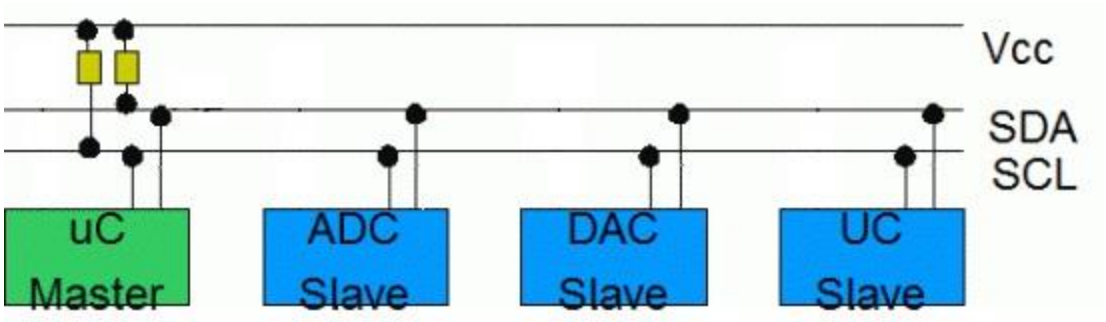
GIAO TIẾP I2C

1. Giới thiệu:

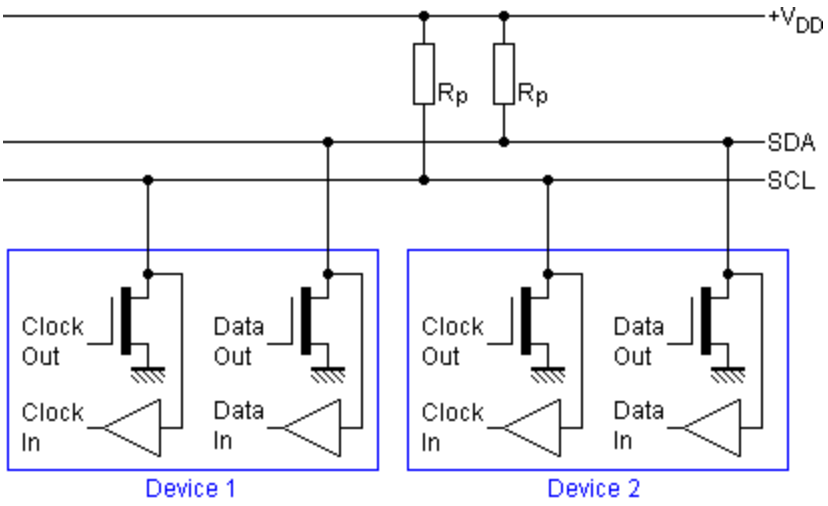
- Đầu năm 1980 Phillips đã phát triển một chuẩn giao tiếp nối tiếp 2 dây được gọi là I2C. I2C là tên viết tắt của cụm từ Inter-Intergrated Circuit. Đây là đường Bus giao tiếp giữa các IC với nhau. I2C mặc dù được phát triển bởi Philips, nhưng nó đã được rất nhiều nhà sản xuất IC trên thế giới sử dụng. 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), MaximDallas, 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... chip 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 - Bus I2C và các thiết bị ngoại vi

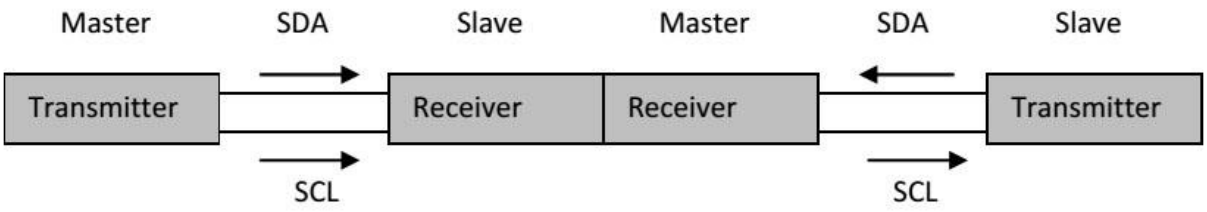
2. Đặc điểm giao tiếp I2C

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



Hình 2 - Kết nối thiết bị vào bus I2C ở chế độ chuẩn (Standard mode) và chế độ nhanh (Fast mode)

- Mỗi dây SDA hãy SCL đều được nối với điện áp dương của nguồn cấp thông qua một điện trở kéo lên (pullup 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 (opendrain hay opencollector). Giá trị của các điện trở này khác nhau tùy vào từng thiết bị và chuẩn giao tiếp, thường dao động trong khoảng 1K đến 4.7k
- Trở lại với hình 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 động như là thiết bị nhận hoặc truyền 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) hãy 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 hay tớ.Tại sao lại có sự phân biệt này ? Đó là vì trên một bus I2C thì quyền điều khiển thuộc về thiết bị chủ. 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 việc giao tiếp.



