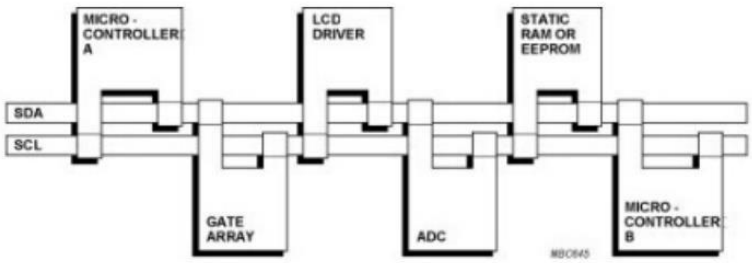
Bài tập lớn môn Kiến trúc máy tính và mạng truyền thông công nghiệp: Giao tiếp I2C (Master – Slave)

I. Giới thiệu chung về I2C(Master – Slave)

Phương pháp Master – Slave (chủ - tớ), một trạm chủ (master) có trách nhiệm chủ động phân chia quyền truy cập bus cho các trạm tớ (slave). Các trạm tớ đóng vai trò bị động chỉ có quyền truy cập bus và gửi tín hiệu đi kh có yêu cầu. Trạm chủ có thể dùng phương pháp hỏi tuần tự theo chu kỳ để kiểm soát toàn bộ hệ thống. Nhờ vậy các trạm tớ có thể gửi các dữ liệu thu thập được từ quá trình kỹ thuật gửi đến trạm chủ cũng như nhận được các thông tin điều khiển từ trạm chủ. Và chuẩn giao tiếp I2C là một chuẩn giao tiếp sử dụng phương pháp này.

Ngày nay trong các hệ thống điện tử hiện đại, rất nhiều IC hay thiết bị ngoại vi cần phải giao tiếp với các IC hay thiết bị khác giao tiếp với thế giới bên ngoài. Với mục tiêu đạt được hiệu quả cho phần cứng tốt nhất với mạch điện đơn giản, Phillips đã phát triển một chuẩn giao tiếp nối tiếp 2 dây được gọi là I2C. I2C là tên viết tắt của cụm từ Inter - Intergrated Circuit Bus giao tiếp giữa các IC với nhau.

I2C mặc dù được phát triển bởi Philips, nhưng nó đã được rất nhiều nhà sản xuất IC trên thế giới sử dụng. I2C trở thành một chuẩn công nghiệp cho các giao tiếp điều khiển, có thể kể ra đây một vài tên tuổi ngoài Philips như: Texas Intrument (TI), Maxim-Dallas, analog Device, National Semiconductor Bus I2C được sử dụng làm bus giao tiếp ngoại vi cho rất nhiều loại IC khác nhau như các loại. Vi điều khiển 8051, PIC , AVR, ARM, chíp nhớ như RAM tĩnh (Static Ram), EEPROM, bộ chuyển đổi tương tự số (ADC), số tương tự (DAC), IC điều khiển LCD, LED…

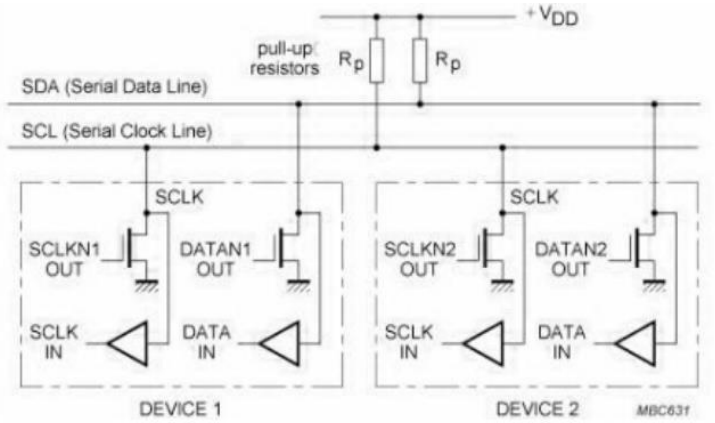


Hình 1.1. BUS I2C và các thiết bị ngoại vi

1. Đặc điểm giao tiếp I2C

Một giao tiếp I2C gồm có 2 dây: Serial Data (SDA) và Serial Clock (SCL). SDA là đường truyền dữ liệu 2 hướng, còn SCL là đường truyền xung đồng hồ

và chỉ theo một hướng. Như hình vẽ trên, khi một thiết bị ngoại vi kết nối vào đường I2C thì chân SDA của nó sẽ nối với dây SDA của bus, chân SCL sẽ nối với dây SCL.



