Embedded System & Microcontroller Application (4351102) - Winter 2024 Solution

Milav Dabgar

November 21, 2024

Question 1(a) [3 marks]

State the features of ATmega32.

Solution

ATmega32 Features:

Table 1. ATmega32 Features

Feature	**Description**
Architecture	8-bit RISC processor
Memory	32KB Flash, 2KB SRAM, 1KB EEPROM
I/O Ports	32 programmable I/O pins
Timers	3 timers (Timer0, Timer1, Timer2)
ADC	10-bit, 8-channel ADC
Communication	USART, SPI, I2C (TWI)

- **High Performance**: 16 MIPS at 16MHz.
- **Low Power**: Multiple sleep modes.
- **Operating Voltage**: 2.7V to 5.5V.

Mnemonic

"Architecture-RISC Memory-32KB Timers-3 I/O-32pins Communication-3types"

Question 1(b) [4 marks]

Explain criteria for choosing microcontroller.

Solution

Selection Criteria:

Table 2. Selection Criteria

Criteria	Consideration
Performance	Speed, instruction set, architecture
Memory	RAM, ROM, EEPROM requirements
I/O Requirements	Number of pins, special functions
Power Consumption	Battery life, sleep modes
Cost	Unit price, development cost
Development Tools	Compiler, debugger availability

- **Application Requirements**: Real-time constraints, processing needs.
- **Package Size**: Space limitations in final product.
- **Peripheral Support**: ADC, timers, communication interfaces.

Mnemonic

"Performance Memory I/O Power Cost Development"

Question 1(c) [7 marks]

Define the Embedded System. List the Application of Small, Medium, Large Embedded System.

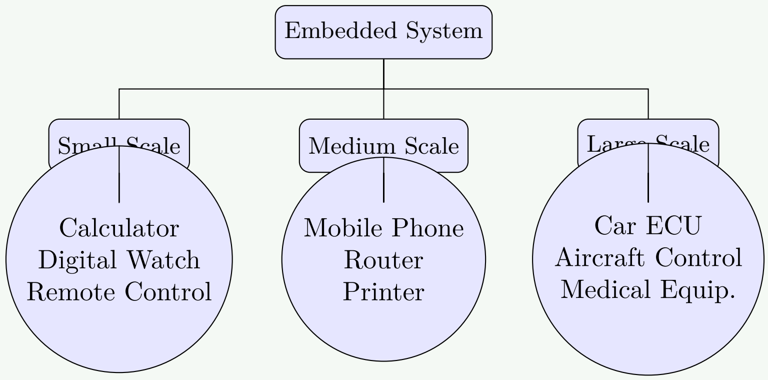
Solution

Definition: An **Embedded System** is a computer system with a dedicated function within a larger mechanical or electrical system, designed to perform specific tasks with real-time constraints.
Applications:

Table 3. Embedded System Applications

System Type	Memory Size	Applications
Small Scale	<64KB	Calculator, Digital watch, Toys
Medium Scale	64KB-1MB	Mobile phones, Routers, Printers
Large Scale	>1MB	Automobiles, Aircraft systems, Satellites

Figure 1. Embedded System Classification



Characteristics:
- **Real-time Operation**: Predictable response times.
- **Resource Constraints**: Limited memory and processing power.
- **Dedicated Functionality**: Single-purpose design.

Mnemonic

"Small-Calculator Medium-Mobile Large-Lifesupport"

