Set B

(83)

April - 24

Page No - 13

Scoring Indicators

COURSENAME: EMBEDDED SYSTEMS AND REAL TIME OPERATING SYSTEM

COURSECODE: 5131

QP

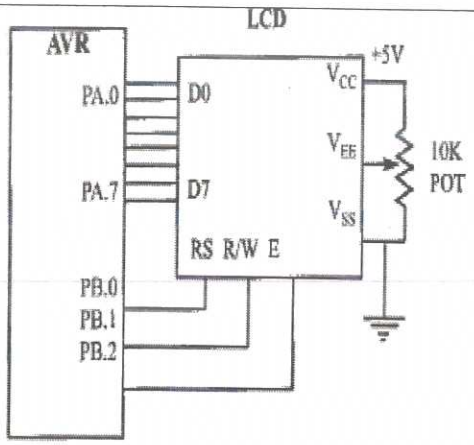
QID: **QID : 2109230281**

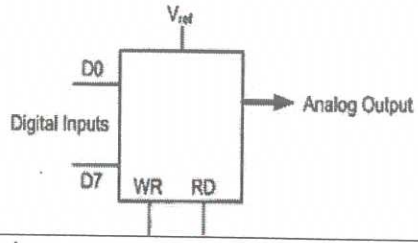
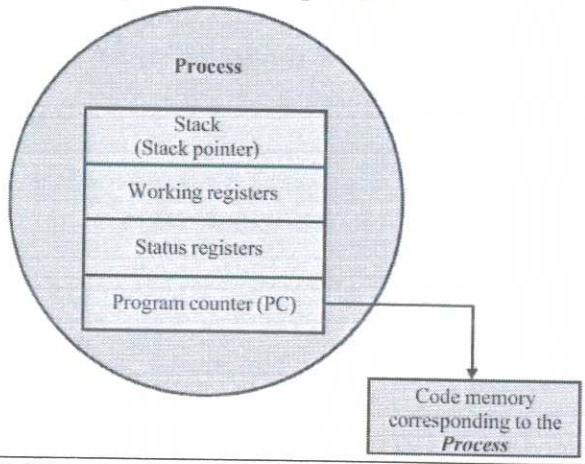
Q No	Scoring Indicators	Split score	Sub Total	Total score
	PARTA			9
I.1	Sensors converts energy from one form to another for any measurement or control purpose. Sensors acts as input device.	1	1	9
I.2	PORTA, PORTB, PORTC, PORTD	1	1	
I.3	DDRB = 0X00 or DDRB = 0b00000000 or DDRB = 0	1	1	
I. 4	Timer/counter, Timer overflow, Timer/counter control, output compare register	1	1	
I.5	The program associated with the interrupt is called the interrupt service routine(ISR) or interrupt handler.	1	1	
I. 6	RS232	1	1	
I. 7	To select the command register or data register	1	1	
I. 8	Used for holding the information corresponding to a task.	1	1	
I. 9	*Thread* is a single sequential flow of control within a process. It is a light weight process.	1	1	
	PARTB			24
II.1	**Micro processor** / **Microcontroller**	1 x 3	3	3
	A silicon chip contains a CPU which is capable of performing arithmetic and logical operations according to a pre-defined set of instructions / An integrated chip contains a CPU, scratchpad RAM, special & general purpose register arrays, on chip ROM, timer and interrupt control units and i/o ports			
	It is a dependant unit / It is a self contained unit			
	General purpose in design and operation / Application-oriented or domain-specific			
	Doesn't contain built in I/O ports. The I/O port functionality to be implemented with external programmable peripheral interface 8255 / Contains multiple built-in I/O ports which can be operated as a single 8 or 16 or 32 bit port or individual port pins			
	Performance is important / Performance is critical			

