

# Subject Name Solutions

4343204 – Summer 2025

Semester 1 Study Material

*Detailed Solutions and Explanations*

## Question 1(a) [3 marks]

Discuss characteristics of real time operating system.

### Solution

Table 1: RTOS Characteristics

Characteristic	Description
<b>Deterministic</b>	Predictable response times
<b>Time Constraints</b>	Hard and soft deadlines
<b>Priority Scheduling</b>	Task execution by priority
<b>Resource Management</b>	Efficient memory and CPU usage

- **Deterministic behavior:** System responds within guaranteed time limits
- **Multitasking support:** Multiple tasks execute concurrently with priority
- **Interrupt handling:** Fast response to external events

### Mnemonic

“RTOS Delivers Tasks Properly”

## Question 1(b) [4 marks]

Describe AVR I/O port registers.

### Solution

Table 2: AVR I/O Port Registers

Register	Function	Access
<b>DDRx</b>	Data Direction Register	Read/Write
<b>PORTx</b>	Port Output Register	Read/Write
<b>PINx</b>	Port Input Register	Read Only

- **DDRx register:** Controls pin direction (0=input, 1=output)
- **PORTx register:** Sets output values or enables pull-up resistors
- **PINx register:** Reads current pin states for input operations

### Mnemonic

“Direction, Port, Pin - DPP”

## Question 1(c) [7 marks]

Compare different AVR microcontrollers and What are the factors to be considered in selecting the microcontroller for embedded system?

## Solution

Table 3: AVR Microcontroller Comparison

Feature	ATmega8	ATmega32	ATmega128
<b>Flash Memory</b>	8KB	32KB	128KB
<b>SRAM</b>	1KB	2KB	4KB
<b>EEPROM</b>	512B	1KB	4KB
<b>I/O Pins</b>	23	32	53
<b>Timers</b>	3	3	4

### Selection Factors:

- **Processing speed:** Clock frequency requirements for application
- **Memory requirements:** Program and data storage needs
- **I/O requirements:** Number of pins needed for interfacing
- **Power consumption:** Battery life considerations for portable devices
- **Cost factor:** Budget constraints and volume requirements
- **Development tools:** Availability of compilers and debuggers

## Mnemonic

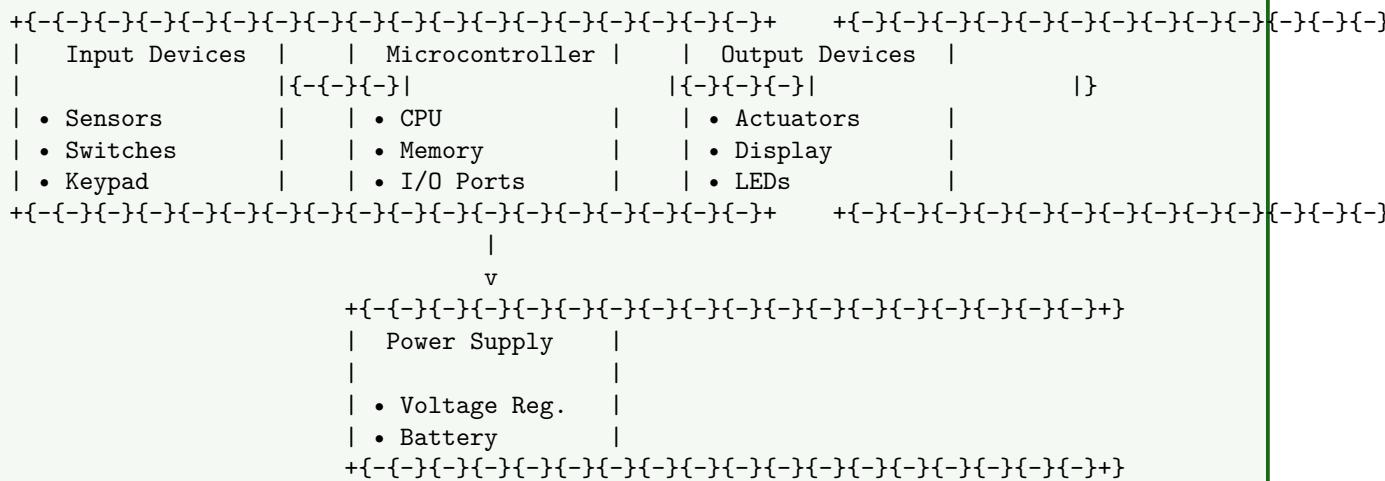
“Speed, Memory, I/O, Power, Cost, Tools - SMIPCT”

## Question 1(c OR) [7 marks]

Draw and explain general block diagram of embedded system.

