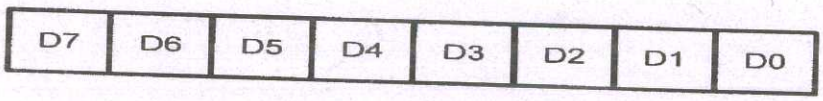
SCHEME OF EVALUATION

(Scoring Indicators)

Revision : 2015 Course Code : 6132

Course Title : MICRO CONTROLLERS

Qn. No		Split up Score	Sub total	Total
	PART- A			
I(1)	AVR,8051,Z8,PIC,68HC08/68HC11 (Any two)	1+1	2 Marks	10 Marks
I(2)	PORTx,DDRx,PINx (Any two)	1+1	2 Marks	
I(3)	Stack is a section of RAM, used by CPU to store information (data or address) temporarily. Because of limited no of registers CPU needs stack.	1+1	2 Marks	
I(4)	Three	2	2 Marks	
I(5)	MAX232	2	2 Marks	
	PART- B			
II(1)	• General purpose registers (GPR) are 8 bit and used to store information temporarily, • There are 32 general purpose registers, R0 to R31 and are located at the lowest location of memory address. • It is same as Accumulator and used by all arithmetic and logical instructions. 	3 +3 Marks	6 Marks	

	PC is used to point the address of the next instruction to be executed. As the CPU fetches the opcode from the program ROM, PC is incremented automatically . As the CPU fetches the opcode from the program ROM, PC is incremented automatically .A 14 bit PC can acess a maximum of 16K program memory locations.Whn the microprocessor is powered up PC has the value 0f 00000 in it and starts to fetch that instruction from 00000 of Program ROM Ex: LDI R16,25 LDI R17,0x34 In this case CPU places value 25 to R16. Then the pc is incremented to point to 00001 which contains instruction LDI R17,0x34 . Depending upon the instruction and its size PC takes the address and repeats the process until all are executed		
II(2)	1. meeting the computing needs of the task at hand efficiently and cost effectively 2. Availability of software and hardware development tools such as compiler, assembler, debugger and emulators. 3. wide availability and reliable sources of the microcontroller	6 marks (2 Marks for each point)	6 Marks
II(3)	1. the crystal frequency 2. The AVR Design	3+3 Marks (Expl)	6 marks
II(4)	Fig: Bitwise Logical operators in C **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 */	4+2 Marks	6Marks
II(5)	When reset, all interrupts are disabled		6

	Interrupts must be enabled by software. The D7 bit of the SREG is responsible for enabling and disabling the interrupts globally. **Steps in enabling an interrupt** To enable any one of the interrupts, we take the following steps: 1. 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. 2. If I = 1, each interrupt is enabled by setting to HIGH the interrupt enable (IE) Flag bit of that interrupt.	6 Marks	Marks
II(6)	**ATmega32 ADC features** The ADC peripheral of the ATmega32 has the following characteristics: (a) It is a 10-bit ADC. (b) It has 8 analog input channels, 7 differential input channels, and 2 differential input channels with optional gain of 10x and 200x. (c) The converted output binary data is held by two special function registers called ADCL (A/D Result Low) and ADCH (A/D Result High). (d) Because the ADCH:ADCL registers give us 16 bits and the ADC data out is only 10 bits wide, 6 bits of the 16 are unused. We have the option of making either the upper 6 bits or the lower 6 bits unused. (e) We have three options for V_{ref}. V_{ref} can be connected to AVCC (Analog V_{cc}), internal 2.56 V reference, or external AREF pin. (f) The conversion time is dictated by the crystal frequency connected to the XTAL pins (Fosc) and ADPS0:2 bits.	6 Marks	6 Marks
II(7)	**Serial Communication** • A serial interface sends data sequentially, one bit at a time. • one data line, so the communication is resource efficient. • On the negative side, throughput, the number of data bits that can be sent by second, is low. **Parallel Communication** • It uses several data lines to transfer more than one bit at a time. • The number of bits that are transferred in parallel wires.	3+3 (Marks)	6 Marks

