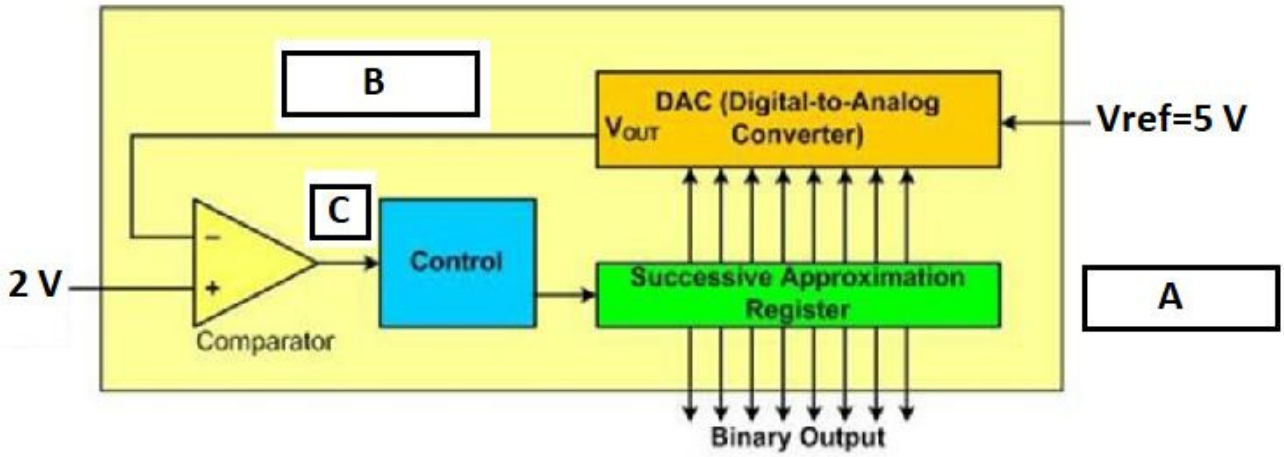
AIN SHAMS UNIVERSITY FACULTY OF ENGINEERING CREDIT HOURS ENGINEERING PROGRAMS		
June 2nd,2018	**Course Code: CSE 345&347**	***Time: 3 Hours***
Real-Time and Embedded Systems Design		
The Exam Consists of **6** Questions in **6** Pages	***Total Marks: 40 Marks***	**1/6**

ورق الأسئلة هو كراسة الأجابة – لابد من تدبيس ورق الأسئلة مع غلاف بيانات الطالب

(Solve in Designated Area - Assume Any Missing Data if Any)

Question (1):(5 Marks)



After each clock pulse, decide what are in the blocks A, B, and C, like values or decisions. "A" has the digital SAR value. "B" is the DAC output value in Volt. "C" could be "GT" (+ve > -ve) and "LT" otherwise. Fill your answer in the following table.

	A	B	C
After Clock Pulse 1			
After Clock Pulse 2			
After Clock Pulse 3			
After Clock Pulse 4			
After Clock Pulse 5			
After Clock Pulse 6			
After Clock Pulse 7			
After Clock Pulse 8			

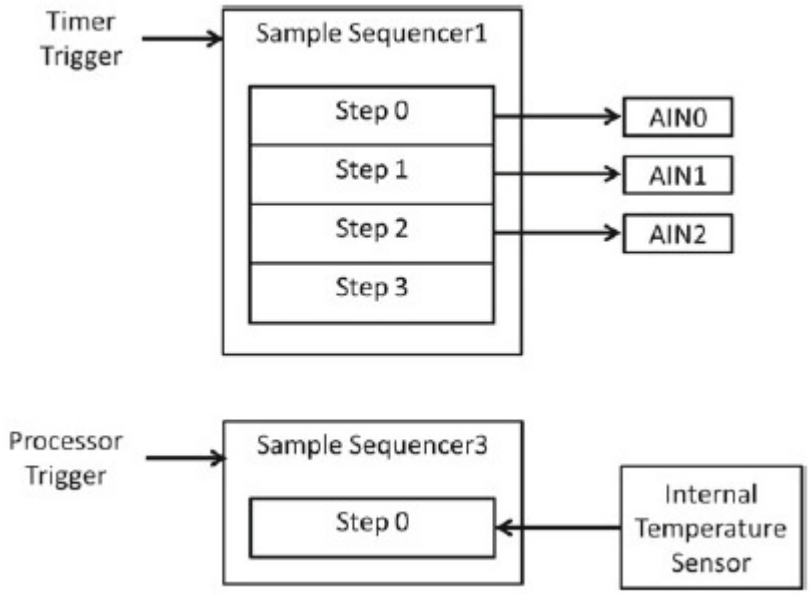
Question (2): (6 Marks)