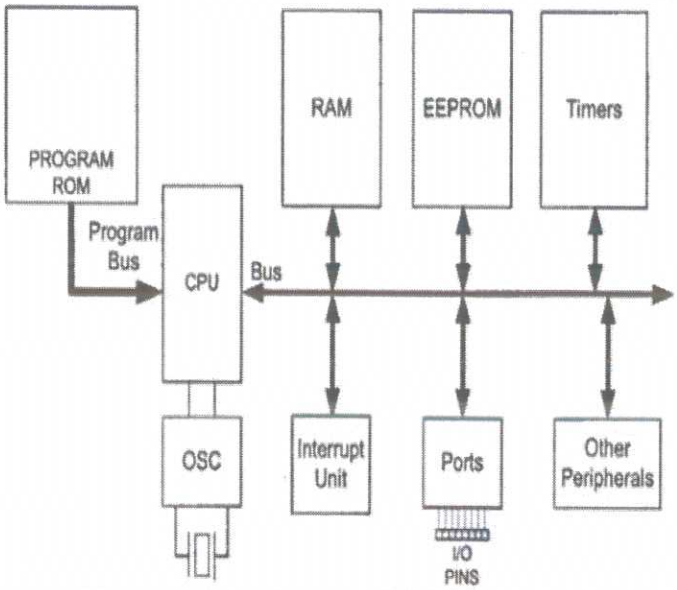
	Limited power savings | Lots of power saving (any three)			
II. 2	Bit D7 D0 SREG | I | T | H | S | V | N | Z | C | C – Carry flag S – Sign flag Z – Zero flag H – Half carry N – Negative flag T – Bit copy storage V – Overflow flag I – Global Interrupt Enable	3	3	3
II. 3	• ***Each port has three I/O registers associated with it.*** • They are designated as **PORTx, DDRx** (Data Direction Register) & **PINx** (Port INput pins). • Each I/O registers is 8 bit wide and each port has a maximum of 8 pins. • Each bit of the I/O registers affects one of the pins • Eg:- For port B, we have PORT B, DDRB & PINB • Each of the ports (A – D) in the ATMEGA 32 can be used for input or output. • DDRx register is used for the purpose of making a given port as input or output port. • **PIN (Port Input pin) Register** • To read the data present at the pins. • To bring the data into CPU from pins , read the contents of the PINx register. • To send data out to pins, use PORTx register. • **PORTx - Port Registers** • ATMega 32 has four ports, each having 8 pins. • Each port is associated with one register. • To use the pins of Portx as input or output, depends on the corresponding bit of DDRx register . • The four registers are – Port A, Port B, Port C and Port D	3	3	3

II. 4	# include <avr/io.h>
	int main (void)
	{
	unsigned char x;
	DDRB = 0xFF;
	for(x = 0; x<=255; x++)
	PORTB = z;
	return 0;
	}
		3	3	3

II. 5	• There are many sources of interrupts, depending on which peripheral is incorporated in to the microcontroller. • Widely used sources of interrupts are: ○ Two interrupts set aside for each of the timers are over flow and compare match.	3	3	3

	○ Three interrupts are set aside for external hardware interrupts. Pins PD2 (PORTD.2), PD3(PORTD.3), and PB2(PORTB.2) are for the external hardware interrupts INT0 ,INT1, and INT2, respectively. ○ Serial communication's USART has three interrupts - one for receive and two interrupts for transmit. ○ The SPI (Serial Peripheral Interface) interrupts. ○ The ADC (analog-to-digital converter).			
II.6	1. It finishes the instruction it is currently executing and saves the address of the next instruction (program counter)on the stack. 2. It jumps toa fixed location n memory called the interrupt vector table. . The Interrupt vector table directs the microcontroller to the address of the interrupt service routine (ISR). 3. The microcontroller starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine, which is RETI (return from interrupt). 4. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted. First, it gets the program counter(PC)address from the stack by popping the top bytes of the stack into the PC. Then it starts to execute from that address.	3	3	3
II. 7		3	3	3
II. 8	• The digital-to- analog converter (DAC) is a device used to convert digital pulses to analog signals. • There are two methods of creating a DAC – *binary weighted and R/2R ladder.* • Most of the integrated circuit DACs, including the MC1408(DAC0808), use R/2R method because it can achieve a much higher degree of precision. • The first criterion for judging a DAC is its resolution,	Explanation – 2 Marks	3	3

