(40) Question 1. Place one letter A-LL for each. (4 points each)

Arms output compare 4.

U) TMSK1 |= 0x10; //812A4
TIE |= 0x10;    //9S12C32

Acknowledges output compare 3.

CC) TFLG1 = 0x08;

Enables interrupts.

A) asm(" cli");

Disables interrupts.

LL) Not here, should be asm(" sei");

Disarms output compare 3.

R) TMSK1&= ~0x08;  //812A4
TIE &= ~0x08;  //9S12C32

Clears all 8 output compare flags.

EE) TFLG1 = 0xFF;
Z) TFLG1 |= 0x10;
AA) TFLG1 |= 0x08;

Specifies channel 4 is output compare.

O) TIOS |= 0x10;

Specifies the period of the OC3 interrupt

JJ) TC3 = TC3+1000; // ISR

Activates TCNT timer at the fastest rate

GG) TSCR |= 0x80;    //812A4
TMSK2 = 0;       //812A4
TSCR1| = 0x80;  //9S12C32
TSCR2 = 0;       //9S12C32

(5) Question 2. C

(5) Question 3. B) CPU bound.

(5) Question 4a. C) I_{OH} > 2mA

(5) Question 4b. R = (5-2V)/2mA = 3V/2mA = 1500 Ω

(5) Question 5. According to the Nyquist Theorem, frequencies 0 to 500 Hz are reliably represented.

(10) Question 6a. Show the C code that defines the finite state machine.

#define north &fsm[0]
#define west &fsm[1]
StateType fsm[2] = {
    {0x09, 30, {north, west, north, west},
     {0x06, 60, {north, west, north, west}}};

(5) Question 6b. You may add global variables as needed.

unsigned short Time;
StateType *Pt;

(5) Question 6c. Show the ritual() function that initializes the finite state machine.

void ritual(void){
    DDRT = 0x0F;     // PT7-6 inputs, PT3-0 outputs
    Pt = north;      // initial state
    PTT = Pt->out;   // initial output
    Time = Pt->wait; // time to wait in initial state
}

(15) Question 6d. Show the machine() function that executes the finite state machine.

void machine(void){unsigned char input;
if(!--Time){
    input = (0xC0&PTT)>>6; // 0,1,2,3
    Pt = Pt->next[input]; // next state
    PTT = Pt->out;     // output
    Time = Pt->wait;   // time to wait in next state
}