SECTION FIVE SIMPLE INTERRUPTS

SECTION OVERVIEW

- Understand how and when interrupts can be used
- Demonstrate the use of simple interrupts
- Explore how to configure interrupts on a microcontroller

PRELAB

Be sure to have completed these prelab questions before your lab. They will help you perform the tasks in this lab and hopefully give you more time to experiment.

1. In computing there are traditionally two ways for a microprocessor to listen to other devices and communicate. These two methods are commonly called 'polling' and 'interrupts.' A large amount of information about these two methods exists. Please describe what each of them is and give a few examples where you would choose one over the other.
2. What is the function for each bit in the following registers in the ATMega128? EICRA, EICRB, and EIMSK. Do not give just a description of each register. You must give specific details of each bit, its possible values, and what those values mean. You can find this information from either the AVR Instruction Set guide, or the ATMega128 Reference Manual both located on the lab web site. HINT: These registers are related to 'external interrupts.'
3. The AVR microcontroller uses 'interrupt vectors' to run code when an interrupt is triggered. What is an interrupt vector? List the memory locations for the following vectors in the AVR microcontroller: Timer/Counter2 Comparison Match, External Interrupt 2, and USART1-Rx Complete.
4. In the AVR microcontroller, like many others, there are several different ways of triggering interrupts. Below is a sample signal being input onto one of the external interrupt pins. List the specific clock cycles or range of clock cycles that the interrupt would trigger on this waveform if the interrupt was set up as: a.) rising edge, b.) Falling Edge, c.) Level High, and d.) Level Low.

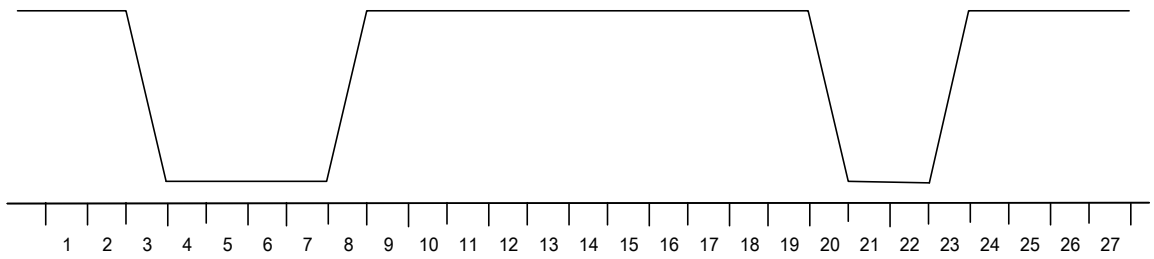

Figure 1: Sample Input to External Interrupt

PROCEDURE

Introduction

Most modern day computing systems use interrupts to communicate with peripheral devices. Interrupts allow the microprocessor(s) to execute instructions until a peripheral device needs attention. Some examples of this are how PC hardware is used to have 'interrupts' that a user would have to define so that maybe a sound card or joystick could ask the microprocessor for attention.
Using interrupts can be tricky and sometimes wasteful. When an interrupt request comes in, the microprocessor has to stop what it is doing, store any special variables and then service the interrupt. Once the interrupt is done, the processor then must reload its original special variables and restart what it was doing. For example if a peripheral wants the microprocessor to store a single byte of data every couple of clock cycles it may try to interrupt the microprocessor every couple of clock cycles. This would force the microprocessor to spend all of its time storing its special variables, servicing the interrupt, reloading the special variables and then starting all over again when the next interrupt comes in. The cost of servicing an interrupt in this manner is called a 'context switch.' This is why many modern computers have coprocessors and peripheral controllers (like DMA for example) to handle these frequent requests.

Interrupting a TekBot

You will need to write a short assembly program that causes your TekBot to move forward. Then when either its right or left whisker is hit it will need to react by interrupting the AVR microcontroller to back up and turn away from the point of impact. This will make your TekBot function as it has before in Lab 1, but using a different method in code. This is an important point to note. There is ALWAYS more than one option/solution. Your job as an engineer is to be able to choose the BEST option based on all pertinent information.

Write your assembly program and download it to your TekBot. Show your TA the operation and the code and have them sign below for credit. Be sure your code is well commented. Skeleton code is available on the webpage.

TA Signature: _____

STUDY QUESTIONS/ REPORT

Write a short summary that details what you did and why, explain any problems you may have encountered, and answer the questions below. Your write up should follow the required format given on the lab web page. Submit a hard copy of your write up and code to your TA by the beginning of class the week following the lab. NO LATE WORK IS ACCEPTED.

Study Questions

1. As an engineer you should be able to justify your design and testing choices. You have implemented this bumper TekBot using two different programming languages (AVR assembly, C) and two different methods of communicating with external inputs (polling, interrupts). Explain the benefits and costs of each of these choices. Some important areas of interest include, but are not limited to, efficiency, speed, cost of context switching, time to code, understandability, etc.
2. Now that you have a basic understanding of how interrupts work, would it be possible to use a timer/counter interrupt to perform the wait loop while the robot is between transitions within the external interrupt? (HINT: The order in which the interrupt is located within the interrupt vector list is also the priority, the lower on the list, the higher the priority.) Give a reasonable argument either way.

Challenge

1. Sometimes your TekBot can get caught in a loop where it is stuck in a corner and it continually backs up, hits the right whisker, then backs up and hits the left whisker, then the right, then left, and so on. Add a 'memory' to your TekBot so that it can detect this problem and when it has hit alternating whiskers five times, it will stop, turn around 180 degrees, and resume forward motion to get out of the corner.

   In addition, correct the simple problem of the TekBot hitting the same wall several times. In this scenario the TekBot hits one of its whiskers, backs up and turns away, but does not turn far enough and hits the same object. In this case you need to make the TekBot backup and turn away twice as far as if the whisker has been hit twice in a row.

   Write your program and keep it well documented. Turn in your code along with your write up and have you TA sign below that they have seen your functioning code.

   TA Signature: _____