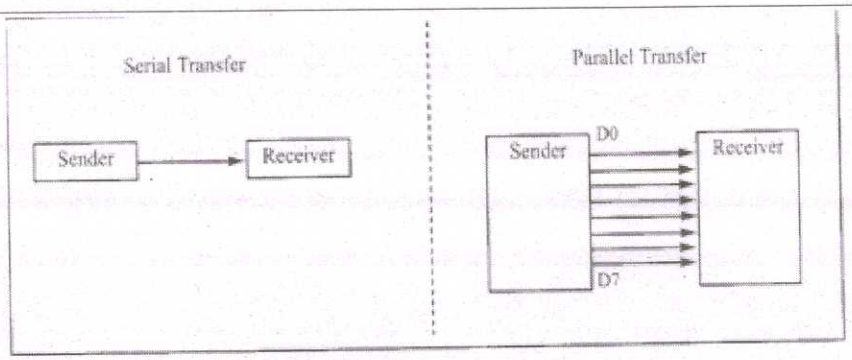
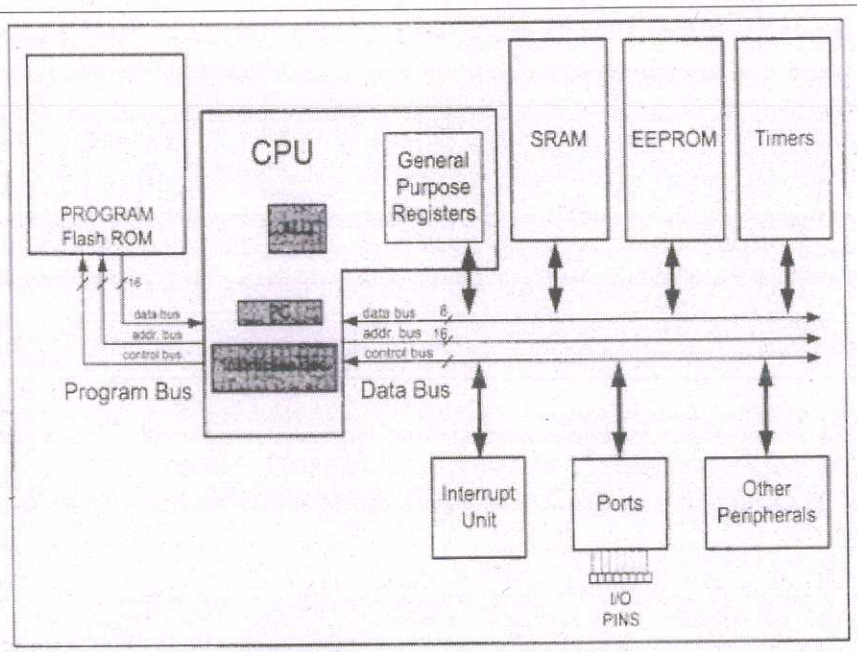


Figure Serial versus Parallel Data Transfer

42 Mrks

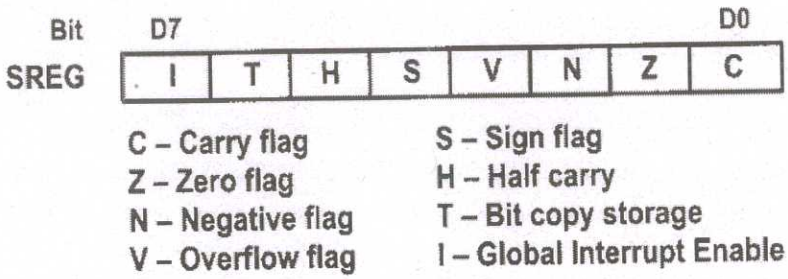
PART- C

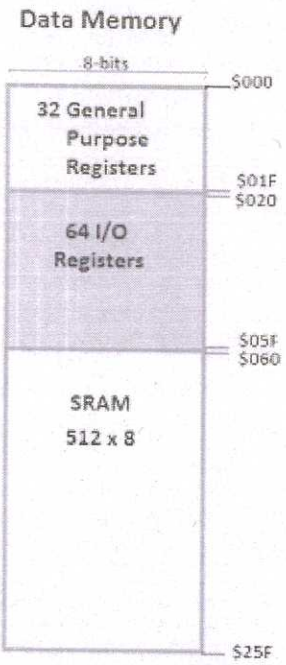
III(a)

