ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN - ĐIỆN TỬ

----------♦♦♦----------

BÁO CÁO BÀI TẬP NHÓM CHƯƠNG 4
HỆ THỐNG ĐIỀU KHIỂN NHÚNG

GVHD: Nguyễn Vĩnh Hảo
Lớp: L01
Nhóm: 5

Sinh viên thực hiện:

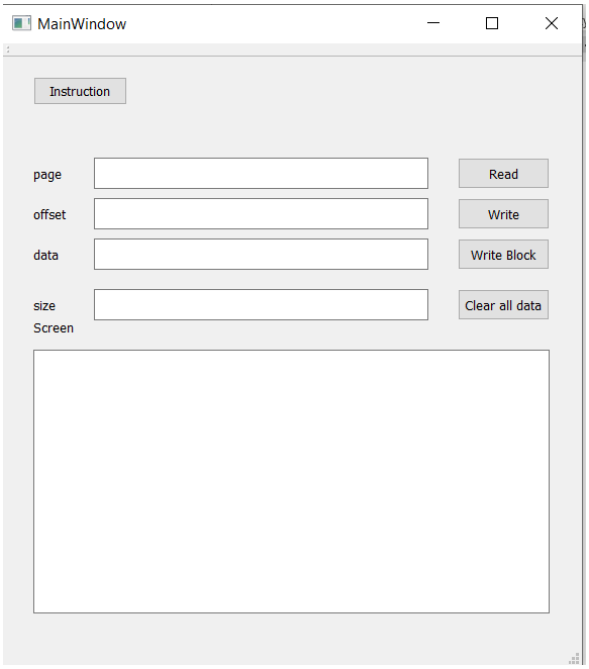
STT	Họ và tên SV	MSSV
1	Trương Hoàng Ngọc	1914360
2	Đoàn Tiến Thông	1915352
3	Nguyễn Minh Mẫn	1910336
4	Đoàn Xuân Khoa	1910262

Tháng 12/2021

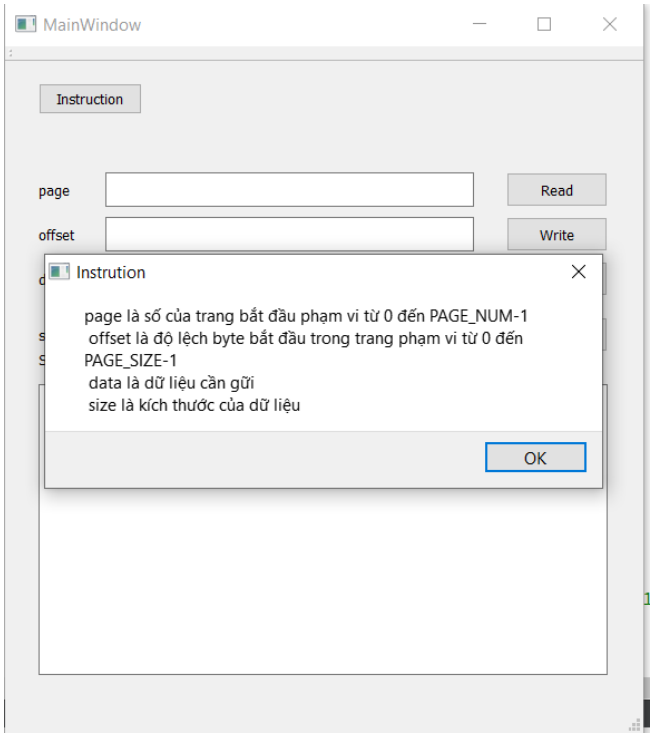
I. THIẾT KẾ GIAO DIỆN

- Giao diện gồm các nút nhấn Read, Write, Write Block, Clear all data, Instruction
  + sssPage, offset, data, size để nhập dữ liệu .
  - page là số của trang bắt đầu. Phạm vi từ 0 đến PAGE_NUM-1.
  - offset là độ lệch byte bắt đầu trong trang. Phạm vi từ 0 đến PAGE_SIZE-1.
  - data là dữ liệu cần gữi.
  - size là kích thước của dữ liệu.

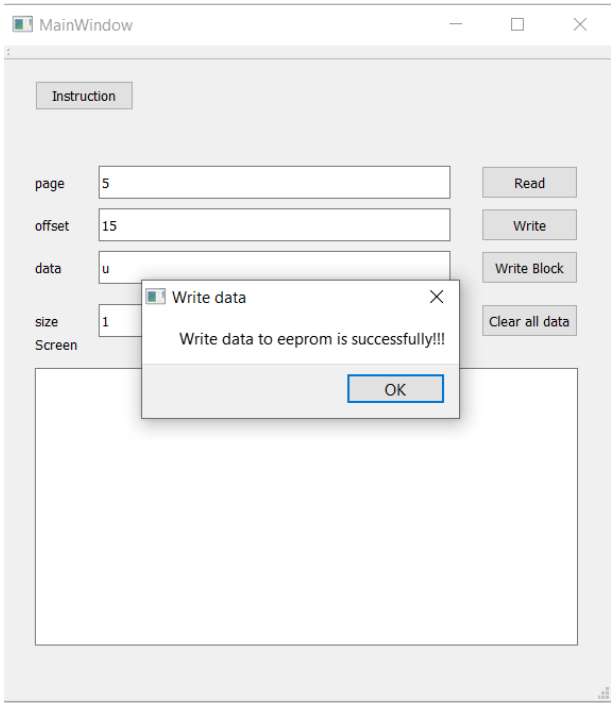
  + Giao điện:



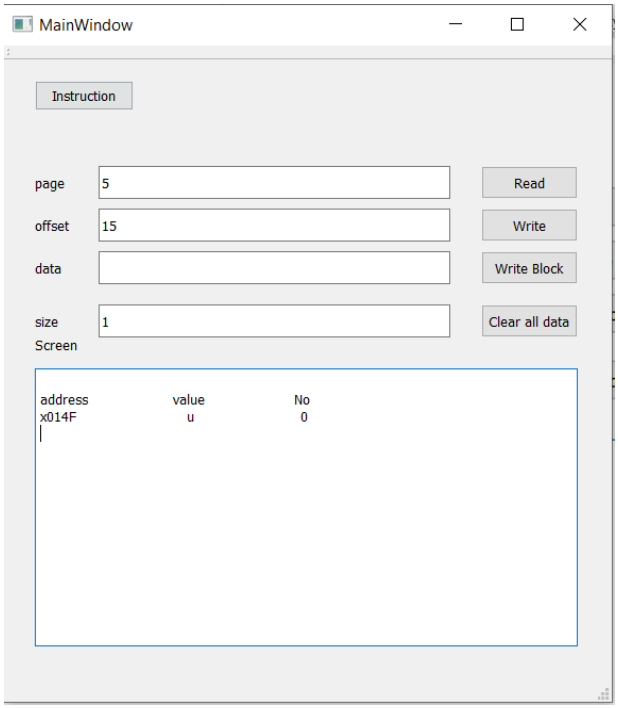
+ Nút intruction



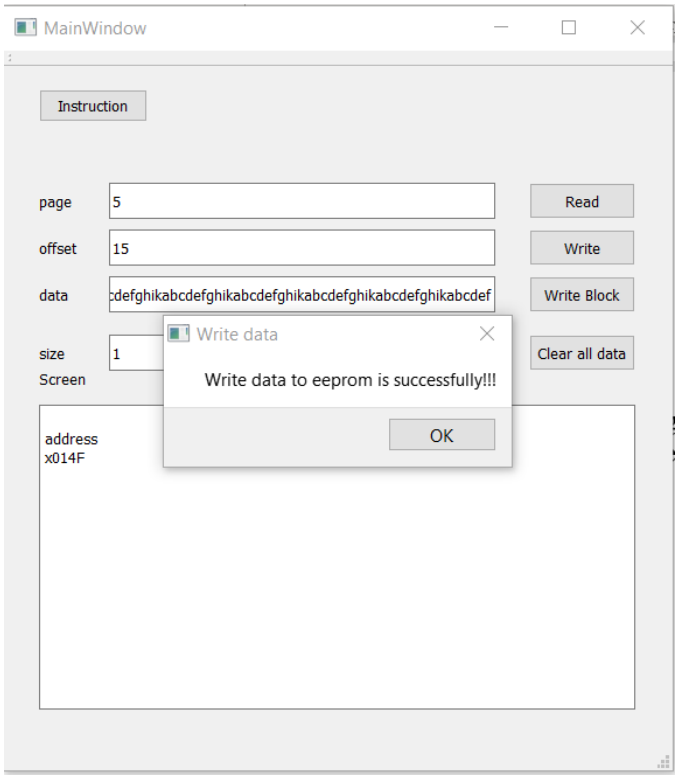
+ Ghi địa chỉ vào một ô nhớ (vd: page = 5, offset = 15,data = u, size =1) sau đó bấm nút Write. Sau khi dữ liệu ghi vào thì hiện lên thông báo "Write data to eeprom is successfully!!!".



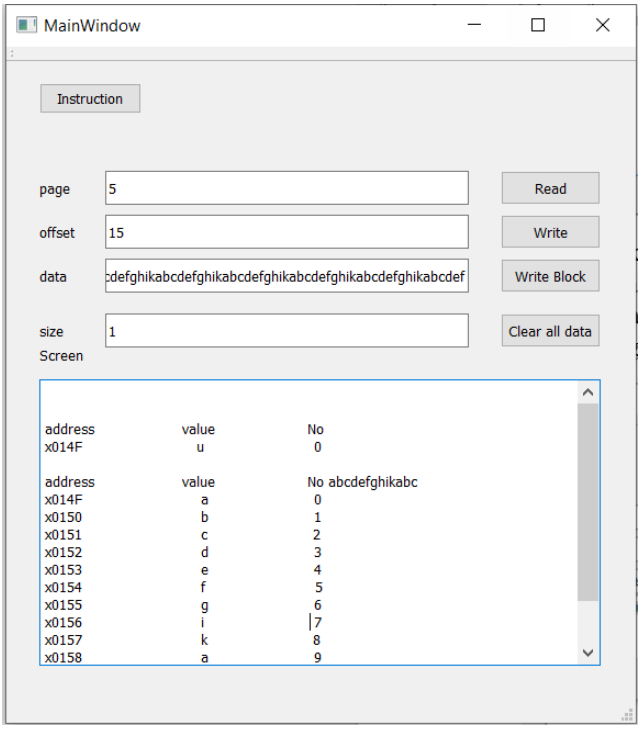
+ Đọc địa chỉ một ô nhớ (vd: page = 5, offset = 15, size =1) sau đó bấm nút Read.