Hình 1.2. Kết nối thiết bị vào bus I2C ở chế độ chuẩn (Standard mode) và chế độ nhanh (Fast mode)

Mỗi dây SDA hay SCL đều được nối với điện áp dương của nguồn cấp thông qua một điện trở kéo lên (pull-up resistor). Sự cần thiết của các điện trở kéo này là vì chân giao tiếp I2C của các thiết bị ngoại vi thường là dạng cực máng hở (open-drain or open - collector). Giá trị của các điện trở này khác nhau tùy vào từng thiết bị và chuẩn giao tiếp thường dao động trong khoảng 1KΩ đến 4.7KΩ.

Nhìn lại hình 1.1, ta thấy có rất nhiều thiết bị (IC) cùng được kết nối vào một bus I2C, tuy nhiên sẽ không xảy ra chuyện nhầm lẫn giữa các thiết bị, bởi mỗi thiết bị sẽ được nhận ra bởi một địa chỉ duy nhất với một quan hệ chủ/tớ tồn tại trong suốt thời gian kết nối. Mỗi thiết bị có thể hoạt đông như là thiết bị nhận dữ liệu hay có thể vừa truyền vừa nhận. Hoạt động truyền hay nhận còn tùy thuộc vào việc thiết bị đó là chủ chủ (master) hay tớ (slave). Một thiết bị hay một IC khi kết nối với bus I2C, ngoài một địa chỉ (duy nhất) để phân biệt, nó còn được cấu hình là thiết bị chủ (master) hay tớ (slave). Tại sao lại có sự phân biệt này ? Đó là vì trên một bus I2C thì quyền điều khiển thuộc về thiết

bị chủ (master). Thiết bị nắm vai trò tạo xung đồng hồ cho toàn hệ thống, khi giữa hai thiết bị chủ/tớ giao tiếp thì thiết bị chủ có nhiệm vụ tạo xung đồng hồ và quản lý địa chỉ của thiết bị tớ trong suốt quá trình giao tiếp. Thiết bị chủ giữ vai trò chủ động, còn thiết bị tớ giữ vai trò bị động trong viêc giao tiếp.

Hình 1.3. Truyền nhận dữ liệu giữa chủ/tớ

Nhìn hình trên ta thấy xung đồng hồ chỉ có một hướng từ chủ đến tớ, còn luồng dữ liệu có thể đi theo hai hướng, từ chủ đến tớ hay ngược lại tớ đến chủ. Về dữ liệu truyền trên bus I2C, một bus I2C chuẩn truyền 8-bit dữ liệu có hướng trên đường truyền với tốc độ là 100Kbits/ s – Chế độ chuẩn (Standard mode). Tốc độ
truyền có thể lên tới 400Kbits/s – Chế độ nhanh (Fast mode) và cao nhất là 3,4Mbits/s – Chế độ cao tốc (High-speed mode).

Một bus I2C có thể hoạt động ở nhiều chế độ khác nhau:

- Một chủ một tớ (one master – one slave)
- Một chủ nhiều tớ (one master – multi slave)
- Nhiều chủ nhiều tớ (Multi master – multi slave)

Dù ở chế độ nào, một giao tiếp I2C đều dựa vào quan hệ chủ/tớ. Giả thiết một một thiết bị A muốn gửi dữ liệu đến thiết bị B, quá trình được thực hiện như sau

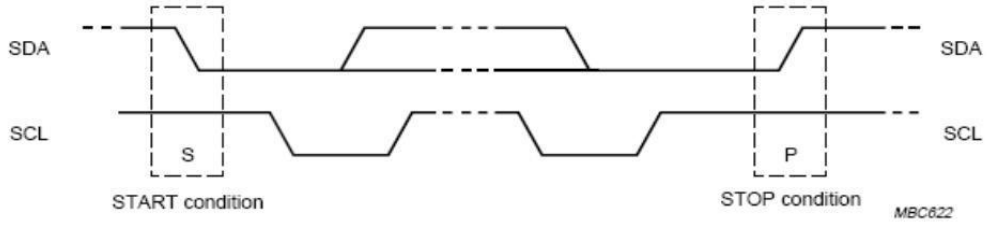
- Thiết bị A (Chủ) xác định đúng địa chỉ của thiết bị B (tớ), cùng với việc xác định địa chỉ, thiết bị A sẽ quyết định việc đọc hay ghi vào thiết bị tớ.
- Thiết bị A gửi dữ liệu tới thiết bị B.
- Thiết bị A kết thúc quá trình truyền dữ liệu.

Khi A muốn nhận dữ liệu từ B, quá trình diễn ra như trên, chỉ khác là A sẽ nhận dữ liệu từ B. Trong giao tiếp này, A là chủ còn B vẫn là tớ. Chi tiết việc thiết lập giao tiếp với một thiết bị nào đó trong mạng I2C.

