Tóm tắt:

Giới thiệu chung về I2C

- *Đặc điểm chung về I2C: giao thức, địa chỉ*
- *Các chế độ hoạt động: master-slave, multi-master*

Module I2C trong Vi điều khiển PIC

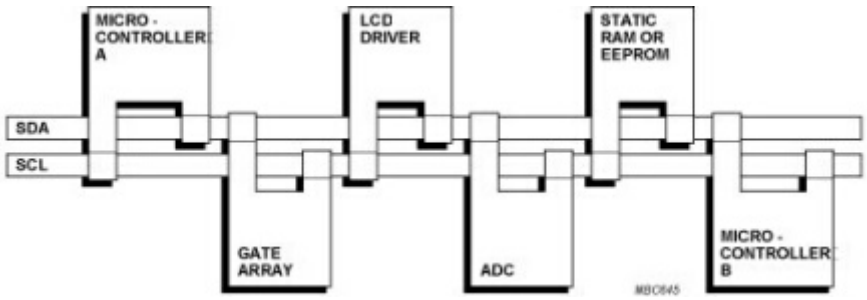
- *Cấu trúc phần cứng*
- *Chế độ hoạt động: Master, Slave, Multi-master*

1. Giới thiệu chung về I2C

Ngày nay trong các hệ thống điện tử hiện đại, rất nhiều ICs hay thiết bị ngoại vi cần phải giao tiếp với các ICs 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.

Lịch sử I2C – Thêm vào đây…(Phần này sẽ thêm sau…)

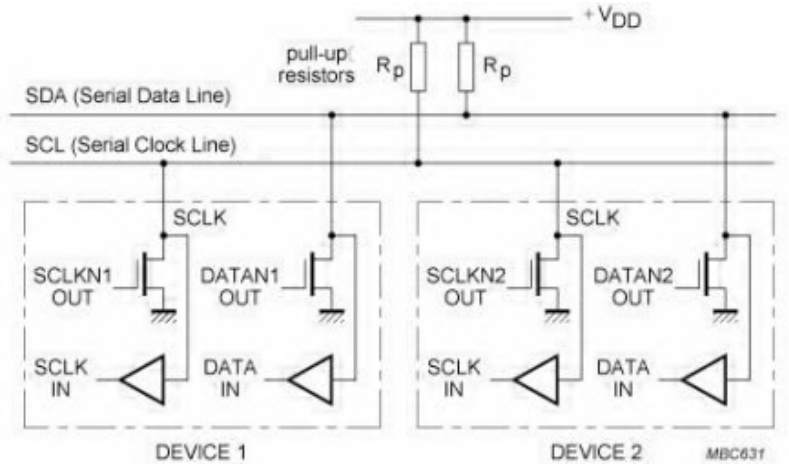
I2C mặc dù được phát triển bởi Philips, nhưng nó đã được rất nhiều nhà sản xuất IC trên thế giới sử dụng. I2C trở thành một chuẩn công nghiệp cho các giao tiếp điều khiển, có thể kể ra đây một vài tên tuổi ngoài Philips như: Texas Intrument (TI), Maxim-Dallas, analog Device, National Semiconductor … 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 tụ (DAC), IC điều khiển LCD, LED…



Hình 1.1. BUS I2C và các thiết bị ngoại vi

1.1. 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.

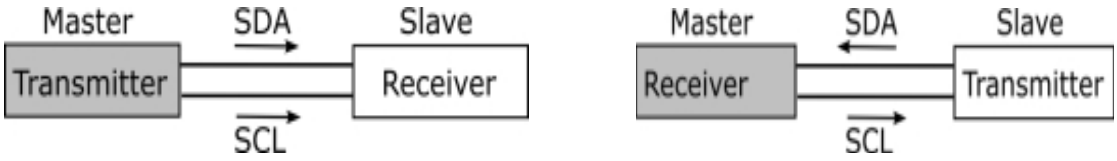


Hình 1.2. Kết nối thiết bị vào bus I2C ở chế độ chuẩn (Standard mode) và chế độ nhanh (Fast mode)

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.