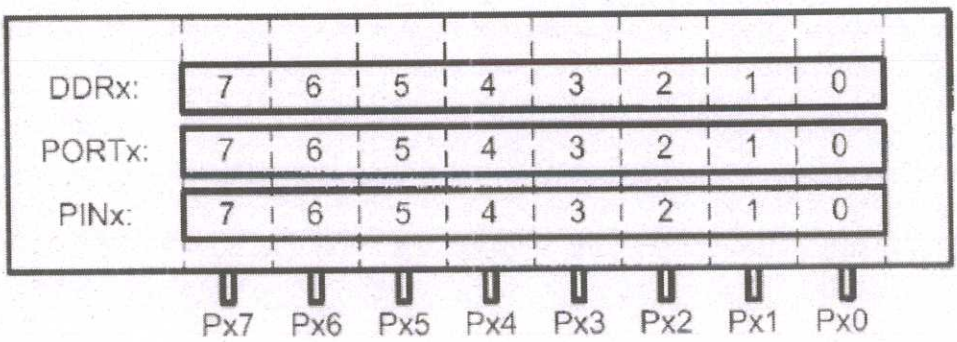


Harvard Architecture in the AVR

6 marks

III(b)	Status Register (SREG) referred as flag register and size is 8 bit. It indicates the current state of the processor. All the bits of flag can be read or written by the prograrn. The bits C, Z, N, V, S and H are conditional flags. It perform conditional branch (jump).  C - The flag is set (C=1) whenever there is a carry out from D7 bit otherwise C=0. It affected after 8 bit addition/subtraction Z - The flag shows the result of arithmetic and logical operation If the result is zero then Z=1 other wise Z=0 N- If D7 bit of result is 0, the number is positive and N=0 , If D7 bit of result is 1, the number is negative and N=1 V-overflow flag used find errors in signed arithmetic operation.This flag is set whenever the result of a signed number operation is too large,causing high order bit to overflow S- sign bit is the result of Exclusive oring of N and V flags H- Half carry flag-is set when there is a carry from D3 to D4 during an add or sub instruction T- Used with BLD (bit load) and BST (bit store) instruction for loading and storing bits from one register to another. I-Setting this Bit enables all the interrupts, resetting this bit disables all interrupts.	8+1 (fig) Marks	15 Marks	120 Marks
IV(a)	• Powerful design • Fixed instruction set size • Large no of registers (32 nos) • Small size instructions • 95 % instruction are executive in one clock cycle	*7 marks*		

	• Separate bus for data and code. • Instructions are implemented using Hardwire method • It use Load/store architecture			
1V (b)	AVR data memory has 3 parts. GPRs(General purpose Registers)- It use 32 byte of data memory space, address location 00-FFin data memory space I/O Memory(SFR)-It is Special function Register dedicated for Status register, timers, serial communication, I/O ports, ADC etc.64 bytes of !/O memory locations are available. Internal Data SRAM-It is used to store data and parameters by AVR programmers and C compilers, Every location can access individually by address and used like scratch pad. 	8 Marks	15 marks	

V(a)	• I/O Ports provide pins to take in/output information form/to the outside world. • The 40 pin AVR has 4 ports of 8 bits , PORTA,PORTB,PORTC,PORTD. • These ports can be used as input and output. • It must be programmed and access bit by bit. Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write.  **DDRx - Port X Data Direction Register** DDRx is an 8-bit register which stores configuration information for the pins of Portx. Writing a **1** in the pin location in the DDRx makes the physical pin of that port an output pin and writing a **0** makes that pin an input pin. **PINx - Port X Input Pins Register** PINx is an 8-bit register that stores the logic value, the current state, of the physical pins on Portx. So to read the values on the pins of Portx, you read the values that are in its PIN register. **PORTx - Port X Data Register** PORTx is an 8-bit register which stores the logic values that currently being outputted on the physical pins of Portx if the pins are configured as output pins. So to write values to a port, you write the values to the PORT register of that port.	8 Marks	15 Marks	
V(b)	#include <avr/io.h> int main(void) { DDRB = 0XFF;	7 marks		

	while(1)
	{
	PORTB = PORTB | 0b00010000;
	PORTB = PORTB & 0b11101111;
	}
	return 0;
	}		

VI(a)	Powerful feature of AVR I/O ports is to access 1 or 2 bits instead of entire 8 bits. Single bit assembly instructions are used to access the individual bits of the port without altering the rest of bits in that port. Example of Single bit Instructions for AVR. **Instruction** **Function** SBI A,b Set Bit b in I/O register CBI A,b Clear Bit b in I/O register SBIC A,b Skip next instruction if Bit b in I/O register is Cleared SBIS A,b Skip next instruction if Bit b in I/O register is Set A is any I/O Register	6 marks	15 marks

VI(b)	#include <avr/io.h>
	int main(void)
	{
	unsigned char temp;
	DDRD =0x00;
	DDRB =0xFF;
	while(1)
	{
	temp=PIND;
	PORTB=temp;
	}
	return 0;
	}
		9 Mark	

VII(a)

Steps in executing an interrupt

Upon activation of an interrupt, the microcontroller goes through the following steps:

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 to a fixed location in 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.