2: Bit Start và Stop

START là điều kiện khởi đầu, báo hiệu bắt đầu của giao tiếp, còn STOP hiệu kết thúc một giao tiếp. Hình dưới đây mô tả điều kiện START và báo
STOP. Ban đầu khi chưa thực hiện quá trình giao tiếp, cả hai đường SDA và

SCL đều ở mức cao (SDA = SCL= HIGH). Lúc này bus I2C được coi là dỗi ("bus free"), sẵn sàng cho một giao tiếp. Hai điều kiện START và STOP là không thể thiếu trong việc giao tiếp giữa các thiết bị I2C với nhau.



Hình 1.4. Điều kiện START và STOP của bus I2C

Điều kiện START: một sự chuyển đổi trạng thái từ cao xuống thấp trên đường SDA trong khi đường SCL đang ở mức cao (cao = 1; thấp = 0) báo hiệu một điều kiện START

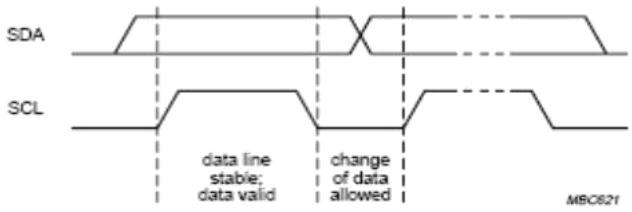
Điều kiện STOP: Một sự chuyển đổi trạng thái từ mức thấp lên cao trên đường SDA trong khi đường SCL đang ở mức cao.

Cả hai điều kiện START và STOP đều được tạo ra bởi thiết bị chủ. Sau tín hiệu START, bus I2C coi như đang trong trang thái làm việc (busy). Bus I2C sẽ rỗi, sẵn sàng cho một giao tiếp mới sau tín hiệu STOP từ phía thiết bị chủ.

Sau khi có một điều kiện START, , trong quá trình giao tiếp, khi có một tín hiệu START được lặp lại thay vì một tín hiệu STOP thì bus I2C vẫn tiếp tục trong trạng thái bận. Tín hiệu START và lặp lại START đều có chức năng giống nhau là khởi tạo một giao tiếp.

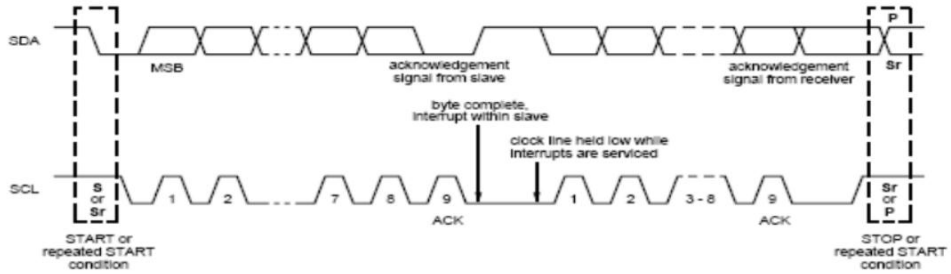
3. Định dạng dữ liệu truyền

Dữ liệu được truyền trên bus I2C theo từng bit, bit dữ liệu được truyền đi tại mỗi sườn dương của xung đồng hồ trên dây SCL, quá trình thay đổi bit dữ liệu xảy ra khi SCL đang ở mức thấp.

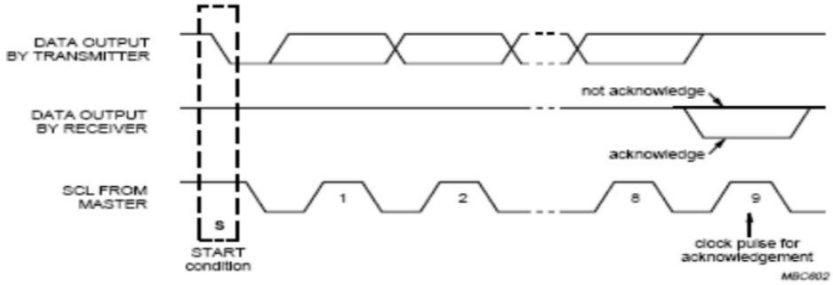


Hình 1.5. Quá trình truyền 1 bit dữ liệu

Mỗi byte dữ liệu được truyền có độ dài là 8 bits. Số lượng byte có thể truyền trong một lần là không hạn chế. Mỗi byte được truyền đi theo sau là một bit ACK để báo hiệu đã nhận dữ liệu. Bit có trọng số cao nhất (MSB) sẽ được truyền đi đầu tiên, các bít sẽ được truyền đi lần lượt. Sau 8 xung clock trên dây SCL, 8 bit dữ liệu đã được truyền đi. Lúc này thiết bị nhận, sau khi đã nhận đủ đủ 8 bít dữ liệu sẽ kéo SDA xuống mức thấp tạo một xung ACK ứng với xung clock thứ 9 trên dây SDA để báo hiệu đã nhận đủ 8 bit. Thiết bị truyền khi nhận được bit ACK sẽ tiếp tục thực hiện quá trình truyền hoặc kết thúc.