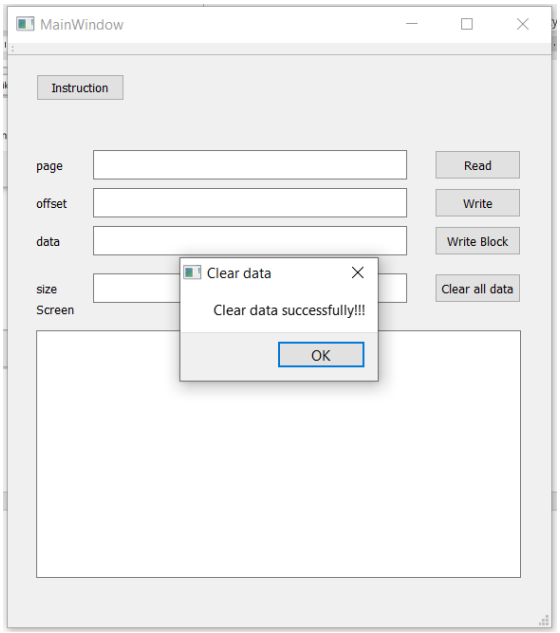
+ Ghi Block 256 bytes (vd: page = 5, offset = 15, size =1, data = abcdefghikabcdefghikabcdefghikabcdefghikabcdefghikabcdefghikabcdefghikabcdefg hikabcdefghikabcdefghikabcdefhikabcdefghikabcdefghikabcdefghikabcdefghikabcdef ghikabcdefghikabcdef).



+ Đọc Block về:



+ **Nút Clear all data** dùng để xóa tất cả dữ liệu.



II. CƠ SỞ LÝ THUYẾT

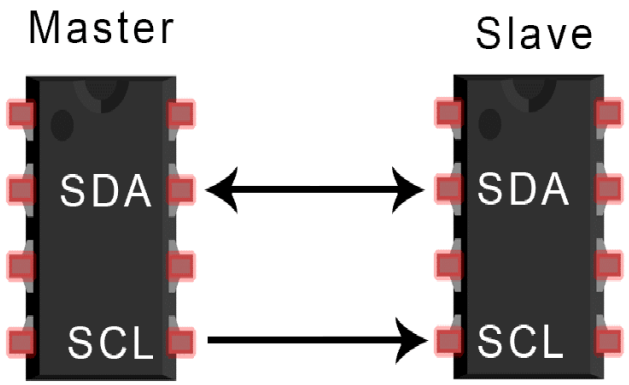
1. Giới thiệu về giao tiếp I2C

I2C là tên viết tắt của cụm từ tiếng anh “Inter-Integrated Circuit”. Nó là một giao thức giao tiếp được phát triển bởi Philips Semiconductors để truyền dữ liệu giữa một bộ xử lý trung tâm với nhiều IC trên cùng một board mạch chỉ sử dụng hai đường truyền tín hiệu.

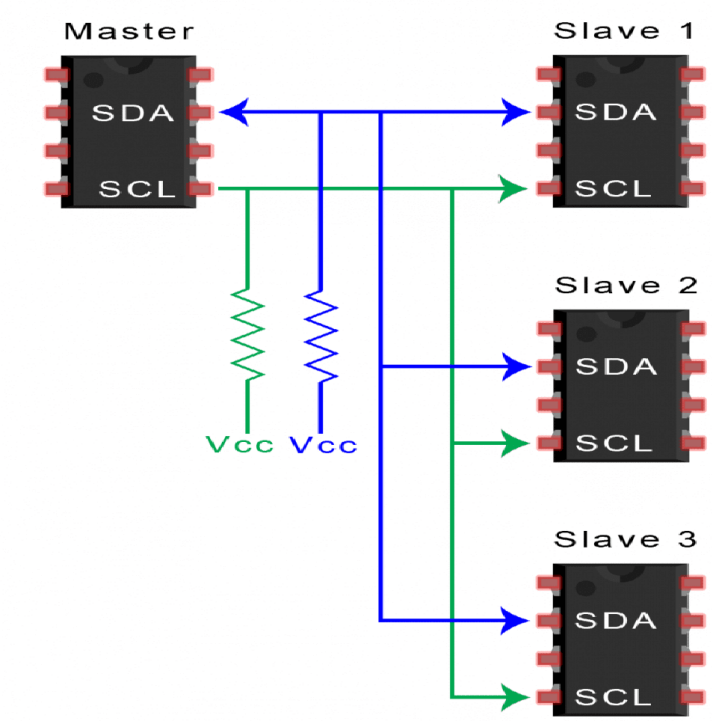
Do tính đơn giản của nó nên loại giao thức này được sử dụng rộng rãi cho giao tiếp giữa vi điều khiển và mảng cảm biến, các thiết bị hiển thị, thiết bị IoT, EEPROMs, v.v …

Đây là một loại giao thức giao tiếp nối tiếp đồng bộ. Nó có nghĩa là các bit dữ liệu được truyền từng bit một theo các khoảng thời gian đều đặn được thiết lập bởi một tín hiệu đồng hồ tham chiếu.