Trở lại với hình 1.1, ta thấy có rất nhiều thiết bị (ICs) 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 (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 chủ 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 thiết bị A muốn gửi dữ liệu đến thiết bị B, quá trình được thực hiện như sau:

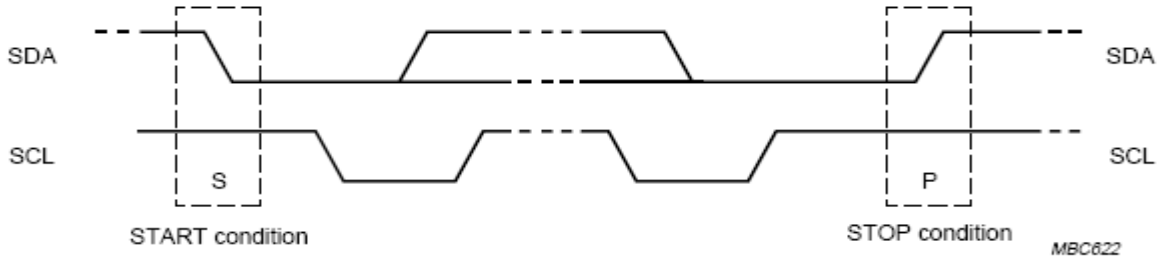
- Thiết bị A (Chủ) xác định đúng địa chỉ của thiết bị B (tớ), cùng với việc xác định địa chỉ, thiết bị A sẽ quyết định việc đọc hay ghi vào thiết bị tớ
- Thiết bị A gửi dữ liệu tới thiết bị B
- Thiết bị A kết thúc quá trình truyền dữ liệu

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 một giao tiếp giữa hai thiết bị sẽ được mô tả chi tiết trong các mục dưới đây.

1.2. START and STOP conditions

START và STOP là những điều kiện bắt buộc phải có khi một thiết bị chủ muốn thiết lập giao tiếp với một thiết bị nào đó trong mạng I2C. START là điều kiện khởi đầu, báo hiệu bắt đầu của giao tiếp, còn STOP báo hiệu kết thúc một giao tiếp. Hình dưới đây mô tả điều kiện START và 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

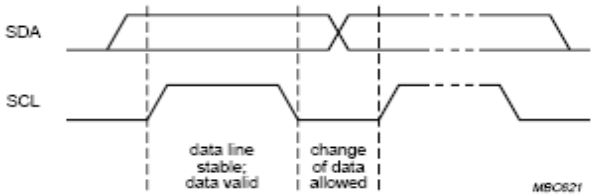
Điều kiện STOP: Một sự chuyển đổi trạng thái từ mức thấp lên cao trên đường SDA trong khi đường SCL đang ở mức cao.

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 qua 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.

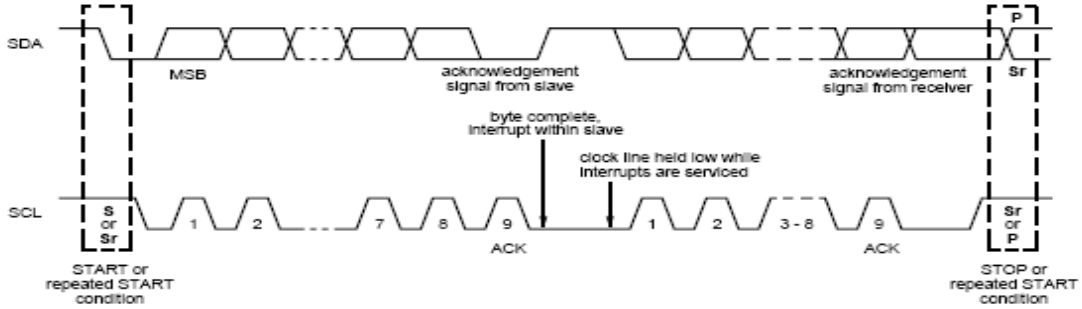
1.3. Định dạng dữ liệu truyền

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.

