

Martin DIS

by Martin White

Submission date: 14-Nov-2024 11:52PM (UTC+0000)

Submission ID: 242729632

File name: 42524_Martin_White_Martin_DIS_523660_47450836.docx (379.41K)

Word count: 2322

Character count: 14518



Distributed Intelligence systems

I2C MULTI-MASTER SYSTEM

Martin White | Embedded Electronics | 20/10/2024

Abstract

This report details the design and analysis of a distributed intelligence system focused on a multi-master I2C configuration for automotive lighting control. Utilizing two PIC18F2525 microcontrollers as masters, the system demonstrates cooperative management of inputs and outputs through shared I2C EEPROM memory and synchronized data handling. The architecture ensures real-time control of various vehicle lights, such as brake, rear, indicator, fog, and reverse lights, showcasing how coordinated microcontroller communication facilitates modular and efficient task distribution. Key elements include the implementation of a handshake protocol to manage shared resource access, interrupt-driven logic for dynamic responses, and strategic use of non-volatile memory for consistent data states. This design underscores the application of quality management principles, such as continuous improvement, operational reliability, and risk-based thinking, reinforcing safety and adaptability in embedded system development. The outcome is a robust, scalable solution for automotive control systems, highlighting the integration of multi-master I2C protocols in real-time, distributed embedded environments.

Research

QUALITY MANAGEMENT SYSTEMS [1]

Quality management systems (QMS) are critical in maintaining standards and driving continuous improvement, improving operational efficiency, and strengthening safety protocols. By setting clear goals and effectively managing resources and processes, a quality management system optimizes resource use and balances immediate and future impacts. Continuous improvement helps adapt to changing conditions and promotes growth. Regular reviews help ensure the effectiveness of the quality management system and guide strategic decisions. Prioritizing employee expertise, aligning outcomes with stakeholder needs, and implementing risk-based thinking promote consistent quality, safety, and stakeholder trust in services.

POINT-TO-POINT TRANSMISSION SYSTEM [2]

Point-to-point networks (PTP) connect two locations for example, over a leased line, ensuring secure, fast, and reliable data transmission without the risks associated with public Internet connections. These networks are ideal for critical

tasks such as file sharing, data backup, VOIP, and video surveillance. PTP networks often use microwave radio links for flexibility, speed, low latency, and high availability. Although fiber optic leased lines are an alternative, they are more expensive, slower to install, and more prone to damage. PTP communication is usually a direct connection, such as a telephone conversation or a modern fiber optic connection.

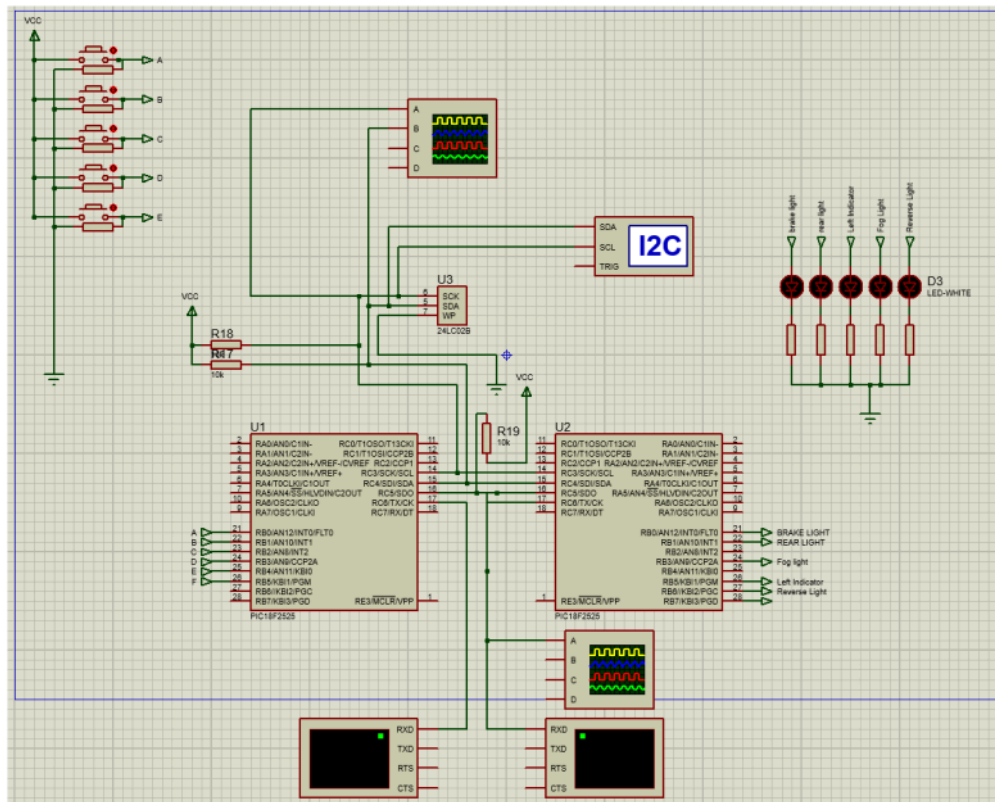
POINT TO MULTIPOINT TRANSMISSION SYSTEM [4]