I2C kết hợp các tính năng tốt nhất của SPI và UART. Với I2C, bạn có thể kết nối nhiều slave với một master duy nhất (như SPI) và bạn có thể có nhiều master điều khiển một hoặc nhiều slave. Điều này thực sự hữu ích khi bạn muốn có nhiều hơn một vi điều khiển ghi dữ liệu vào một thẻ nhớ duy nhất hoặc hiển thị văn bản trên một màn hình LCD.



Một Master với một Slave



Một Master với một Slave

Giống như giao tiếp UART, I2C chỉ sử dụng hai dây để truyền dữ liệu giữa các thiết bị:

SDA (Serial Data) - đường truyền cho master và slave để gửi và nhận dữ liệu.

SCL (Serial Clock) - đường mang tín hiệu xung nhịp.

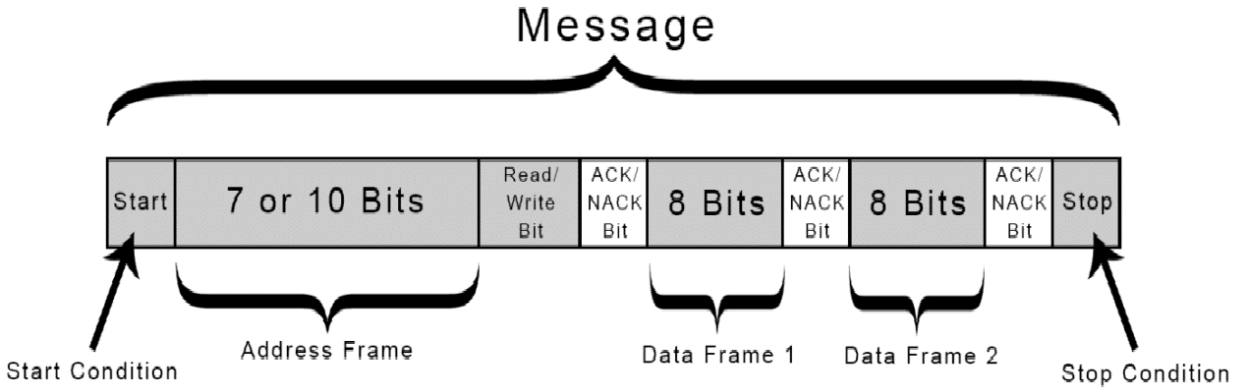
I2C là một giao thức truyền thông nối tiếp, vì vậy dữ liệu được truyền từng bit dọc theo một đường duy nhất (đường SDA).

Giống như SPI, I2C là đồng bộ, do đó đầu ra của các bit được đồng bộ hóa với việc lấy mẫu các bit bởi một tín hiệu xung nhịp được chia sẻ giữa master và slave. Tín hiệu xung nhịp luôn được điều khiển bởi master.

Hoạt động:

Với I2C, dữ liệu được truyền trong các tin nhắn. Tin nhắn được chia thành các khung dữ liệu. Mỗi tin nhắn có một khung địa chỉ chứa địa chỉ nhị phân của địa chỉ slave

và một hoặc nhiều khung dữ liệu chứa dữ liệu đang được truyền. Thông điệp cũng bao gồm điều kiện khởi động và điều kiện dừng, các bit đọc / ghi và các bit ACK / NACK giữa mỗi khung dữ liệu:



Điều kiện khởi động: Đường SDA chuyển từ mức điện áp cao xuống mức điện áp thấp trước khi đường SCL chuyển từ mức cao xuống mức thấp.

Điều kiện dừng: Đường SDA chuyển từ mức điện áp thấp sang mức điện áp cao sau khi đường SCL chuyển từ mức thấp lên mức cao.

Khung địa chỉ: Một chuỗi 7 hoặc 10 bit duy nhất cho mỗi slave để xác định slave khi master muốn giao tiếp với nó.

Bit Đọc / Ghi: Một bit duy nhất chỉ định master đang gửi dữ liệu đến slave (mức điện áp thấp) hay yêu cầu dữ liệu từ nó (mức điện áp cao).

Bit ACK / NACK: Mỗi khung trong một tin nhắn được theo sau bởi một bit xác nhận / không xác nhận. Nếu một khung địa chỉ hoặc khung dữ liệu được nhận thành công, một bit ACK sẽ được trả lại cho thiết bị gửi từ thiết bị nhận.