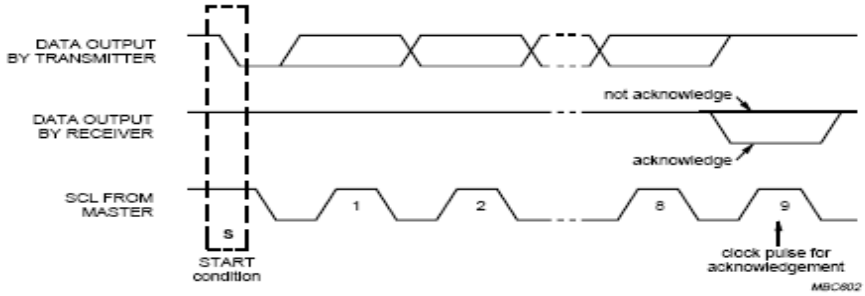


Hình 1.5. Quá trình truyền 1 bit dữ liệu

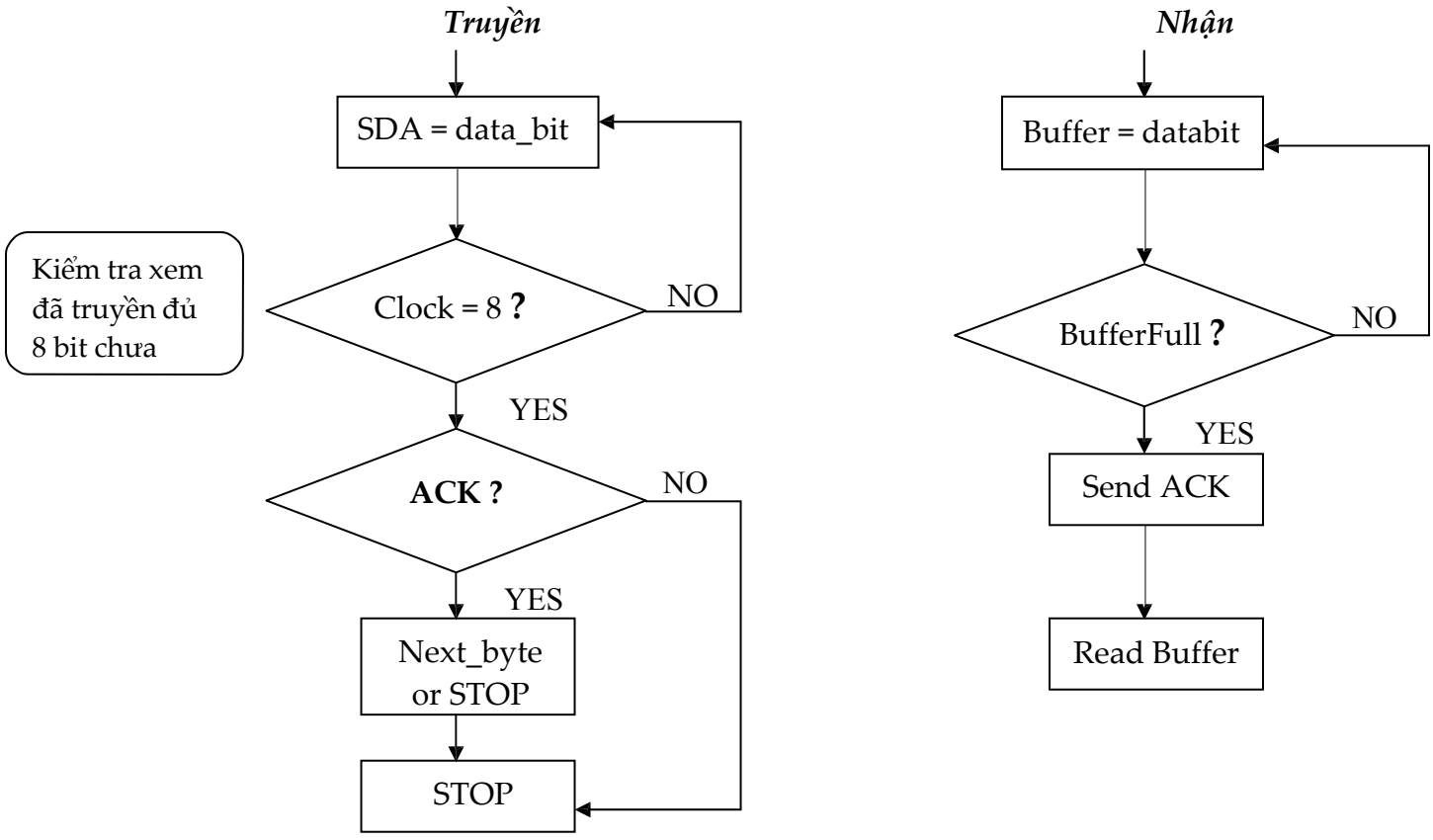
Mỗi byte dữ liệu được truyền có độ dài là 8 bits. Số lượng byte có thể truyền trong một lần là không hạn chế. Mỗi byte được truyền đi theo sau là một bit ACK để báo hiệu đã nhận dữ liệu. Bit có trọng số cao nhất (MSB) sẽ được truyền đi đầu tiên, các bít sẽ được truyền đi lần lượt. Sau 8 xung clock trên dây SCL, 8 bit dữ liệu đã được truyền đi. Lúc này thiết bị nhận, sau khi đã nhận đủ 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

