(4) Question 1. We can hear sounds from about 20 to 20 kHz, thus we say the frequency components of sound exist from 20 to 20 kHz. The Nyquist Theorem requires you to sample at a rate strictly larger than twice the maximum frequency components of the signal. Thus in this case, we must sample larger than 400 Hz. Recall in lab the distance signal was 0 to 2 Hz, and we sampled at 5 Hz.

(4) Question 2. E) The motor will not spin at all because we did not wait in between each output to the motor.

(4) Question 3. Write a subroutine to sample ADC channel 5 of the 9S12DP512.

```
ADC_In movb #$85,ATD0CTL5           ;start ADC channel 5
    brclr ATD0STAT0,#$80,adcwait ; wait for SCF
ldy   ATD0DR0
rts
```

(8) Question 4. 0 maps to 0 and 4095 maps to 4000. When Vin is 4 V, the ADC is 4095, software converts 4095 into 4000. Multiply by 4000, then divide by 4095

```
ldy  #4000
emul
ldx  #4095
ediv
```

(4) Question 5. The trick is the sty instruction overwrites the data saved on the stack by the stx instruction. RegX will be $3455

(4) Question 6. The order is D) 1,6,3,2,5,4,7

1) There is data in the receive data register and the hardware sets the flag bit (e.g., RDRF=1)
6) The CCR, A, B, X, Y, PC are pushed on the stack
3) The I bit in the CCR is set by hardware
2) The SCI vector address is loaded into the PC
5) The software reads SCI1SR1
4) The software reads SCI1DRL
7) The software executes rti

(10) Question 7. Write software that increments a 16-bit global variable every 2 msec.

```
org  $0800
Count rmb  2  ;incremented every 2 msec
org  $4000
main  lds  #$4000
    movw #0,Count
    movb #$80,TSCR1  ;enable TCNT
    bset TIOS,#$08   ;activate output compare
    bset TIE,#$08    ;arm OC3
    ldd TCNT
    addd #50
    std TC3          ;first interrupt right away
    cli
    bra *
OC3han movb #$08,TFLG1 ;ack
    ldd TC3
    addd #16000 ;2ms=16000*125ns
    std TC3 ;next interrupt
    ldx Count
```
inx
stx Count
rti
org $FFE8
fdb OC3han
org $FFFE
fdb main

(5) Question 8. Idle is high, start is low, bits go 0,1,2,3,4,5,6,7, stop is high, idle is high

<table>
<thead>
<tr>
<th>t</th>
<th>t-1</th>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
<th>900</th>
<th>1000</th>
<th>1100</th>
<th>1200</th>
<th>us</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

(5) Question 9. Transmits 1 to 2, then 2 to 1 taking a total of 2 ms.

(4) Question 10. The most common mistake was to think \( V_{out} \) is 5V when the digital output was 111. Let \( n = 4b_2 + 2b_1 + b_0 \) be the 3-bit number on PT2,PT1,PT0. The voltage on PT2 is \( 5*b_2 \). The voltage on PT1 is \( 5*b_1 \). The voltage on PT0 is \( 5*b_0 \). We will define all currents going from left to right. The current through the 10k\( \Omega \) is \( (5*b_2-V_{out})/10k\Omega \). The current through the 20k\( \Omega \) is \( (5*b_1-V_{out})/20k\Omega \). The current through the left 40k\( \Omega \) is \( (5*b_0-V_{out})/40k\Omega \). The current through the right 40k\( \Omega \) is \( V_{out}/40k\Omega \). Adding the currents up we get

\[
\frac{(5*b_2-V_{out})}{10k\Omega} + \frac{(5*b_1-V_{out})}{20k\Omega} + \frac{(5*b_0-V_{out})}{40k\Omega} = \frac{V_{out}}{40k\Omega}.
\]

Multiply by 40k\( \Omega \)

\[
4(5*b_2-V_{out}) + 2(5*b_1-V_{out}) + (5*b_0-V_{out}) = V_{out}
\]

Solving for \( V_{out} \)

\[
4*5*b_2 + 2*5*b_1 + 5*b_0 = 8*V_{out}
\]

Substituting the definition of \( n \)

\[
V_{out} = 5*n/8
\]

<table>
<thead>
<tr>
<th>PT2</th>
<th>PT1</th>
<th>PT0</th>
<th>Vout (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.625</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1.250</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1.875</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2.500</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3.125</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3.750</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4.375</td>
</tr>
</tbody>
</table>

(8) Question 11. You will be execute one instruction and answering questions.

Part a) 07 is the machine code for \texttt{bsr}

Part b) The return address is \$4005, which is pushed on the stack. To push, first decrement SP then store. The order in this table does not matter. In fact, the 9S12 will store the \$4005 as a 16-bit value in one bus cycle

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3FFD</td>
<td>$05</td>
</tr>
<tr>
<td>$3FFC</td>
<td>$40</td>
</tr>
</tbody>
</table>
Part c) The SP points to valid data on top of the stack, \( SP = \$3FFC \)

Part d) The PC is incremented after each fetch (PC=$4005), then the PC relative value of $02 is added to the PC to get the target address of the subroutine, PC = $4007

(5) Question 12. \( R = \frac{(5-V_d-V_{OL})}{I_d} = \frac{(5-1.5-0.5)}{0.03} = \frac{3V}{0.03A} = 100 \Omega \).

(10) Question 13. Basically, we wait for the input to be 1, wait for it to be zero, wait for the input to be 1, wait for it to be zero, wait for the input to be 1, then wait for it to be zero. To debounce we run the FSM at any time between 10 and 100 ms.

(10) Question 14.

Part a) Hand execute and draw a stack picture

\[
\begin{align*}
\text{Pt set 4} &; 16\text{-bit pointer to 8-bit data} \\
\text{L1 set -1} &; 8\text{-bit local variable}
\end{align*}
\]

Part b) A pointer is on the stack, so we need an indirection to get the data

\[
\text{ldaa} \ [\text{Pt},x] 
\]

We could have performed the indirection in multiple instructions

\[
\text{ldy} \ \text{Pt},x \\
\text{ldaa} \ 0,y
\]

(15) Question 15. The FIFO is full when Cnt is 8.

\[
\text{Fifo\_Put} \ \text{ldab} \ \text{Cnt} \ ; \text{number of elements currently saved, 0 to 8} \\
\text{cmpb} \ #8 \\
\text{bhs} \ \text{full} \ ; \text{full if Cnt is 8} \\
\text{NotFull} \ \text{ldx} \ \#\text{Fifo} \\
\text{ldab} \ \text{PutI} \ ; \text{RegB is 0 to 7} \\
\text{staa} \ b,x \ ; \text{save in Fifo} \\
\text{inc} \ \text{Cnt} \ ; \text{one more saved} \\
\text{incb} \ ; \text{next place to put} \\
\text{andb} \ \#$07 \ ; \text{0 to 7} \\
\text{stab} \ \text{PutI} \\
\text{ldab} \ \#1 \ ; \text{success} \\
\text{bra} \ \text{done} \\
\text{full} \ \text{clrb} \ ; \text{failure} \\
\text{done} \ \text{rts}
\]