Scoring Indicators

COURSE NAME: Embedded System and Real time operating system

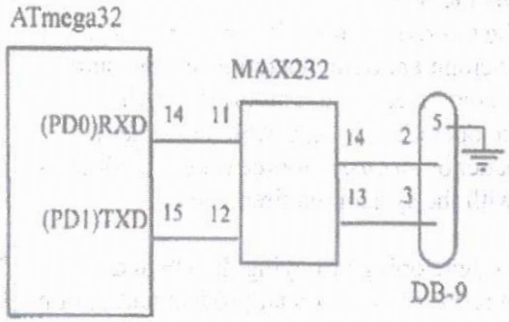
COURSE CODE :5131 QID: 2109230282

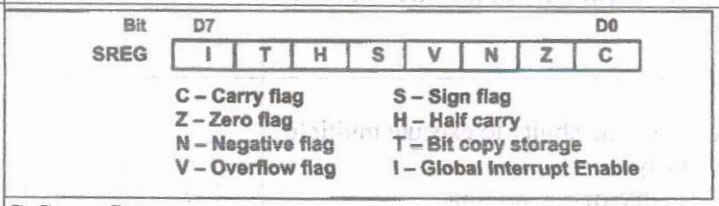
Revision:2021

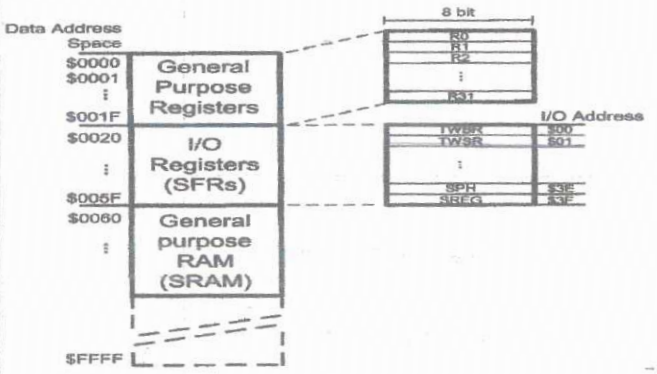
Q No	Scoring Indicators	Split score	Sub Total	Total score
I	**PART A**			9
I. 1	1.Consumer Electronics 2. Household appliances 3. Home automation and security 4. Telecom 5. Healthcare 6. Card Readers 7. Wearable devices	List any two: ½ marks each	1	
I. 2	Harvard Architecture	1	1	
I. 3	0xFF	1	1	
I. 4	DDRD=0xFF;	1	1	
I. 5	16 bit	1	1	
I. 6	Input pin	1	1	
I. 7	1s(high)	1	1	
I. 8	Kernel	1	1	
I. 9	Thread	1	1	
	PART B			24
II. 1	1.Data collection/Storage/Representation 2.Data Communication 3.Data (Signal)processing 4.Minitoring 5.Control 6.Application Specific User Interface	Any three: 1 mark each	3	
II. 2	DDRx (Data Direction Register): Configuring a given port as input or output PORTx (Port Output Pin): To write the data to the pins PINx (Port Input Pin): To read the data present at the pins.	1 1 1	3	

II. 3	`#include <avr/io.h>` `//standard AVR header`
	`int main(void)`
	`{`
	`  unsigned char temp;`
	
	`  DDRB = 0x00;` `//Port B is input`
	`  DDRC = 0xFF;` `//Port C is output`
	
	`  while(1)`
	`  {`
	`  temp = PINB;`
	`  PORTC = temp;`
	`  }`
	`  return 0;`
	`}`
		1
		
		2
			3	

II. 4	**AND OR EX-OR Inverter** **A B A&B A|B A^B Y=~B** 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0 The following shows some examples using the C bit-wise operators: 1. 0x35 & 0x0F = 0x05 /* ANDing */ 2. 0x04 | 0x68 = 0x6C /* ORing */ 3. 0x54 ^ 0x78 = 0x2C /* XORing */ 4. ~0x55 = 0xAA /* Inverting 55H */	Listing:2 Examples:1	3	
II. 5	Some of the most widely used sources of interrupts in the AVR are given below: 1. There are at least two interrupts set aside for each of the timers, one for ***overflow*** and another ***for compare match***. 2. Three interrupts are set aside for external hardware interrupts. Pins PD2, PD3 and PB2 are for external hardware interrupts INT0, INT1 and INT2, respectively. 3. Serial communications USART has three interrupts, one for receive and two interrupts for transmit. 4. The SPI (Serial Peripheral Interface) interrupt. 5. The ADC.	Any 3: 3 marks	3	
II. 6	• To send data to LCD, Make pins RS=1 and R/W=0. • Then puta data on the data pins (D0-D7) and send a high-to-low pulse to the E pin to enable the internal latch of the LCD. • After sending data wait for100 microseconds to let the LCD module to write the data on the screen.	3	3	