## Solution

### Diagram:



### Components:

- **Input section:** Sensors and switches provide data to system
- **Processing unit:** Microcontroller executes program and controls operations
- **Output section:** Displays results and controls external devices
- **Power supply:** Provides regulated power to all components
- **Memory:** Stores program code and data permanently
- **Communication:** Interfaces with external systems via serial/wireless

## Mnemonic

“Input, Process, Output, Power, Memory, Communication - IPOPMC”

## Question 2(a) [3 marks]

Compare SRAM with EEPROM of ATMega32.

### Solution

Table 4: SRAM vs EEPROM Comparison

Parameter	SRAM	EEPROM
<b>Size</b>	2KB	1KB
<b>Volatility</b>	Volatile	Non-volatile
<b>Access Speed</b>	Fast	Slow
<b>Write Cycles</b>	Unlimited	100,000 cycles

- **Data retention:** SRAM loses data on power-off, EEPROM retains data
- **Usage purpose:** SRAM for variables, EEPROM for configuration data

### Mnemonic

“SRAM is Fast but Forgets, EEPROM Endures”

## Question 2(b) [4 marks]

List Timer/counter 0 operation mode and explain anyone.

### Solution

Table 5: Timer0 Operation Modes

Mode	Name	Description
<b>0</b>	Normal	Count up to 0xFF, overflow
<b>1</b>	PWM Phase Correct	PWM with phase correction
<b>2</b>	CTC	Clear Timer on Compare
<b>3</b>	Fast PWM	High frequency PWM

### Normal Mode Explanation:

- **Counter operation:** Counts from 0x00 to 0xFF continuously
- **Overflow flag:** TOV0 flag set when counter overflows to 0x00
- **Interrupt generation:** Can generate interrupt on overflow condition

### Mnemonic

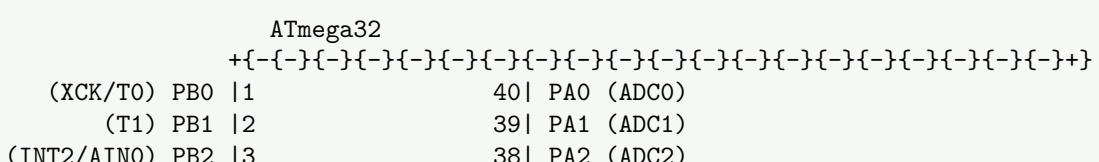
“Normal Counts, PWM Pulses, CTC Clears”

## Question 2(c) [7 marks]

With a sketch, identify and write function of each pins of ATmega32.

### Solution

#### Diagram: ATmega32 Pin Configuration



(OC0/AIN1)	PB3	4	37   PA3 (ADC3)
(SS)	PB4	5	36   PA4 (ADC4)
(MOSI)	PB5	6	35   PA5 (ADC5)
(MISO)	PB6	7	34   PA6 (ADC6)
(SCK)	PB7	8	33   PA7 (ADC7)
	RST	9	32   AREF
	VCC	10	31   GND
	GND	11	30   AVCC
XTAL2		12	29   PC7 (TOSC2)
XTAL1		13	28   PC6 (TOSC1)
(RXD)	PDO	14	27   PC5 (TDI)
(TXD)	PD1	15	26   PC4 (TDO)
(INT0)	PD2	16	25   PC3 (TMS)
(INT1)	PD3	17	24   PC2 (TCK)
(OC1B)	PD4	18	23   PC1 (SDA)
(OC1A)	PD5	19	22   PC0 (SCL)
(ICP1)	PD6	20	21   PD7 (OC2)

+{ -{-} {-} {-} {-} {-} {-} {-} {-} {-} {-} {-} {-} {-} {-} {-} {-} {-} {-} {-} +}

#### Pin Functions:

- **Port A:** 8-bit ADC input pins (PA0-PA7)
- **Port B:** SPI communication and timer functions
- **Port C:** JTAG interface and I2C communication
- **Port D:** UART communication and external interrupts
- **Power pins:** VCC, GND, AVCC for analog supply
- **Crystal pins:** XTAL1, XTAL2 for external oscillator

#### Mnemonic

“Analog-A, Bus-B, Communication-C, Data-D”

### Question 2(a OR) [3 marks]

Explain data memory organization of ATmega32.

#### Solution

Table 6: ATmega32 Memory Organization

Memory Type	Address Range	Size
<b>Registers</b>	0x00-0x1F	32 bytes
<b>I/O Registers</b>	0x20-0x5F	64 bytes
<b>Internal SRAM</b>	0x60-0x25F	2048 bytes

- **General purpose registers:** R0-R31 for arithmetic operations
- **I/O memory space:** Control registers for peripherals
- **Internal SRAM:** Variable storage during program execution

#### Mnemonic

“Registers, I/O, SRAM - RIS”

### Question 2(b OR) [4 marks]

Draw TIFR and TCCR registers of timer/counter 0.