Các bước truyền dữ liệu I2C

- Master gửi điều kiện khởi động đến mọi slave được kết nối bằng cách chuyển đường SDA từ mức điện áp cao sang mức điện áp thấp trước khi chuyển đường SCL từ mức cao xuống mức thấp.
- Master gửi cho mỗi slave địa chỉ 7 hoặc 10 bit của slave mà nó muốn giao tiếp, cùng với bit đọc / ghi.
- Mỗi slave sẽ so sánh địa chỉ được gửi từ master với địa chỉ của chính nó. Nếu địa chỉ trùng khớp, slave sẽ trả về một bit ACK bằng cách kéo dòng SDA xuống thấp

cho một bit. Nếu địa chỉ từ master không khớp với địa chỉ của slave, slave rời khỏi đường SDA cao.

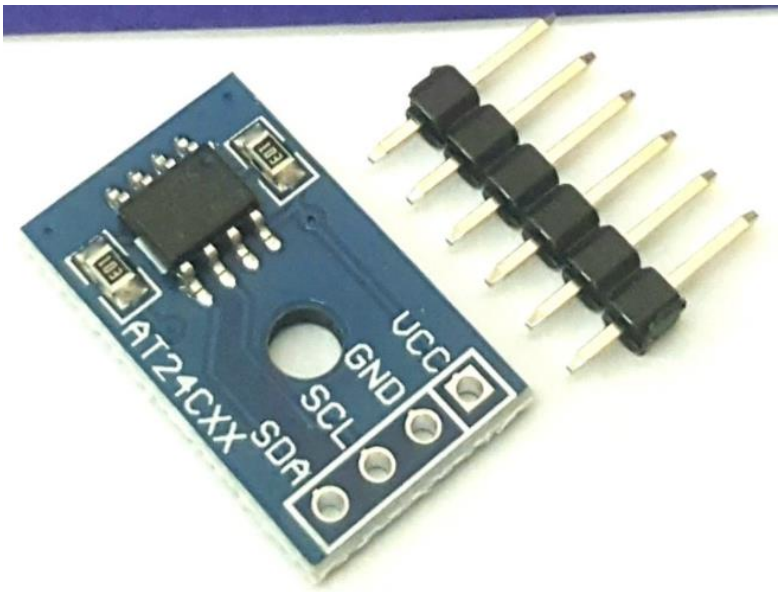
- Master gửi hoặc nhận khung dữ liệu.
- Sau khi mỗi khung dữ liệu được chuyển, thiết bị nhận trả về một bit ACK khác cho thiết bị gửi để xác nhận đã nhận thành công khung.
- Để dừng truyền dữ liệu, master gửi điều kiện dừng đến slave bằng cách chuyển đổi mức cao SCL trước khi chuyển mức cao SDA.

2. EEPROM

EEPROM (Bộ nhớ chỉ đỏ có thể lập trình được bằng điện) Đây là một loại bộ nhớ ROM, tức là bộ nhớ không bay hơi, trong đó dữ liệu sẽ được lưu trữ vĩnh viễn, ngay cả khi nguồn điện bị loại bỏ. Điều đó đặt chúng ở phía bên kia của RAM (Bộ nhớ truy cập ngẫu nhiên), bộ nhớ này sẽ mất tất cả dữ liệu của chúng khi chúng không được cấp nguồn.

Ưu điểm của bộ nhớ EEPROM, ngoài dữ liệu được lưu trữ không thay đổi, thì còn có thể đọc dữ liệu từ nó và cũng có thể xóa và ghi dữ liệu vào nó. Để xóa dữ liệu, cần có điện áp tương đối cao và các EEPROM đời đầu cần có nguồn điện áp cao bên ngoài. Các phiên bản sau của chip nhớ này đã có thêm nguồn cấp cho EEPROM và kết hợp nguồn điện áp cao trong chip EEPROM. Bằng cách này, thiết bị bộ nhớ có thể chạy từ một nguồn, do đó giảm đáng kể chi phí của một mạch tổng thể sử dụng EEPROM và đơn giản hóa thiết kế.

Mạch EEPROM 24C256 Giao Tiếp I2C

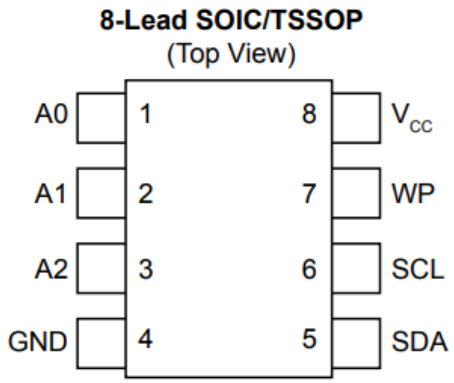


Mạch EEPROM 24C256 giao tiếp I2C được sử dụng cho các ứng dụng cần lưu trữ dữ liệu nhỏ trên vi điều khiển với bộ nhớ 262.144 bits, đặc điểm của bộ nhớ EEPROM là có thể lưu trữ dữ liệu ngay cả khi mất điện và khả năng ghi xóa lên đến 1 triệu lần với thời gian lưu trữ dữ liệu lên đến 40 năm (theo datasheet), mạch EEPROM 24C256 sử dụng giao tiếp I2C nên rất dễ giao tiếp và sử dụng với chỉ 2 chân giao tiếp, mạch có thiết kế nhỏ gọn, linh kiện chất lượng tốt, độ bền cao.