II. 7		3	3	
II. 8	Hard Real-time • Strictly adhere to the time constraints for a task . • Must meet the deadline for a task without any slippage. • Missing any deadline may produce catastrophic results for Hard Real-time system • Eg: Air bag Control System, Anti-lock brake system for vehicles Soft Real-time: • Doesn't guarantee meeting deadlines, but offer best effort to meet deadline • Missing deadlines for tasks are acceptable • Eg: Automatic Teller machine, Audio-video playback system	1 ½ 1 ½	3	
II.9	**Multiprocessing** • *Multiprocessing* describes the ability to execute multiple processes simultaneously. • Systems which are capable of performing multiprocessing, are known as *multiprocessor* systems. • *Multiprocessor* systems possess multiple CPUs and can execute multiple processes simultaneously. **Multitasking** • The ability of an operating system to hold multiple processes in memory and switch the processor (CPU) from executing one process to another process is known as *multitasking.* • Multitasking creates the illusion of multiple tasks executing in parallel. • Multitasking involves the switching of CPU from executing one task to another.	1 ½ 1 ½	3	

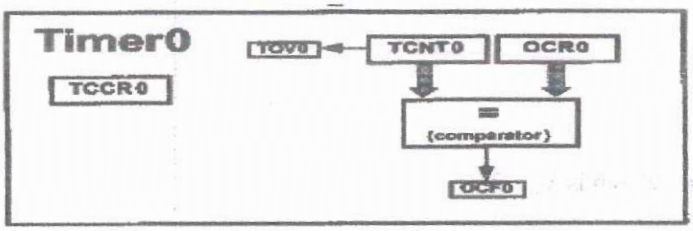
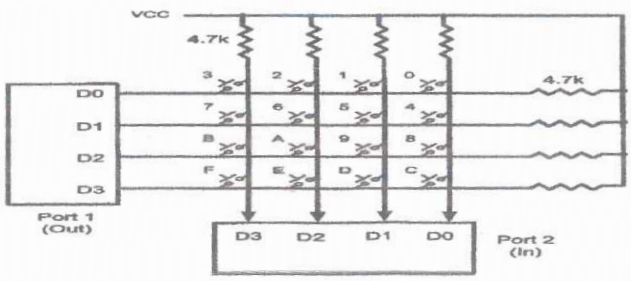
II.10	1. Custom Developed or Off the Shelf • Depending on the OS requirement, it is possible to go for the complete development of an operating system suiting the embedded system needs or use an off the shelf. • Readily available operating system, which is either a commercial product or an Open-Source product, which is in close match with the system requirements 2. Cost • The total cost for developing or buying the OS and maintaining it in terms of commercial product and custom build needs to be evaluated before taking a decision on the selection of OS. 3. Development and Debugging Tools Availability • The availability of development and debugging tools is a critical decision-making factor in the selection of an OS for embedded design. 4. After Sales • For a commercial embedded RTOS, after sales in the form of e-mail, on-call services, etc. for bug fixes, critical patch updates and support for production issues, etc. should be analyzed thoroughly. 5 Ease of Use • How easy it is to use a commercial RTOS is another important feature that needs to be considered in the RTOS selection	Any three: 1 mark each	3	
	PART C			
III	 C-Carry flag • This flag is set whenever there is a carry out from the D7 bit. • This flag bit is affected after an 8-bit addition or subtraction. Z-Zero flag • This flag reflect the result of an arithmetic or logic operation • If the result is 0 then Z=1 • If the result is not = 0 then Z=0 N-Negative flag • Binary representation of signed numbers uses D7 as the sign bit. • Negative flag reflects the result of an arithmetic operation. • If the D7 bit of the result is 0, then N=0 and the result is positive. • If the D7 bit of the result is 1, then N=1 and the result is negative. V-Overflow flag • This flag is set whenever the result of an signed number operation is too large . • Overflow flag is used to detect errors in signed arithmetic operation S-Sign flag • This flag is the result of exclusive ORing of N &V flags	2 5	7	7

	H-Half carry • If there is a carry from D3 to D4 during ADD or SUB operation, this bit is set, otherwise it is cleared. • This bit is used by instruction that performs BCD operation T-Bit Copy Storage • Used with BLD (bit load) and BST (bit store)instructions for loading and storing bit from one register to another. I-Global InterruptEnable • Setting this bit enabled all the interrupt. • Resetting this disables all interrupts			
IV.	 Data memory composed of three parts: - • GPRS • I/O memory • Internal data SRAM GPRs • It uses 32 bytes of data memory space. • They always take the address location $00 - $1F in the data memory space, regardless of the AVR chip number I/O MEMORY(SFRs) • functions of I/O memory are fixed by the CPU designer at the time of design. • it is used for control of the microcontroller or peripheral. • the AVR I/O memory is made of 8-bit registers. • it is used for specific functions such as status register, timers, serial communication etc. • all the AVR ha at least 64bytes of I/O memory locations Internal data SRAM • Internal data is widely used for storing data and parameters by AVR programmers and C compilers. • Generally, that is called ***scratch pad***. • Each location of SRAM can be accessed by its address. • Each location is 8 bits wide. • Size of data SRAM can vary from chip to chip	4 3	7	7

