**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

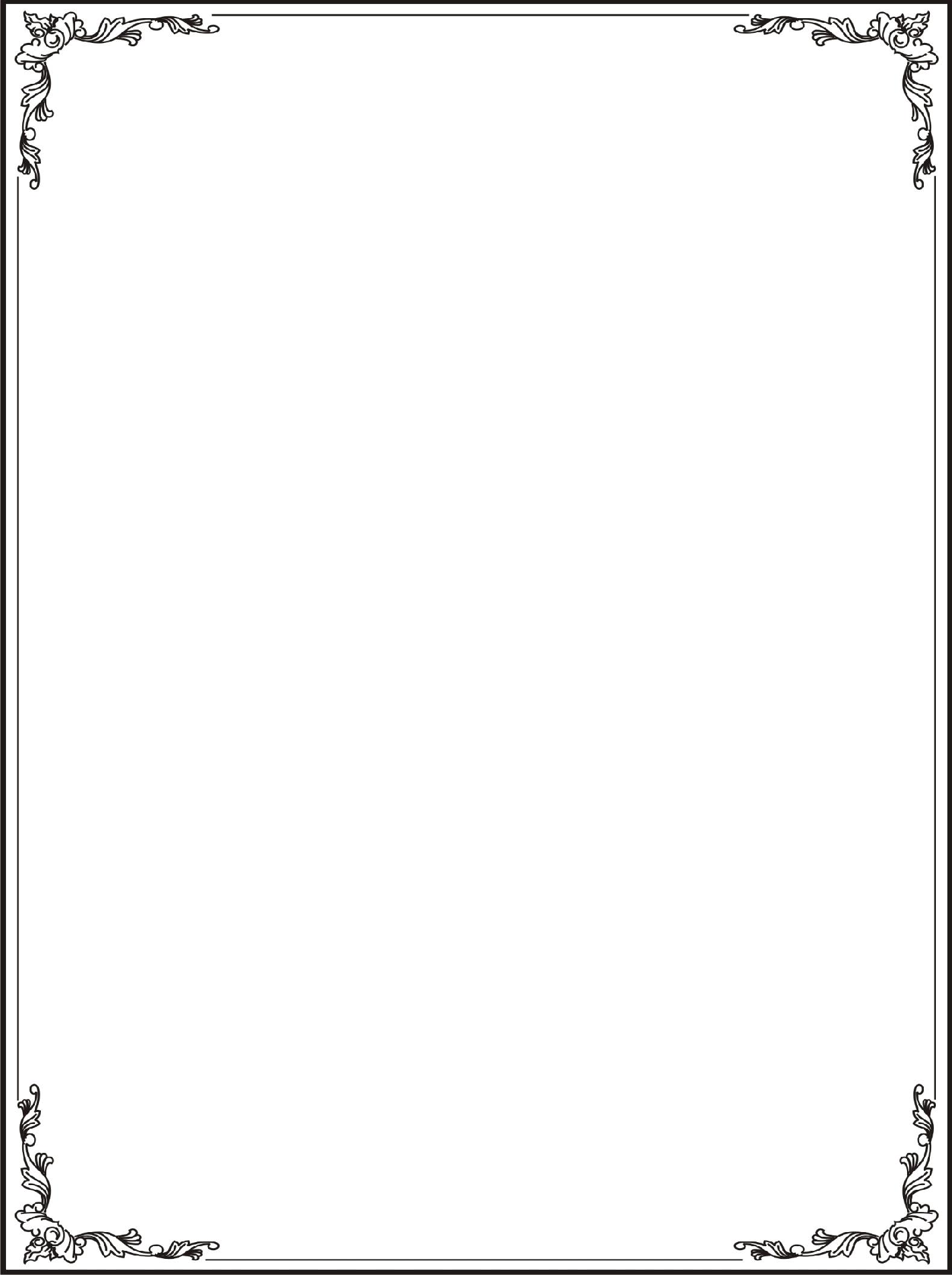
**THỰC TẬP KIẾN TRÚC VÀ TỔ CHỨC MÁY TÍNH**

GVHD: Huỳnh Hoàng Hà

HTSV: NGUYỄN ĐÌNH QUẢNG

MSSV: 21139044

**TP. Hồ Chí Minh - 10/2023**



|  |  |  |  |
| --- | --- | --- | --- |
| **Câu** | **Đã chạy / chưa?** | **Ý nghĩa chương trình** | **Ghi chú** |
| EEPROM 24C02 | Đã chạy | 8051 giao tiếp với EEPROM 24C02 hiển thị ra màn hình LCD 1604 |  |