## Solution

## Diagram: Timer0 Registers

TIFR (Timer Interrupt Flag Register)  
 $+ \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\}$   
 $| \{ - \} | \{ - \} | \{ - \} | \{ - \} | \{ - \} | \{ - \} | \{ - \} | \{ - \} | \{ - \} | \{ - \} | \{ - \} | \{ - \} | \{ - \}$   
 $+ \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\} + \{-\} \{-\}$   
 7    6    5    4    3    2    1    0

## Bit Functions:

- **TOV0:** Timer0 overflow flag bit
  - **OCF0:** Timer0 output compare match flag
  - **CS02:CS00:** Clock select bits for prescaler
  - **WGM01:WGM00:** Waveform generation mode bits

## Mnemonic

“TIFR shows Flags, TCCR Controls Clock”

**Question 2(c OR) [7 marks]**

Draw and explain general block diagram of AVR microcontroller.

## Solution

## Diagram: AVR Architecture



### Components:

- **CPU core:** Executes instructions and controls system operation
  - **Program memory:** Stores application code in non-volatile flash
  - **Data memory:** Temporary storage for variables and stack
  - **ALU:** Performs arithmetic and logical operations

- **Register file:** 32 general-purpose working registers
- **I/O system:** Interfaces with external hardware components
- **Peripherals:** Built-in modules like timers, UART, ADC

### Mnemonic

“CPU Controls Program, Data, I/O, Peripherals - CPDIP”

### Question 3(a) [3 marks]

Write an AVR C program to toggle all the bits of Port B continuously with a 10 ms delay.

#### Solution

```
\#include <avr/io.h>
\#include <util/delay.h>

int main()
\{
    DDRB = 0xFF;           // Set Port B as output

    while(1)
    \{
        PORTB = 0xFF;      // Set all bits high
        \_delay\_\_ms(10);  // 10ms delay
        PORTB = 0x00;       // Set all bits low
        \_delay\_\_ms(10);  // 10ms delay
    \}
\}
```

#### Key Points:

- **DDRB = 0xFF:** Configures all Port B pins as outputs
- **PORTB toggle:** Alternates between 0xFF and 0x00

### Mnemonic

“DDR Direction, PORT Output”

### Question 3(b) [4 marks]

Explain function of MAX232.

#### Solution

Table 7: MAX232 Functions

Function	Description
<b>Level Conversion</b>	TTL to RS232 voltage levels
<b>Charge Pump</b>	Generates $\pm 10V$ from $+5V$ supply
<b>Line Drivers</b>	Two transmit drivers
<b>Line Receivers</b>	Two receive receivers

- **Voltage conversion:** Converts 0-5V TTL to  $\pm 12V$  RS232 levels
- **Serial communication:** Enables microcontroller to communicate with PC
- **Dual channel:** Supports two-way communication simultaneously

## Mnemonic

“MAX232 Makes Microcontroller Meet PC”

### Question 3(c) [7 marks]

Write AVR C program to toggle all the bits of PORTC continuously with some delay. Use timer 0, mode 0 and no prescaler options to generate delay.

#### Solution

```
\#include <avr/io.h>

void timer0\_delay()
\{
    TCNT0 = 0;           // Initialize counter
    TCCR0 = 0x01;        // No prescaler, normal mode
    while(!(TIFR \& (1<1>TOV0))); // Wait for overflow
    TIFR |= (1<1>TOV0); // Clear overflow flag
    TCCR0 = 0;           // Stop timer
\}

int main()
\{
    DDRC = 0xFF;         // Port C as output

    while(1)
    \{
        PORTC = 0xFF;    // All bits high
        for(int i=0; i<100; i++)
            timer0\_delay(); // Multiple delays

        PORTC = 0x00;    // All bits low
        for(int i=0; i<100; i++)
            timer0\_delay(); // Multiple delays
    \}
\}
```

#### Key Features:

- **Timer0 normal mode:** Counts from 0 to 255 then overflows
- **No prescaler:** Timer runs at system clock speed
- **Overflow detection:** TOV0 flag indicates timer overflow
- **Delay generation:** Multiple timer cycles create visible delay

## Mnemonic

“Timer Counts, Overflow Flags, Generate Delays”

### Question 3(a OR) [3 marks]

Write AVR C program to store #30h into location 0X011F of EEPROM.

