June 19th., 2019 | **Course Code: CSE345 & CSE347** | **Time: 3 Hours**

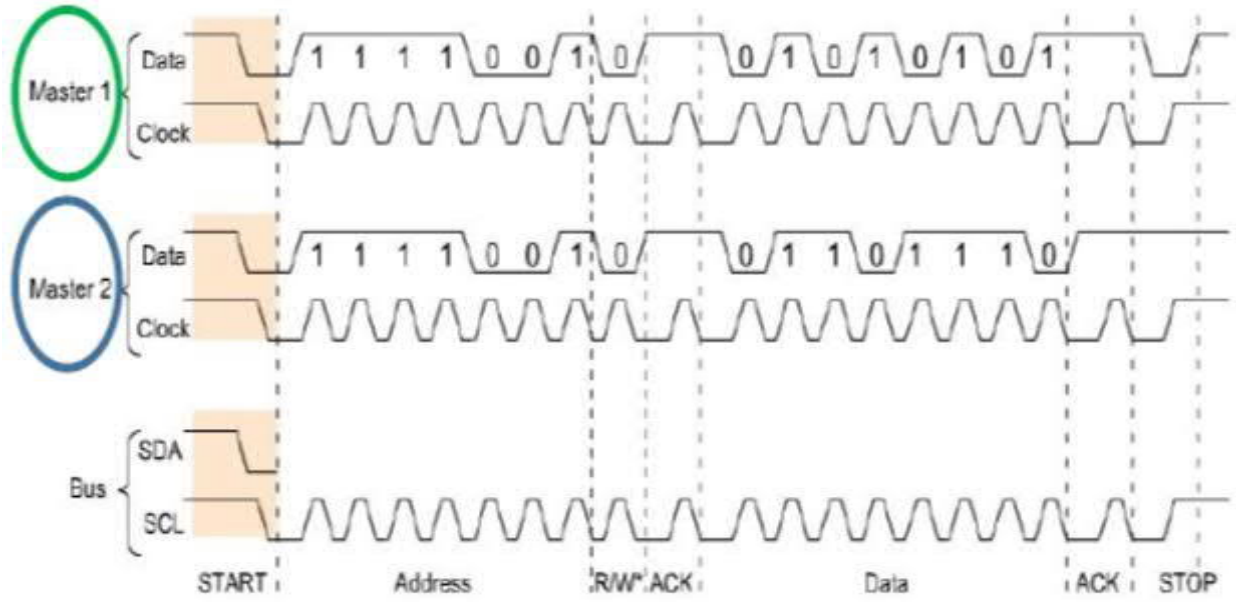
Real Time - Embedded System Design

The Exam Consists of **6** Questions in **4** Pages | **Total Marks: 40 Marks**

هذه ورقة إجابة أيضا – على كل طالب تدبيسها من الناحية اليسرى في الغلاف الرسمي المعد لذلك – وكتابة بيانات الطالب عليه

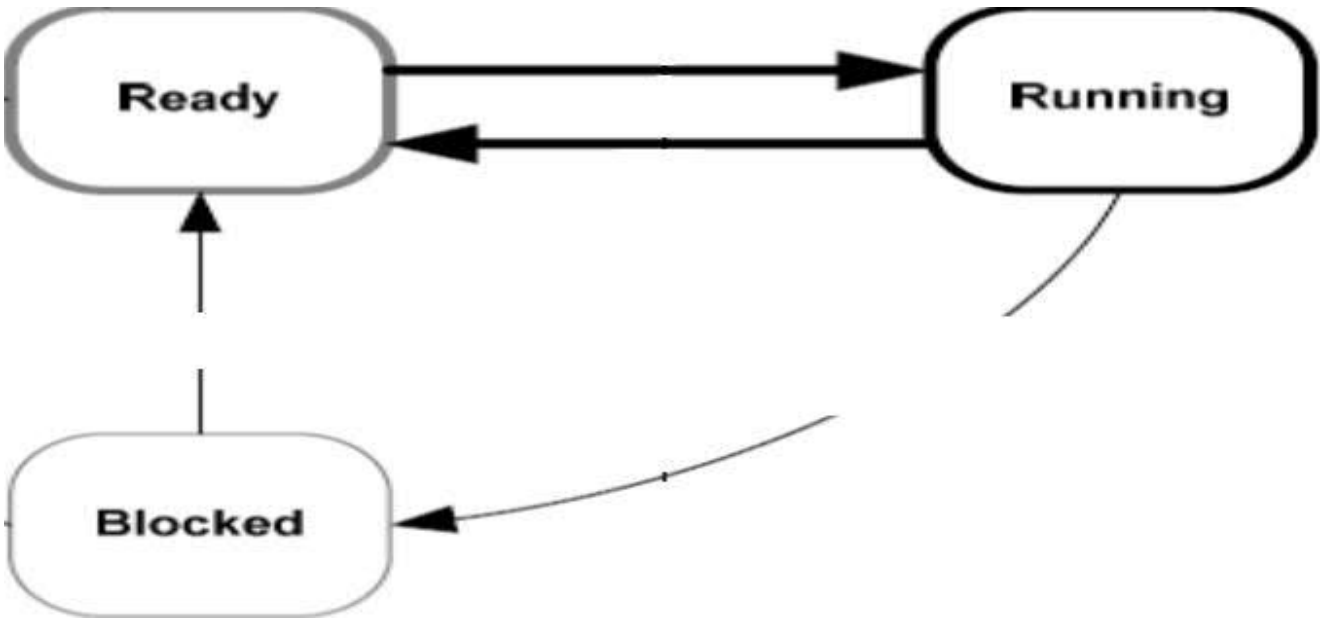
Question (1): (7 Marks)

In the figure below, there are two masters communicating on an I2C bus. Complete the SDA line bits switching.