Hình 1.6. Dữ liệu truyền trên bus I2C



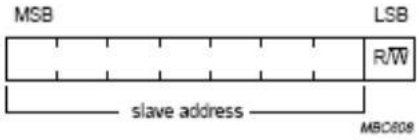
Hình 1.7. Bit ACK trên bus I2C

Một byte truyền đi có kèm theo bit ACK là điều kiên bắt buộc, nhằm đảm bảo cho quá trình truyền nhận được diễn ra chính xác. Khi không nhận được đúng

địa hay khi muốn kết thúc quá trình giao tiếp, thiết bị nhận sẽ gửi một xung Not-ACK(SDA ở mức cao) để báo cho thiết bị chủ biết, thiết bị chủ sẽ tạo xung xung STOP để kết thúc hay lặp lại một xung START để bắt đầu quá trình mới.

4. Định dạng địa chỉ thiết bị

Mỗi thiết bị ngoại vi tham gia vào bus i2c đều có một địa chỉ duy nhất, nhằm phân biệt giữa các thiết bị với nhau. Độ dài địa chỉ là 7 – bit, điều đó có nghĩa là trên một bus I2C ta có thể phân biệt tối đa 128 thiết bị. Khi thiết bị chủ muốn giao tiếp với ngoại vi nào trên bus I2C, nó sẽ gửi 7 bit địa chỉ của thiết bị đó ra bus ngay sau xung START. Byte đầu tiên được gửi sẽ bao gồm 7 bit địa chỉ và một bít thứ 8 điều khiển hướng truyền đó mà có sự phản hồi tương ứng đến con chủ.

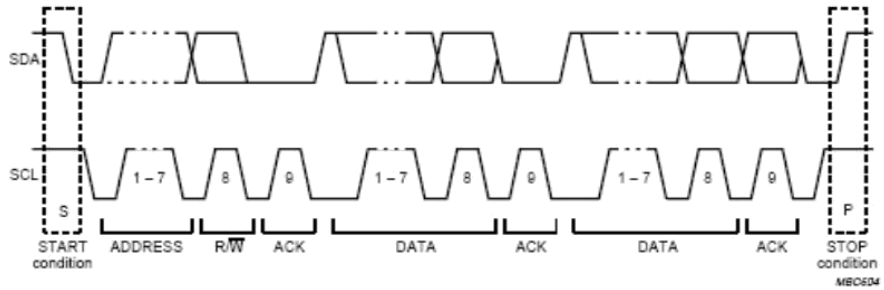


Hình 1.8. Cấu trúc byte dữ liệu đầu tiên

Mỗi một thiết bị ngoại vi sẽ có một địa chỉ riêng do nhà sản xuất ra nó quy định. Địa chỉ đó có thể là cố định hay thay đổi. Riêng bit điều khiển hướng sẽ quy định chiều truyền dữ liệu. Nếu bit này bằng "0" có nghĩa là byte dữ liệu tiếp theo sau sẽ được truyền từ chủ đến tớ, còn ngược lại nếu bằng "1" thì các byte đầu tiên sẽ là dữ liệu từ tớ gửi đến chủ. Việc thiết lập giá trị cho bit này do con chủ thi hành, con tớ sẽ tùy theo giá trị đó mà có sự phản hồi tương ứng đến con chủ. Việc thiết lập giá trị cho bit này do con chủ thi hành, con tớ sẽ tùy theo giá trị.

5. Truyền dữ liệu trên bus I2C, chế độ Master-Slave

Việc truyền dữ liệu diễn ra giữa con chủ và con tớ. Dữ liệu truyền có thể theo 2 hướng, từ chủ đến tớ hay ngược lại. Hướng truyền được quy định bởi bit thứ 8 trong byte đầu tiên được truyền đi.



Hình 1.9. Quá trình truyền dữ liệu

Truyền dữ liệu từ chủ đến tớ (ghi dữ liệu). Thiết bị chủ khi muốn ghi dữ liệu đến con tớ, quá trình thực hiện là:

- Thiết bị chủ tạo xung START.

- Thiết bị chủ gửi địa chỉ của thiết bị tớ mà nó cần giao tiếp cùng với bit = 0 ra bus và đợi xung ACK phản hồi từ con tớ.

- Khi nhận được xung ACK báo đã nhận diện đúng thiết bị tớ, con chủ bắt đầu gửi dữ liệu đến cho con tớ theo từng byte một. Theo sau mỗi byte này đều là một xung ACK. Số lượng byte truyền là không hạn chế.

- Kết thúc quá trình truyền, con chủ sau khi truyền byte cuối sẽ tạo xung STOP báo hiệu kết thúc.