The Figure below shows the example of configuration of Sample Sequencer 1 and Sample Sequencer 3. Complete the following C-Snippet to achieve this purpose. Few comments are only given. Do your best to recall TIVA-Ware APIs, arguments and necessary constants definitions.

```
/* Necessary Sequences OFF */




/* Necessary Sequences ON */
```



Question (3): (5 Marks)

CIRCLE THE CORRECT ANSWER

1. The number of bytes per I2C transfer is defined as the time period between a valid _____ and_____ condition and is unrestricted, but each data byte has to be followed by an acknowledge bit, and data must be transferred _____ first.
   a) START, LSB, STOP
   b) STOP, START, MSB
   c) LSB, START, STOP
   d) START, STOP, MSB

2. In each I2C module, after the START condition, a slave address is first transmitted. This address is_____ bits long followed by a _____ bit.
   a) 8, START
   b) 8, STOP
   c) 7, R/S
   d) 8, R/S

3. If a slave is servicing an internal interrupt, which of the following will it do to avoid losing data?
   a) Terminate the interrupt.
   b) Stop the clock and reset the system.
   c) Stretch the clock until the interrupt servicing is complete.
   d) Stop servicing the interrupt and get the data from the bus.
   e) Turn itself into the bus master

4. When must data be stable for a correct I²C bus transaction?
   a) When the clock is low
   b) When the clock is in high-impedance state
   c) When the clock transitions high to low
   d) When the clock is high
   e) When the clock transitions low to high
   f) Anytime is acceptable

5. The I²C bus can connect several devices at the same time to its clock and data lines in a wired-AND format. Which kind of outputs must a device have to operate on the I²C bus?
   a) Tri-state
   b) Open collector
   c) Emitter follower
   d) Source follower
   e) Open drain
   f) a and c only
   g) b and e only

AIN SHAMS UNIVERSITY FACULTY OF ENGINEERING CREDIT HOURS ENGINEERING PROGRAMS		
June 2nd,2018	**Course Code: CSE 345&347**	***Time: 3 Hours***
Real-Time and Embedded Systems Design		
The Exam Consists of **6** Questions in **6** Pages	***Total Marks: 40 Marks***	**4/6**

Question (4):(8 Marks)

The figure below is a snap shot from a debugging session. In the following table, document your expectation of the hitting-order of Breakpoints (designated by line number). Consider multiple hits of a break point, inside a loop, as ONLY 1 HIT.

```
Breakpoint 77
Breakpoint 79
Breakpoint 82
Breakpoint 84
Breakpoint 86
Breakpoint 95
Breakpoint 98
Breakpoint 100

63  int main( void )
64  {
65    xMutex = xSemaphoreCreateMutex();
66    if( xMutex != NULL )
67    {
68      xTaskCreate( prvPrintTask, "Print1", 240, "Task 1 **\n", 1, NULL );
69      xTaskCreate( prvPrintTask, "Print2", 240, "Task 2 --\n", 2, NULL );
70      vTaskStartScheduler();
71    }
72  }
73  static void prvNewPrintString( const portCHAR *pcString )
74  {
75  static char cBuffer[ mainMAX_MSG_LEN ];
76  int i,j;
77    xSemaphoreTake( xMutex, portMAX_DELAY );
78    {
79      sprintf( cBuffer, "%s", pcString );
80      for (i=1;i<1000000;i++)
81      {
82        j++;
83      }
84      printf( cBuffer );
85    }
86    xSemaphoreGive( xMutex );
87  }
88  static void prvPrintTask( void *pvParameters )
89  {
90  char *pcStringToPrint;
91  int i,j;
92    pcStringToPrint = ( char * ) pvParameters;
93    for( ;; )
94    {
95      prvNewPrintString( pcStringToPrint );
96      for (i=1;i<1000000;i++)
97      {
98        j++;
99      }
100     vTaskDelay( 255 );
101   }
102 }
```