#### Solution

```
\#include <avr/io.h>
\#include <avr/eeprom.h>

int main()
```

```
\{
    eeprom\_write\_byte((uint8\_t*)0x011F, 0x30);
    return 0;
\}
```

#### Alternative Method:

```
\#include <avr/io.h>

int main()
\{
    while(EECR \& (1<0>EEWE));      // Wait for previous write
    EEAR = 0x011F;                  // Set address
    EEDR = 0x30;                   // Set data
    EECR |= (1<0>EEMWE);         // Master write enable
    EECR |= (1<0>EEWE);          // Write enable
\}
```

#### Mnemonic

“Address, Data, Master, Write - ADMW”

### Question 3(b OR) [4 marks]

Discuss different data types for programming AVR in C.

#### Solution

Table 8: AVR C Data Types

Data Type	Size	Range
<b>char</b>	1 byte	-128 to 127
<b>unsigned char</b>	1 byte	0 to 255
<b>int</b>	2 bytes	-32768 to 32767
<b>unsigned int</b>	2 bytes	0 to 65535
<b>long</b>	4 bytes	$-2^{31} \text{ to } 2^{31} - 1$
<b>float</b>	4 bytes	IEEE 754 format

- **Memory efficiency:** Choose smallest suitable data type
- **Unsigned types:** Use when negative values not needed
- **Integer arithmetic:** Faster than floating-point operations

#### Mnemonic

“Choose Correct Size for Memory Efficiency”

### Question 3(c OR) [7 marks]

Write AVR C programs for serial data transmission.

#### Solution

```
\#include <avr/io.h>

void uart\_init(unsigned int baud)
\{
```

```

UBRRH = (unsigned char)(baud{}8);
UBRRL = (unsigned char)baud;
UCSRB = (1{}TXEN);           // Enable transmitter
UCSRC = (1{}URSEL)|(3{}UCSZ0); // 8{-bit data}
\}

void uart\_transmit(unsigned char data)
\{
    while(!(UCSRA \& (1{}UDRE))); // Wait for empty buffer
    UDR = data;                 // Send data
\}

void uart\_send\_string(char *str)
\{
    while(*str)
    \{
        uart\_transmit(*str++);
    \}
\}

int main()
\{
    uart\_init(51);             // 9600 baud at 8MHz

    while(1)
    \{
        uart\_send\_string("Hello World\r\n");
        for(long i=0; i<100000; i++); // Delay
    \}
\}

```

#### Key Components:

- **Baud rate setting:** UBRR registers set communication speed
- **Transmit enable:** TXEN bit enables UART transmitter
- **Data transmission:** UDR register holds data to transmit
- **Buffer check:** UDRE flag indicates transmit buffer empty

#### Mnemonic

“Init, Enable, Check, Transmit - IECT”

### Question 4(a) [3 marks]

Explain ADMUX register.

#### Solution

Table 9: ADMUX Register Bits

Bit	Name	Function
<b>REFS1:0</b>	Reference Select	Voltage reference selection
<b>ADLAR</b>	Left Adjust	Result left adjustment
<b>MUX4:0</b>	Channel Select	ADC input channel selection

- **Reference voltage:** Selects internal/external voltage reference
- **Result format:** ADLAR bit adjusts 10-bit result alignment
- **Channel selection:** MUX bits choose which ADC pin to read

## Mnemonic

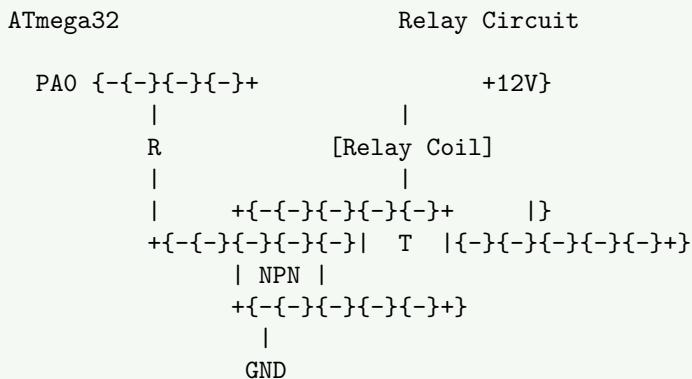
“Reference, Adjust, Channel - RAC”

### Question 4(b) [4 marks]

**Draw and explain Interfacing Relay with ATmega32.**

## Solution

## Diagram: Relay Interfacing



T = BC547 Transistor

R = 1K Resistor

## Components:

- **Transistor switch:** BC547 NPN transistor acts as electronic switch
  - **Base resistor:**  $1\text{K}\Omega$  limits base current from microcontroller
  - **Relay coil:** 12V relay operates external high-power devices
  - **Protection diode:** Freewheeling diode protects from back EMF

## Mnemonic

## “Micro Controls Transistor Controls Relay”

### Question 4(c) [7 marks]

**Draw and explain TWI registers in AVR.**

## Solution

## Diagram: TWI Register Structure

## TWCR (TWI Control Register)

+{--}{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}{-}+  
| TWINT | TWFA | TWSTA | TWSTO | TWWC | TWEN | {-} | TWIE | }