Point-to-multipoint communication (P2MP) involves a central device transmitting data to multiple receivers simultaneously and is commonly used in wireless networks such as Wi-fi and cellular, video conferencing, and broadcasting. Unlike point-to-point connections that connect only two devices, P2MP enables efficient, scalable, and cost-effective data distribution via protocols such as unicast and multicast. Although it offers simplicity, resource efficiency, and reliability with built-in redundancy, it also faces challenges such as limited bandwidth, potential interference, and security concerns. P2MP systems often use different antennas, such as omnidirectional and sector antennas to optimize coverage for various applications.

MULTI-MASTER TRANSMISSION SYSTEM [5]

In I2C environments with multiple masters, each device must work together by detecting when the bus is in use. This involves following an arbitration logic: When two devices communicate simultaneously, the one that sends more zeros or works slower wins, causing the other to shut down immediately. Additionally, devices must detect a communication in progress and wait for a stop condition before initiating their transmission. For successful operation, all connected masters must support the multi-master functionality. A single-master device that lacks these capabilities, when used in conjunction with a multi-master, can disrupt communications, leading to unpredictable results.

Design



The circuit diagram depicts a multi-master I2C system designed for automotive lighting control, featuring two PIC microcontrollers (U₁ and U₂), an I2C EEPROM (U₃), input switches, and LED indicators simulating various vehicle lights such as brake, rear, and indicator lights. Each microcontroller functions as an I2C master, capable of reading from and writing to shared EEPROM memory, demonstrating a cooperative multi-master setup where both can manage data transfer. The microcontrollers are responsible for reading button inputs and controlling outputs to the vehicle lights, with U₁ focusing on processing user inputs from switches (labeled A–E) and updating the I2C memory with the corresponding light status. U₂ reads this data from memory, interprets it, and drives the output to control the lights connected to its port B. The I2C EEPROM serves as shared, non-volatile storage that allows the microcontrollers to synchronize the control of lights by writing or reading data, ensuring consistent system operation.

The system exemplifies how two microcontrollers can coordinate within a multi-master I2C setup to share control tasks. U1 acts as an interface for user-triggered actions, writing button states to the shared memory, while U2 monitors and processes this data to execute light control functions. This architecture simulates an automotive environment where multiple controllers can work independently yet collaboratively to manage different vehicle functions, enhancing modular control. The use of I2C with shared EEPROM ensures that data remains consistent between the controllers, supporting scenarios where actions such as indicator flashing or turning on brake lights are distributed across different microcontrollers. The memory-based communication enables reliable and responsive control, showcasing the system's capability to handle real-time lighting changes efficiently in automotive applications.

Development

To design the vehicle lighting control system, I chose the PIC18F2525 microcontroller, as it was used in our lectures, allowing me to apply my understanding of multi-master I2C communication. An essential addition to this design was the handshake (hs) control signal, which coordinates I2C access between devices, ensuring smooth communication without conflicts.

The system controls the rear-left corner lights: brake lights, rear lights, the left indicator, fog lights, and reverse lights. Each button press triggers a specific case in the code, adjusting the `local_output` variable that holds the light states. This state is then stored in memory via the I2C protocol, using `write_to_memory` and `read_from_memory` functions. The handshake line (hs) in these functions ensures conflict-free memory access.

In the main loop, button inputs are processed to toggle lights. For example, pressing the brake button checks if the light is on by examining the corresponding bit in `local_output`. If off, the system turns it on, saves the state in memory, and outputs a corresponding character (like 'A' for the brake light). This process repeats for each light, maintaining consistent state tracking.

