ECED 3204 Microprocessor

Final Reference Solution

Date: December 10 2016 **Room: Sexton Gym**

Time: 12:00pm-3:00pm

Student name	
ID	

1) Implement Macro for following subroutine and write the macro call as well.

[10points]

```
SwapRegs:   push    r16
            mov     r16,r17
            pop     r17
            ret
```

solution:

```
            .macro      swapreg
            push        @0
            mov         @0, @1
            pop         @1
            .endmacro

                swapreg     r16, r17
```

2) Write an ASM program to find the total elements of an array of 8-bit integers that have bit 1,3,5,7 set to 1. Assume the array is in the program memory.

[10points]

Solution: To find the number of elements with bit 1, 3, and 5, 7 set to 1, we need to use the mask 0x10101010 to AND with each number. If the result is equal to 0x10101010, then the number satisfies the requirement and causes the element to increment by 1. The following program performs the desired operation:

```
            .include <m644Adef.inc>
            .def    count = r16
            .def    lpcnt = r17
            .def    temp=r18
            .equ    NN = 30             ; array count
            .cseg
            .org    0x00
            rjmp    start
            .org    0xF6
start:      ldi     ZL,low(array<<1)    ; use Z to point to the array element
            ldi     ZH,high(array<<1)   ;       "
            ldi     count,0
            ldi     lpcnt,NN        ; initialize loop count to NN
loop:       lpm     temp,z+
            andi    temp,0xAA       ; check bit 1,3,5,7
            cpi     temp,0xAA       ; are these 4 bits all 1's?
            brne    next
            inc     count
```

```
next:   dec     lpcnt
        brne    loop
done:   rjmp    done
array:  .db     11,12,13,14,15,16,17,18,19,20
        .db     21,22,23,24,25,26,27,28,29,30
        .db     31,32,33,34,35,36,37,38,39,40
// end of program
```

3) Read following program and determine the value of r0, r16, 17, r18, r19 when the program is done.

[10points]

```
        .include <m644Adef.inc>
        .equ    N1 = 0
        .equ    N2 = 30
        .def    ii = r16
        .def    count = r17
        .def    tmp = r18
        .macro setPointer
        ldi     @0,low(@2)
        ldi     @1,high(@2)
        .endmacro
        .cseg
        .org    0x00
        rjmp    start
        .org    0xF6
start:  clr     ii              ; initialize ii to N1
        clr     count           ; initialize count to 0
        ldi     tmp,50          ; value to be compared
        setPointer ZL,ZH,(array<<1); use the Z register as a pointer to the array
floop:  cpi     ii,N2           ; check looping condition
        breq    done
        lpm     r0,z+           ; read the next array element
        cp      r0, tmp         ; is the element less than 100?
        brge    next            ; if yes, then branch to increment count
        inc     count
next:   inc     ii
        rjmp    floop
done:   rjmp    done
array:  .db     10,11,12,13,14,15,16,17,18,19
        .db     50,21,22,23,24,25,26,27,28,29
        .db     30,31,32,33,34,35,36,37,38,39
// End of program
```

Solution:
r0=39
r16=30
r17=29
r18=50

4) Write a sequence of C statements to configure PCINT1, PCINT3, PCINT5, PCINT11, PCINT13, PCINT15, PCINT22 and PCINT23 to interrupt the MEGA CPU.

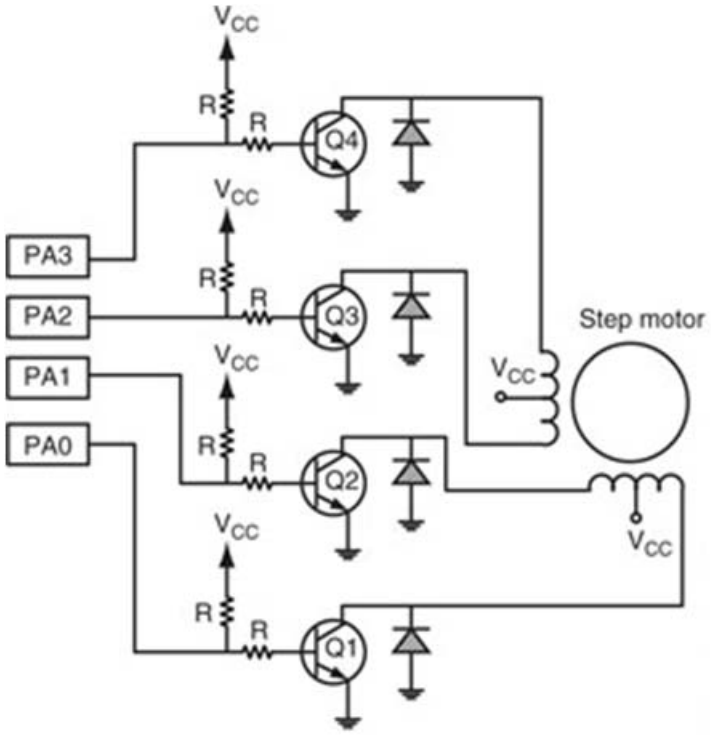
[10points]