+{-{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}

TUSB (TUI Status Bar)

IWSR (TWI Status Register)

| TWS7 | TWS6 | TWS5 | TWS4 | TWS3 | {- | TWPS1 | TWPS0 | }  
+ {- {-} {-} {-} {-} + {-} {-} {-} {-} + {-} {-} {-} {-} + {-} {-} {-} {-} + {-} {-}

7        6        5        4        3        2        1        0

## TWDR (TWI Data Register)

## Register Functions:

- **TWCR:** Controls TWI operation and interrupt handling
  - **TWSR:** Provides status information and prescaler setting
  - **TWDR:** Holds data for transmission/reception
  - **TWAR:** Sets slave address when operating as slave
  - **TWBR:** Sets bit rate for TWI communication
  - **TWINT:** Interrupt flag cleared by writing 1
  - **Start/Stop:** TWSTA and TWSTO control I2C conditions

## Mnemonic

## “Control, Status, Data, Address, Bit Rate - CSDAB”

**Question 4(a OR) [3 marks]**

**Explain ADCSRA register.**

### Solution

Table 10: ADCSRA Register Bits

Bit	Name	Function
<b>ADEN</b>	ADC Enable	Enables ADC module
<b>ADSC</b>	Start Conversion	Starts ADC conversion
<b>ADATE</b>	Auto Trigger	Enables auto trigger mode
<b>ADIF</b>	Interrupt Flag	ADC conversion complete flag
<b>ADIE</b>	Interrupt Enable	Enables ADC interrupt
<b>ADPS2:0</b>	Prescaler	Sets ADC clock prescaler

- **ADC control:** ADEN enables ADC, ADSC starts conversion
  - **Interrupt system:** ADIF flag set when conversion complete

## Mnemonic

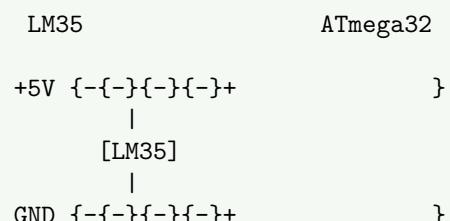
## “Enable, Start, Trigger, Interrupt, Prescale - ESTIP”

**Question 4(b OR) [4 marks]**

Draw and explain interfacing of LM35 with ATmega32.

### Solution

## Diagram: LM35 Interfacing



Vout { -{-} {-} + {-} {-} {-} {-} {-} {-} {-} PA0 (ADC0)}

Temperature Sensor  
Output: 10mV/^

#### Connection Details:

- **Power supply:** LM35 requires +5V and ground connections
- **Output voltage:** Produces 10mV per degree Celsius
- **ADC input:** Connect LM35 output to ADC channel (PA0)
- **Temperature calculation:**  $= (ADC\_Value \times 5000mV) / (1024 \times 10mV)$

#### Code Example:

```
float temp = (adc\_read() * 5.0 * 100.0) / 1024.0;
```

#### Mnemonic

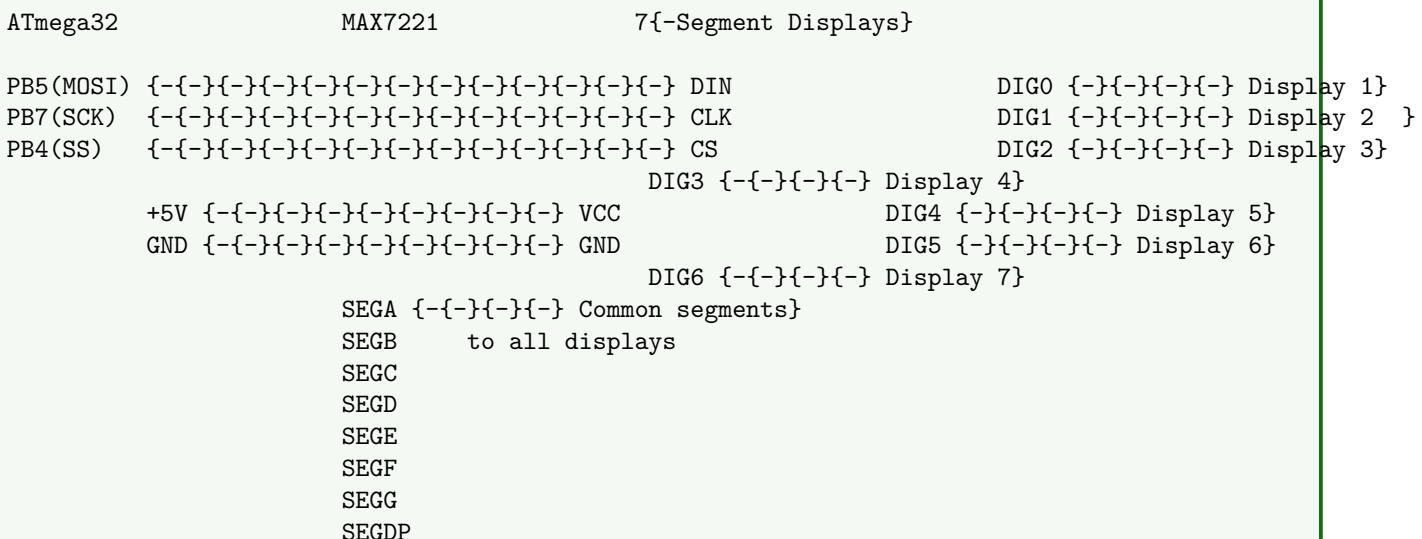
“LM35 gives 10mV per degree”