Additionally, a TIMER2 interrupt provides a flashing mechanism for the indicators, toggling the flash state periodically. Together, the handshake function and I2C protocol improve system reliability, making this lighting control system well-suited for real-world automotive use.

Analysis

MASTER

The C code demonstrates the functionality of a microcontroller (U₁) in a multi-master I²C system used for automotive lighting control. The microcontroller reads input from buttons.

Main Logic: The `main()` function continuously checks button inputs and toggles the corresponding light status. Each button press sets or clears a specific bit in the `local_output` variable, representing the state of each light. Depending on the button pressed, a unique character (`txbyte`) is assigned to indicate the action (e.g., 'A' for brake lights ON, 'a' for brake lights OFF). The `txbyte` is then printed via `printf()` and written to an EEPROM at address `#0` using the `write_to_memory()` function. This ensures that changes in light states are temporarily stored in shared memory, later being overwritten by the next input, enabling synchronization between U₁ and other controllers.

I²C Communication: The `write_to_memory()` function handles I²C communication by waiting for the handshake line (`hs`) to be high. The point of this function is to prevent the two Masters from talking over each other. The microcontroller lowers the line, initiates an I²C start condition, and writes the device and memory addresses followed by the data (`txbyte`). After a short delay, the function stops the I²C transaction and releases the `hs` line, completing the write process.

Reading Memory: The `read_from_memory()` function allows the microcontroller to read stored data from the EEPROM. It waits for the `hs` signal, lowers it to begin communication, sends the device and address bytes, and then reads the data. The function concludes by stopping the I²C communication and releasing the `hs` line.

Reading Buttons: The `read_buttons()` function polls port B for button input and returns a byte representing which button was pressed. The function includes a delay for button debounce to avoid multiple readings from a single press.

Role in Multi-Master I²C System: U₁ operates as a master that reads user inputs and writes the light status to shared EEPROM. This setup allows another master microcontroller (i.e., U₂) to read from the same EEPROM and control outputs accordingly. The use of a handshake line (`hs`) ensures proper coordination between the masters, preventing simultaneous access and potential data corruption. Through this approach, both microcontrollers can synchronize their actions, demonstrating how a multi-master I²C system enables collaborative control in an automotive context.

SLAVE.

The C code demonstrates the functionality of a microcontroller (U2) in a multi-master I2C system used for automotive lighting control. The microcontroller reads input from buttons and updates the status of various lights based on user interaction.

Main Logic: The `main()` function continuously checks the current state of the buttons by reading data from the EEPROM at address 0x00 using the `read_from_memory()` function. If a change in button state is detected (i.e., `but` differs from `old_but`), the new state is printed using `printf()` and stored in the EEPROM at address 0x10 via the `write_to_memory()` function. The function `process_input()` is then called to set or clear specific bits in `local_output` and `control_byte`, representing the status of each light. For example, pressing 'A' sets the bit for brake lights, while pressing 'a' clears it. This process ensures that the light status reflects user inputs. The old button state (`old_but`) is updated to match `but` after processing.

I2C Communication: The `write_to_memory()` function manages I2C communication to write data safely to the EEPROM. The function begins by waiting for the handshake line (`hs`) to be high, ensuring that the EEPROM is ready for communication. The microcontroller then lowers the `hs` line, initiates an I2C start condition, and sends the chip address, memory address, and data byte (e.g., `but`). A short delay follows, after which the I2C stop condition is sent, and the `hs` line is released, signaling the end of the transaction. This approach prevents conflicts between masters, enabling smooth data writing.

Reading Memory: The `read_from_memory()` function allows the microcontroller to retrieve stored data from the EEPROM. It begins by waiting for the `hs` line to be high, then lowers it to initiate communication. The function sends the chip and memory addresses, delays for synchronization, and then reads the data byte. The I2C stop condition is sent, and the `hs` line is released, completing the read process. This ensures that the microcontroller can read previously stored button states, facilitating coordinated control.

Timer and Flashing Logic: The code includes a Timer2 interrupt (`TIMER2_isr()`) that toggles the flash variable every 40 ms, creating a flashing effect for lights such as indicators. The `handle_flash()` function uses the flash state to toggle `pin_b5`, which controls the right indicator light if activated.

