Topic 7 : I2C Interface



Problem 1 : Design two embedded systems using ATMEGA32 (working at 8Mhz) communicate together by I2C serial interface. One system work as the Master I2C Transmitter, One system work as the I2C Slave Receiver. The Master I2C Microcontoller has 8 buttons, The Slave I2C Microcontoller has LEDs. Wrrite the C program to control the two system, When button0 is pressed the Master system will send character ‘0’ to slave system, When button1 is pressed the Master system will send character ‘1’ to slave system . When the slave system receiving new character from the I2C interface the system will toggle the relevant LED ( for example when the character received is ‘0’ LED0 will be toggled, when the character received is ‘1’ LED1 will be toggled).

Problem 2 : Convert the C Program of Master I2C Microcontroller into AVR Assembly code.

**Problem 1: I2C Communication Between Master and Slave**

To achieve communication between the Master I2C Transmitter and the I2C Slave Receiver using ATMEGA32 microcontrollers, follow these steps:

1. **Initialize I2C (TWI) on Both Microcontrollers:**
   * Set up the I2C interface on both the Master and Slave microcontrollers.
   * Configure the I2C clock frequency and enable the I2C module.
2. **Master Microcontroller (Transmitter):**
   * Read button states (you can replace this with your actual button reading logic).
   * If button0 is pressed, send character ‘0’ to the Slave.
   * If button1 is pressed, send character ‘1’ to the Slave.
3. **Slave Microcontroller (Receiver):**
   * Receive data from the Master via I2C.
   * When a new character is received:
     + If the character is ‘0’, toggle LED0.
     + If the character is ‘1’, toggle LED1.

Below is a simplified C program for the Master (Transmitter) and Slave (Receiver). You’ll need to adapt it to your specific hardware setup and button reading logic:

// Master (Transmitter)

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

void I2C\_Master\_Init() {

// Initialize I2C as Master

// Set clock frequency and enable I2C

// ...

}

void I2C\_SendChar(char data) {

// Send data via I2C

// ...

}

int main() {

// Initialize I2C as Master

I2C\_Master\_Init();

while (1) {

// Read button states (replace with actual logic)

if (button0\_pressed) {

I2C\_SendChar('0');

} else if (button1\_pressed) {

I2C\_SendChar('1');

}

\_delay\_ms(100); // Delay for stability

}

return 0;

}

// Slave (Receiver)

void I2C\_Slave\_Init() {

// Initialize I2C as Slave

// Set address and enable I2C

// ...

}

ISR(TWI\_vect) {

// Handle received data from Master

char receivedChar = TWDR;

if (receivedChar == '0') {

// Toggle LED0

} else if (receivedChar == '1') {

// Toggle LED1

}

}

int main() {

// Initialize I2C as Slave

I2C\_Slave\_Init();

sei(); // Enable global interrupts

while (1) {

// Your other tasks (e.g., LED control) here

}

return 0;

}

**Problem 2: Convert to AVR Assembly Code**

Converting the C program to AVR Assembly code is more complex. You’ll need to use AVR assembly instructions and directives. I recommend using an AVR assembly toolchain (such as avra) to convert the logic from the C program into assembly instructions. The exact assembly code will depend on your specific requirements and the toolchain you choose. Remember to adapt the assembly code to your specific hardware setup and button reading logic. 😊👍