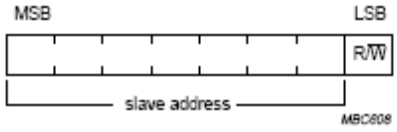


Hình 1.8. Lưu đồ thuật toán quá trình truyêng nhận dữ liệu

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 chỉ 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 STOP để kết thúc hay lặp lại một xung START để bắt đầu quá trình mới.

1.4. 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.

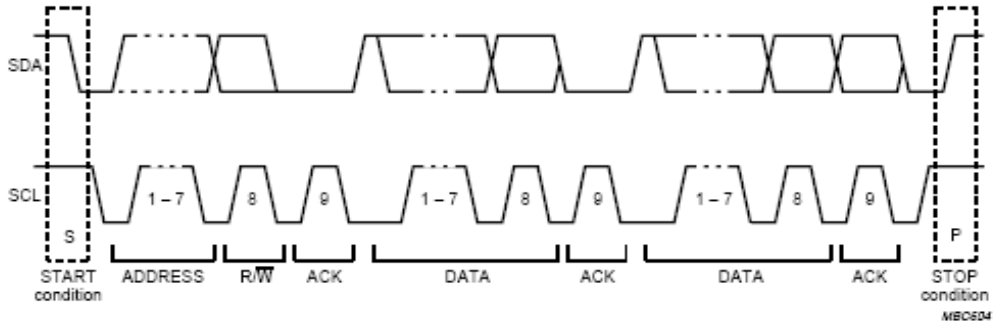


Hình 1.8. Cấu trúc byte dữ liệu đầu tiên

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 theo sau byte đầu tiên sẽ là dữ liệu từ con tớ gửi đến con 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.

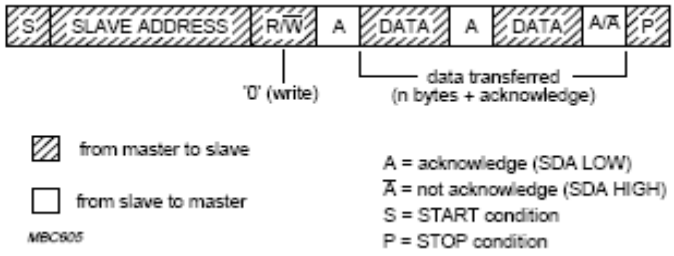
1.5. Truyền dữ liệu trên bus I2C, chế độ Master-Slave

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 R/$\overline{W}$ trong byte đầu tiên được truyền đi.