I.Các linh kiện điện tử trong bài thực hành:

1. EEPROM 24C02:

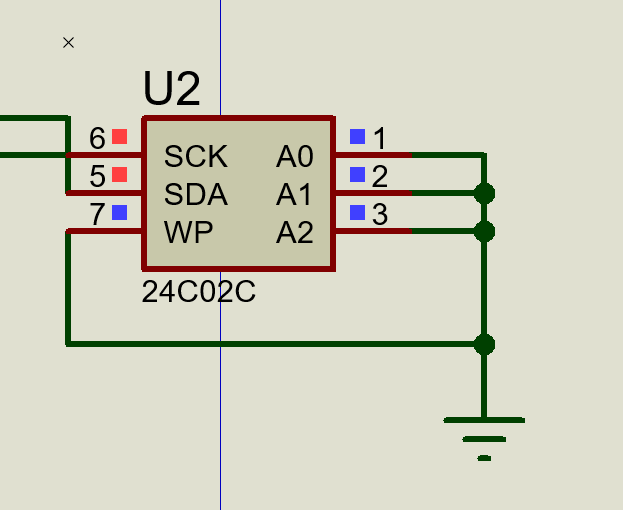
Dùng để lưu trữ dữ liệu cho vi điều khiển 8051. dung lượng lưu trữ 256 byte.

Giao tiếp I2C.

Hình ảnh 24C02 thực tế



Sơ đồ kết nối mạch:



Chức năng của các chân:

SCK - P2-1: chân tín hiêu xung clock.

SDA - P2-0: chân dữ liệu, địa chỉ cho các thiết bị I/O.

WP - GND: được sử dụng để bảo vệ việc khả năng ghi dữ liêụ vào chip.

A0, A1, A2: là các chân địa chỉ đầu vào.

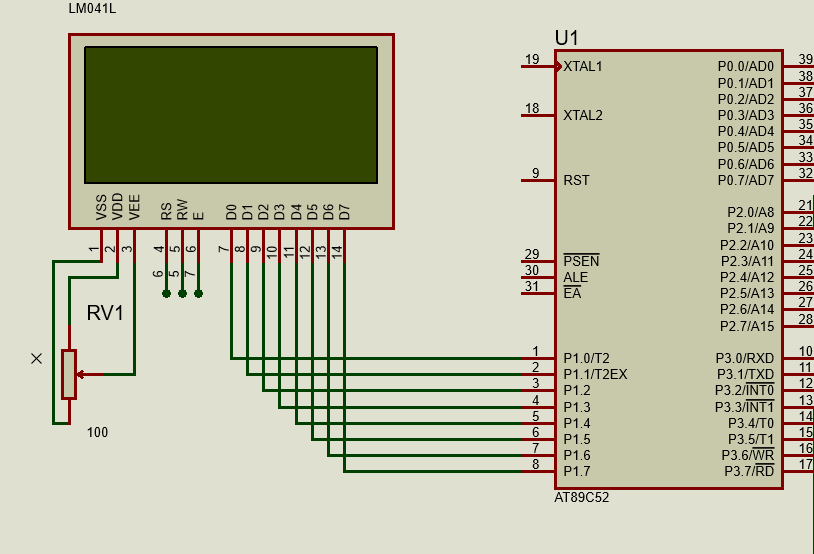
2. Màn hình LCD 1604:

Dùng để hiển thị dữ liệu được lưu trong bộ nhớ EEPROM.

Hình ảnh LCD 1604 thực tế:



Sơ đồ kết nối:



**II. EEPROM MODULE 24C02:**

- Giao tiếp I2C: là giao tiếp truyền được sử dụng để kết nối các vi mạch, thiết bị trong một hệ thống điện tử, mỗi vi mạch đều có một địa chỉ riêng để kết nôi. Giao tiếp theo kiểu Master ( 8051) và Slave (24C02).

- Thiết bị Master là thiết bị điều khiển giao tiếp và khởi tạo các hoạt động truyền trên bus I2C. Master có khả năng gửi lệnh và yêu cầu dữ liệu từ các Slave được kết nốt trong hệ thống.

- Thiết bị Slave: đây là thiết bị nhận lệnh từ Master và phản hội dữ liệu theo yều cầu từ Master.

+Tổng quát Quy trình đọc dữ liệu từ thiết bị ngoài

-Start I2C

-Gửi địa chỉ thiết bị.