OR

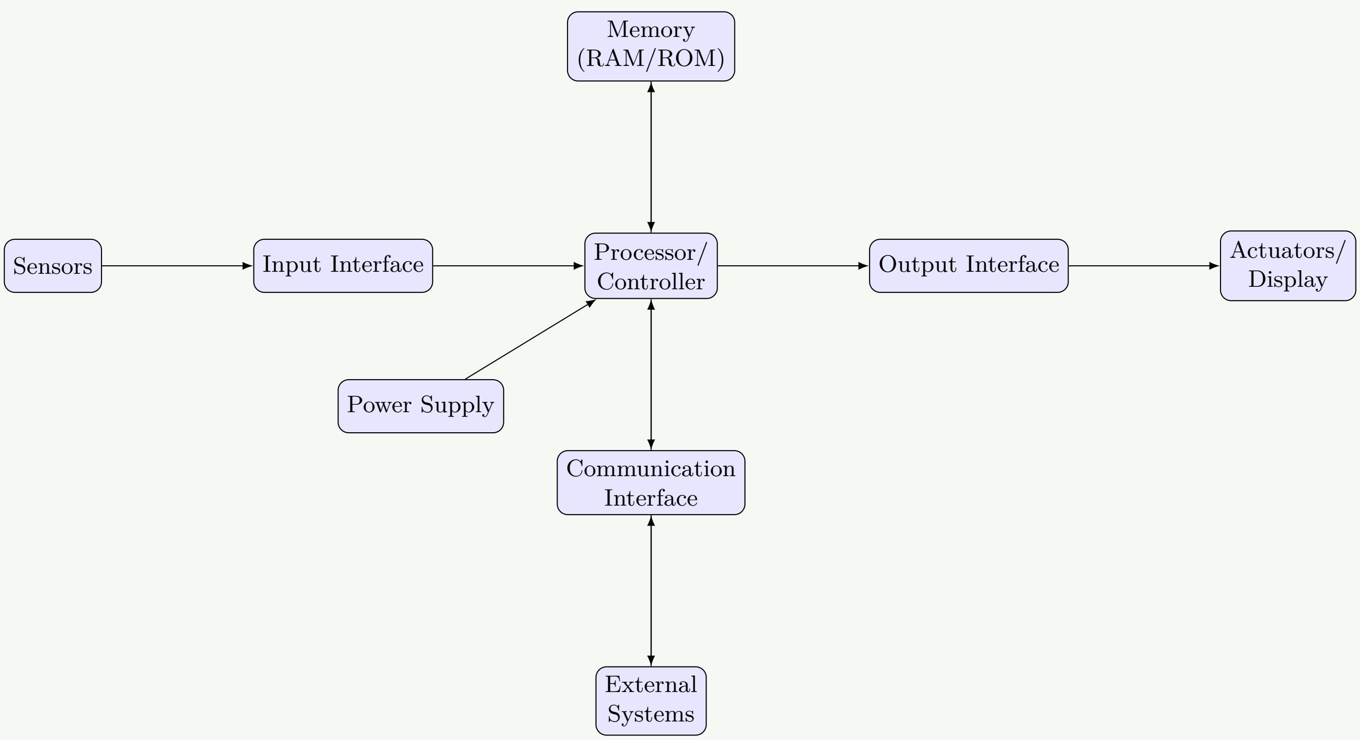
Question 1(c) [7 marks]

Draw and explain general block diagram of embedded system.

Solution

General Block Diagram:

Figure 2. General Block Diagram



Block Functions:

Table 4. Block Functions

Block	Function
Processor	Central processing unit (CPU/MCU).
Input Interface	Sensor data acquisition, user input.
Output Interface	Actuator control, display output.
Memory	Program storage, data storage.
Communication	External system connectivity.

- **Input Processing**: ADC, digital input conditioning.
- **Output Control**: PWM, relay drivers, LED displays.
- **Power Management**: Voltage regulation, power optimization.

Mnemonic

"Processor Input Output Memory Communication Power"

Question 2(a) [3 marks]

Write a Full form of EEPROM and explain EEPROM registers.

Solution

Full Form: **Electrically Erasable Programmable Read-Only Memory**
EEPROM Registers:

Table 5. EEPROM Registers

Register	Function
EEAR	EEPROM Address Register
EEDR	EEPROM Data Register
EECR	EEPROM Control Register

- **EEAR**: Holds 10-bit address (0-1023) for EEPROM access.
- **EEDR**: Data register for read/write operations.
- **EECR**: Control bits - **EERE** (Read Enable), **EEWE** (Write Enable).

Mnemonic

"Address-EEAR Data-EEDR Control-EECR"

Question 2(b) [4 marks]