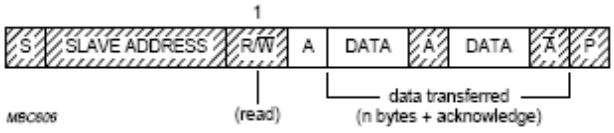
Hình 1.9. Quá trình truyền dữ liệu

- 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 $R/\overline{W} = 0$ 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 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.



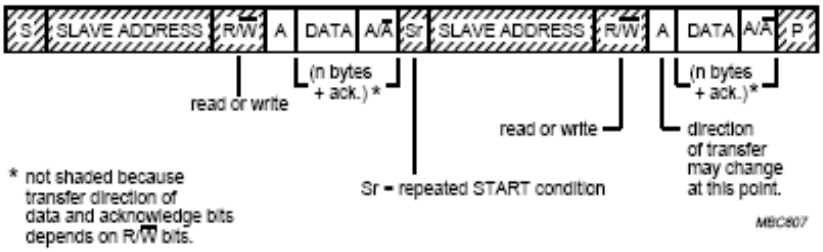
Hình 1.10. Ghi dữ liệu từ chủ đến tớ

- Truyền dữ liệu từ tớ đến chủ (đọc dữ liệu): Thiết bị chủ muốn đọc dữ liệu từ thiết bị tớ, quá trình thực hiện như sau:
    - Khi bus rỗi, thiết bị chủ tạo xung START, báo hiệu bắt đầu giao tiếp
    - Thiết bị chủ gửi địa chỉ của thiết bị tớ cần giao tiếp cùng với bit $R/\overline{W} = 1$ và đợi xung ACK từ phía thiết bị tớ
    - Sau xung ACK đầu tiên, thiết bị tớ sẽ gửi từng byte ra bus, thiết bị chủ sẽ nhận dữ liệu và trả về xung ACK. Số lượng byte không hạn chế
    - Khi muốn kết thúc quá trình giao tiếp, thiết bị chủ gửi xung Not-ACK và tạo xung STOP để kết thúc.



Hình 1.11. Đọc dữ liệu từ thiết bị tớ

- 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.



Hình 1.12.Quá trình phối hợp đọc/ghi dữ liệu

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.

1.6. 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.

Vấn đề này sẽ được bàn thảo sau.

2. Module I2C trong Vi điều khiển 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 16F84, 16F628 thì 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 trinh diều khiển 2 chân bất kỳ của Vi điều khiển để nó thực hiện giao tiếp I2C (các hàm START, STOP, WRITE, READ). Trong bài viết này ta đề cập đến việc sử dụng giao tiếp I2C của các loại PIC có tích hợp khối I2C sẵn trong nó, mà cụ thể là Vi điều khiển PIC16F877A.

2.1. Đặc điểm phần cứng

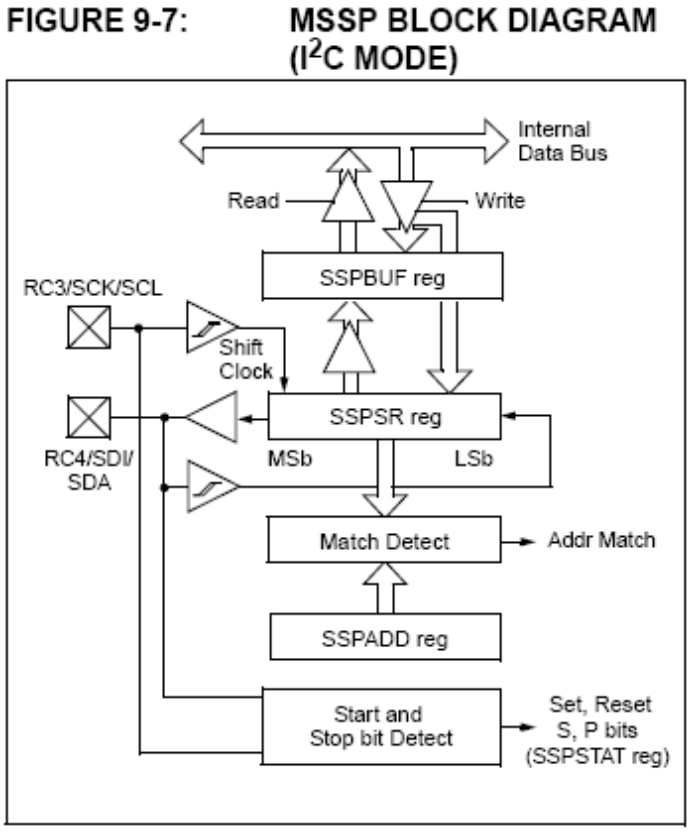
Hình dưới đây chỉ ra cấu trúc phần cứng của khối điều khiển Giao tiếp nối tiếp đồng bộ (MSSP) hoạt động ở chế độ I2C. Khối I2C có dầy đủ chức năng, hoạt động ở cả 2 chế độ là MASTER (chủ) và SLAVE (tớ), có ngắt xảy ra khi có điều kiện START hay STOP xảy ra, nhằm định rõ đường I2C có dỗi hay khônong (chức năng Multi-master). Chế độ địa chỉ có thể là 7-bit hay 10-bit.