Role in Multi-Master I2C System: U2 operates as a master that reads user inputs and updates the light control state, storing this information in the shared EEPROM. This setup allows other master microcontrollers (e.g., U1) to read from the same EEPROM and synchronize their actions. The use of a shared hs line ensures that only one microcontroller accesses the EEPROM at a time, preventing data corruption. This coordination mechanism is crucial in a multi-master I2C system, enabling both microcontrollers to manage shared resources efficiently and collaboratively control the automotive lighting system.

Conclusion

The multi-master I2C system detailed in this report presents an advanced and cooperative approach to automotive lighting control using dual PIC18F2525 microcontrollers. This design highlights the effective implementation of a shared-resource environment, where both microcontrollers operate as masters, coordinating real-time data handling and task distribution. Through shared I2C EEPROM memory and a handshake protocol, the architecture ensures synchronized access and robust communication, enhancing reliability and responsiveness.

A key feature is the handshake line (hs) that prevents concurrent EEPROM access, reducing data corruption and enabling smooth, conflict-free interactions. This coordination demonstrates how distributed control systems maintain data consistency and stability, even with multi-master communication complexities. The use of interrupt-driven logic further improves the system's responsiveness to changing inputs, a crucial attribute for real-time automotive applications where safety and speed are critical.

The integration of input reading and output control functions showcases system modularity, managing vehicle lights like brake, rear, indicator, fog, and reverse lights seamlessly. U1 processes user inputs through button presses and updates the shared EEPROM, while U2 reads this data, processes it, and updates the lights. This division of tasks, supported by synchronized data handling, demonstrates efficient distributed intelligence without compromising performance.

Strategic use of non-volatile memory ensures persistent data storage, maintaining system state during power cycles or disruptions. This feature keeps vehicle lights consistent after resets, aligning with automotive safety standards. Timer2 interrupts are used to create periodic flashing effects for indicator lights, illustrating seamless integration of timer-based functions within the multi-master setup.

Applying quality management principles reinforces the system's focus on continuous improvement, reliability, and risk-based thinking. Structured processes, systematic testing, and iterative refinement guide the development to meet safety and adaptability standards. The cooperative multi-master configuration also allows for scalability, enabling future system enhancements or adding more microcontrollers to control other subsystems without significant redesign.

Overall, the multi-master I2C system effectively demonstrates how two microcontrollers can manage complex tasks collaboratively, ensuring responsive, coordinated control. This approach optimizes resource allocation and supports modular expansion. The result is a robust, scalable solution suited for automotive applications where synchronized communication and real-time data management are essential. This project exemplifies the successful use of multi-master I2C protocols in embedded systems, showcasing how collaborative microcontroller designs can enhance the performance and reliability of modern automotive control systems.

References

[1]

"Engineering Council," *www.engc.org.uk*.

<https://www.engc.org.uk/quality/quality-policy/>

[2]

D. Jones, "What is a Point to Point Network?," *APC Solutions*, Nov. 23, 2020.

<https://www.apcsolutionsuk.com/point-to-point-network/>

[3]

"Point to Point Communication Explained," *melita.io*.

<https://www.melita.io/help/general-information/iot-wiki/point-to-point-communication-explained/>

[4]

“What is Point-to-Multipoint Communication?,” *Sanny Telecom*, May 18, 2024. <https://www.sannytelecom.com/what-is-point-to-multipoint-communication/>

[5]

“MultiMaster,” *I2C Bus*, Sep. 19, 2016. <https://www.i2c-bus.org/multimaster/>

Appendices

FINAL GRADE

GENERAL COMMENTS

72/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

GRADING FORM: SMU GENERIC_GJ

MARTIN WHITE

72.35

RESEARCH-INFORMED LITERATURE



You have explained what point to point and point to multipoint and multimaster are correctly. 68%

13.6

KNOWLEDGE AND UNDERSTANDING



The design of your circuit is suitable for the task 75%

18.75

ANALYSIS



You have completed the programs for both devices and the circuit works well. 70%

17.5

PRACTICAL APPLICATION AND DEP.



Your explanation is good and acknowledges that both pics speak to memory IC in multimaster mode and not directly to each other 78%

11.7

SKILLS FOR PROFESSIONAL PRAC.



The report is well presented with sections to explain the code and include a reasonable number of references. 72%

10.8