ECE 375

Computer Organization and Assembly Language Programming

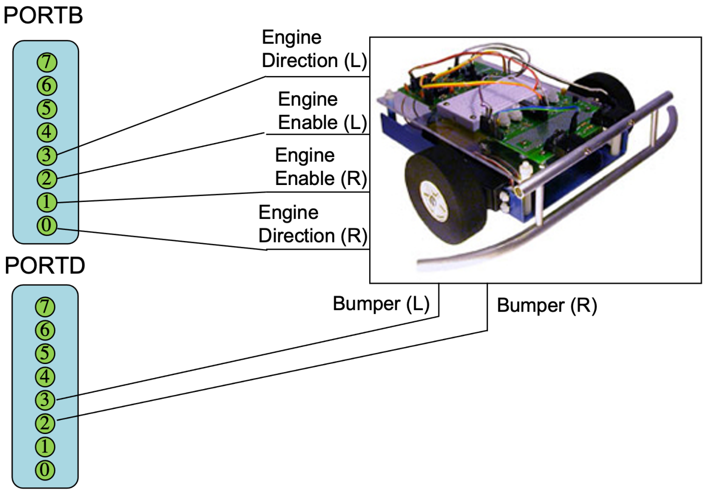
Fall 2022

Assignment #3

[25 pts]

1- Consider the AVR code segment shown below that initializes I/O and interrupts for Tekbots shown below (with some information missing).

```
.include “m128def.inc”
.def mpr = r16
.org $0000
      rjmp  INIT
.org  ____________________(i)
      rjmp  HitRight
.org  ____________________(ii)
      rjmp  HitLeft
      …
.org $0056
INIT: ______________________________(1)        ; Control engine
      ______________________________(2)        ;
      ______________________________(3)        ; Detect whiskers
      ______________________________(4)        ;
      ______________________________(5)        ; Enable pull-up resisters for L/R bumpers
      ______________________________(6)        ;
      ______________________________(7)        ; Detect on the proper edge
      ______________________________(8)        ;
      ______________________________(9)        ; Turn on interrupts for L/R bumpers
      ______________________________(10)       ;
      sei                                      ; Turn on global interrupt
```



(a) Fill in the lines 1-2 with the necessary code to set Data Directional Register x to control engine enable and engine direction for both left and right wheels.

(b) Fill in the lines 3-4 with the necessary code to set Data Directional Register x to detect left and right bumper movements.

(c) What are the addresses needed in the lines (i) and (ii) to properly control the execution of interrupt service routines for left and right bumpers. Fill in the lines 5-6 to enable the pull-up resisters for these whiskers.

(d) Fill in the lines 7-8 with the necessary code to set External Input Sense Control to detect bumper hits (i.e., interrupts) on a falling edge.

(e) Fill in the lines 9-10 to enable interrupts for whisker movements.

[25 pts]

2- Consider the WAIT subroutine for Tekbot discussed in class that waits for 1 sec. and returns. Rewrite the WAIT subroutine so that it waits for 1 sec. using the 16-bit Timer/Counter1 with the highest possible resolution. Assume that the system clock frequency is 8 MHz and the Timer/Counter1 is operating under the CTC mode. This is done by doing the following:

(a) Timer/Counter1 is initialized to operate in the CTC mode.

(b) The WAIT subroutine loads the proper value into OCR1A and waits until OCF1A is set. Once OCF1A is set, it is cleared and the WAIT subroutine returns.

Use the skeleton code shown below. Also, show the necessary calculations for determining *value* and *prescale*. Note that your code may not use any other GPRs besides mpr.

```
.include "m128def.inc"
.def mpr = r16
...
.ORG    $0000
        RJMP  Initialize
.ORG    $0046                ; End of interrupt vectors
Initialize:
        ...
        ...Your code goes here...
        ...
WAIT:
        ...
        ...Your code goes here...
        ...
        RET
```

[25 pts]

3- Consider the AVR code segment shown below (with some missing information) that configures Timer/Counter0 for Fast PWM operation, and modifies the Fast PWM duty cycle whenever a specific button on Port D is pressed.

(a) Fill in lines (1-2) with the instructions necessary to configure Timer/Counter0 for Fast PWM mode to toggle OC0A.

(b) Fill in lines (3-4) with the instructions necessary to set the prescale value to 8.

(c) Based on the prescale value used in part (b), what is the frequency of the PWM signal (f_{PWM}) being generated by Timer/Counter0? Assume the system clock frequency is 8 MHz.

(d) Fill in lines (5-6) to provide the compare value for Timer/Counter0 so that the initial duty cycle is 0%.

(e) What would be the value necessary for the variable step to increase the duty cycle by 10% each time the DUTY_STEP subroutine is executed? Ignore the case when/if the compare value overflows.

```
.include    "m32U4def.inc"
.def        mpr = r16
.def        temp = r17
.equ        step = ___

INIT:
     …
     ; stack pointer is initialized
     …

     ; I/O ports
     ldi    mpr, 0b10000000     ; set Port B, pin 7 (OC0A) as output
     out    DDRB, mpr

     ldi    mpr, 0b00000000     ; set pin 0 as input
     out    DDRD, mpr
     ldi    mpr, 0b00000001     ; enable pull-up resistor for pin 0
     out    PORTD, mpr

     ; Timer/Counter0
     ; Fast PWM mode, non-inverting, prescale = 8
     __________________________(1)
     __________________________(2)
     __________________________(3)
     __________________________(4)

     ; Initial compare value for PWM output
     __________________________(5)
     __________________________(6)

MAIN:
     sbis  PIND, 0
     rcall DUTY_STEP
     rjmp  MAIN

DUTY_STEP:
     push  mpr
     push  temp

     in    mpr, ________       ; read the current PWM compare value
     ldi   temp, step
     add   mpr, temp           ; add step value to compare value
     out   ________, mpr       ; write new PWM compare value

     pop   temp
     pop   mpr
     ret                       ; return
```

[25 pts]

4- Write a subroutine `initUSART1` to configure ATmega32U4 USART1 to operate as a transmitter and sends a data every time USART1 Data Register Empty interrupt occurs. The transmitter operates with the following settings:

- 8 data bits, 1 stop bit, and odd parity
- 9,600 Baud rate
- Transmitter enabled
- Normal asynchronous mode operation
- Interrupt enabled

Assume the system clock is 8 MHz. The skeleton code is shown below:

```
.include “m32U4def.inc”
.def mpr = r16
.ORG $0000
     RJMP  initUSART1
…
.ORG $0034
     JMP   SendData
…
.ORG $0056
initUSART1:
     …
     …Your code goes here…
     …
Main:
     ld    mpr, X+          ; Send first data
     sts   UDR1, mpr
Loop:
     RJMP Loop

SendData:
     ld    mpr, X+          ; Send next data
     sts   UDR1, mpr
     reti
```