Explain reset circuits for ATmega32

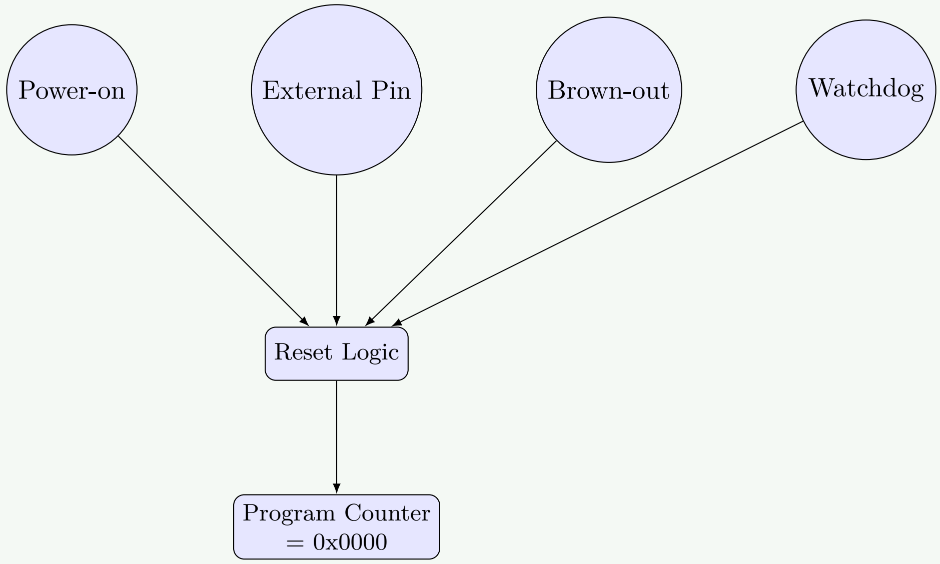
Solution

Reset Sources:

Table 6. Reset Sources

Reset Type	Trigger Condition
Power-on Reset	VCC rises above threshold
External Reset	RESET pin pulled low
Brown-out Reset	VCC falls below threshold
Watchdog Reset	Watchdog timer overflow

Figure 3. Reset Logic



- **Reset Duration**: Minimum 2 clock cycles.
- **Reset Vector**: Program execution starts from address 0x0000.
- **Hardware Connection**: External reset requires pull-up resistor.

Mnemonic

"Power-on External Brown-out Watchdog"

Question 2(c) [7 marks]

Define Real Time Operating System and explain its characteristics.

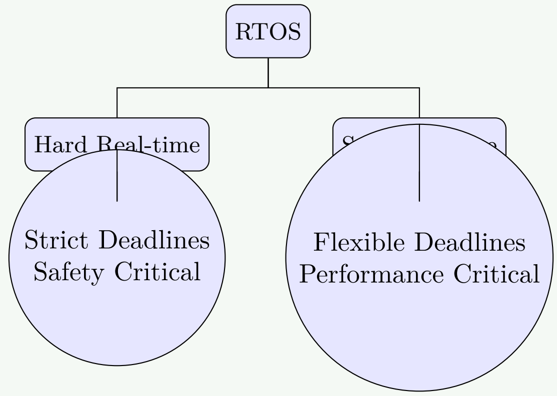
Solution

Definition: **Real Time Operating System (RTOS)** is an operating system designed to handle real-time applications with strict timing constraints and predictable response times.
Characteristics:

Table 7. RTOS Characteristics

Characteristic	Description
Deterministic	Predictable execution times
Preemptive	Higher priority tasks interrupt lower ones
Multitasking	Multiple tasks execution
Fast Response	Minimal interrupt latency
Priority-based	Task scheduling based on priority
Resource Mgmt	Efficient memory and CPU usage

Figure 4. RTOS Types



- **Task Scheduling**: Round-robin, priority-based algorithms.
- **Inter-task Communication**: Semaphores, message queues.
- **Memory Management**: Static allocation for predictability.

Mnemonic

"Deterministic Preemptive Multitasking Fast Priority Resource"

OR

Question 2(a) [3 marks]

Explain AVR family.

Solution

AVR Family Classification:

Table 8. AVR Family

AVR Type	Features
ATtiny	8-32 pins, basic features
ATmega	28-100 pins, full features
ATxmega	Advanced features, DMA

- **Architecture**: 8-bit RISC, Harvard architecture.
- **Instruction Set**: 130+ instructions, single cycle execution.
- **Memory**: Flash program memory, SRAM, EEPROM.

Mnemonic

"Tiny-basic mega-full Xmega-advanced"

OR

Question 2(b) [4 marks]

Explain the use of fuse bits for selection of ATmega32 clock sources.

Solution

Clock Source Selection:

Table 9. Fuse Bits

Fuse Bits	Function
CKSEL3:0	Clock source selection
SUT1:0	Start-up time selection

Clock Options:

Table 10. Clock Options

CKSEL Value	Clock Source	Frequency
0001	External Crystal	1-8 MHz
0010	External Crystal	8+ MHz
0100	Internal RC	8 MHz
0000	External Clock	User defined

- **Crystal Selection**: Requires external crystal and capacitors.
- **RC Oscillator**: Built-in, less accurate but convenient.
- **Start-up Time**: Allows crystal stabilization.