Solution: The statements that configure the specified pins to interrupt on pin change is as follows:

```
PCMSK2  = 0xC0;    // enable PCINT22, PCINT23
PCMSK1  = 0xA8;    // enable PCINT11, PCINT13, PCINT15
PCMSK0  = 0x2A;    // enable PCINT1, PCINT3, PCINT5
PCICR   = 0x07;    // enable interrupts PCINT2~PCINT0
sei(); // enable interrupt globally
```

5) Write a subroutine that rotates the stepper motor shown in figure clockwise rotation in half step sequence.
Assume the subroutine for delay is available **delayby10ms.**

[10points]



The subroutine to rotate the stepper motor counterclockwise full-step is as follows:

```
        .include <m644Adef.inc>

        .equ    HSTEP1 = 0x08   ; values to rotate a half step
        .equ    HSTEP2 = 0x0C;  "
        .equ    HSTEP3 = 0x04   ;    "
        .equ    HSTEP4 = 0x06   ;    "
        .equ    HSTEP5 = 0x02   ;    "
        .equ    HSTEP6 = 0x03   ;    "
        .equ    HSTEP7 = 0x01   ;    "
        .equ    HSTEP8 = 0x09   ;    "

HSccw:  ldi     r20, HSTEP1
        out     PORTA,r20
        rcall   wait10ms
        ldi     r20, HSTEP2
        out     PORTA,r20
        rcall   wait10ms
        ldi     r20, HSTEP3
        out     PORTA,r20
        rcall   wait10ms
        ldi     r20, HSTEP4
        out     PORTA,r20
        rcall   wait10ms
        ldi     r20, HSTEP5
        out     PORTA,r20
        rcall   wait10ms
        ldi     r20, HSTEP6
        out     PORTA,r20
        rcall   wait10ms
        ldi     r20, HSTEP7
        out     PORTA,r20
        rcall   wait10ms
        ldi     r20, HSTEP8
        out     PORTA,r20
        rcall   wait10ms
        ret
```

6) Write an AVR assembly program to generate a periodic square wave with 2-kHz frequency and 50% duty cycle using the Timer 1 CTC mode of the ATmega644 device. Square wave is on OC1A pin and use ICR1 to hold the TOP value assuming that fclk_I/O = 16 MHz. select clk_IO as clock input to Timer1 and prescaler to 1.

[10points]

Solution: By setting the prescaler to 1, the TOP value to be loaded into the ICR1 register is

$TOP = (16 \times 10^6 \div 1) \div (2 \times 2 \times 10^3) = 4{,}000$

The assembly program that generates this waveform is as follows:

```
        .include <m644Adef.inc>
        .def    tmp = r16
        .cseg
        .org    0x00
        rjmp    start
        .org    0xF6
start:  ldi     r16,low(RAMEND)
        out     SPL,r16
        ldi     r16,high(RAMEND)
        out     SPH,r16
        ldi     r16,0x20        ; configure PD5/OC1A pin for output
        sts     DDRD,r16        ;    "
        ldi     tmp,0x40        ; configure Timer1 to CTC mode (WGM13:0)
        sts     TCCR1A,tmp      ; and make OC1A pin toggle
        clr     tmp             ; force timer1 to count up from 0
        sts     TCNT1L,tmp      ;    "
        sts     TCNT1H,tmp      ;    "
        ldi     tmp,high(4000)  ; place 4000 as the TOP value
        sts     ICR1H,tmp       ;    "
        ldi     tmp,low(4000)   ;    "
        sts     ICR1L,tmp       ;    "
        ldi     tmp,0x19        ; clk_IO as the clock input to timer 1 and start
        sts     TCCR1B,tmp      ; to generate the 2-kHz square wave with 50% duty cycle
again:  jmp     again
```

