OS. Give reasons why this may or may not be preferable to allowing an exception interrupt to occur on an actual divide by zero situation in a user program.

Programming software is a much EASIER approach for handling divide by zero errors. The user can define code to handle the error gracefully, or just give a default value.

With this approach there is no reliance on the OS or even exceptions. Thus overhead reduced.

A hardware safety net is still needed to ensure system integrity — an security - if malicious divide by 0 code is ran by a bad actor.

Software approach: easy & Flexible, not always implemented

hardware approach: complex, BUT MORE SECURE.