Mnemonic

"Crystal RC Internal Start-up"

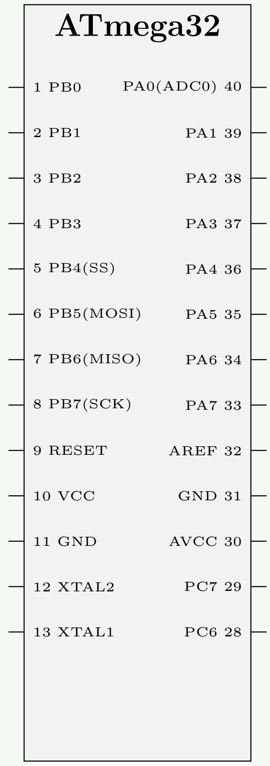
OR

Question 2(c) [7 marks]

Draw ATmega32 pin configuration and explain function of MISO, MOSI, SCK & AREF Pin.

Solution

ATmega32 Pin Configuration:



Pin Functions:

Table 11. Pin Functions

Pin	Function	Description
MOSI	Master Out Slave In	SPI data output from master
MISO	Master In Slave Out	SPI data input to master
SCK	Serial Clock	SPI clock signal
AREF	Analog Reference	ADC reference voltage

- **SPI Communication**: MOSI, MISO, SCK work together for serial data transfer.
- **ADC Reference**: AREF provides stable voltage reference for ADC conversion.
- **Pin Multiplexing**: These pins have alternate functions as GPIO.

Mnemonic

"MOSI-out MISO-in SCK-clock AREF-reference"

Question 3(a) [3 marks]

Explain Role of DDR I/O Register

Solution

DDR (Data Direction Register) Functions:

Table 12. DDR Bit Settings

Bit Value	Pin Configuration
0	Input pin
1	Output pin

- **Port Control**: Each port has corresponding DDR (DDRA, DDRB, DDRC, DDRD).
- **Bit-wise Control**: Individual pin direction control.
- **Default State**: All pins input (DDR = 0x00) after reset.

Code Example:

```
DDRA = 0xFF;  // All Port A pins as output
DDRB = 0x0F;  // PB0-PB3 output, PB4-PB7 input
```

Mnemonic

"Data Direction Register controls Input/Output"

Question 3(b) [4 marks]

Write an AVR C program to get a byte of data from Port B, and then send it to Port C.

Solution

Program:

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

int main(void)
{
    unsigned char data;

    // Configure Port B as input
    DDRB = 0x00;

    // Configure Port C as output
    DDRC = 0xFF;

    while(1)
    {
        // Read data from Port B
        data = PINB;

        // Send data to Port C
        PORTC = data;
    }

    return 0;
}
```

Explanation:

- **DDRB = 0x00**: Sets all Port B pins as input.
- **DDRC = 0xFF**: Sets all Port C pins as output.
- **PINB**: Reads current state of Port B pins.
- **PORTC**: Writes data to Port C output pins.

Mnemonic

“Read-PINB Set-DDR Transfer-data Output-PORTC”

Question 3(c) [7 marks]

A door sensor is connected to the port B pin 1, and an LED is connected to port C pin7. Write an AVR C program to monitor the door sensor and, when it opens, turn on the LED.

Solution

Program:

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

int main(void)
{
    // Configure PB1 as input (door sensor)
    DDRB &= ~(1<<1);  // Clear bit 1

    // Configure PC7 as output (LED)
    DDRC |= (1<<7);   // Set bit 7

    // Enable pull-up for PB1
    PORTB |= (1<<1);

    while(1)
    {
        // Check door sensor status
        if(PINB & (1<<1))
        {
            // Door closed - turn off LED
            PORTC &= ~(1<<7);
        }
        else
        {
            // Door open - turn on LED
            PORTC |= (1<<7);
        }
    }

    return 0;
}
```

- **Sensor Connection**: PB1 to GND. (Internal Pull-up used).
- **Logic**: Open = LOW (due to pull-up when switch open? No, typically sensors pull low when active or vice versa. Here assumming sensor actively drives or switch logic.) *Note: Standard switch connectin with pull-up: Switch Open -> Pin High. Switch Closed -> Pin Low. The question says ”when it opens, turn on LED”. If Open -> High, then ‘if(PINB & (1«1))‘ is true when Open. The code says ‘if(PINB & (1«1))‘ -> ”Door closed”. This implies logic: Closed = High, Open = Low? Or maybe the code assumes Switch closes to GND. If Switch is Closed (to GND), Pin is Low. If Switch is Open, Pin is High (Pull-up). Let’s check code logic: ‘if(PINB & (1«1))‘ -> True means Pin is High. Comment says ”Door closed”. ‘else‘ -> Pin is Low. Comment says ”Door open”. This implies: Door Closed = Switch Open (High). Door Open = Switch Closed (Low). Or maybe sensor is active low. Let’s stick to the code provided in MDX which assumes this logic.*