Quá trình kết hợp ghi và đọc dữ liệu: giữa hai xung START và STOP, thiết bị chủ có thể thực hiện việc đọc hay ghi nhiều lần, với một hay nhiều thiết bị. Để thực hiện việc đó, sau một quá trình ghi hay đọc, thiết bị chủ lặp lại một xung START và lại gửi lại địa chỉ của thiết bị tớ và bắt đầu một quá trình mới. Chế độ giao tiếp Master-Slave là chế độ cơ bản trong một bus I2C, toàn bộ bus được quản lý bởi một master duy nhất. Trong chế độ này sẽ không xảy ra tình trạng xung đột bus hay mất đồng bộ xung clock vì chỉ có một master duy nhất có thể tạo xung clock.

6. Chế độ Multi-Master

Trên bus I2C có thể có nhiều hơn một master điều khiển bus. Khi đó bus I2C sẽ hoạt động ở chế độ Multi-Master. Chế độ này được hiểu là trên trong trên cùng một bus có thể hiểu hơn một thiết bị làm Slave có thể trở thành một Master nếu nó có khả năng trở thành Master ở một thời điểm nào đó. Tuy nhiên nếu sử dụng một IC điều khiển các chip nhớ thì chế độ Multi – Master không ồn tại vì các chip nhớ được thiết kế là Slave, không có khả năng trở thành Master.

II: Modul I2C với PIC

1. Modul I2C với PIC

Với những tiện ích đem lại, khối giao tiếp I2C đã được tích hợp cứng trong khá nhiều loại vi điều khiển khác nhau. Trong các loại Vi điều khiển PIC dòng Mid-range phổ biến tại Việt Nam, chỉ từ 16F88 mới có hỗ trợ phần cứng I2C, còn các loại chip khác không có. Với những loại Vi điều khiển không có hỗ trợ phần cứng giao tiếp I2C, để sử dụng ta có thể dùng phần mềm lập trình, khi đó ta sẽ viết một chương trình điều khiển 2 chân bất kỳ của Vi điều khiển để nó thực hiện giao tiếp I2C (các hàm START, STOP, WRITE, READ).

Trong việc lập trình cho PIC có rất nhiều phần mềm viết chương trình như CCS, AMS,Mplab… nhưng tôi chỉ đề cập đến phần giao tiếp I2C sử dụng Mplab.

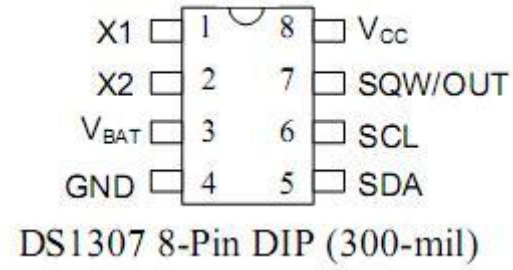
Trong Mplab, cần quan tâm đến những câu lệnh sau:

- OpenI2C(MASTER,SLEW_OFF); // khởi tạo I2C, chọn chip làm chủ hay tớ
- SSPADD=0x31; // đặt tốc đọ xung clock(SCL)
- StartI2C();// bắt đầu I2C
- IdleI2C(); // ACK, đợi phản hồi
- WriteI2C(noi dung can ghi);// ghi dữ liệu
- StopI2C();// dừng I2C
- RestartI2C();// bắt đầu lại I2C
- AckI2C(); // ACK phản hồi đi
- NotAckI2C(); // NotAck phản hồi đi.

2. Ví dụ sử dụng modul I2C của PIC 18f4520 với ds1307(đồng hồ thời gian thực).

DS1307 là chip đồng hồ thời gian thực, khái niệm thời gian thực ở đây được dùng với ý nghĩa thời gian tuyệt đối mà con người đang sử dụng, tính bằng giây, phút, giờ… DS1307 là một sản phẩm của Dallá Semiconductỏ. Chip này có 7 thanh ghi8 bit chứa thời gian là: giây, phút, giờ, thứ, ngày, tháng, năm. Ngoài ra DS1307 còn có một thanh ghi điều khiển ngõ ra phụ và 56 thanh ghi trống có thể dùng như RAM. DS1307 được đọc và thông qua giao diện I2C nên

cấu tạo bên ngoài rất đơn giản. DS1307 xuất hiện ở hai gói SOIC và DIP có 8 chân.



Hình 2.1 Hình ảnh DS1307

Các chân của DS1307 được mô tả như sau:

- X1 và X2 là hai ngõ kết nối với một thạch anh 32768KHz làm nguồ tạo dao động cho chip.

- Vbat : cực dương của một pin 3V nuôi chip.