7 marks

VII(b)

Steps to program Timer0 in Normal mode

To generate a time delay using Timer0 in Normal mode, the following steps are taken:

1. Load the TCNT0 register with the initial count value.
2. Load the value into the TCCR0 register, indicating which mode (8-bit or 16-bit) is to be used and the prescaler option. When you select the clock source, the timer/counter starts to count, and each tick causes the content of the timer/counter to increment by 1.
3. Keep monitoring the timer overflow flag (TOV0) to see if it is raised. Get out of the loop when TOV0 becomes high.
4. Stop the timer by disconnecting the clock source, using the following instructions:

```
LDI   R20,0x00
OUT   TCCR0,R20   ;timer stopped, mode=Normal
```

5. Clear the TOV0 flag for the next round.
6. Go back to Step 1 to load TCNT0 again.

8 marks

15 marks

VIII(a)

Basic Registers of Timers

Timer Registers are located in I/O Register memory

1.Timer/Counter Register (TCNTn)-It is a counter, and count with each

pulse.We can read or write value to this register

2. Timer Overflow flag (TOVn)- When timer overflows this flag will be set.

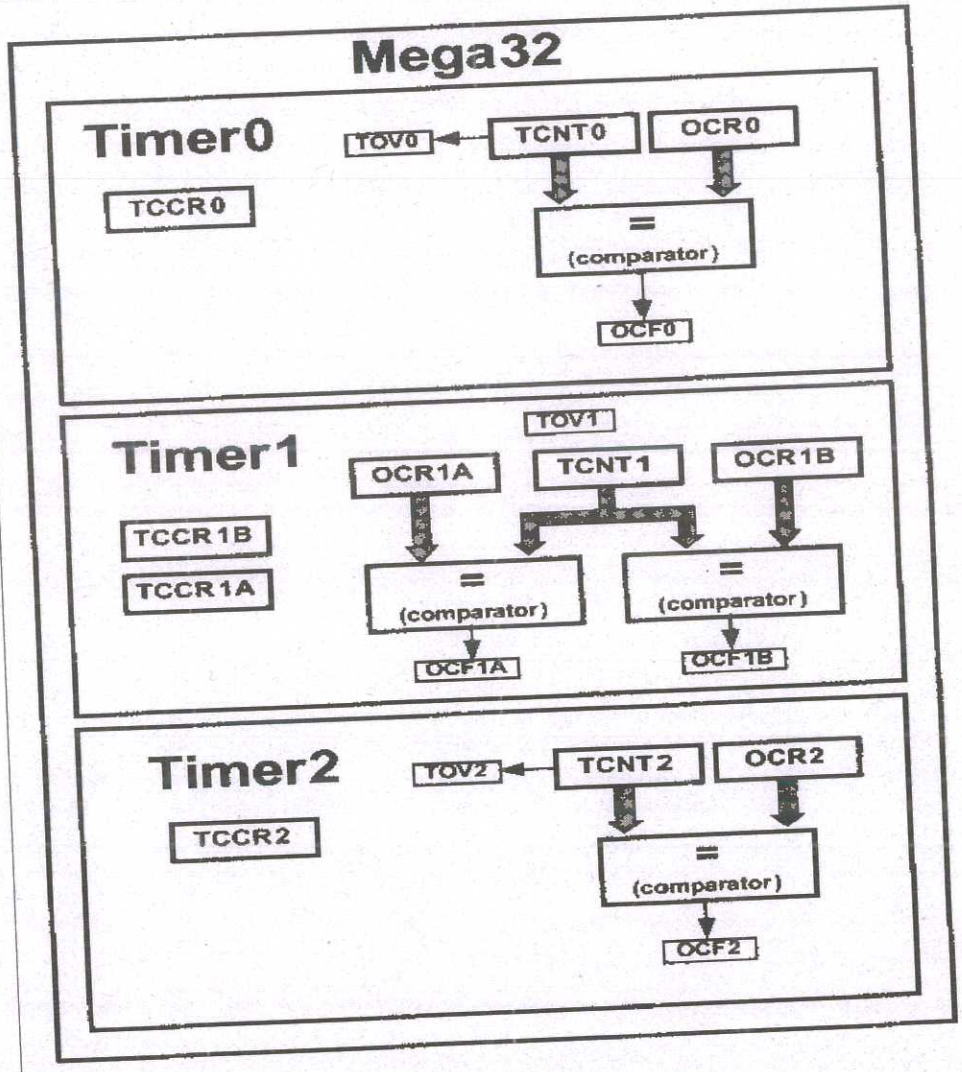
3.Timer/counter Control Register (TCCRn)-This register is for setting the modes of operations .