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.

3. Chế độ hoạt động (tốc độ truyền):

Các bus I2C có thể hoạt động ở ba chế độ, hay nói cách khác các dữ liệu trên bus I2C có thể được truyền trong ba chế độ khác nhau

1. Chế độ tiêu chuẩn (Standard mode)
2. Chế độ nhanh (Fast mode)
3. Chế độ cao tốc High-Speed (Hs) mode

Chế độ tiêu chuẩn:

1. Đây là chế độ tiêu chuẩn ban đầu được phát hành vào đầu những năm 80
2. Nó có tốc độ dữ liệu tối đa 100kbps
3. Nó sử dụng 7-bit địa chỉ, và 112 địa chỉ tớ

Tăng cường hoặc chế độ nhanh:

1. Tốc độ dữ liệu tối đa được tăng lên đến 400 kbps.
2. Để ngăn chặn gai tiếng ồn, Ngõ vào của thiết bị Fast-mode là Schmitt-triggered.
3. chân SCL và SDA của một thiết bị tớ I ² C ở trạng thái trở kháng cao khi không cấp nguồn.

Chế độ cao tốc (High-Speed):

Chế độ này đã được tạo ra chủ yếu để tăng tốc độ dữ liệu lên đến 36 lần nhanh hơn so với chế độ tiêu chuẩn. Nó cung cấp 1,7 Mbps (với Cb = 400 pF), và 3.4Mbps (với C> b = 100pF).

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)

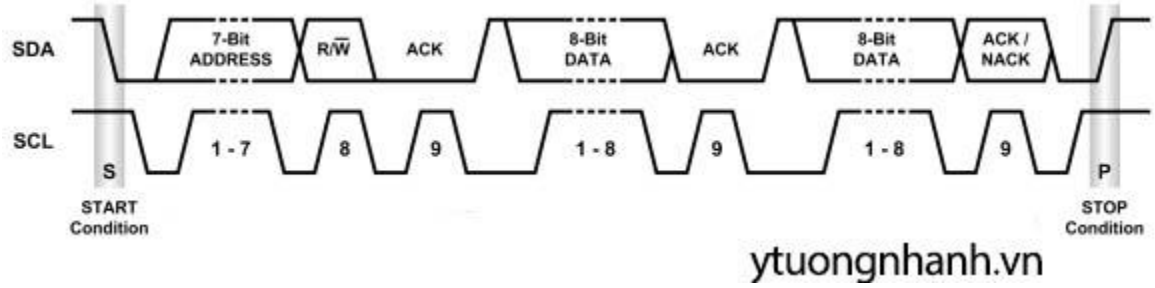
Dù ở chế độ nào, một giao tiếp I2C đều dựa vào quan hệ chủ/tớ. Giả thiết một thiết bị A muốn gửi dữ liệu đến thiết bị B, quá trình được thực hiện như sau:

- 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.

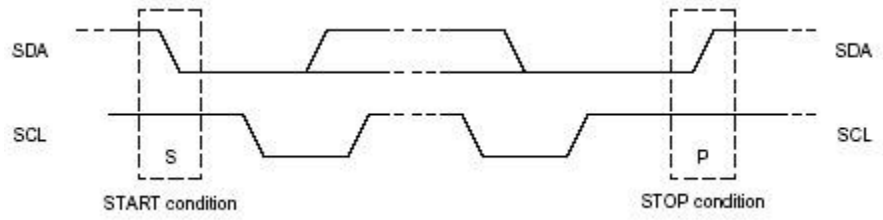
Trình tự truyền bit trên đường truyền:



1. Thiết bị chủ tạo một điều kiện start. Điều kiện này thông báo cho tất cả các thiết bị tớ lắng nghe dữ liệu trên đường truyền
2. Thiết bị chủ gởi địa chỉ của thiết bị tớ mà thiết bị chủ muốn giao tiếp và cờ đọc/ghi dữ liệu (nếu cờ thiết lập lên 1 byte tiếp theo được truyền từ thiết bị tớ đến thiết bị chủ, nếu cờ thiết lập xuống 0 thì byte tiếp theo truyền từ thiết bị chủ đến thiết bị tớ).
3. Khi thiết bị tớ trên bus I2C có địa chỉ đúng với địa chỉ mà thiết bị chủ gửi sẻ phản hồi lại bằng một xung ACK.
4. Giao tiếp giữa thiết bị chủ và tớ trên bus dữ liệu bắt đầu. Cả chủ và tớ đều có thể nhận hoặc truyền dữ liệu tùy thuộc vào việc truyền thông là đọc hay viết. Bộ truyền gửi 8 bit dữ liệu tới bộ nhận, Bộ nhận trả lời với một bit ACK.
5. Để kết thúc quá trình giao tiếp, thiết bị chủ tạo ra một điều kiện stop.

4. Điều kiện START và STOP (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 trên bus 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 rỗi ("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.



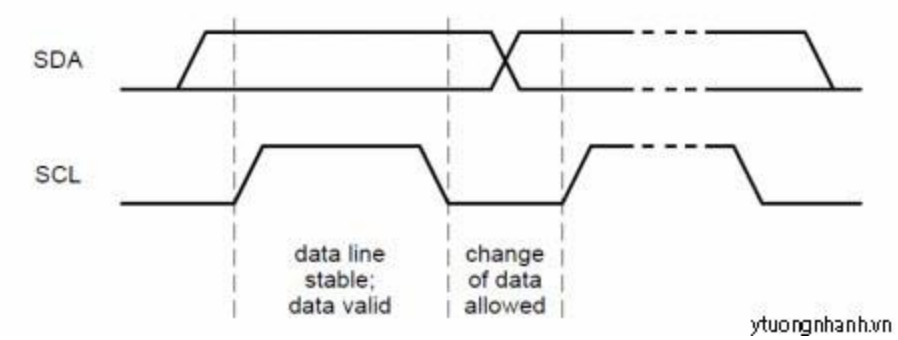
Điều kiện START: một sự chuyển đổi trạng thái từ cao xuống thấp trên đường SDA trong khi đường SCL đang ở mức cao (cao = 1; thấp = 0) báo hiệu một điều kiện START

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 trạng 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 (Repeated START) đều có chức năng giống nhau là khởi tạo một giao tiếp.

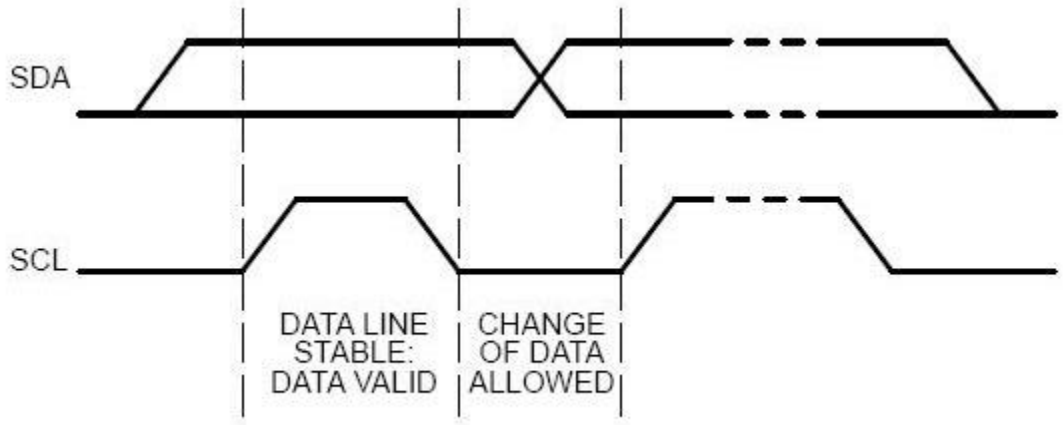
Truyền dữ liệu:

Mỗi xung clock có một bit dữ liệu được truyền. Mức tín hiệu SDA chỉ được thay đổi khi xung clock đang ở mức thấp, và ổn định khi xung clock ở mức cao. Thiết bị tớ có thể lấy mẫu dữ liệu khi xung clock ở mức cao.