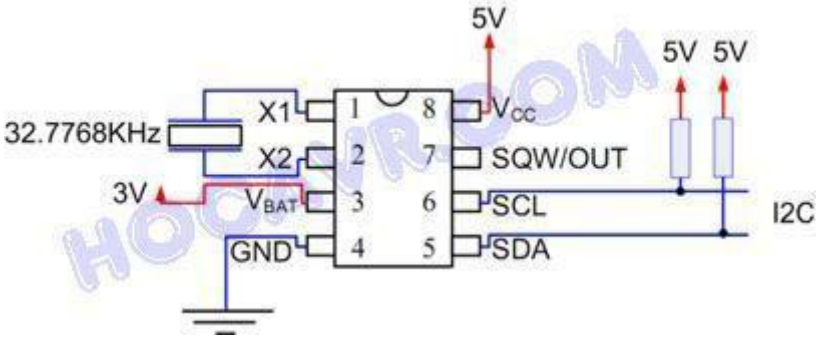
- GND: chân mas chung cho cả pin 3V và Vcc.

- Vcc: nguồn cho giao diện I2C, thường là 5V và dùng chung với vi điều khiển. Chú ý nếu là Vcc không nguồn nhưng Vbat được cấp thì DS1307 vẫn dang hoạt động nhưng không đọc và ghi được.

- SQUW/OUT: một ngõ phụ tạo xung vuông, tần số của xung có thể được lập trình.

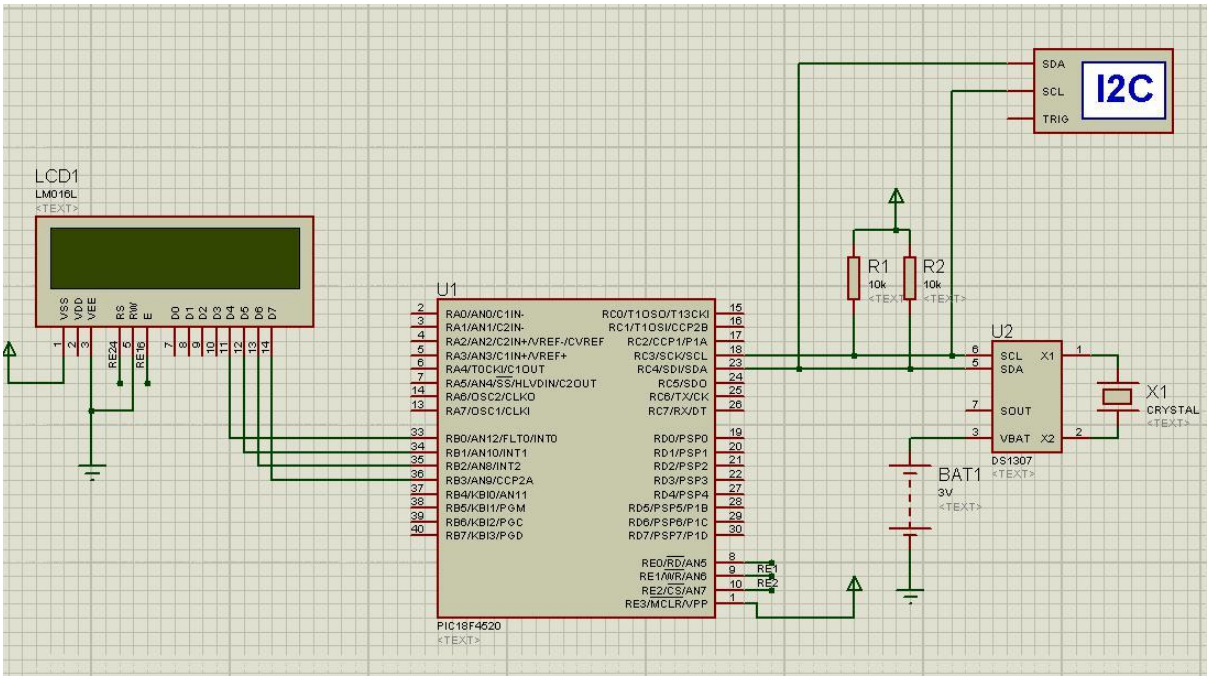
- SCL và SDA là hai đường giao xung nhịp và dữ liệu của giao diện I2C.

Có thể kết nối DS1307 bằng một mạch điện đơn giản sau:



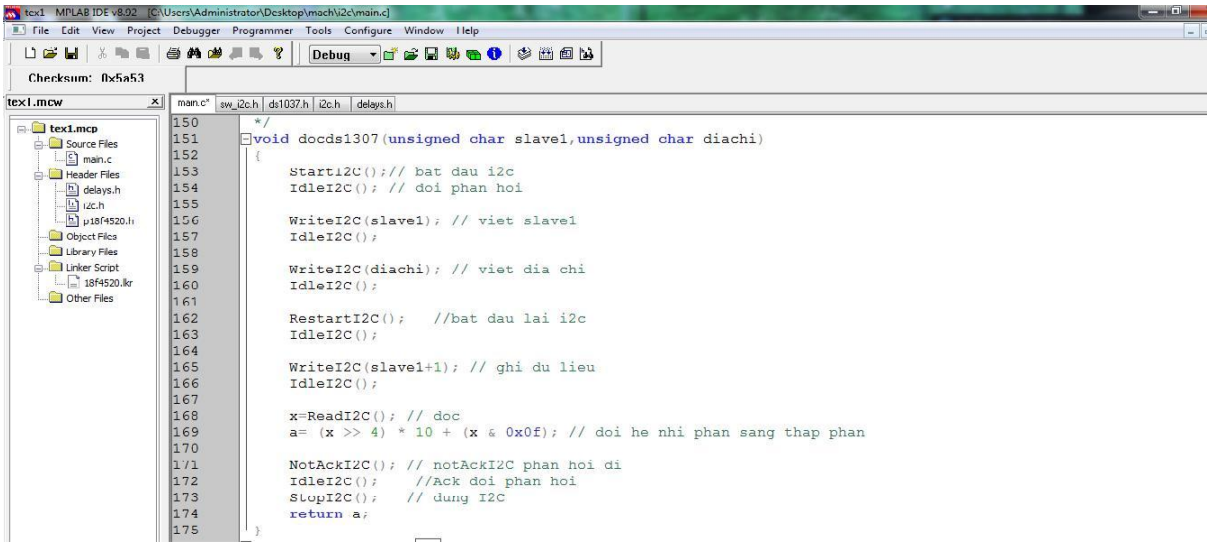
Hình 2.2: Mạch ứng dụng đơn giản của DS1307

Đây là một mạch giữa PIC 18f4520 và DS1307 được vẽ trong phần mềm proteus.

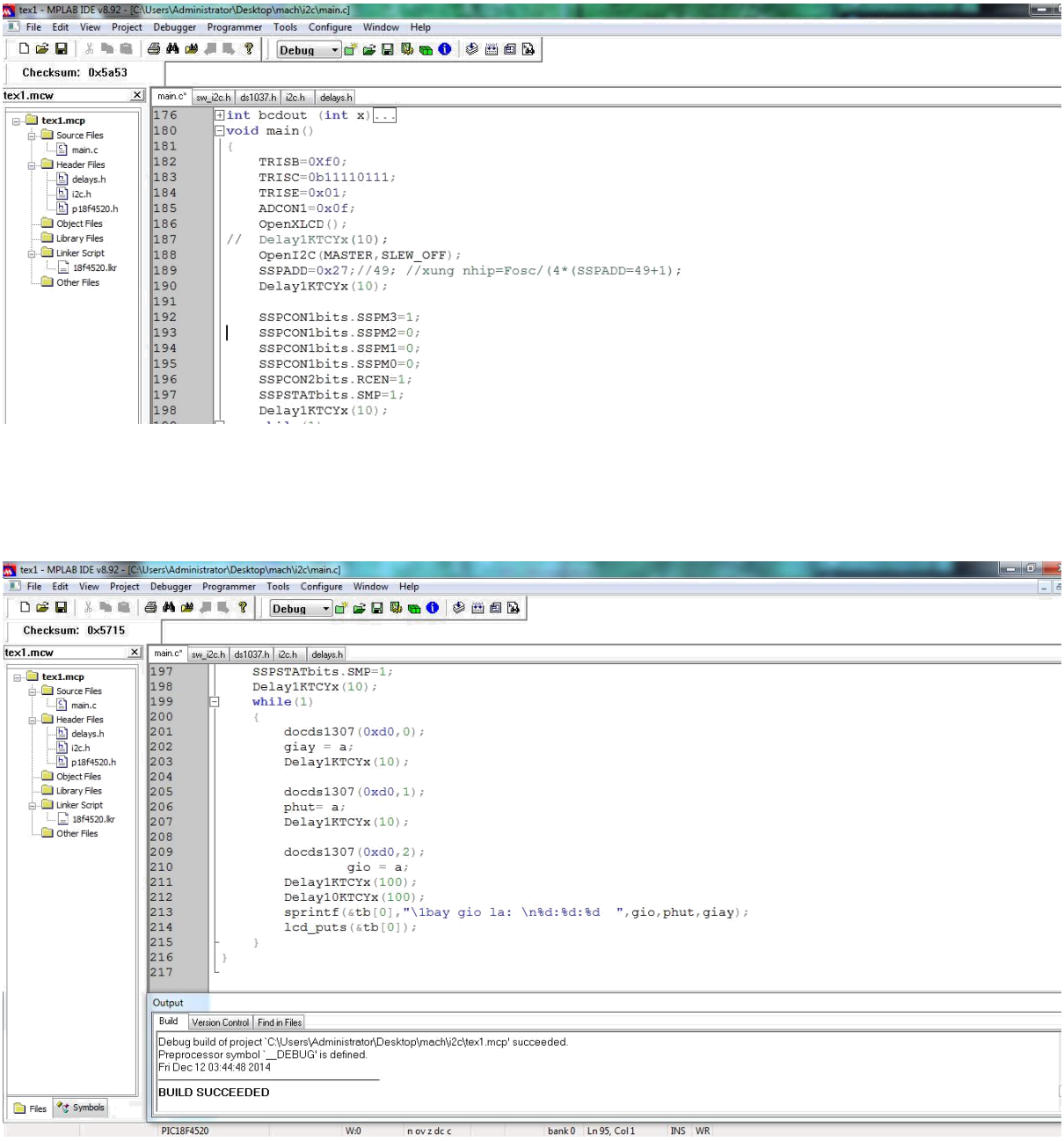


Hình 2.3: mạch PIC 18f4520 và DS1307.