4.Output Compare Register (OCRn)- The content is compared with the content of the TCNTn. When they are equal the OCF n flag will be set.

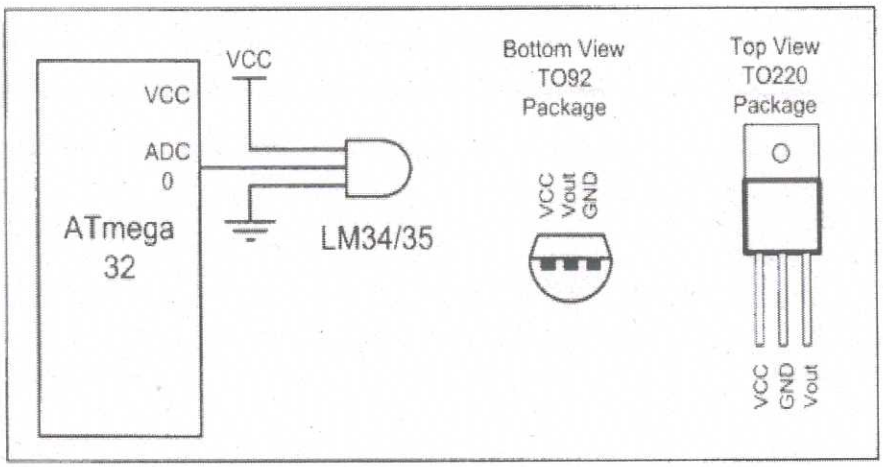
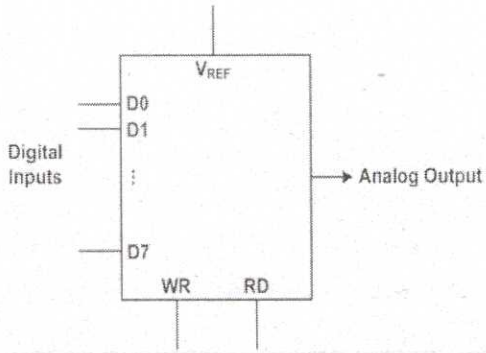
5. Output Compare Flag(OCFn)-It compare the content of OCRn and TCNTn register

9 marks

15 marks



VIII(b)	**Sources of interrupts in AVR** 1. Timer Interrupt 1) overflow 2)compare match 2. External Hardware Interrupt- Pins PD2, PD3,PB2 are external hard ware interrupts INT0,INT1 and INT2 3. Serial Communication USART has three interrupts -one for receive and two for transmit , 4. The SPI Interrupt	6 Marks (Expln)		
IX(a)	**Pin Description** Vcc - +5 V power supply , Vss- Ground , VEE- power supply to control contrast Rs- Register select, If RS=0, the instruction command code register is selected, allowing user to send commands. If Rs=1, data register is selected to send data to be displayed on the LCD R/W- Read/write, If R/w=1 for reading and R/W =0 for writing E- Enable, to latch information presented to its data pins a high to low pulse must be applied to pin E D0-D7- 8 bit data pins, to send information to the LCD or read the contents of the LCD's internal registers **Advantages of LCD** 1. The declining prices of LCDs. 2. The ability to display numbers, characters, and graphics. This is in contrast to LEDs, which are limited to numbers and a few characters. 3. Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU (or in some other way) to keep displaying the data. 4. Ease of programming for characters and graphics.	4+4 Marks		
IX b.	• LM35 is a temperature sensor which can measure temperature in the range of -55°C to 150°C. • It is a 3-terminal device that provides analog voltage proportional to the temperature. Higher the temperature, higher is the output voltage. • The output analog voltage can be converted to digital form using ADC so that a microcontroller can process it.	3+4	15 Marks	