Hit Order	Break Point No.	Hit Order	Break Point No.	Hit Order	Break Point No.
1		***11***		***21***	
2		***12***		***22***	
3		***13***		***23***	
4		***14***		***24***	
5		***15***		***25***	
6		***16***		***26***	
7		***17***		***27***	
8		***18***		***28***	
9		***19***		***29***	
10		***20***		***30***	

Question (5): (8 Marks)

The figure below is a snap shot from FreeRTOS application using tasks communicating through a queue. Fill the given table with task states (Ready, Running, or Blocked) in consecutive time slots Tx. Show the content of the queue at the end of each slot assuming initial HEX value of (00 00 00 00 – 00 00 00 00 – 00 00 00 00). "Sender 2" is the first running task just after scheduler-start in T1.

```
60  int main( void )
61  {
62      xQueue = xQueueCreate( 3, sizeof( long ) );
63      xTaskCreate( vSenderTask, "Sender1", 240, ( void * ) 100, 2, NULL );
64      xTaskCreate( vSenderTask, "Sender2", 240, ( void * ) 200, 2, NULL );
65      xTaskCreate( vReceiverTask, "Receiver", 240, NULL, 1, NULL );
66      vTaskStartScheduler();
67  }
68  static void vSenderTask( void *pvParameters )
69  {
70  long lValueToSend;
71  const portTickType xTicksToWait = 100 / portTICK_RATE_MS;
72  lValueToSend = ( long ) pvParameters;
73    for( ;; )
74    {
75      xQueueSendToBack( xQueue, &lValueToSend, xTicksToWait );
76      taskYIELD();
77    }
78  }
79  static void vReceiverTask( void *pvParameters )
80  {
81  long lReceivedValue;
82    for( ;; )
83    {
84      xQueueReceive( xQueue, &lReceivedValue, 0 );
85      vPrintStringAndNumber( "Received = ", lReceivedValue );
86    }
87  }
```

	Sender 1	Sender 2	Receiver	Queue Content at The End of Tx Period
T1		Running		C8 00 00 00 00 00 00 00 00 00 00 00
T2				
T3				
T4				
T5				
T6				
T7				
T8				
T9				
T10				
T11				
T12				
T13				
T14				
T15				
T16				

AIN SHAMS UNIVERSITY FACULTY OF ENGINEERING CREDIT HOURS ENGINEERING PROGRAMS		
June 2nd,2018	**Course Code: CSE 345&347**	***Time: 3 Hours***
Real-Time and Embedded Systems Design		
The Exam Consists of **6** Questions in **6** Pages	***Total Marks: 40 Marks***	**6/6**

Question (6):(8 Marks)

The figure below is a snap shot from a debugging session. Show the sequence of execution timing diagram starting just after vTaskStartScheduler. Tasks should be shown down-up vertically according to their priorities (e.g. Highest priority should be on the top). Also show the debug (printf) viewer.

```
 82  int main( void )
 83  {
 84      vSemaphoreCreateBinary( xBinarySemaphore );
 85      if( xBinarySemaphore != NULL )
 86      {
 87          prvSetupSoftwareInterrupt();
 88          xTaskCreate( vHandlerTask, "Handler", 240, NULL, 1, NULL );
 89          xTaskCreate( vPeriodicTask, "Periodic", 240, NULL, 3, NULL );
 90          vTaskStartScheduler();
 91      }
 92  }
 93  static void vHandlerTask( void *pvParameters )
 94  {
 95      xSemaphoreTake( xBinarySemaphore, 0 );
 96      for( ;; )
 97      {
 98          xSemaphoreTake( xBinarySemaphore, portMAX_DELAY );
 99          vPrintString( "Handler task - Processing event.\n" );
100      }
101  }
102  static void vPeriodicTask( void *pvParameters )
103  {
104      for( ;; )
105      {
106          vTaskDelay( 500 / portTICK_RATE_MS );
107          vPrintString( "Periodic task - About to generate an interrupt.\n" );
108          mainTRIGGER_INTERRUPT();
109          vPrintString( "Periodic task - Interrupt generated.\n\n" );
110      }
111  }
112  void vSoftwareInterruptHandler( void )
113  {
114  portBASE_TYPE xHigherPriorityTaskWoken = pdFALSE;
115      xSemaphoreGiveFromISR( xBinarySemaphore, &xHigherPriorityTaskWoken );
116      mainCLEAR_INTERRUPT();
117      portEND_SWITCHING_ISR( xHigherPriorityTaskWoken );
118  }
```

End of Questions