-Gửi địa chỉ dữ liệu.

-Gửi lại địa chỉ thiết bị và lệnh read.

-Đọc dữ liệu.

-Stop I2C.

+Tổng quá quy trình viết dữ liệu từ thiết bị ngoài

-Start I2C

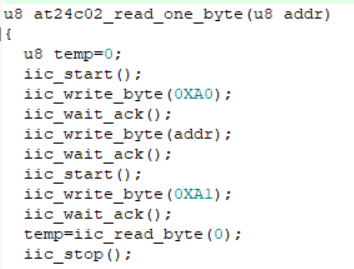
-Gửi địa chỉ thiết bị.

-Gửi địa chỉ dữ liệu.

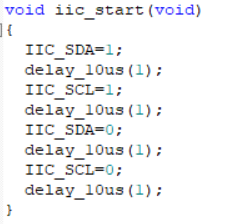
-Ghi dữ liệu.

-Stop I2C.

+Giải thích Code quá trình đọc dữ liệu từ 8051

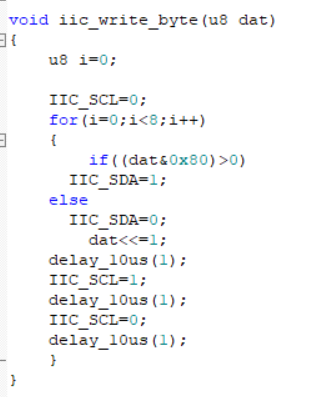


-Bắt đầu quá trình truyền thông I2C bắt cách gọi hàm iic\_start().



Hàm này sẽ set mức logic của hai chân SDA và SCL từ mức logic 1 xuống mức logic 0 để bắt đầu quá trình I2C.

-Gửi địa chỉ thiết bị ngoài muốn đọc dữ liệu: sử dụng hàm iic\_write\_byte(u8 dat)



Hàm này dùng để gửi địa chỉ: cho phép ghi dữ liệu. Việc này giúp vi điều khiển chỉ định vị trí đầu tiên muốn đọc.

Đặt mức 0 cho chân SCL, sử dụng dòng for để gửi 8 bit của dữ liệu.

Trong vòng lặp for kiểm tra từng bit bên trái cùng của byte nếu bit đó bằng 1 thì đặt SDA lên mức cao và ngược lại đặt xuống mức 0.

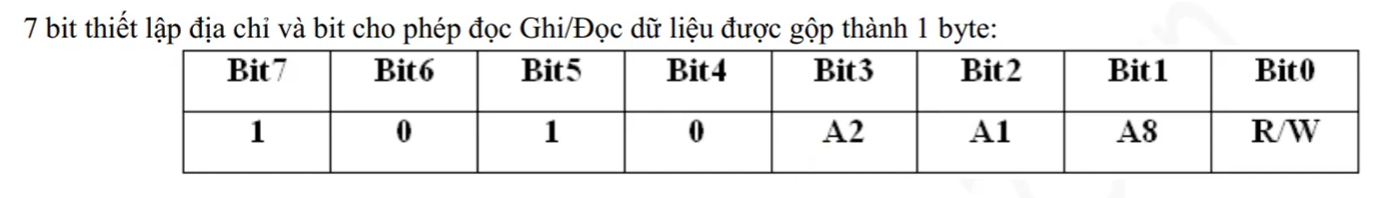
Sau đó lập trình chân SCL từ mức 1 xuống mức 0 để tạo xung truyền dữ liệu.