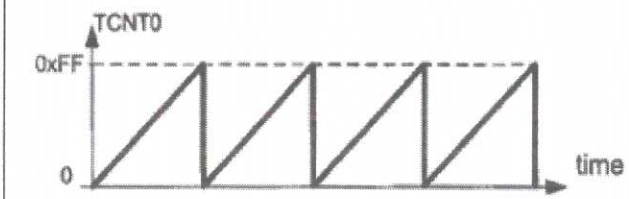
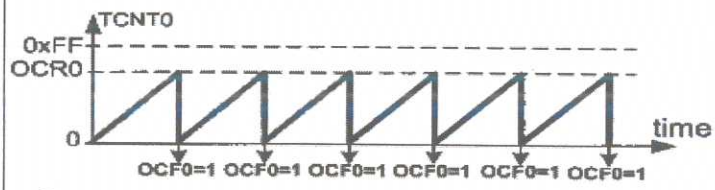
	which is a function of the number of binary inputs. • The common ones are 8, 10 , and 12 bits. • The number of data bit inputs decides the resolution of the DAC because the number of analog output levels is equal to 2^n, where n is the number of data bit inputs.  	Diagram – 1 Mark		
II. 9	A process mimics a processor in properties and holds a set of registers, process status, a Program Counter (PC) to point to the next executable instruction of the process, a stack for holding the local variables associated with the process and the code corresponding to the process.  	Explanation – 2 Marks Diagram – 1	3	3
II. 10	**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 ofCPU from executing one task to another. I	3	3	3
	PARTC			42

III	 **Program ROM** ▪ ROM is used to store program and is called program ROM or code ROM ▪ AVR has 8M of program ROM space. ▪ Program ROM size vary from 1K to 256K. ▪ AVR was one of the first microcontrollers to use on-chip Flash memory for program storage. ▪ Flash memory is ideal for fast development because flash memory can erase the data within seconds. **Data RAM and EEPROM** ▪ RAM space is for data storage. ▪ AVR has maximum of 64K of data RAM space. ▪ RAM space has three components ▪ General purpose registers – 32 general purpose registers. ▪ I/O memory ▪ Internal SRAM ▪ SRAM and I/O memory space varies from chip to chip. ▪ Internal SRAM space is used for a read/write scratch pad. ▪ EEPROM has to store critical data that need not be changed very often. **I/O Pins** ▪ AVR has 3 to 86 pins for I/O. ▪ Number of I/O depends on the number of pins in the package itself. **Peripherals** ▪ AVR comes with ▪ ADC – It is 10 bit and the number of ADC channel varies and can be upto 16 depending on the number of pins in the package. ▪ Timers – AVR can have 6 timers ▪ USART – These peripherals allows to connect	Diagram – 3 Marks Explanation – 4 Marks	7	7

