MicroControllers

Lab4

William L. Cole & Romeo Djeulong

22/11/2018

Circuitry

- PD2 to Button 1
- PD3 to Button 2
- PB0 to Red LED
- PB1 to Yellow LED
- PB2 to Green LED

Program

Task 1

Without the use of interrupts, the program for task 1 checks to see if button 1 is activated, with a simple if-statement. If it is, the Red LED is turned on using the '|=' operator.

The program also checks if Button 2 is activated, in the same manner, and if it is, the program turns the Red LED off, using the '&=' operator and a negation of the previous activation [i.e. "~(1<<PB0)"].

In order to blink the Yellow LED, the program simply activates the Yellow LED with the '|=' operator, followed by a delay of 250 milliseconds, followed by a deactivation of the LED, followed by another 250 millisecond delay. This creates a blink cycle of half a second. However, without the interrupts, the buttons for the Red LED are now unresponsive.

Task 2

In order to overcome the issues of *Task 1*, EICRA and EIMSK are initialized in the init. The if-statements that checked the activation of Buttons 1 & 2 are omitted, but their behavior is transferred to the INT0 interrupt (for Button 1) and INT1 interrupt (for Button 2). As INT0 and INT1 interrupts are attached to the same ports that the buttons were previously wired to (PD2 & PD3 respectively), the buttons now call the interrupt behavior when activated. So, when Button 1 is pressed, INT0 turns the Red LED on, and when Button 2 is pressed, INT1 turns the Red LED off.

Task 3

In order to add blinking behavior of the Green LED, the TIMSK1 is initialized in the init. The timer prescaler is set to 256 by activating the CS12 bit of TCCR1B.

Every time the counter rolls over, the Timer Overflow is activated, toggling PB2 with the ‘^=’ operator.

This only allows for a 50% duty cycle, however. So the LED is always off for the same amount of time that it’s on. Were the blink more rapid (i.e. a PMW) the light would only be able to achieve a 50% brightness.

Task 4

Allowing for more flexible blinking behavior, a compare interrupt is added to the program. The comparison value is set in main, just after the init: [OCR1A = 1000;]. The operator in the overflow timer is changed from a toggle to just an activation [i.e. from ‘^=’ to ‘|=’]. Then the compare interrupt deactivates the LED. So, the behavior is now that the Green LED is turned on every time the counter rolls over, then when 1000 counts pass (at a 256 precaler) the LED is turned back off. One can increase or decrease the duty cycle by changing the value of ORCR1A.