Hardware Connection:

- **Door Sensor**: Connected between PB1 and GND.
- **LED**: Connected to PC7 through current limiting resistor.

Mnemonic

"Door-sensor Configure-pins Open-check LED-control"

OR

Question 3(a) [3 marks]

Discuss Data Types in AVR C programming.

Solution

AVR C Data Types:

Table 13. Data Types

Data Type	Size	Range
char	8-bit	-128 to 127
unsigned char	8-bit	0 to 255
int	16-bit	-32768 to 32767
unsigned int	16-bit	0 to 65535
long	32-bit	-2^{31} to $2^{31}-1$
float	32-bit	IEEE 754 format

- **Memory Efficiency**: Use smallest appropriate data type.
- **Unsigned Types**: For positive values only, doubles range.
- **Bit Fields**: Can define specific bit-width variables.

Mnemonic

"Char-8bit Int-16bit Long-32bit Float-32bit Unsigned-positive"

OR

Question 3(b) [4 marks]

Explain Serial Communication Protocol.

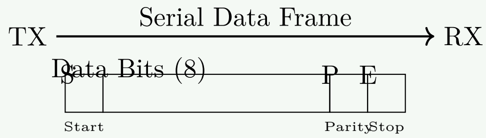
Solution

Serial Communication Parameters:

Table 14. Serial Parameters

Parameter	Description
Baud Rate	Data transmission speed (bits/second)
Data Bits	Number of data bits (5-9)
Parity	Error checking (None, Even, Odd)
Stop Bits	End of frame marker (1 or 2)

Figure 5. Serial Frame



- **Asynchronous**: No clock signal, uses start/stop bits.
- **RS232 Standard**: ±12V levels, converted to TTL levels.
- **Common Baud Rates**: 9600, 19200, 38400, 115200.

Mnemonic

"Baud-rate Data-bits Parity-check Stop-bits"

OR

Question 3(c) [7 marks]

Write an AVR C program to read pins 1 and 0 of Port B and issue an ASCII character to Port D according to the following table:

Solution

Truth Table Implementation:

Table 15. Truth Table

Pin1	Pin0	Input Value	ASCII Output
0	0	0x00	'0' (0x30)
0	1	0x01	'1' (0x31)
1	0	0x02	'2' (0x32)
1	1	0x03	'3' (0x33)

Program:

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

int main(void)
{
    unsigned char input;

    // Configure PB1 and PB0 as input
    DDRB &= ~((1<<1)|(1<<0));

    // Configure Port D as output
    DDRD = 0xFF;

    // Enable pull-ups for PB1 and PB0
    PORTB |= (1<<1)|(1<<0);

    while(1)
    {
        // Read PB1 and PB0
        input = PINB & 0x03;  // Mask other bits

        switch(input)
        {
            case 0x00:  // Pin1=0, Pin0=0
```

```
                PORTD = '0';  // ASCII '0' = 0x30
                break;

            case 0x01:  // Pin1=0, Pin0=1
                PORTD = '1';  // ASCII '1' = 0x31
                break;

            case 0x02:  // Pin1=1, Pin0=0
                PORTD = '2';  // ASCII '2' = 0x32
                break;

            case 0x03:  // Pin1=1, Pin0=1
                PORTD = '3';  // ASCII '3' = 0x33
                break;
        }
    }