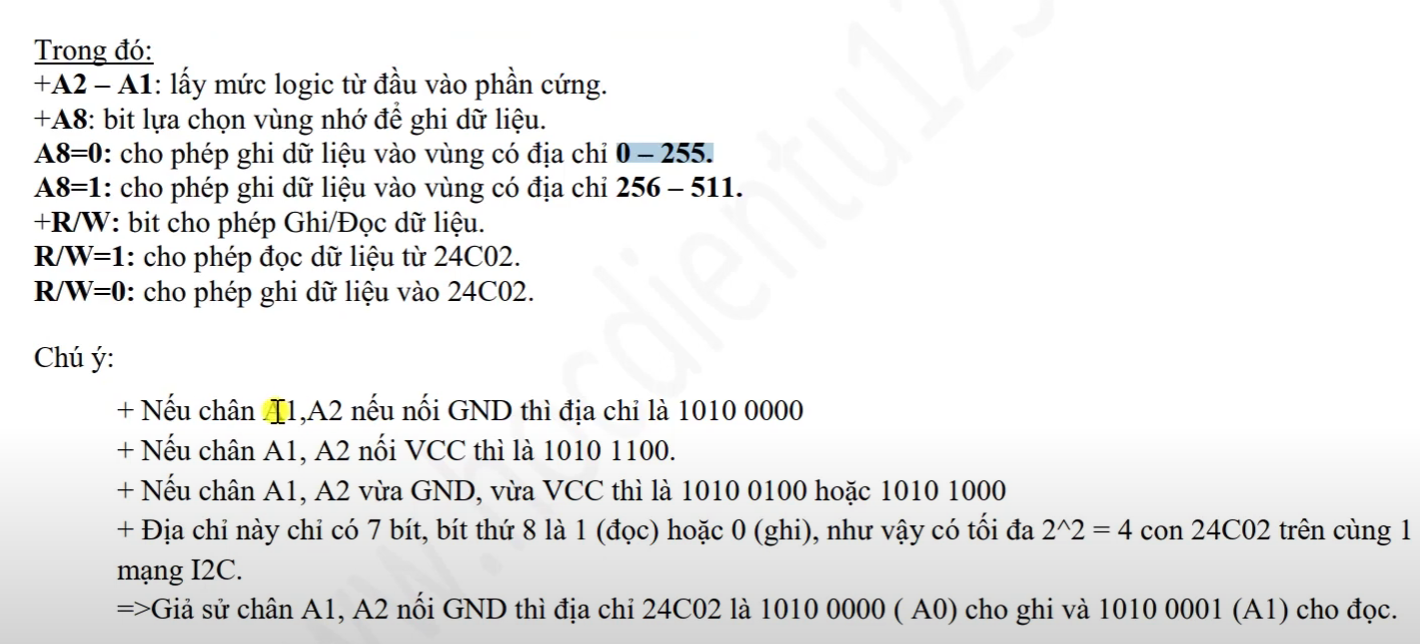
-Kiểm tra xem tín hiệu ACK() đã được nhận hay chưa bằng hàm iic\_wait\_ack(). Nếu không nhận được tín hiệu ACK() thì sẽ stop bằng cách gọi hàm ii2\_stop().

-Sau đó gửi địa chỉ dữ liệu bằng hàm iic\_write\_byte(addr) với addr là địa chỉ EEPROM 0. Sau đó đợi ACK để biết quá trình gửi dữ liệu thành công.

-Bắt đầu lại quá trình I2C bằng hàm ii2\_start().

-Gửi địa chỉ thiết bị bằng hàm iic\_write\_byte(0XA1) ) 0XA1: 10100001

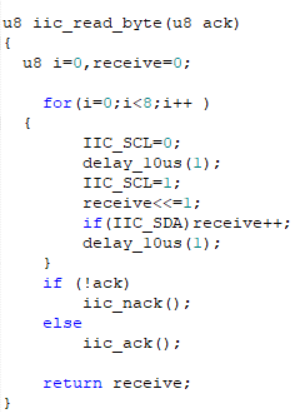




nên trong chương trình khi gửi địa chỉ 10100001 module 24C02 sẽ cho phép đọc dữ liệu từ 24C02.

-Đợi ACK để biết việc gửi dữ liệu thành công hay chưa.

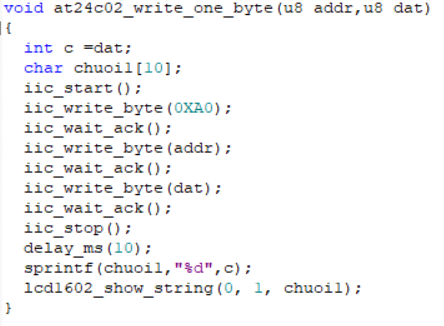
-Và đọc dữ liệu bằng hàm iic\_read\_byte(0)



Hàm này nhận một thông số đầu vào là ack. Thực hiện vòng for để đọc 8 bit dữ liệu và một hàm kiể tra nếu như SDA có mức logic 1 thì sẽ cộng thêm 1 vào dữ liệu đọc được. Đặt mức logic của chân SCL từ 0 lên 1 để tạo 1 xung cho 8051. Dữ liệu sẽ được đọc bằng cách dịch trái 1 bit. Nếu chân SDA bằng 1 thì biến receive sẽ được tăng lên 1.

Hàm iic\_stop() để kết thúc truyền thông I2C.