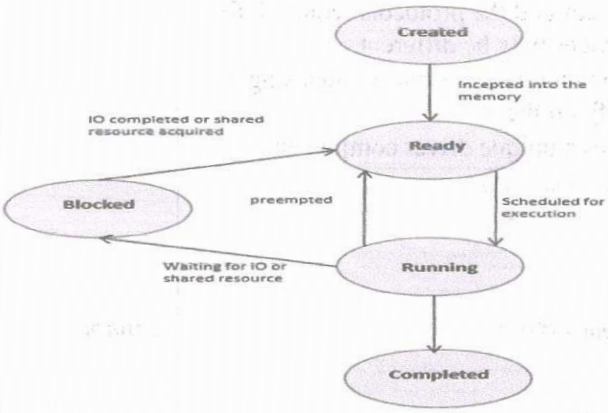
V.	`#include <avr/io.h>  //standard AVR header`
	`int main(void)`
	`{`
	`  DDRC = 0;  //Port C is input`
	`  DDRB = 0xFF;  //Port B is output`
	`  DDRD = 0xFF;  //Port D is output`
	`  unsigned char temp;`
	`  while(1)`
	`  {`
	`  temp = PINC;  //read from PINB`
	`  if ( temp < 100 )`
	`  PORTB = temp;`
	`  else`
	`  PORTD = temp;`
	`  }`
	`  return 0;`
	`}`
		Port configuration: 2 marks
		
		Logic: 5 marks
			7	7

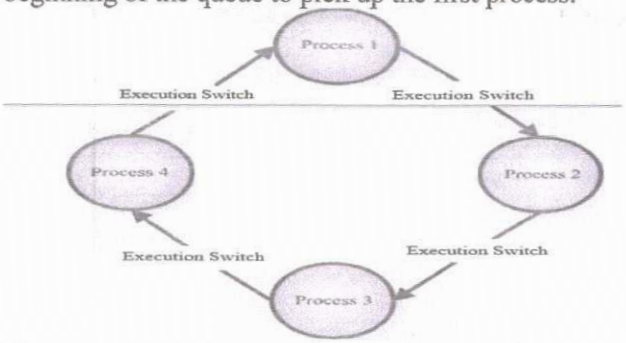
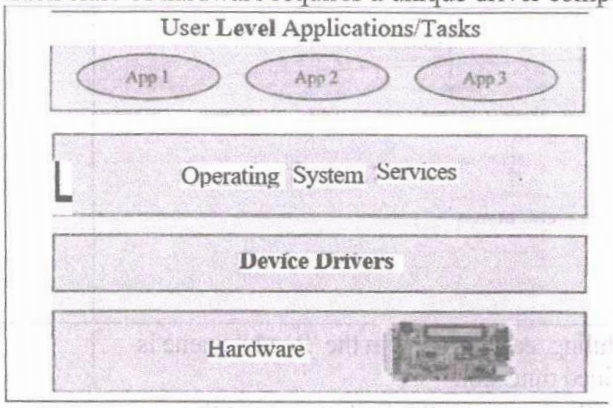
VI.	Inverting Operator(~)
	`#include <avr/io.h>  //standard AVR header`
	`int main(void)`
	`{`
	`  DDRB = 0xFF;  //Port B is output`
	`  PORTB = 0xAA;`
	`  while (1)`
	`  PORTB = ~ PORTB;  //toggle PORTB`
	`  return 0;`
	`}`
	EX-OR Operator(^)
	`#include <avr/io.h>  //standard AVR header`
	`int main(void)`
	`{`
	`  DDRB = 0xFF;  //Port B is output`
	`  PORTB = 0xAA;`
	`  while (1)`
	`  PORTB = PORTB ^ 0xFF;`
	`  return 0;`
	`}`
		3 ½
		
		3 ½
			7	7

VII.	`#include<avr/io.h>`
	`int main(void)`
	`{`
	`unsigned char x,y;`
	`unsigned char mybyte=0x42;`
	`DDRB=DDRC=0xFF;`
	`x=mybyte & 0x0F;`
	`PORTC=x | 0x30;`
	`y=mybyte & 0xF0;`
	`y=y>>4;  //shift right to lower 4 bits`
	`PORTD=y | 0x30`
	`}`
	Output:
	PORTC=0x32
	PORTD=0x34
		Port Configuration:2
		