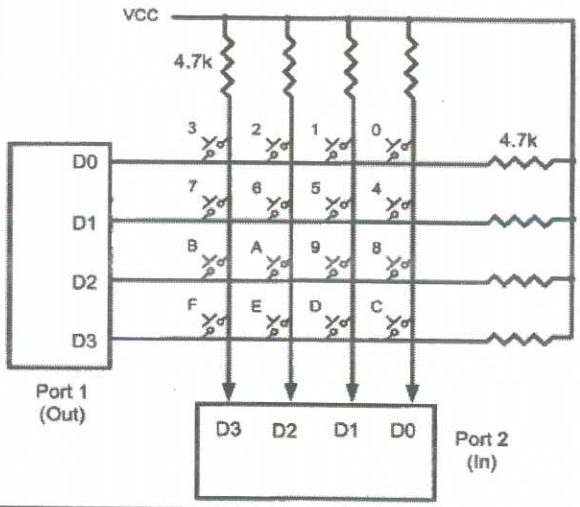
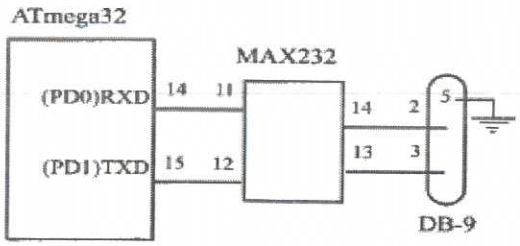
	AVR based system to serial ports.			
IV	(a) General Purpose Registers o Registers used to store data or information temporarily. • The information could be a byte of data to be processed, or an address pointing to the data to be fetched. • GPR's are used for arithmetic and logic operations. • GPR's are 8-bit registers. • The 8-bit ranges from D7 (MSB) to D0 (LSB). • Any data larger than 8 bits must be broken into 8-bit chunks before it is processed. • AVR has 32 general purpose registers and all are 8 bits. They are named as R_0 to R_{31} and located in the lowest location of memory address. GPRs are same as accumulator in microprocessors. They can be used by all arithmetic and logic instructions. (b) Program Counter ▪ It point to the address of the next instruction to be executed. ▪ The CPU fetches the opcode from the program ROM, the program counter is incremented automatically to point to the next instruction. ▪ Wider the PC, CPU can access more memory locations. ▪ i.e., a 14-bit program counter can access a maximum of 2^{14} = 16K program memory locations. ▪ Each flash memory location in AVR is 2 bytes wide. ▪ A 16 bit PC has a code space of 64K, which occupies the address in the range of 0000 to $FFFF ▪ The PC in AVR family is up to 22 bit wide and can access 000000 to $3FFFFF addresses (a total of 4M locations) (c) Data Memory The data memory is composed of three parts: o GPRs (General Purpose Register) o I/O Memory o Internal SRAM	2 + 2 +3	7	7

	 **GPRs** • It use 32 bytes of data memory space and take the address location $00 to $1F in the data memory space. **I/O Memory (Special Function Registers)** • It is dedicated to specific functions - **status register, timers, serial communication, I/O ports, ADC** and so on. • AVRs have **64 bytes of I/O memory location called standard I/O memory.** **Internal Data SRAM** • Internal data SRAM is used for **storing data and parameters.** • Each location can be **accessed directly by its addresses.**			
V	The most powerful feature of C language is its ability to perform bit manipulation. The logic operators are - AND(&&), OR(||) and NOT (!) Bit-wise operators C also supports bit-wise operators. Those are o AND(&) o OR(|) o EX-OR(^) o Inverter(~) o Shift right(>>) o Shift left(<<)	4 + 3	7	7

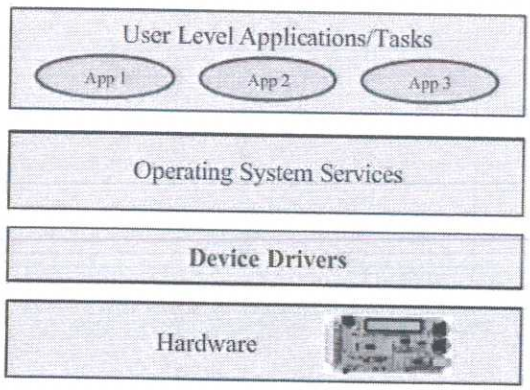
VI	#include<avr/io.h>
	int main(void)
	{
	unsigned char x, y;
	unsigned char data =0x47;
	DDRB = DDRC = 0XFF;
	x = data & 0x0F;
	PORTB = x | 0x30;
	y = data & 0xF0
	y = y>>4;
	PORTC = y | 0x30;
	return 0;
	}
		7	7	7