+Giải thích code quá trình ghi dữ liệu từ 8051

-

-Bắt đầu quá trình truyền thông I2C.

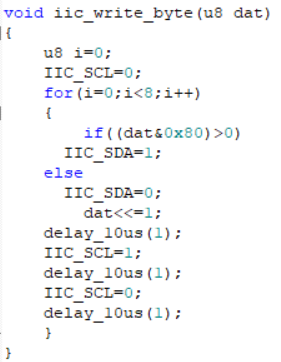
-Gửi địa chỉ thiết bị.

-Đợi ack để chờ cho việc gửi dữ liệu thành công.

-Gửi địa chỉ dữ liệu bằng hàm iic\_write\_byte(addr);

-Đợi ack để chờ cho việc gửi dữ liệu thành công.

-Ghi dữ liệu bằng hàm iic\_write\_byte(dat);



Chân SCL xuất ra mức 0, sử dụng một vòng for để gửi hết 8 bit dữ liệu. Kiểm tra nếu bit bên trái cùng của dữ liệu bằng chân SDA xuất ra mức 1 ngược lại thì xuất ra mức 0.

Đặt chân SCL lên 1 và cho xuống mức 0 để cấp xung cho việc đọc dữ liệu.

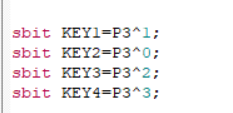
-Đợi ack để chờ cho việc gửi dữ liệu thành công.

-Dừng quá trình tuyền thông I2C bằng hàm iic\_stop().

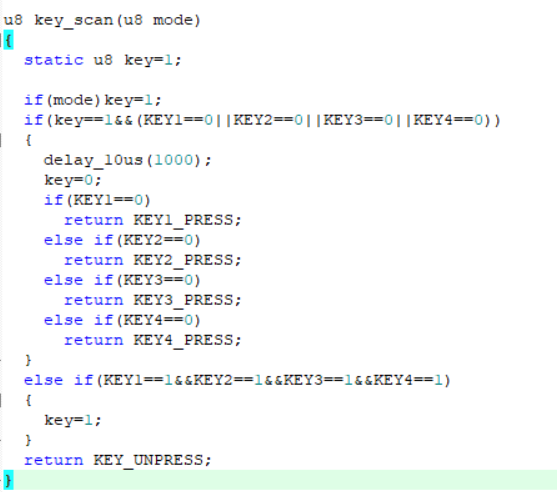
III. Nút nhấn:

- Dùng để thực hiện các chức năng đọc, ghi dữ liệu vào bộ nhớ EEPROM, tăng giảm giá trị muốn lưu vào bộ nhớ.

- Nút nhấn dùng để tăng giá trị:

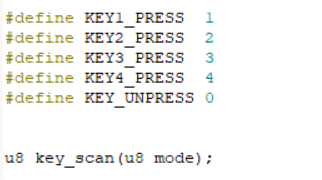


Cấu hình các nút nhấn ứng với các chân trên vi điều khiển 8051.



-Hàm này nhận một dữ liệu đầu vào là mode:

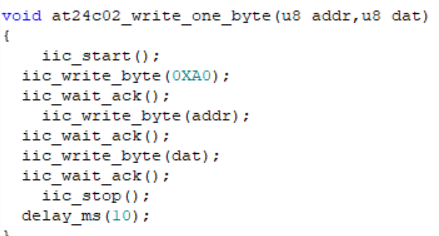
-Kiểm tra nếu key = 1 và 1 trong 4 nút nhấn được nhấn ( nút nhấn tích cực mức thấp 0 ) thì sẽ trả về KEY\_PRESS được định nghĩa trong file key.h, nếu không key sẽ bằng 1.



-KEY1 nút nhấn thực hiện chức năng ghi dữ liệu vào EEPROM.



-Câu lệnh này sẽ thực hiện việc ghi dữ liệu vào EEPROM, có hai tham số cần truyền vào cho hàm đó là địa chỉ EEPROM và byte dữ liệu.



Bước 1: bắt đầu giao thức truyền thông I2C bằng hàm iic\_start();

Bước 2: Gửi địa chỉ 0xA0 với bit cuối cùng 0 điều này cho phép ghi dữ liệu vào EEPROM bằng hàm iic\_write\_byte();

Bước 3: đợi tín hiệu ACK trả về để nhận biết tín hiệu có được gửi thành công bằng hàm iic\_wait\_ack().

Bước 4: Gửi địa chỉ của EPPROM bằng hàm iic\_write\_byte(addr).

