ME477, Lab 5

Ali Jones

2/18/2022

Description:

This program demonstrates the use of an interrupt thread, a subfunction that is external to the microcomputer and not directly called but can briefly override the main program functionality. In this case, the main function of the program is to count from 1 to 60 seconds. When interrupted (by the press of a push-button switch), the LCD screen displays the string "interrupt_" while continuing to count.

```
main
|_ Irq_RegisterDiIrq()     - reserves interrupt, configures DI and IRQ
|_ pthread_create()         - creates new thread to service interrupt
|_ Irq_UnregisterDiIrq()    - unregisters interrupt
|_ wait()                   - waits for 5ms
|
DI_Irq_Thread               - performs function in response to interrupt
|_ printf_lcd()             - prints to LCD screen
|_ Irq_Wait()               - waits for IRQ number or ready signal
```

Testing:

1. Run the program
2. The LCD screen will begin counting up in 1 second intervals
3. Press the push-button switch
4. Alongside the count, the LCD screen will display the word "interrupt_"

Results:

With the first circuit setup, the interrupt routine worked as expected; pressing the switch serviced the interrupt without altering the `main()` program. Due to the nature of the electronics, however, the circuit 'bounced' several times during each button press, creating more falling edges than expected and calling the interrupt several times. This bouncing behavior can be seen in the figures on the following page. The transition between voltages took 10-20 microseconds to settle. The oscilloscope images below show upwards of 5 bounces, and the LCD screen sometimes displayed more.

To solve this problem, the wiring was replaced by a "debouncing" circuit. This circuit uses two NAND gates that, regardless of the number of bounces, only produces one voltage transition at DIO0. Graphs of the circuit voltage vs time can be seen in figure 2, below.

With the debouncing circuit implemented, the interrupt thread worked successfully, printing one "interrupt_" with each press of the push button. The main program was still left unchanged; it simply registered the thread and channel with the function `Irq_RegisterDiIrq()`, created a Posix thread to control the interrupt behavior, and (after the rest of the tasks in `main()`) terminated and unregistered the thread. The thread itself mostly consisted of a loop that continuously paused to wait for the interrupt signal and, at the end of the pause, checked to see if an interrupt had been requested.

Figure 1. Examples of the circuit bounce when pressing (top, oscilloscope goes from high to low) and releasing (bottom, oscilloscope goes from low to high) the switch

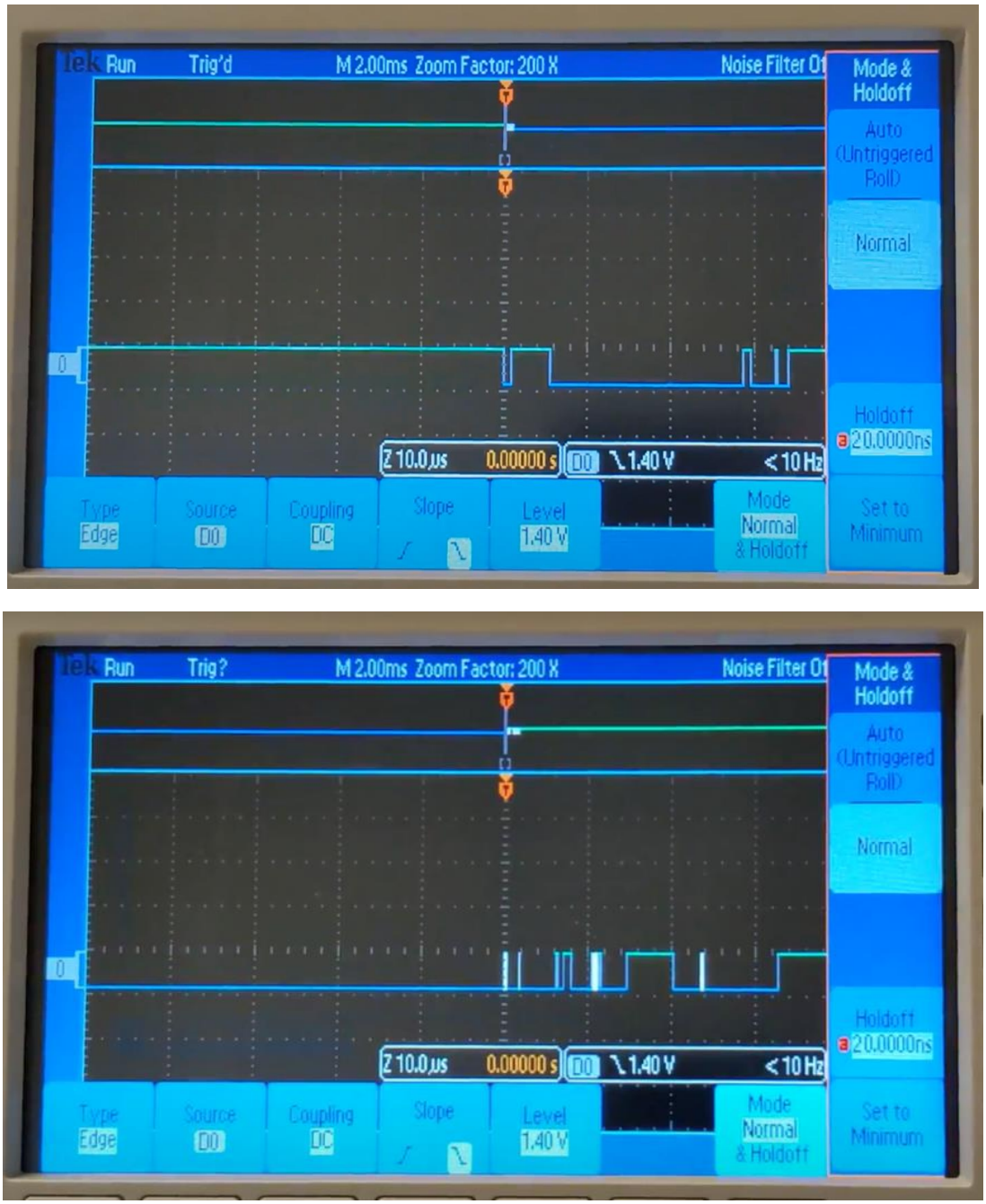

Figure 2. Analysis of the debouncing circuit