Mạch trên sử dụng LCD hiển thị thời gian thực từ ds1307, chip PIC được coi như một Master và gọi slave (ds1307) trả lời. Và đây là chương trình của modul I2C PIC 18f4520.

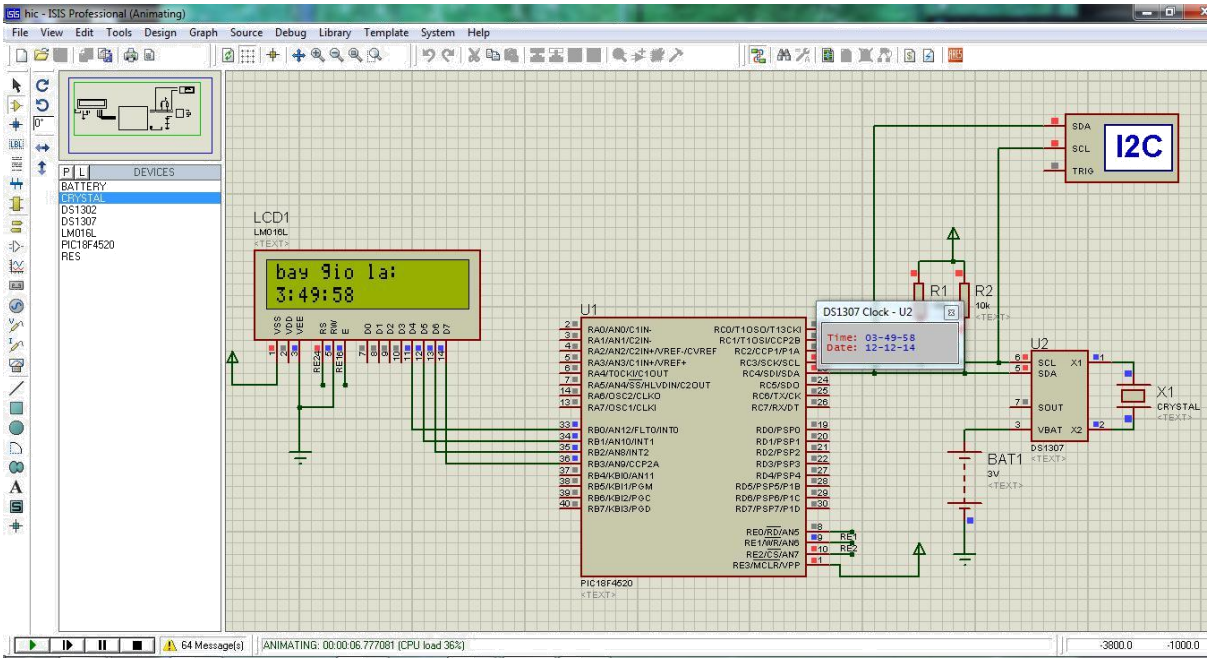


```
*/
void docds1307(unsigned char slave1,unsigned char diachi)
{
    StartI2C();// bat dau i2c
    IdleI2C(); // doi phan hoi

    WriteI2C(slave1); // viet slave1
    IdleI2C();

    WriteI2C(diachi); // viet dia chi
    IdleI2C();

    RestartI2C();   //bat dau lai i2c
    IdleI2C();

    WriteI2C(slave1+1); // ghi du lieu
    IdleI2C();

    x=ReadI2C(); // doc
    a= (x >> 4) * 10 + (x & 0x0f); // doi he nhi phan sang thap phan

    NotAckI2C(); // notAckI2C phan hoi di
    IdleI2C();    //Ack doi phan hoi
    StopI2C();   // dung I2C
    return a;
}
```



Hình 2.4: Chương trình với modul I2C

Với chương trình này, trên phần mềm proteus sẽ hiển thị kết quả mô phỏng là:



Hình 2.5: Kết quả mô phỏng proteus

Đây chỉ là một ví dụ rất nhỏ và đơn giản trong việc ứng dụng giao tiếp I2C. Ngoài ra còn sử dụng I2C giao tiếp giữa chip này với chip khác, giao tiếp chip với IC, … Vì vậy nó khá phổ biến và quan trọng trong vi điều khiển.