(4) Question 1. List four experimental parameters to evaluate the quality of your new DAC.
1) The DAC *range* is the maximum and minimum DAC output
2) The DAC *resolution* is the smallest distinguishable change in output. The most dominate factor affecting resolution is *noise*.
3) The DAC *precision* is the number of different output values it can generate. Since, resolution is dominated by noise, and range is another term for signal, precision is very similar to *signal to noise ratio*.
4) The DAC *accuracy* is (Actual - Ideal) / Ideal where Ideal is the desired output.
5) A DAC is *monotonic* if an increase in digital value always causes an increase in analog value.
6) The *speed* of a DAC is how fast the output changes after the input changes. Similarly, the *bandwidth* of a DAC is the fastest sine wave that can be created.

(6) Question 2. 10kΩ in parallel with 10kΩ is 5kΩ. V = 5V*(5kΩ/25kΩ) = 1V

(10) Question 3. Write a C function to sample ADC channels 3 and 4 of the 9S12DP512.

```
unsigned short ADC_In34(void){
  ATD0CTL5 = 0x93; // start at Channel 3 and do multiple channels
  while((ATD0STAT0&0x80)==0){}; // wait for SCF
  if(ATD0DR0>ATD0DR1) return ATD0DR0;
  return ATD0DR1;
}
```

(15) Question 4. Write three C functions that operate the SCI0 module.

(5) Part a) Write an initialization function that turns on SCI0 with a baud rate of 38400 bits/sec.

```
//------------------------SCI0_Init-----------------------
// Initialize Serial port SCI0, baud rate 38400
// Input: none
// Output: none
// assumes a module clock frequency of 8 MHz
void SCI0_Init(void){
#define E 8000000
#define BAUD 38400
  SCI0BD = E/16/BAUD; // 500,000/38,400 = 13
  SCI0CR1 = 0;        // 8-bit
  SCI0CR2 = 0x0C;     // enable
}
```

(5) Part b) Write a C function that outputs one ASCII character.

```
//------------------------SCI0_OutChar-----------------------
// Wait for buffer to be empty, output 8-bit to serial port
// busy-waiting synchronization
// Input: 8-bit data to be transferred
// Output: none
#define TDRE 0x80
void SCI0_OutChar(char data){
  while((SCI0SR1&TDRE) == 0){}; // TDRE
  SCI0DRL = data;
}
```

(5) Part c) Write a C function that outputs an ASCII string to the SCI0.

```
//------------------------SCI0_OutString-----------------------
// Output String (NULL termination), busy-waiting synchronization
// Input: pointer to a NULL-terminated string to be transferred
// Output: none
void SCI0_OutString(char *pt){
  while(*pt){  // stop if data is zero
    SCI0_OutChar(*pt);
    pt++;
```

```
  }
}
```

(5) Question 5. Write assembly code that implements RegD= 0.3456*RegX.

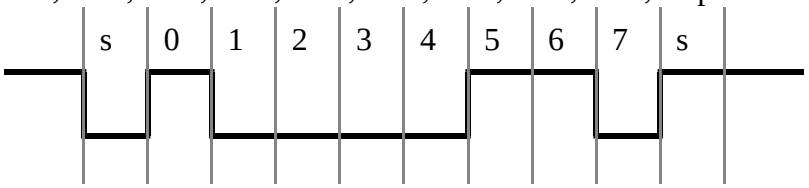
```
  tfr X,Y    ; input in X
  ldd #3456
  emul
  ldx #10000
  ediv
  tfr Y,D    ; result in D
```

(5) Question 6. Consider the result of executing the following two 9S12 assembly instructions.
First convert to unsigned. -50 is equivalent to -50+256=206. 156-206= -50 does not fit, so C=1
First convert to signed. 156 is equivalent to 156-256 = -100. -100- -50 = -50 fits, so V=0.

(5) Question 7. Give the simplified memory cycles produced

R/W	Addr	Data
R	$4200	$07
R	$4201	$0E
W	$3FF0	$42
W	$3FF1	$02

(5) Question 8. Start bit, bit 0, bit 1, bit 2, bit 3, bit 4, bit 5, bit 6, bit 7, stop bit



(5) Question 9. For each application choose the data structure that best matches.

Application	Data structure
A data structure used to implement buffered I/O.	fifo
A data structure used to store calibration data	table
A data structure used to implement a local variables	stack
A data structure used to implement a debugging dump	buffer
A data structure used to implement a Mealy machine	linked list

(20) Question 10. Write assembly code to implement the following FSM.

(5) Part a) Show the assembly code defining the FSM graph in ROM.

```
     org  $0800
Pt   rmb  2              ; pointer to current state
     org  $4000          ; Put linked list in ROM
Stop fcb  7              ; Output
     fdb  Stop,Go,Go,Go  ; Next
Go   fcb  3
     fdb  Stop,Go,Go,Turn
Turn fcb  5
     fdb  Stop,Go,Stop,Go
```

(7) Part b) Show the assembly subroutine to initialize the FSM controller

```
Init bset DDRT,#$07    ; PT2-0 are outputs
     bclr DDRH,#$03    ; PH0,PH1 are inputs
     ldy  #Stop
     sty  Pt           ; Pt is the State pointer
     ldab 0,y          ; RegB is output value for this state
     stab PTT          ; Perform the first output to PTT
     movb #$80,TSCR1   ; enable TCNT
     movb #$03,TSCR2   ; TCNT is 1 MHz
     bset TIOS,#$01    ; output compare 0
```

```
     bset TIE,#$01     ; arm output compare 0
     ldd  TCNT
     addd #1000        ; first output occurs for 1ms
     std  TC0
     cli               ; enable all interrupts
     rts
```

(8) Part c) Show the output compare 0 ISR including interrupt vector that runs the FSM.

```
OC0ISR movb #$01,TFLG1 ; acknowledge by clearing C0F
       ldy  Pt         ; RegY points to the current state
       ldab PTH        ; Read input
       andb #$03       ; just interested in bits 1,0
       lslb            ; 0,1,2,3 converted to 0,2,4,6
       aby             ; add 0,2,4,6 depending on input
       ldy  1,y        ; Next state depending on input
       ldab 0,y        ; RegB is output value for this state
       stab PTT        ; Perform the output to PTT
       sty  Pt
       ldd  TC0
       addd #1000      ; output occurs for 1ms
       std  TC0
       rti
       org  $FFEE
       fdb  OC0ISR
```

(10) Question 11. In this question, you will translate the C code line by line into 9S12 assembly.

```
void Run(void){                  Run
                                    movw #D1,2,-sp  ; push address of D1
  Add(&D1,&D2);                     movw #D2,2,-sp  ; push address of D2
                                    bsr  Add
                                    leas 4,sp       ; balance stack
}                                   rts
```

```
void Add(char *p1,
     char *p2){

  *p2 = *p2 + *p1;




}
```

```
p1  set  2
p2  set  4
Add
    ldx  p1,sp  ; X is p1
    ldy  p2,sp  ; Y is p2
    ldd  0,y    ; D is *p2
    addd 0,x    ; D is *p2 + *p1
    std  0,x    ; return by reference
    rts
```

(10) Question 12. For each definition fill in the term that best matches.

Definition	Term
Activate an individual trigger.	arm
The information transfer rate	bandwidth
how close it measures the desired parameter	accuracy
A process that fixes the inputs to a system	stabilization
storing most significant byte exists first	big endian
Establishing an upper bound.	ceiling
Clearing the interrupt trigger flag	acknowledge
An error that occurs after a right shift	drop out
presence does not affect	nonintrusive
A digital logic output that has two states low and HiZ.	open collector