VII	**Normal Mode** Normal Mode The content of counter/timer increments with each clock. It counts up to a maximum of 0xFF.When it rolls over from 0xFF to 0x00, it sets a high flag called TOV0  **CTC Mode** • The OCR0 register is used with CTC mode. • In the CTC mode, the timer is incremented with a clock. • But it counts up until the content of the TCNT0 register becomes equal to the content • The OCF0 flag is located in the TIFR register.  •	3 + 4	7	7
VIII	(a) Enabling and Disabling an Interrupt Upon reset, all interrupts are disabled (masked). The interrupts must be enabled (unmasked)by software. The D7 bit (I flag) of the SREG(Status Register)register is responsible for enabling and disabling the interrupts. The I bit disable all the interrupts easily.Use the instruction CLI (Clear Interrupt) to make I = 0 during the operation of a critical task. Steps in Enabling an Interrupt To enable an interrupt: Bit D7(I) of the SREG register must be set to HIGH to allow the interrupts to happen.This is done with the "SEI" (Set Interrupt) instruction. If I= 1, each interrupt is enabled by setting to HIGH the interrupt enable (IE) flag bit for that interrupt. If I=0,no interrupt will be responded Some I/O registers holding the interrupt enable bits. Eg: TIMSK(Timer Interrupt Mask Register) contains interrupt enable bits for Timers. (b) Interrupt Priority If two interrupts are activated at the same time, the interrupt with the higher priority is served first. The priority of each interrupt is related to the address of	5+2	7	7

	that interrupt in the interrupt vector. The interrupt that has a lower address, has a higher priority. When the AVR begins to execute an ISR, it disables the I bit of the SREG register,causing all the interrupts to be disabled, and no other interrupt occurs while serving the interrupt. When the RETI instruction is executed, the AVR enables the I bit, causing the other interrupts to be served.			
IX	• Keyboards are the most widely used input device in microcontroller. • The keyboards are organized in a matrix of rows and columns. • The CPU accesses both rows and columns through ports. • Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microcontroller. • When a key is pressed, a row and a column make a contact; otherwise, there is no connection between rows and columns. <u>Scannig and identifying the key</u> • The rows are connected to an output port and the columns are connected to an input port. • If no key has been pressed, the input port will yield 1s for all columns since they are all connected to high(V_{CC}). • If all the rows are grounded and a key is pressed, one of the column 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. • To detect a key press, the microcontroller grounds all rows by providing 0 to the output latch , and then it reads the columns. • If the data read from the columns is D3-D0 = 1111, no key has been pressed and the process continues until a key press is detected. • If one of the column bits has a zero, this means that a key press has occurred. • After a key press is detected, the microcontroller will go through the process of identifying the key. • Starting with the top row, the microcontroller grounds it by providing a low to row D0 only; then it reads the columns. • If the data read is all 1s,no key in that row is activated and the process is moved to the next row. • It grounds the next row, reads the columns, and checks for any zero. This process continues until the row is identified. • After identification of the row in which the key has been pressed and find out which column the pressed key belongs	Explanation - 4 Marks Diagram – 3 Marks	7	7

	to. • After identifying the row and column, the microcontroller will identify which key has been pressed.  			
X	• RS232 is not compatible with TTL. • A line driver (voltage converter) called **MAX232** is required to convert RS232 voltage levels to TTL levels and vice versa. **RX and TX pins in the ATmega32** • The ATmega32 has two pins for transferring and receiving data serially. • These two pins are called TX and RX and are part of the Port D group (PD0 and PD1) of the 40-pin package. • Pin15 of the ATmega32 is assigned to TX and pin 14 is designated as RX. • These pins are TTL compatible, so they require a line driver MAX232 to make them RS232 compatible.  MAX232 • The MAX232 converts RS232 voltage levels to TTL voltage levels, and vice versa. • The advantage is that it uses a +5V power source, which is the same as the source voltage for the AVR. • With a single power supply, power up both the AVR and MAX232. • The MAX232 has two sets of line drivers for transferring and receiving data. • The line drivers used for TX are called T1 and T2 while		7	7