Thông số kỹ thuật:

- IC chính: AT24C256
- Điện áp sử dụng: 2.7~5.5VDC
- Điện áp giao tiếp: TTL 2.7~5.5VDC.
- Chuẩn giao tiếp: I2C
- Kiểu bộ nhớ: EEPROM
- Dung lượng bộ nhớ: 262.144 bits
- Tích hợp trở kéo cho 2 chân giao tiếp I2C trên mạch.
- Kích thước: 19 x 11mm

IC AT24C256



Các chân được mô tả dưới bảng sau:

Name	8-Lead SOIC	8-Lead TSSOP	8-Pad UDFN(1)	8-Ball VFBGA	Function
A0(2)	1	1	1	1	Device Address Input
A1(2)	2	2	2	2	Device Address Input
A2(2)	3	3	3	3	Device Address Input
GND	4	4	4	4	Ground
SDA	5	5	5	5	Serial Data
SCL	6	6	6	6	Serial Clock
WP(2)	7	7	7	7	Write-Protect
VCC	8	8	8	8	Device Power Supply

AT24C256 được tổ chức nội bộ dưới dạng 512 trang với 64 byte mỗi trang.

Bảng 6.1 là byte Device Addressing

Table 6-1. Device Addressing

Package	Device Type Identifier				Hardware Slave Address Bits			R/$\overline{W}$ Select
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SOIC, TSSOP, UDFN, VFBGA	1	0	1	0	A2	A1	A0	R/$\overline{W}$

Đối với tất cả các hoạt động ngoại trừ việc đọc địa chỉ hiện tại, hai byte Word Address 8 bit phải được truyền tới thiết bị ngay sau byte Device Addressing. Các byte Word Address bao gồm địa chỉ từ của mảng bộ nhớ 15 bit và được sử dụng để chỉ định vị trí byte nào trong EEPROM để bắt đầu đọc hoặc ghi.

First Word Address Byte chứa bảy bit Quan trọng nhất của địa chỉ từ (A14 đến A8) ở các vị trí bit từ sáu đến không, như được thấy trong Bảng 6-2. Bit 7 của First Word Address Byte là bit "không quan tâm" vì nó nằm ngoài phạm vi 256 - Kbit có thể định địa chỉ. Sau khi hoàn thành First Word Address Byte, AT24C256C sẽ trả về một ACK.

Table 6-2. First Word Address Byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	A14	A13	A12	A11	A10	A9	A8

Tiếp theo, Second Word Address Byte được gửi đến thiết bị cung cấp tám bit còn lại của địa chỉ từ (từ A7 đến A0). Sau khi hoàn thành Second Word Address Byte, AT24C256C sẽ trả về một ACK. Xem Bảng 6-3 để xem xét các vị trí bit này..

Table 6-3. Second Word Address Byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A7	A6	A5	A4	A3	A2	A1	A0

Ghi:

Tất cả các hoạt động ghi cho AT24C256C bắt đầu bằng việc gửi chính điều kiện Bắt đầu, tiếp theo là byte Device Addressing với bit R / W được đặt thành logic '0', sau đó là Word Address Byte. (Các) giá trị dữ liệu được ghi vào thiết bị ngay sau Word Address Byte.

Đọc:

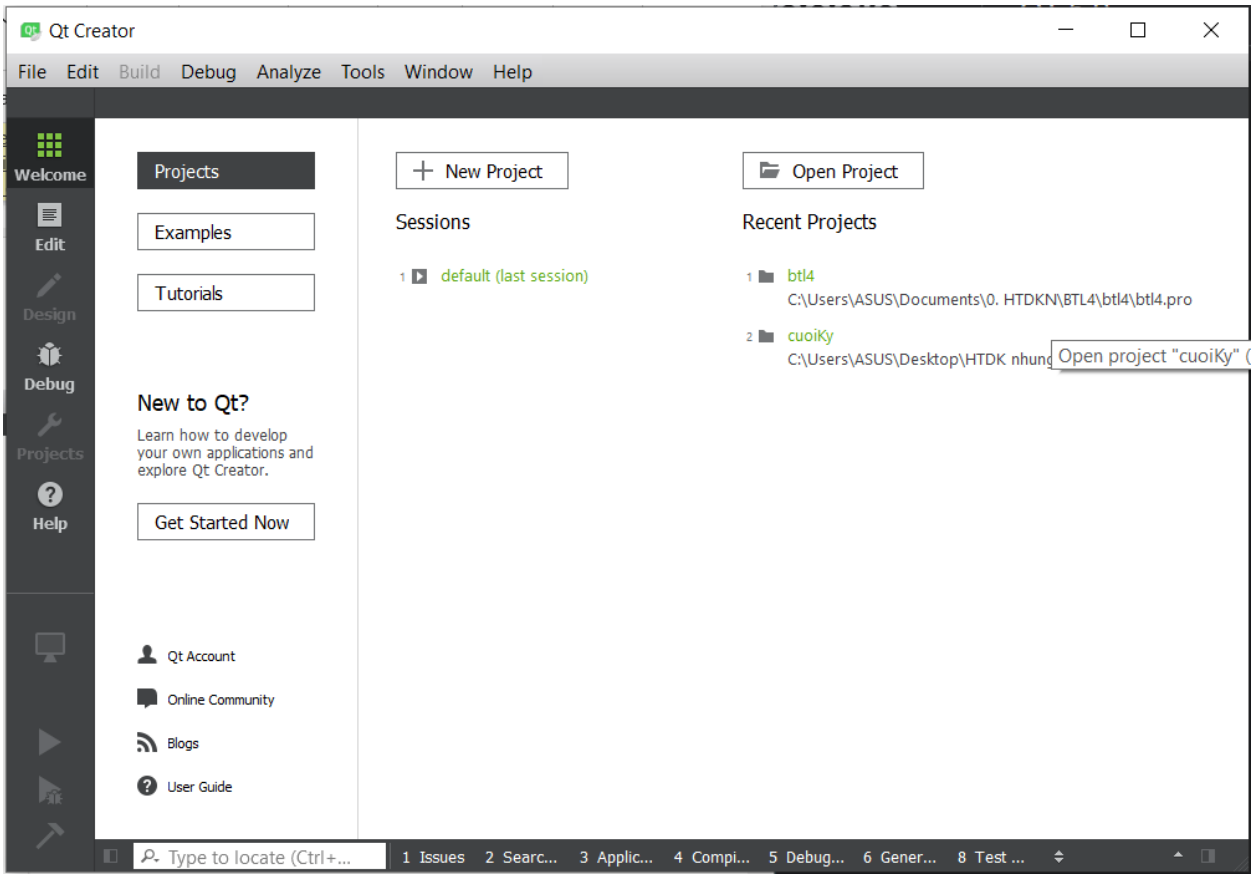
Các thao tác đọc được bắt đầu giống như các thao tác ghi với ngoại lệ là bit Chọn đọc / ghi trong byte địa chỉ vùng nhớ phải là logic '1'. Có ba thao tác đọc:

• Đọc địa chỉ hiện tại

• Đọc địa chỉ ngẫu nhiên

• Đọc tuần tự

3. PHẦN MỀM QT 5



Phần mềm QT 5.9

Qt là 1 ứng dụng đa nền tảng (phát triển ứng dụng trên các platform khác nhau: Windows, Linux/X11, iOS, Android, ...), 1 framework UI (cho phép tạo ứng dụng với giao diện đồ hoạ). Ứng dụng tạo ra bởi Qt có thể chạy trên desktop, mobile hay trên các hệ thống nhúng.

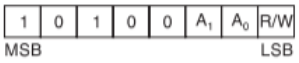
Hiện nay, Qt được xây dựng có cả bản thương mại và open source. Qt cho phép tạo giao diện 1 cách đơn giản là kéo thả.

Qt cho phép hiện thực hàm tạo và xử lý sự kiện bằng C++, và xây dựng nhiều class để hỗ trợ cho việc phát triển ứng dụng với giao diện đồ hoạ.

Từ các đặc điểm của phần mềm, tiến hành xây dựng dao diện đồ họa (GUI) cho dự án.

4. PHẦN CODE NHÚNG

Figure 7. Device Address



1. Địa chỉ của eeprom 24c256 là 0xA0.

Memory Organization

AT24C128/256, 128K/256K SERIAL EEPROM: The 128K/256K is internally organized as 256/512 pages of 64-bytes each. Random word addressing requires a 14/15-bit data word address.

2. Eeprom 24c256 có 512 trang, mỗi trang có 64 bytes.

```
// EEPROM ADDRESS (8bits)
#define EEPROM_ADDR 0xA0

// Define the Page Size and number of pages
#define PAGE_SIZE 64      // in Bytes
#define PAGE_NUM  512     // number of pages
```

3. Tạo ra các hàm EEPROM_Write, EEPROM_Read, EEPROM_PageErase

+ Hàm EEPROM_Write dùng để viết dữ liệu vào EEPROM.

void EEPROM_Write (uint16_t page, uint16_t offset, uint8_t *data, uint16_t size);

Hàm này nhận các đối số sau

- @ page là số của trang bắt đầu. Phạm vi từ 0 đến PAGE_NUM-1
- @ offset là độ lệch byte bắt đầu trong trang. Phạm vi từ 0 đến PAGE_SIZE-1
- @ data là con trỏ tới dữ liệu để ghi theo byte
- @ size là kích thước của dữ liệu