	Figure shows the pin configuration of the LM34/LM35 temperature sensor and the connection of the temperature sensor to the ATmega32.  **Figure LM34/35 Connection to AVR and Its Pin Configuration**	Marks		
X a.	• Digital to Analog converter converts digital pulses to analog signals. • The number of data input decides the resolution of DAC. • 8-bit DAC called DAC 0808 uses weighted register method to convert digital data into the analog form and is efficient. • Analog output maximum is equal to 2^n, where n = no of input bits. So DAC 0808 has discrete voltage or current values. DAC Block Diagram  	4Marks (expl) 4 mark (fig)		

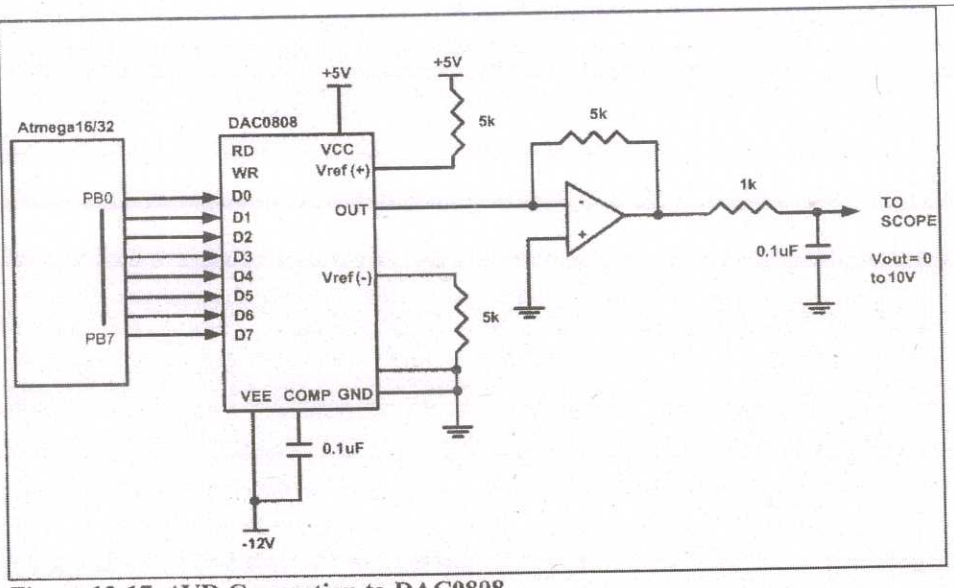


Figure 13-17. AVR Connection to DAC0808

15 Marks

X. b

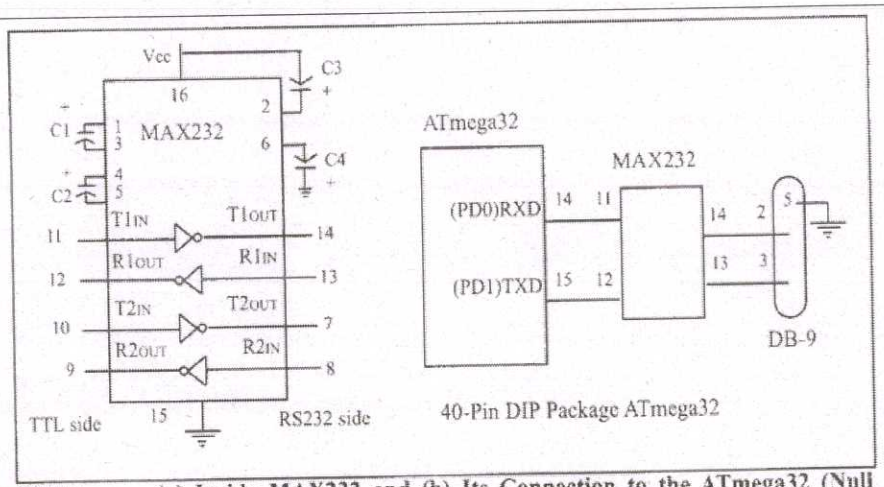


Figure (a) Inside MAX232 and (b) Its Connection to the ATmega32 (Null Modem)

(7 Marks)