(4) **Question 1.** Precision (alternatives) is range divided by resolution. Therefore the number of bits is \( \log_2(200/0.1) = \log_2(2000) = 11 \text{ bits} \)

(4) **Question 2.** The digital result is \((V_{\text{in}} - V_{\text{min}}) \times (N_{\text{max}} - N_{\text{min}}) / (V_{\text{max}} - V_{\text{min}}) + N_{\text{min}}\). In this case the ADC result = \(0.625 \times 256/2.5 = 64\) (or = \(0.625 \times 255/2.5 = 64\))

(2) **Question 3.** C = 0 (because it fits 160 - 140 = 20)

(2) **Question 4.** V = 1 (because it doesn’t fit -90 - (-40) = -130

(4) **Question 5.** The value = \(I \times \Delta = 1152/256 = 4.5\)

(4) **Question 6.** J) Hardware sets it when there is no data in the transmit data register, because TDRE means transmit data register empty.

(4) **Question 7.** D) Hardware sets it when there is data in the receive data register, because RDRF means receive data register full

(4) **Question 8.** The sequence length determines how many samples will be taken. The MULT bit is zero, so the same channel is sampled multiple times. ATDDR0 always received the first conversion. C) Channel 5 is sampled three times and the results are placed in ATDDR0-2.

(4) **Question 9.** C (enable) D (arm) G (trigger)

(4) **Question 10.** C) overflow, because 123 * N may not fit into a 16-bit temporary result

(4) **Question 11.** $6D$ $85$

(5) **Question 12.** $4000\ 6C31\ \text{std 2,x+}$

\[
\begin{align*}
R\ &\text{$4000\ \text{opcode fetch}} \\
R\ &\text{$4001\ \text{operand fetch}} \\
W\ &\text{$3900\ \text{data write}} \\
W\ &\text{$3901\ \text{data write}}
\end{align*}
\]

(5) **Question 13.** \(R = (5-2-0.5V)/20mA = 125 \text{ ohms}\)

(5) **Question 14.** One side of the switch is grounded and the other side has a 10k pullup to +5V

(15) **Question 15.**

Part a) Save RegY, setup up RegY to point into the stack, and allocates

\[
\begin{align*}
pshy & \\
tsy & \\
\text{leas} & -4,sp
\end{align*}
\]

Part b) Draw a stack picture, SP equals $3EF8$, Y equals $3EFC$

Part c) Show the symbolic binding

\[
\text{front} \ \text{equ} \ -4 \\
\text{back} \ \text{equ} \ -2
\]

Part d) \(\text{back} = 2000;\)

\[
\begin{align*}
\text{movw} & \ #2000,\text{back},Y \\
\text{ldd} & \ \text{back},Y \\
\text{lsld} & \ \text{front},Y \\
\text{std} & \ \text{front},Y
\end{align*}
\]

Part e) \(\text{front} = 2*\text{back};\)

\[
\begin{align*}
\text{ldd} & \ \text{back},Y \\
\text{lsld} & \ \text{front},Y
\end{align*}
\]

Part f) Deallocate the two 16-bit local variables, and restore Y.

\[
\begin{align*}
\text{leas} & \ 4,sp \ \text{(or \ tys)} \\
\text{puly} & \\
\end{align*}
\]
(15) Question 16. Mealy finite state machine. Part a) Show the ROM-based FSM data structure

```assembly
happy fcb 3,2 ; Outputs if input 0,1
fdb hungry,happy ; Next states if input 0,1
hungry fcb 7,3
fdb hungry,sleepy
sleepy fcb 4,8
fdb happy,hungry
```

Part b) Show the software.

```assembly
bset DDRM,#$0F ; PM3-0 are outputs
bclr DDRT,#$01 ; PT0 is an input
ldx #happy ; RegX is the State pointer
FSM ldab PTT ; Read input
andb #$01 ; just interested in bit 0
ldaa PTM ; Perform the output
anda #$30 ; retain bits 4,5 to be friendly
oraa B,x ; RegB is Output value for this state
staa PTM ; Output
lslb ; 2 bytes per 16 bit address
abx ; add 0,2 depending on input
ldx 2,x ; Next state depending on input
bra FSM
```

(15) Question 17. System that uses the RTI periodic interrupt to create waveform on PT7 output.

```assembly
org $3800
Count rmb 1 ; 0 when PT7 high, 1,-1 when PT7 low
org $4000
main lds #$4000
bset DDRT,#$80 ; PT7 output
movb #-1,Count ; interrupt counter
movb #$44,RTICTL ; 5.12ms
movb #$80,CRGINT ; arm RTIF
cli ; enable interrupts
loop bra loop
RTIhan movb #$80,CRGFLG ; acknowledge clear RTIF
inc Count
bne low
high bset PTT,#$80 ; PT7 is now high
bra done
low bclr PTT,#$80 ; PT7 is now low
ldaa Count
cmpa #2
bne done
movb #-1,Count ; goes ...-1,0,1,-1,0,1,...
done rti
org $FFF0
fdb RTIhan
org $FFFE
fdb main
```