## Question 4(c OR) [7 marks]

Draw and explain interfacing of multiple 7-segment displays using MAX7221 with ATmega32.

#### Solution

##### Diagram: MAX7221 Interfacing



#### Features:

- **SPI communication:** Uses serial peripheral interface for control
- **Multiple displays:** Controls up to 8 seven-segment displays
- **Automatic scanning:** MAX7221 handles multiplexing automatically
- **Brightness control:** Software-controlled brightness levels
- **Decode mode:** Built-in BCD to 7-segment decoder
- **Low component count:** Reduces external components needed

#### Key Registers:

- **Decode mode register:** Enables/disables BCD decoding
- **Intensity register:** Controls display brightness
- **Scan limit register:** Sets number of active displays
- **Shutdown register:** Normal operation or shutdown mode

## Mnemonic

## “SPI Sends Serial Data to Multiple Displays”

### Question 5(a) [3 marks]

**Explain SPCR register.**

## Solution

Table 11: SPCR Register Bits

Bit	Name	Function
<b>SPIE</b>	Interrupt Enable	Enables SPI interrupt
<b>SPE</b>	SPI Enable	Enables SPI module
<b>DORD</b>	Data Order	LSB/MSB first selection
<b>MSTR</b>	Master/Slave	Selects master or slave mode
<b>CPOL</b>	Clock Polarity	Clock idle state selection
<b>CPHA</b>	Clock Phase	Clock edge for data sampling
<b>SPR1:0</b>	Clock Rate	SPI clock rate selection

- **SPI enable:** SPE bit must be set to enable SPI functionality
  - **Master mode:** MSTR bit determines if device is master or slave

## Mnemonic

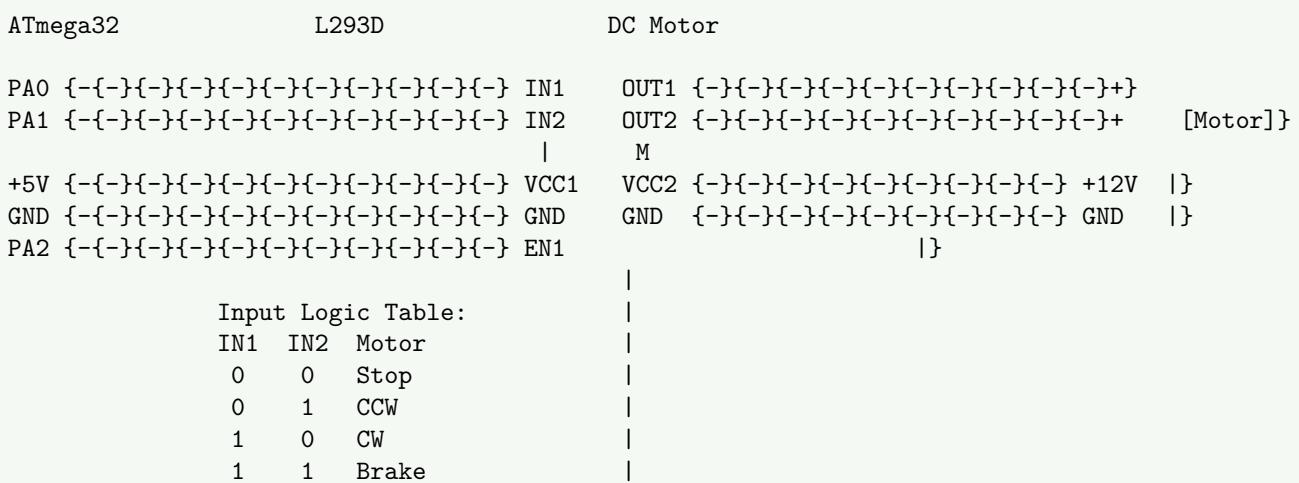
“Interrupt, Enable, Data, Master, Clock settings - IEDMC”

### Question 5(b) [4 marks]

Draw circuit diagram to interface DC motor with ATmega32 using L293D motor driver.