Khối I2C có 6 thanh ghi điều khiển hoạt động, đó là:

- SSPCON: Thanh ghi điều khiển
- SSPCON2: Thanh ghi điều khiển thứ 2
- SSPSTAT: Thanh ghi trạng thái

- SSPBUF: Thanh ghi bộ đệm truyền nhận
- SSPSR: Thanh ghi dịch
- SSPADD: Thanh ghi địa chỉ

Các thanh ghi SSPCON, SSPBUF, SSPADD và SSPSON2 có thể truy cập đọc/ghi được. Thanh ghi SSPSR không thể truy cập trực tiếp, là thanh ghi dich dữ liệu ra hay vào. Các thanh ghi SSPCON, SSPCON2 và SSPSTAT được định địa chỉ bit, mỗi bit có chức năng riêng. Ý nghĩa của từng thanh ghi và của mỗi bit trong từng thanh ghi đã được đề cập kỹ trong tài liệu Datasheet của PIC và trong tài liệu TUT04.02.PVN.MAFD của bạn Mạnh, tôi không đề cập thêm ở đây. Trong tài liệu nayg tôi sẽ tập trung vào việc sử dụng khối I2C của PIC ở các chế độ Master và Slave trong phần mềm biên dịch C cho PIC là CCS



Hình2.1. Cấu trúc khối I2C trong PIC

2.2. Cách thức sử dụng Module I2C trong CCS

Trong việc lập trình cho PIC sử dụng giao tiếp I2C của nó trong các ứng dụng, người lập trình có thể thực hiện một cách dễ dàng với trình dịch CCS. Nói dễ dàng ở đây là chỉ về mặt cú pháp lệnh, ta không cần sử dụng nhiều câu lệnh khó nhớ như trong lập trình ASM.

Việc khởi tạo, chọn chế độ hoạt động và thực hiện giao tiếp của I2C đã có các hàm dựng sẵn của CCS thực hiện. Các hàm liệt kê dưới đây là của phiên bản CCS 3.242, đó là:

- i2c_isr_state(): Thông báo trạng thái giao tiếp I2C
- i2c_start(): Tạo điều kiện START
- i2c_stop(): Tạo điều kiện STOP
- i2c_read(): Đọc giá trị từ thiết bị I2C, trả về giá trị 8 bit
- i2c_write(): Ghi giá trị 8 bit đến thiết bị I2C

Để sử dụng khối I2C ta sử dụng khai báo sau:

```
#use i2c(chế_độ, tốc độ, sda = PIN_C4, scl=PIN_C3)
```

- Chế độ: Master hay Slave
- Tốc độ: Slow (100KHz) hay Fast (400KHz)
- SDA và SCL là các chân i2c tương ứng của PIC

Sau khai báo trên, ta có thể sử dụng các hàm nêu trên để thực hiện, xử lý các giao tiếp i2c với các thiết bị ngoại vi khác.

Còn tiếp nữa…

3. Kết luận