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	A14	A13	A12	A11	A10	A9	A8

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A7	A6	A5	A4	A3	A2	A1	A0

Page Address -> 0 to 511

Byte Address -> 0 to 63

Code hàm EEPROM_Write

```
103  void EEPROM_Write (uint16_t page, uint16_t offset, uint8_t *data, uint16_t size)
104  {
105
106    // Find out the number of bit, where the page addressing starts
107    int paddrposition = log(PAGE_SIZE)/log(2);
108
109    // calculate the start page and the end page
110    uint16_t startPage = page;
111    uint16_t endPage = page + ((size+offset)/PAGE_SIZE);
112
113    // number of pages to be written
114    uint16_t numofpages = (endPage-startPage) + 1;
115    uint16_t pos=0;
116
117    // write the data
118    for (int i=0; i<numofpages; i++)
119    {
120      /* calculate the address of the memory location
121       * Here we add the page address with the byte address
122       */
123      uint16_t MemAddress = startPage<<paddrposition | offset;
124      uint16_t bytesremaining = bytestowrite(size, offset);  // calculate the remaining bytes to be written
125
126      HAL_I2C_Mem_Write(EEPROM_I2C, EEPROM_ADDR, MemAddress, 2, &data[pos], bytesremaining, 1000);  // write the data to the EEPROM
127
128      startPage += 1;  // increment the page, so that a new page address can be selected for further write
129      offset=0;   // since we will be writing to a new page, so offset will be 0
130      size = size-bytesremaining;  // reduce the size of the bytes
131      pos += bytesremaining;  // update the position for the data buffer
132
133      HAL_Delay (5);  // Write cycle delay (5ms)
134    }
135  }
136
```

- Ở đây, trước tiên chúng ta tìm số bit , nơi địa chỉ trang bắt đầu sử dụng log (PAGE_SIZE) / log (2) . Bit này sẽ phụ thuộc vào Kích thước trang của bạn.
- Sau đó đặt trang bắt đầu và trang kết thúc, đồng thời tính toán số trang sẽ được viết.
- Bây giờ bên trong vòng lặp for, chúng ta sẽ tính toán vị trí bộ nhớ Để làm điều này, chúng ta sẽ dịch chuyển trang bắt đầu theo vị trí bit của trang (6, nếu kích thước trang là 64 byte) Ví dụ: nếu trang bắt đầu là 5, nó sẽ được chuyển bằng 6 tức là 5 << 6, chúng tôi sẽ thêm phần bù vào đây để thiết lập vị trí bộ nhớ cuối cùng.
- Tiếp theo tính toán các byte còn lại
- Bây giờ ta sẽ ghi dữ liệu vào EEPROM bằng hàm HAL_I2C_MEM_WRITE().
- Sau đó sẽ cập nhật các thông số ngay bây giờ và cuối cùng là độ trễ 5 ms cho chu kỳ ghi 5 ms này là thời gian cần thiết để EEPROM ghi dữ liệu vào bộ nhớ.

+ Hàm EEPROM_Read dùng để đọc dữ liệu từ eeprom có cấu trúc tương tượng như EEPROM_Write

Code hàm EEPROM_Read

```
143  void EEPROM_Read (uint16_t page, uint16_t offset, uint8_t *data, uint16_t size)
144  {
145    int paddrposition = log(PAGE_SIZE)/log(2);
146
147    uint16_t startPage = page;
148    uint16_t endPage = page + ((size+offset)/PAGE_SIZE);
149
150    uint16_t numofpages = (endPage-startPage) + 1;
151    uint16_t pos=0;
152
153    for (int i=0; i<numofpages; i++)
154    {
155      uint16_t MemAddress = startPage<<paddrposition | offset;
156      uint16_t bytesremaining = bytestowrite(size, offset);
157      HAL_I2C_Mem_Read(EEPROM_I2C, EEPROM_ADDR, MemAddress, 2, &data[pos], bytesremaining, 1000);
158      startPage += 1;
159      offset=0;
160      size = size-bytesremaining;
161      pos += bytesremaining;
162    }
163  }
```

+ Hàm EEPROM_PageErase dùng để xóa 1 trang trong EEPROM

```
165  void EEPROM_PageErase (uint16_t page)
166  {
167    // calculate the memory address based on the page number
168    int paddrposition = log(PAGE_SIZE)/log(2);
169    uint16_t MemAddress = page<<paddrposition;
170
171    // create a buffer to store the reset values
172    uint8_t data[PAGE_SIZE];
173    memset(data,0x00,PAGE_SIZE);
174
175    // write the data to the EEPROM
176    HAL_I2C_Mem_Write(EEPROM_I2C, EEPROM_ADDR, MemAddress, 2, data, PAGE_SIZE, 1000);
177
178    HAL_Delay (5);  // write cycle delay
179  }
180
```

- Ở đây, trước tiên ta cũng sẽ tính toán vị trí bộ nhớ bằng cách sử dụng trang.
- Sau đó tạo một bộ đệm, nơi lưu trữ dữ liệu và ghi dữ liệu vào vị trí bộ nhớ thao tác ghi này được giới hạn trong một trang duy nhất.
- Trong trường hợp muốn xóa toàn bộ ROM, hãy gọi hàm này theo vòng lặp for.