## Solution

## Diagram: DC Motor Interfacing



## Components:

- **L293D driver:** Provides current amplification for motor control
  - **Power supplies:** +5V for logic, +12V for motor power
  - **Control signals:** IN1, IN2 determine motor direction
  - **Enable pin:** EN1 controls motor on/off and speed (PWM)

## Mnemonic

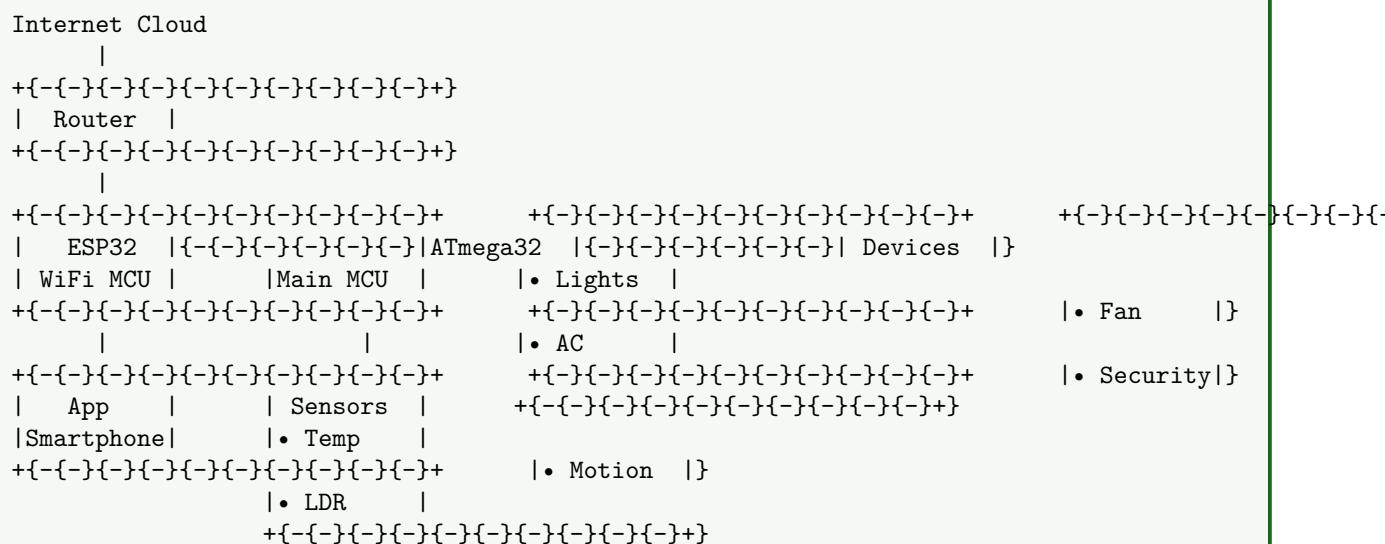
“Logic controls Direction, Enable controls Speed”

### Question 5(c) [7 marks]

**Explain IoT based Home Automation System.**

## Solution

## Diagram: IoT Home Automation System



### **System Components:**

- **Internet connectivity:** WiFi module connects system to internet
  - **Mobile application:** User interface for remote control and monitoring
  - **Sensor network:** Temperature, motion, light sensors for automation
  - **Control devices:** Relays control home appliances and lights
  - **Central controller:** Microcontroller processes commands and sensor data
  - **Cloud services:** Store data and enable remote access

#### **Features:**

- **Remote control:** Control appliances from anywhere via internet
  - **Automation:** Automatic control based on sensor readings
  - **Energy saving:** Smart scheduling reduces power consumption
  - **Security monitoring:** Motion sensors and cameras for safety
  - **Data logging:** Historical data storage for analysis

## Mnemonic

“Internet connects Phones to Home Devices - IPHD”

**Question 5(a OR) [3 marks]**

**Explain SPSR register.**

### Solution

Table 12: SPSR Register Bits

Bit	Name	Function
<b>SPIF</b>	Interrupt Flag	SPI transfer complete flag
<b>WCOL</b>	Write Collision	Data collision error flag
<b>SPI2X</b>	Double Speed	Doubles SPI clock rate

- **Transfer complete:** SPIF flag indicates SPI transmission finished
  - **Collision detection:** WCOL flag shows write collision occurred
  - **Speed control:** SPI2X doubles communication speed when set

## Mnemonic

## “Flag, Collision, Speed - FCS”

**Question 5(b OR) [4 marks]**

Draw and explain pin diagram of L293D motor driver IC.

