ECE 375
Computer Organization and Assembly Language Programming
Winter 2015
Assignment #3

[25 pts]

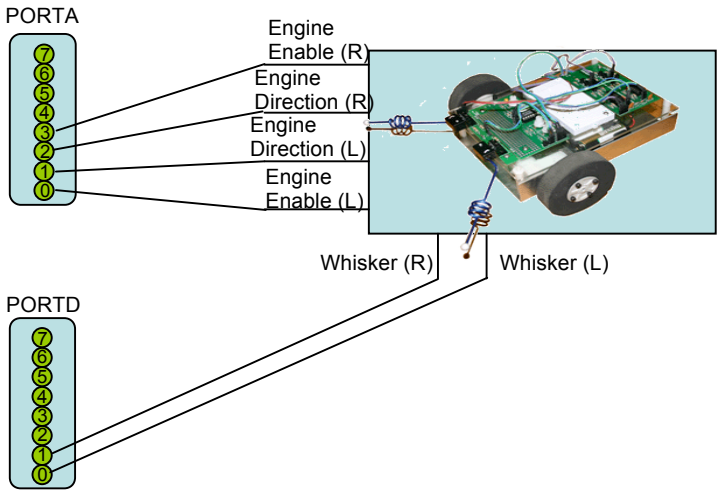
1- Consider the AVR code segment shown below that initializes and handles interrupts.
   (a) Explain in words what the code accomplishes when it is executed. That is, explain what it does and how it does it.
   (b) Write and explain the interrupt initialization code (lines (1)-(7)) necessary to make the interrupt service routine (starting at ISR:) work properly. More specifically,
      (i) Fill in lines 1-2 with the necessary code to set the interrupt in question to detect an interrupt on a rising edge.
      (ii) Fill in lines 3-4 with the necessary code to mask out all other interrupts except the interrupt in question.
      (iii) Fill in lines 5-6 with the necessary code to set the port in question for input.
      (iv) Fill in line 7 to enable interrupt.

```
.include "m128def.inc"
.def mpr = r16                    ; Multi-purpose register
.def count = r17                  ; Assume R17 is initially 0

.ORG $0000
START: RJMP INIT
.ORG $0002
       JMP ISR
.ORG $0046
INIT:  _____________________(1)
       _____________________(2)
       _____________________(3)
       _____________________(4)
       _____________________(5)
       _____________________(6)
       _____________________(7)
       LDI   XH, high(CTR)
       LDI   XL, low(CTR)
       LDI   YH, high(DATA)
       LDI   YL, low(DATA)
WAIT:  RJMP  WAIT
.ORG   0x100F
ISR:   IN    mpr, PINA
       ST    Y+, mpr
       INC   count
       ST    X, count
       RETI
.DSEG
CTR:   .BYTE 1
DATA:  .BYTE 256
```

[25 pts]

2- Consider the AVR code segment shown below that initializes I/O and interrupts for Tekbot shown below.

```
.include "m128def.inc"
.def mpr = r16
.org    $0000
        rjmp  INIT
        …

INIT:   _____________________(1)  ;
        _____________________(2)  ;
        _____________________(3)  ;
        _____________________(4)  ;
        _____________________(5)  ;
        _____________________(6)  ;
        _____________________(7)  ;
        _____________________(8)  ;
        _____________________(9)  ;
        _____________________(10) ;
        sei                       ; Turn on interrupts
```



(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 whisker movements.

(c) Fill in the lines 5-6 to enable the pull-up resisters for left and right whiskers.

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

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

[25 pts]

3- Write an AVR assembly code that waits for 1 sec using the 8-bit Timer/Counter0 with the system clock frequency of 16 MHz operating under Normal mode. This is done by doing the following:

(1) Timer/Counter0 is initialized to count for 10 ms and then interrupts on an overflow;

(2) The main part of the program simply loops, and for each iteration, a check is made to see if the loop has reach 100 iterations; and

(3) On each interrupt, Timer/Counter0 is reloaded to interrupt again in 10 ms.

Use the skeleton code shown below:

```
.include "m128def.inc"
.def mpr = r16
```

```
.def counter = r17
…
.ORG     $0000
    RJMP  Initialize
.ORG     $0020          ; Timer/Counter0 overflow interrupt vector
    RCALL Reload_Counter
    RETI
.ORG     $0046          ; End of interrupt vectors
Initialize:
    …
    …Your code goes here…
    …

LOOP:
    …
    …Your code goes here…
    …

Reload_counter:
    …
    …Your code goes here…
    …
    RET
```

[25 pts]

4- Write a subroutine `initUSART1` to configure ATmega128 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, 2 stop bits, and even parity
- 9,600 Baud rate
- Transmitter enabled
- Normal asynchronous mode operation
- Interrupt enabled

Assume the system clock is 16 MHz. The skeleton code is shown below:

```
.include "m128def.inc"
.def mpr = r16
.ORG $0000
     RCALL initUSART1
     RJMP Main
…
.ORG $003E
     RCALL SendData
     RETI
…
.ORG $0046
Main:
     RCALL SendData
Loop:
     RJMP Loop

initUSART1:
     …
     …Your code goes here…
     …
     ret

SendData:
     …
     …Your code goes here…
     …
     ret
```