    return 0;
}
```

Mnemonic

"Mask-inputs ASCII-conversion Truth-table Switch-case"

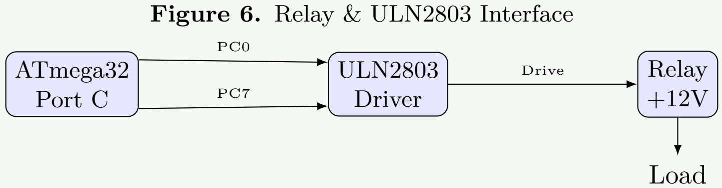
Question 4(a) [3 marks]

Draw interfacing diagram of relay and relay driver ULN2803 with ATmega32

Solution

Relay Interface Diagram:

Figure 6. Relay & ULN2803 Interface



- **ULN2803**: Darlington transistor array, current amplification.
- **Protection Diodes**: Built-in flyback diodes for inductive loads.
- **Relay Coil**: Requires 12V, controlled by ULN2803 output.

Mnemonic

"ULN-driver Port-control Current-amplify"

Question 4(b) [4 marks]

Write steps of programming the A/D converter using polling method

Solution

ADC Programming Steps:

Table 16. ADC Steps

Step	Action
1	Configure ADMUX register (reference, channel)
2	Configure ADCSRA register (enable, prescaler)
3	Start conversion (set ADSC bit)
4	Wait for conversion complete (poll ADIF flag)
5	Read result from ADCL and ADCH

Code:

```
// Step 1: Configure ADMUX
ADMUX = (1<<REFS0);  // AVCC reference, channel 0

// Step 2: Enable ADC with prescaler
ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

// Step 3: Start conversion
ADCSRA |= (1<<ADSC);

// Step 4: Wait for completion
while(!(ADCSRA & (1<<ADIF)));

// Step 5: Read result
result = ADC;  // Combined ADCL and ADCH
```

Mnemonic

"Configure-ADMUX Configure-ADCSRA Start-conversion Wait-complete Read-result"

Question 4(c) [7 marks]

Explain I2C-Two Wire Serial Interface (TWI) Protocol in detail.

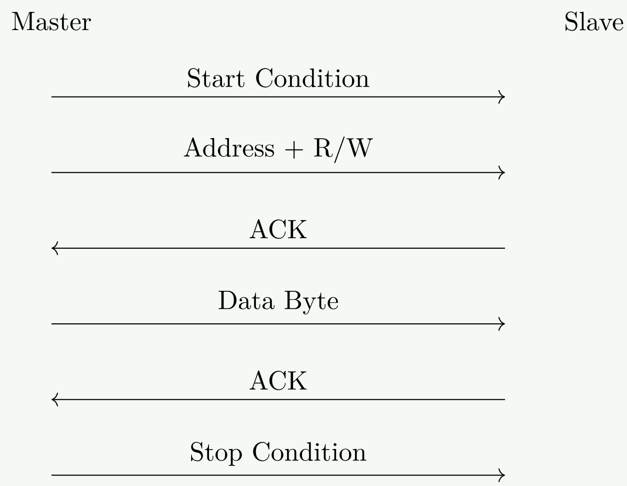
Solution

I2C Protocol Features:

Table 17. I2C Features

Feature	Description
Two Wires	SDA (Data) and SCL (Clock)
Multi-master	Multiple masters can control bus
Addressing	7-bit or 10-bit device addresses
Bidirectional	Data flows both directions

Figure 7. I2C Sequence



- **Start Condition**: SDA goes low while SCL is high.
- **Address Frame**: 7-bit address + R/W bit.
- **Data Frame**: 8-bit data + ACK/NACK.
- **Stop Condition**: SDA goes high while SCL is high.

Registers: TWCR, **TWDR**, **TWAR**, **TWSR**.

Mnemonic

"Start-Address-Data Control-Status-Address"

OR

Question 4(a) [3 marks]

Explain any one PWM mode for controlling speed of DC motor by using 8-bit timer

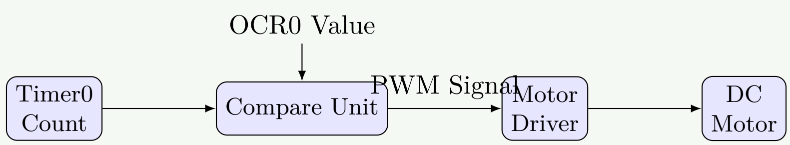
Solution

Fast PWM Mode (Mode 3):

Table 18. Fast PWM

Parameter	Value
WGM bits	WGM01=1, WGM00=1
TOP value	0xFF (255)
Resolution	8-bit
Frequency	$f_{clk}/(256 \times prescaler)$

Figure 8. PWM Motor Control



- **Duty Cycle Control**: OCR0 value determines motor speed.
- **Motor Control**: Higher duty cycle = higher speed.

Mnemonic

"Fast-PWM Timer0 OCR0-control"

OR

Question 4(b) [4 marks]

Write steps for reading data from an SPI device

Solution

SPI Read Steps:

Table 19. SPI Steps

Step	Action
1	Configure SPI control register (SPCR)
2	Set SS pin low to select slave
3	Write dummy data to SPDR
4	Wait for transmission complete (SPIF flag)
5	Read received data from SPDR
6	Set SS pin high to deselect slave

Code:

```
// Configure SPI
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);

// Select slave
PORTB &= ~(1<<SS);

// Send dummy byte
SPDR = 0xFF;

// Wait for complete
while(!(SPSR & (1<<SPIF)));

// Read data
data = SPDR;

// Deselect slave
PORTB |= (1<<SS);
```

Mnemonic

"Configure Select Write-dummy Wait Read-data Deselect"

OR

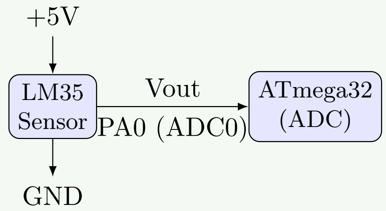
Question 4(c) [7 marks]

Draw and explain interfacing diagram of LM35 with ATmega32.

Solution

LM35 Interface:

Figure 9. LM35 Interface



Specifications:

Table 20. LM35 Specs

Parameter	Value
Output	10mV/°C
Range	0°C to 100°C
Supply	4V to 30V
Accuracy	±0.5°C

Calculation:

$$Temp = \frac{ADC \times 5000mV}{1024 \times 10mV/°C}$$

Mnemonic

"Voltage-output ADC-conversion Reference-5V Calculation-formula"

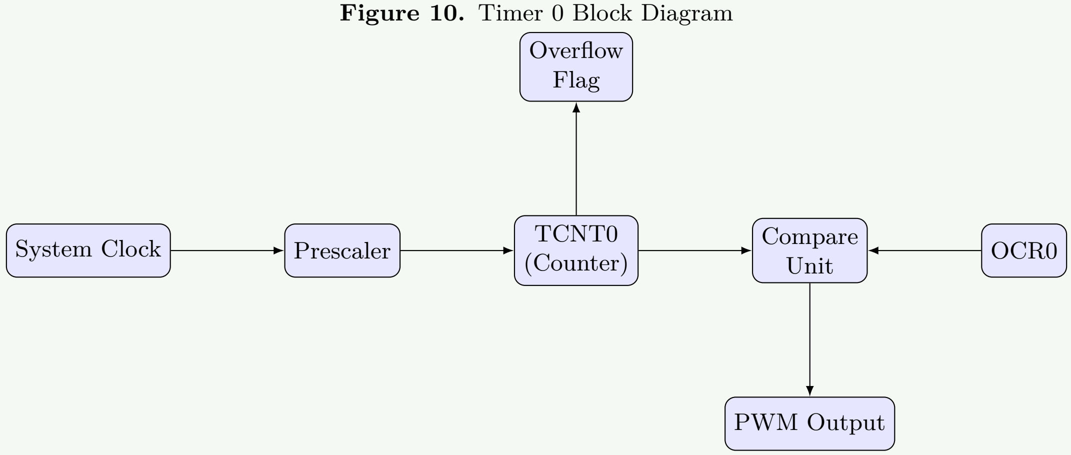
Question 5(a) [3 marks]

Draw Timer 0 Working Block diagram.

Solution

Timer 0 Block Diagram:

Figure 10. Timer 0 Block Diagram



- **Prescaler**: Divides clock by 1, 8, 64, 256, 1024.
- **Counter**: 8-bit up counter (0-255).

- **Compare Unit**: Matches TCNT0 with OCR0.

Mnemonic

"Prescaler Counter Compare Overflow"

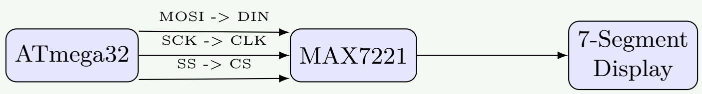
Question 5(b) [4 marks]

Draw Interfacing of MAX7221 to ATmega32.

Solution

MAX7221 Interface:

Figure 11. MAX7221 Interface



- **Display Driver**: 8-digit 7-segment LED driver.
- **SPI Interface**: Uses DIN, CLK, CS pins.
- **Features**: Brightness control, BCD decoding.

Mnemonic

"SPI-interface Current-control Decode-mode Initialize-setup Scan-limit"

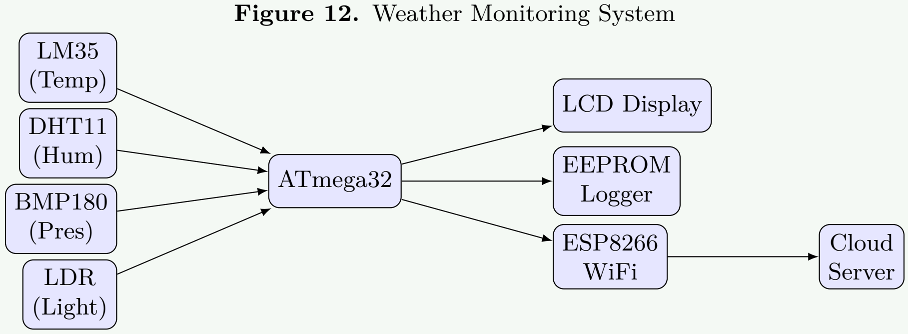
Question 5(c) [7 marks]

Explain Weather Monitoring System.

Solution

Weather Monitoring System:

Figure 12. Weather Monitoring System



Components:

Table 21. Components

Component	Function
LM35	Temperature measurement
DHT11	Humidity measurement
BMP180	Pressure measurement
ESP8266	WiFi connectivity for remote access

- **Real-time**: Continuous monitoring and display.
- **Remote Access**: Data uploaded to cloud.
- **Alerts**: Warning on threshold breach.

Mnemonic

"Sensors Monitoring Alert Remote Temperature Weather"

OR

Question 5(a) [3 marks]

Draw and explain Timer/Counter Control Register 0(TCCR0)

Solution

TCCR0 Register:

Table 22. TCCR0 Layout

7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

- **FOC0**: Force Output Compare.
- **WGM01:00**: Waveform Generation Mode (Normal, PWM, CTC).
- **COM01:00**: Compare Output Mode.
- **CS02:00**: Clock Select (Prescaler settings).

Mnemonic

"Force Waveform Compare Clock-Select"

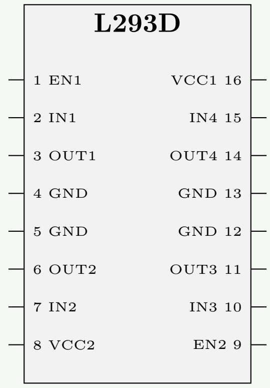
OR

Question 5(b) [4 marks]

Explain the function of motor driver L293D.

Solution

L293D Motor Driver:



- **Features**: Dual H-Bridge, 600mA per channel.
- **Operation**: Controls direction and speed (PWM).
- **Supply**: VCC1 logic (5V), VCC2 motor (up to 36V).

Mnemonic

"Dual-channel H-bridge Input-control Enable-PWM"

OR

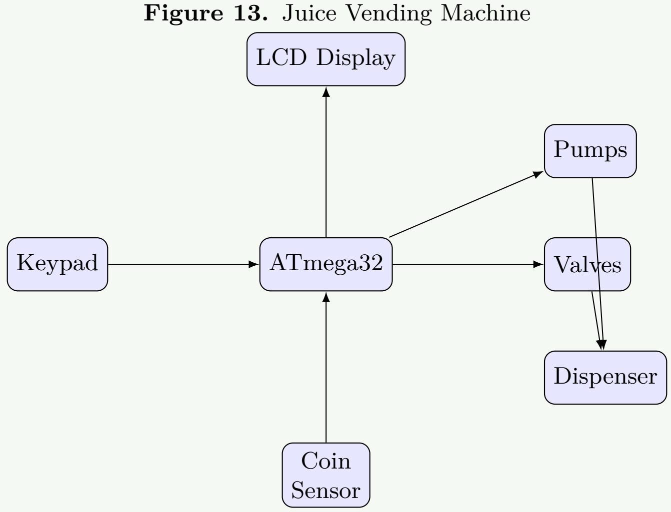
Question 5(c) [7 marks]

Explain Automatic Juice vending machine.

Solution

Automatic Juice Vending Machine:

Figure 13. Juice Vending Machine



Operation:

1. **Selection**: User selects juice via Keypad.
2. **Payment**: Coin sensor validates payment.
3. **Processing**: MCU activates pumps/valves for mixing.
4. **Dispensing**: Juice is dispensed, message on LCD.

Features: Multiple flavors, inventory monitoring, automated cleaning.

Mnemonic

"Juice-selection User-interface Mixing-control Payment-system Sensors-monitoring"