7) Write a subroutine and a C function to configure the USART0 on Port D of the MEGA644A to operate with the following setting assuming that f_{OSC} = 32 MHz:

[10points]

- one start bit, eight data bits, one stop bit, no parity
- 38,400 baud rate
- Enable transmission and reception
- Disable multiprocessor mode
- Asynchronous double speed mode operation
- Disable interrupts

Solution:

```
void initUSARTC0 (void)
{
    DDRD    = 1<<PD1; // configure TXD0/PD1, RXD0/PD0 (0x02)
    UBR0H   =0;   // set up baud rate to 38400
    UBR0L   =103;       //      (0x67)
    UCRR0A  = 1<<U2X0;      // disable USART0 interrupts, (0x02)
```

UCSR0B =(1<<RXEN0) |(1<< TXEN0) // enable transmission, reception (0x19),
USAR0C = (1<<UCSZ01) | (1<<UCSZ00) ; // select asynchronous USART mode, disable parity, 1 stop bit, (0x06)

8) Configure the MEGA SPI module with the specified setting:

[10points]

Master mode with all interrupts disabled
SCK signal idle high and sample data on the leading edge
Data shifted least significant bit first
Baud rate set to 1MHZ assuming f_{SYS} = 16 MHz

```
        ldi     r20, 1<< SPE | 1<<DORD | 1<<MSTR | 1<< CPOL | 1<< SPR0; master mode, SCK signal idle high and sample data on the leading edge (0x79)
        out     SPCR,r20        ; disable SPI interrupts, shift data lsb first, baud rate = 1 MHz
        clr     r20
        out     SPSR,r20        ; clear SPI2X flag
        in      r20,SPDR        ; clear all SPI flags
        ldi     r20,(1<<PB4) | (1<<PB5) |(1<<PB7)     ; configure SS/PB4, MOSI/PB5, MISO/PB6, SCK/PB7 (0xB0)
        out     DDRB,r20        ; pin direction
```

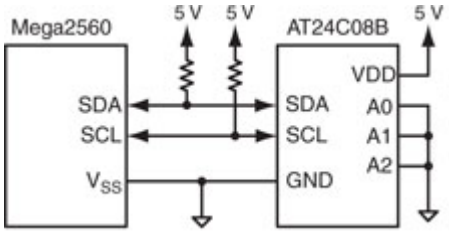
9) For following connection between Mega250 and AT2408B, write a ASM subroutine to configure the TWI module to operate with the following parameters, assuming that the Mega device is running with a 10MHz

[10points]

200 KHz shift rate
Enable TWI
Enable TWI interrupt

Solution:

```
InitTWI:    ldi     r16,17
            sts     TWBR,r16
            lds     r16,TWSR
            andi    r16,oxFC
            sts     TWSR,r16
            ldi     r16,0x05
            sts     TWCR,r16
            ldi     r16,0
            out     DDRD,r16
            ret
```



10) Write a sequence of AVR instructions to configure the AVR MEGA2560 ADC to operate with the following settings:

[10points]

- Select differential mode input
- Select ADC9 and ADC8 as positive and negative inputs with gain set to 10
- Disable auto-triggering
- Set ADC clock prescaler to 32 assuming that the MEGA2560 uses a 16-MHz crystal oscillator to generate its system clock
- Select AVCC as its reference voltage
- Result left-justified
- Disable digital input buffers of the unused analog inputs
- Enable ADC interrupt

Solution:

The assembly subroutine that performs the specified setting is as follows:

```
initADC: ldi   r20,0x8D      ; enable ADC, set ADC clock prescaler to 32, disable
         sts   ADCSRA,r20    ; auto triggering, enable ADC interrupt
         ldi   r20, 0x08
         sts   ADCSRB,r20    ; ADC result left justified, MUX5=1
         ldi   r20,0x69      ; set ADC9 and ADC8 pins as pos and neg input
         sts   ADMUX,r20     ;    "
         lds   r20,DDRF      ; configure PF0/ADC0, PF1/ADC1 pins for input
         andi  r20,0xFC      ;    “
         sts   DDRF,r20      ;    “
         ldi   r20,0xFF      ; disable digital input buffers of the unused analog input
         sts   DIDR0,r20     ;    "
         ldi   r20,0xFC      ;    "
         sts   DIDR1,r20     ;    "
         ret
```