		Logic :5
			7	7

VIII.	**1.TCNTn Register((timer/counter)** • In AVR, for each of the timers, there is a TCNTn (timer/counter) register. • ATmega32 have TCNT0, TCNT1 and TCNT2. • The TCNTn register is a counter. • Upon reset, the TCNTn contains zero. • It counts up with each pulse. • The content of the timers/counters can be accessed using TCNTn. • We can load a value into the TCNTn register or read its value. **2. TCCRn (timer/counter control register)** • Each timer also has the TCCRn (timer/counter control register) register for setting modes of operation. • For example, you can specify Timer0 to work as a timer or a counter by loading proper values into the TCCR0. **3. OCRn (output compare register) register** • Each timer also has an OCRn (output compare register) register. • The content of the OCRn is compared with the content of the TCNTn. • When they are equal the OCFn (output compare flag) flag will be set. **4. TIFR (Timer/Couter Interrupt Flag Register** • Each timer has a TOVn (Timer Overflow) flag. • When a timer overflows, its TOVn flag will be set. • It's a part of TIFR (Timer/Couter Interrupt Flag Register) 	Explanation of any two:3 ½ marks each Figure:Optional	7	7
IX.	**Scanning and identifying the key** • A 4×4 matrix keyboard is connected to two ports. • The rows are connected to an output port and the columns are connected to an input port. 	4	7	7

	• If no key has been pressed, reading input port will yield **1s** for all columns since they are all connected to high (VCC). • If all rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground. • It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed.	3		
X.	• ADC (Analog to Digital Converter) is the most widely used devices for data acquisition. • Digital computers use binary data, but the most of the physical quantiles (Temperature, pressure etc.) are analog signals. • ADC convert the analog signals to digital numbers so that microcontroller can read and process them. **Major Characteristics of ADC** **1.Resolution: -** • It is a function of bits used to represent digital value at its output. • The ADC has n-bit resolution, where 'n' is the number of bits used to represent the digital value. **2.V_{ref}:-** • It is the input voltage used for reference in ADC. • The reference voltage and the number of steps in the o/p will determine the step size of ADC. • Step size=V_{ref}/ number of steps. **3.Conversion time: -** • It is defined as the time required to convert an analog signal into digital output. **4.Digital data output: -** $D_{out} = V_{in}$/step size V_{in} =Analog input voltage Step size is the smallest change, which is V_{ref}/256	2 marks Any two: 5 marks (2 ½ marks each)	7	7

XI.	Functions Of an Operating system 1.Process management 2. Primary memory management. 3.File system management, 4. I/O system management, 5. Secondary storage management 6. Protection system 7. Interrupt handler	Listing:1 mark Explanation Of any 3: 6marks	7	7
XII.	The state at which a process is being created is referred as '**Created State**'. The state, where a process is incepted into the memory and awaiting the processor time for execution, is known as '**Ready State**' '**Running state**' is the state at which the process execution happens (execution of instructions). '**Blocked State/Wait State**' refers to a state where a running process is temporarily suspended from execution and does not have immediate access to resources. A state where the process completes its execution is known as '**Completed State**'.  	Transition Diagram:4 marks Explanation of each state:3marks	7	7
XIII.	• In Round Robin scheduling, each process in the 'Ready' queue is executed for a pre-defined time slot. • The execution starts with picking up the first process in the 'Ready' queue (It is executed for a pre-defined time and when the pre-defined time elapses or the process completes (before the pre-defined time slice), the next process in the 'Ready' queue is selected for execution. • This is repeated for all the processes in the 'Ready' queue. • Once each process in the 'Ready' queue is executed for the pre-defined time period, the scheduler comes back and picks the first process in the 'Ready' queue again for execution. • The sequence is repeated. • The Round Robin scheduling is similar to the FCFS scheduling			

	with preemption. • Preemption is added to switch the execution between the processes in the 'Ready' queue. • The 'Ready' queue can be considered as a circular queue in which the scheduler picks up the first process for execution and moves to the next till the end of the queue and then comes back to the beginning of the queue to pick up the first process.  	5 2	7	7
XIV.	• Device driver is a piece of software that acts as a bridge • between the operating system and the hardware. • Device drivers are responsible for initiating and managing the communication with the hardware peripherals. • They are responsible for establishing the connectivity,initializing the hardware and transferring data. • An embedded product may contain different types of hardware components like Wi-Fi module, File systems, Storage device interface, etc. • The initialization of these devices and the protocols required for communicating with these devices may be different. • All these requirements are implemented in drivers and a single driver will not be able to satisfy all these. • Each class of hardware requires a unique driver component.  •	5marks 2 marks	7	7