	the line drivers for RX are designated as R1 and R2. • In many applications only one of each is used.			
XI	**Task/Process management** • Deals with setting up the memory space for the tasks, loading the task's code into the memory space, allocating system resources, setting up a Task Control Block (TCB) for the task and task/process termination/deletion. • A Task Control Block (TCB) is used for holding the information corresponding to a task. **Task/Process Scheduling** • Deals with sharing the of CPU among various tasks/processes. • A kernel application called 'Scheduler' handles the task scheduling. • Scheduler is nothing but an algorithm implementation, which performs the efficient and optimal scheduling of tasks. **Task/Process Synchronization** Deals with synchronising the concurrent access of a resource, which is shared across multiple tasks and the communication between various tasks. **Error/Exception Handling** • Deals with registering and handling the errors occurred/exceptions raised during the execution of tasks. • Insufficient memory, timeouts, deadlocks, deadline missing, bus error, divide by zero, unknown instruction execution, etc. are examples of errors/exceptions. **Memory Management** • RTOS makes use of 'block' based memory allocation technique, instead of the usual dynamic memory allocation techniques used by the GPOS, RTOS kernel uses blocks of fixed size of dynamic memory and the block is allocated for a task on a need basis. • The blocks are stored in a "Free Buffer Queue'. • To achieve predictable timing and avoid the timing overheads, most of the RTOS kernels allow tasks to access any of the memory block's without any memory protection. **Interrupt Handling** • Deals with the handling of various types of interrupts. • Interrupts provide RealTime behaviour to systems. Interrupts inform the processor that an external device or an associated task requires immediate attention of the CPU. • Most of the RTOS kernel implements 'Nested Interrupts' architecture. Interrupt nesting allows the pre-emption (interruption) of an Interrupt Service Routine (ISR), servicing an interrupt, by a high priority interrupt. **Time Management**	Explain any 3	7	7

	Accurate time management is essential for providing precise time reference for all applications. The time reference to kernel is provided by a high-resolution Real-Time Clock (RTC) hardware chip (hardware timer). The hardware timer is programmed to interrupt the processor/controller at a fixed rate. This timer interrupt is referred as 'Timer tick'. The 'Timer tick' is taken as the timing reference by the kernel.			
XII	1. **CPU Utilisation**: CPU utilisation is a direct measure of how much percentage of the CPU is being utilised. 2. **Throughput:** It is the number of processes executed per unit of time. The throughput for a good scheduler should always be higher. 3. **Turnaround Time:** It is the amount of time taken by a process for completing its execution. It includes the time spent by the process for waiting for the main memory, time spent in the ready queue, time spent on completing the I/O operations, and the time spent in execution. 4. **Waiting Time:** It is the amount of time spent by a process in the 'Ready' queue waiting to get the CPU time for execution. The waiting time should be minimal for a good scheduling algorithm. 5. **Response Time:** It is the time elapsed between the submission of a process and the first response. For a good scheduling algorithm, the response time should be as least as possible	7	7	7
XIII	Device Driver 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, initialising the hardware and transferring data. All these requirements are implemented in drivers. The implementation of driver is OS dependent. A device driver implements the following: . 1. Device (Hardware) Initialisation and Interrupt configuration 2. Interrupt handling and processing 3. Client interfacing (Interfacing with user applications)	3+2+2	7	7

	(b) Task Communication In a multitasking system, multiple tasks/processes mn concurrently (in pseudo parallelism) and each process may or may not interact between. Based on the degree of interaction, the processes running on an OS are classified as ; Co-operating Processes: In the co-operating interaction model one process requires the inputs from other processes to complete its execution. Real-Time Operating System (RTOS) based Embedded System Design Competing Processes: The competing processes do not share anything among themselves but they share the system resources. The competing processes compete for the system resources such as file, display device, etc. Co-operating processes exchanges information and communicate through the following methods. Co-operation through Sharing: The co-operating process exchange data through some shared resources. Co-operation through Communication: No data is shared between the processes. But they commu¬ nicate for synchronisation. The mechanism through which processes/tasks communicate each other is known as Inter Pro¬ cess/Task Communication (IPC). Inter Process Communication is essential for process co-ordination. The various types of Inter Process Communication (IPC) mechanisms adopted by process are kernel (Operating System) dependent. (c) Task Synchronization In a multitasking environment, multiple processes run concurrently (in pseudo parallelism) and share the system resources. Imagine a situation where two processes try to access display hardware connected to the system or two processes try to access a shared memory area where one process tries to write to a memory location when the other process is trying to read from this. The act of making processes aware of the access of shared resources by each process to avoid conflicts is known as ' Task/Process Synchronisation'. Various synchronisation issues may arise in a multitasking environment if processes are not synchronised properly.			
XIV	The requirement to choose a RTOS are functional and non-functional requirements	4+3	7	7