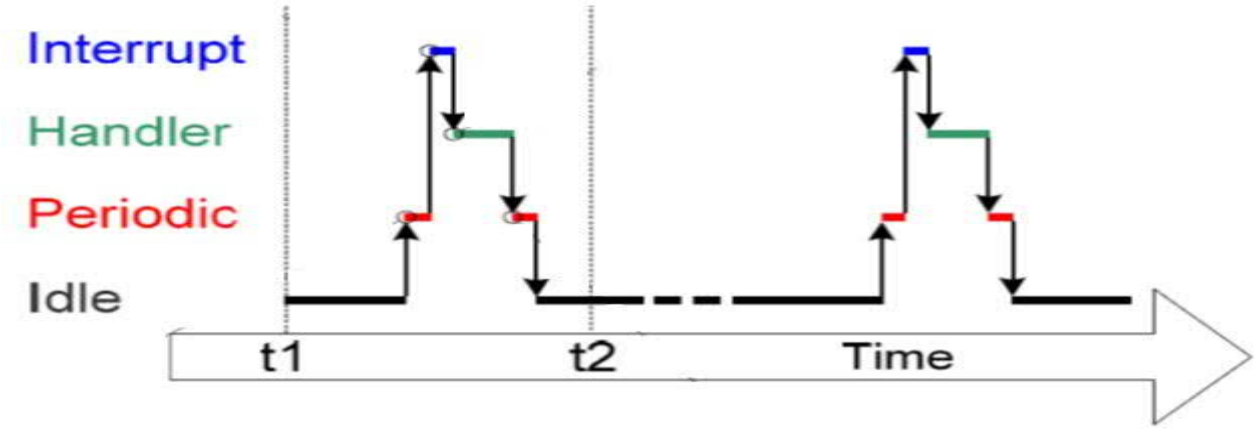
Question (2): (7 Marks)

The figure below shows a subset of RTOS task state machine. On "Cut" arrows, give all conditions of transitions from one state to another.



Question (3): (7 Marks)

The following figure is the timing diagram of a FreeRTOS application. Write the C-code that could achieve this interrupt handling. Assume any missing data if any.



Question (4): (7 Marks) (Assume Missing Data if Any)
Write the code to reproduce the following "Dead Lock" scenario where Task A and Task B both need to acquire mutex X and mutex Y in order to perform an action:
1. Task A executes and successfully takes mutex X.
2. Task A is pre-empted by Task B.
3. Task B successfully takes mutex Y before attempting to also take mutex X
4. If Task A continues executing, it will attempt to take mutex Y

Question (5): (5 Marks)
Complete the following:

a) Normally, queues are used as ---------- buffers where data is written to ---------- of the queue and removed from ---------- of the queue.

b) A task, that is blocked on queue read, will be moved automatically from the ---------- to the ---------- if the ---------- before data becomes available.

c) If the queue-blocked tasks have equal priority, and the chance comes, then the task that ---------- will be unblocked.

d) During "queue write", the block time is the maximum time the task should be held in the ------ state to wait for ---------- on the queue, should the queue already be ----------.

Question (6): (7 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 numbers; 75, 86 and 87), and the content of designated heap.

```
59  int main( void )
60  {
61    xQueue = xQueueCreate( 2, sizeof( long ) );
62    xTaskCreate( vSenderTask, "Sender1", 240, ( void * ) 100, 2, NULL );
63    xTaskCreate( vSenderTask, "Sender2", 240, ( void * ) 200, 2, NULL );
64    xTaskCreate( vReceiverTask, "Receiver", 240, NULL, 1, NULL );
65    vTaskStartScheduler();
66  }
67  static void vSenderTask( void *pvParameters )
68  {
69  long lValueToSend;
70  portBASE_TYPE xStatus;
71  const portTickType xTicksToWait = 100 / portTICK_RATE_MS;
72    lValueToSend = ( long ) pvParameters;
73    for( ;; )
74    {
75      xStatus = xQueueSendToBack( xQueue, &lValueToSend, xTicksToWait );
76      taskYIELD();
77    }
78  }
79  static void vReceiverTask( void *pvParameters )
80  {
81  long lReceivedValue;
82  portBASE_TYPE xStatus;
83  const portTickType xTicksToWait = 100 / portTICK_RATE_MS;
84    for( ;; )
85    {
86      xStatus = xQueueReceive( xQueue, &lReceivedValue, xTicksToWait );
87      vPrintStringAndNumber( "Received = ", lReceivedValue );
88    }
89  }
```

Hit Order	Break Point No.	Content of Designated Heap
1		
2		
3		
4		
5		
6		
7		

End of Questions