Session 24

LIN protocol and its advantages

The LIN (Local Interconnect Network) protocol is a low-cost, low-speed serial communication protocol designed primarily for automotive applications. Its main advantages include its simplicity, low cost, and suitability for non-critical systems within vehicles, making it ideal for features like window controls, mirror adjustments, and interior lighting.

Here's a more detailed breakdown:

Advantages of LIN Protocol:

- **Low Cost:**

  LIN's simplicity allows for cost-effective implementation using standard UART modules in microcontrollers, making it a budget-friendly solution for certain automotive applications.

- **Simplified Wiring:**

  LIN uses a single wire for communication, reducing wiring complexity, weight, and installation costs.

- **Event-triggered Communication:**

  LIN supports event-triggered communication, meaning messages are sent only when necessary, optimizing bandwidth usage and power consumption.

- **Master-Slave Architecture:**

  The master node controls communication, ensuring predictable and deterministic behavior, crucial for real-time control systems.

- **Suitable for Non-Critical Systems:**

  LIN is well-suited for applications where high bandwidth and fault tolerance are not essential, such as body electronics, interior lighting, and seat controls.

- **Integration with CAN Bus:**
  LIN can be integrated with the more robust CAN bus to create a hierarchical network, where LIN handles less critical tasks and CAN handles critical ones.

Applications in Vehicles:

- **Body Control Modules:** LIN is used to control door locks, window lifts, mirror adjustments, sunroofs, and other body functions.
- **Interior Lighting:** LIN enables features like dimming, color changes, and automatic on/off for interior lights.

- **Seat Controls:** LIN can manage seat position motors and control panels.
- **Climate Control Systems:** LIN can be used to control various aspects of the climate control system.
- **Sensors and Actuators:** LIN facilitates communication between sensors and actuators in various systems, like engine cooling fans and grille shutters.

In essence, LIN offers a cost-effective and efficient solution for managing non-critical functions in vehicles, complementing the more powerful CAN bus and contributing to overall vehicle functionality and cost optimization, according to automotive experts.

Sample Pseudocode

```
#include <stdint.h>

#include <stdbool.h>


// Constants

#define LIN_BREAK_FIELD_DURATION 13  // In bit times (approximate)

#define LIN_SYNC_BYTE 0x55


// Function prototypes (hardware specific)

void UART_SendByte(uint8_t byte);

void UART_SendBreak(void);

void Delay_us(uint32_t us);


// Compute LIN Identifier parity bits (P0 and P1)

uint8_t ComputeLINParity(uint8_t id) {

    // Calculate parity bits P0 and P1 based on 6-bit ID (bits 0-5)

    uint8_t p0 = ((id >> 0) ^ (id >> 1) ^ (id >> 2) ^ (id >> 4)) & 0x01;
```

```
    uint8_t p1 = ~(((id >> 1) ^ (id >> 3) ^ (id >> 4) ^ (id >> 5))) & 0x01;
    return (id & 0x3F) | (p0 << 6) | (p1 << 7);
}

// Master sends LIN frame: Break + Sync + Identifier + Data + Checksum
void LIN_MasterSendFrame(uint8_t id, uint8_t* data, uint8_t dataLen) {
    UART_SendBreak();               // Send Break field (dominant low)
    UART_SendByte(LIN_SYNC_BYTE);  // Send Sync byte

    uint8_t idWithParity = ComputeLINParity(id);
    UART_SendByte(idWithParity);   // Send Identifier with parity

    // Send data bytes
    for (uint8_t i = 0; i < dataLen; i++) {
        UART_SendByte(data[i]);
    }

    // Calculate checksum (classic checksum: sum of identifier and data bytes)
    uint16_t sum = idWithParity;
    for (uint8_t i = 0; i < dataLen; i++) {
        sum += data[i];
        if (sum > 255) sum -= 255;
    }
    uint8_t checksum = (uint8_t)(255 - sum);
```

```
    UART_SendByte(checksum);       // Send checksum
}

// Dummy functions for illustration (replace with MCU-specific code)
void UART_SendByte(uint8_t byte) {
    // Wait for TX buffer empty, then load byte to UART data register
}

void UART_SendBreak(void) {
    // Send dominant low for at least 13 bit times
    // Could be done by lowering TX pin manually or configuring UART break frame
}

int main(void) {
    uint8_t data[4] = {0x12, 0x34, 0x56, 0x78};
    while(1) {
        LIN_MasterSendFrame(0x12, data, 4);
        // Delay between frames
        // Delay_ms(1000);
    }
}
```