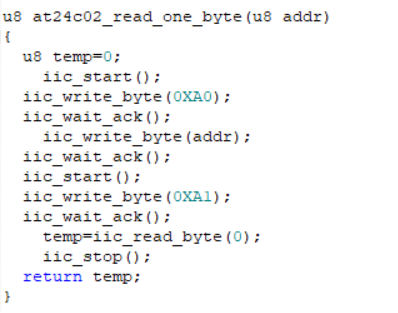
Bước 5: đợi tín hiệu ACK trả về bằng hàm iic\_wait\_ack().

Bước 6: Tiến hành ghi dữ liệu vào bộ nhớ EEPROM bằng hàm iic\_write\_byte(dat).

Bước 7: đợi tín hiệu ACK trả về bằng hàm iic\_wait\_ack();

Bước 8: kết thúc giao thức truyền thông.

-KEY2 nút nhấn thực hiện chức năng đọc dữ liệu từ EEPROM.



Bước 1: Bắt đầu giao thức truyền thông I2C bằng hàm iic\_start().

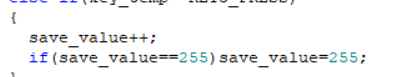
Bước 2: Gửi địa chỉ 0xA1 với bit cuối bằng 1 cho phép đọc dữ liệu từ bộ nhớ EEPROM bằng hàm iic\_write\_byte().

Bước 3: Đợi tín hiệu ACK trả về cho biết đã gửi thành công bằng hàm iic\_wait\_ACK().

Bước 4: Lưu dữ liệu đọc được vào biến temp.

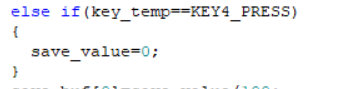
Bước 5: kết thúc quá trình truyền thông iic\_stop().

Key 3:



-Dùng để tăng giá trị biến save\_value cho đến khi đạt giá trị tối đa là 255.

Key 4:

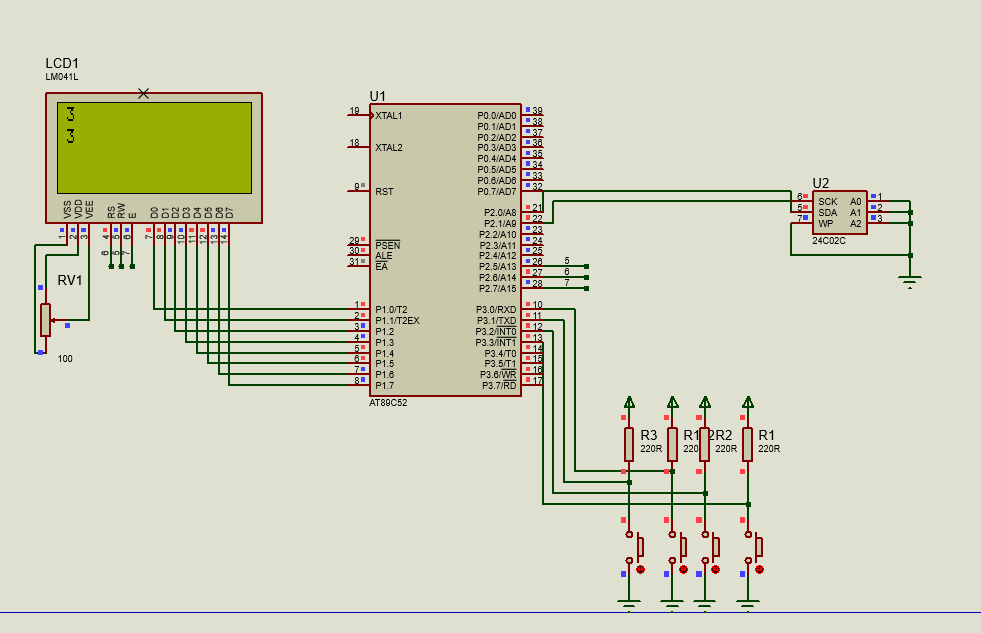
-

Chức năng: dùng để reset giá trị biến save\_value về 0.

**III. Chức năng của hệ thống:**

Được dùng để đọc và ghi dữ liệu vào bộ nhớ EEPROM và hiển thị giá trị lên màn hình LCD.

**IV. Hình ảnh mạch trên mô phỏng Proteus:**



Dòng đầu tiên để hiển thị giá trị dữ liệu.

Dòng thứ hai để hiển thị dữ liệu được lưu trong EEPROM.