### Solution

## Diagram: L293D Pin Configuration

### Pin Functions:

- **Enable pins (EN1, EN2):** Control motor on/off and speed via PWM
  - **Input pins (IN1-IN4):** Logic inputs from microcontroller
  - **Output pins (OUT1-OUT4):** High current outputs to motors
  - **Power supply (VCC1):** +5V logic supply for IC operation
  - **Motor supply (VCC2):** +12V supply for motor power
  - **Ground pins:** Multiple ground connections for heat dissipation

## Features:

- **Dual H-bridge:** Can control two DC motors simultaneously
  - **Current capacity:** 600mA per channel, 1.2A peak
  - **Protection:** Built-in flyback diodes for motor protection

## Mnemonic

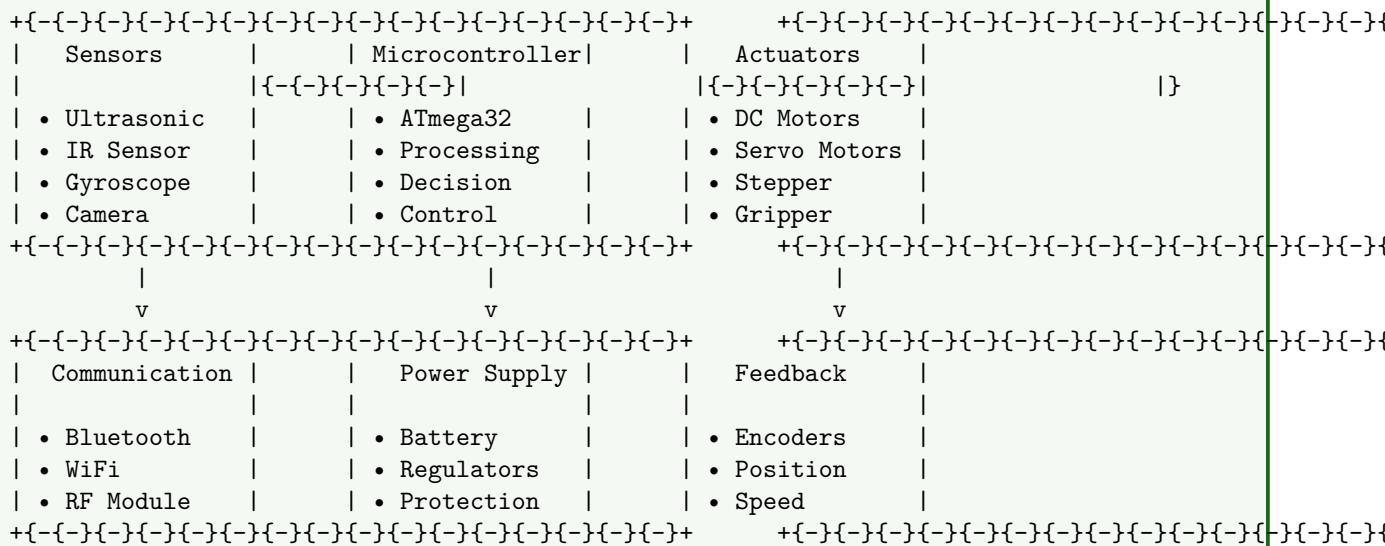
“Enable, Input, Output, Power - EIOP”

**Question 5(c OR) [7 marks]**

**Explain Motorised Control Robotics System.**

## Solution

### Diagram: Robotics Control System



### System Components:

Table 13: Robotics System Elements

Component	Function	Examples
<b>Sensors</b>	Environment sensing	Ultrasonic, IR, Camera
<b>Controller</b>	Decision making	ATmega32, Arduino
<b>Actuators</b>	Physical movement	Motors, Servos
<b>Communication</b>	Remote control	Bluetooth, WiFi
<b>Power</b>	Energy supply	Battery, Regulators
<b>Feedback</b>	Position sensing	Encoders, Gyroscope

### Control Algorithm:

- Sense:** Collect data from environment using sensors
- Process:** Analyze sensor data and make decisions
- Act:** Control motors and actuators based on decisions
- Feedback:** Monitor actual movement and adjust control
- Communicate:** Send status and receive commands remotely

### Applications:

- Autonomous navigation:** Robot moves independently using sensors
- Object manipulation:** Gripper controlled for pick and place tasks
- Remote operation:** Manual control via wireless communication
- Path following:** Line following or predetermined route navigation
- Obstacle avoidance:** Dynamic path planning around obstacles

### Programming Structure:

```

while(1) \{
    read\_sensors();
    process\_data();
    make\_decision();
    control\_motors();
    check\_feedback();
    communicate\_status();
\}
  
```

### Mnemonic

“Sense, Process, Act, Feedback, Communicate - SPACF”