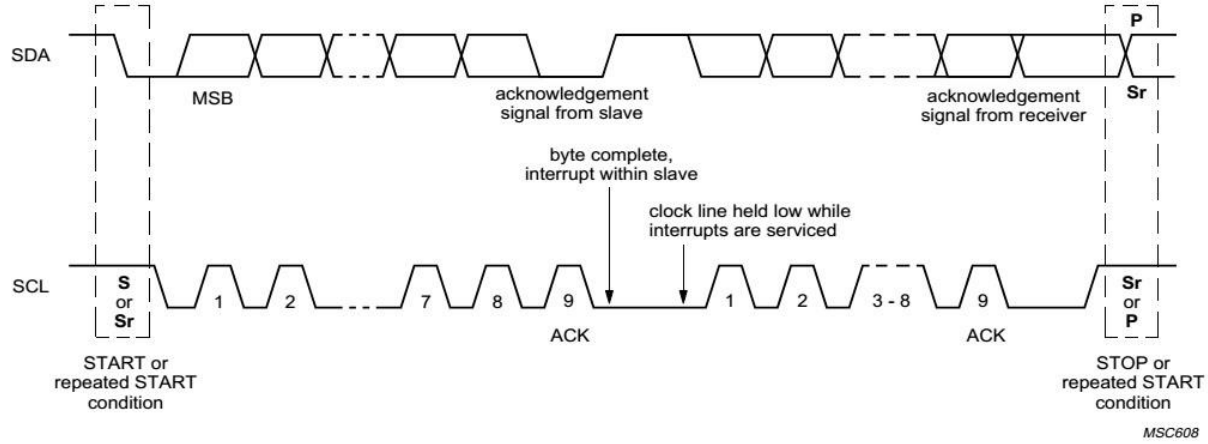
Định dạng dữ liệu truyền

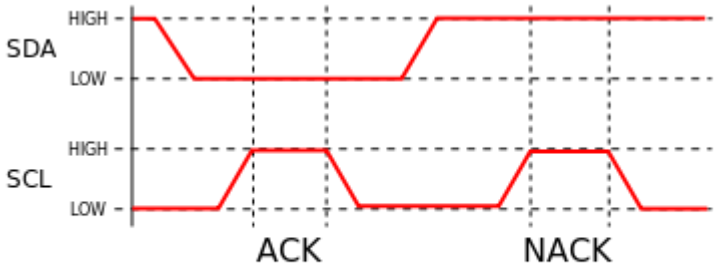
Dữ liệu được truyền trên bus I2C theo từng bit, bit dữ liệu được truyền đi tại mỗi cạnh lên của xung đồng hồ trên dây SCL, quá trình thay đổi bit dữ liệu xảy ra khi SCL đang ở mức thấp.



Hình 3 - Quá trình truyền 1 bit dữ liệu

Mỗi byte dữ liệu được truyền có độ dài là 8 bit. 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 bit sẽ được truyền đi làn 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.

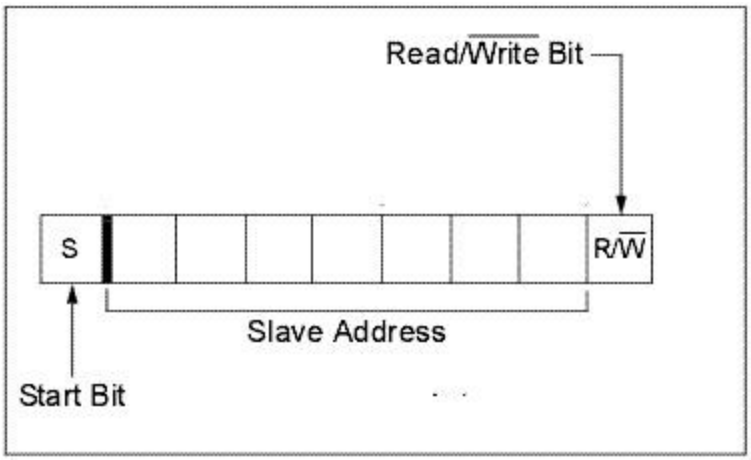




Một byte truyền đi có kèm theo bit ACK là điều kiện bắt buộc, nhằm đảm bảo cho quá trình truyền nhận được diễn ra chính xác. Khi không nhận được đúng địa chỉ hay khi muốn kết thúc quá trình giao tiếp thiết bị nhận sẽ gửi một xung Not-ACK(SDA ở mức cao) để báo cho thiết bị chủ biết, thiết bị chủ sẽ tạo xung STOP để kết thúc hay lặp lại một xung START để bắt đầu quá trình mới.

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.

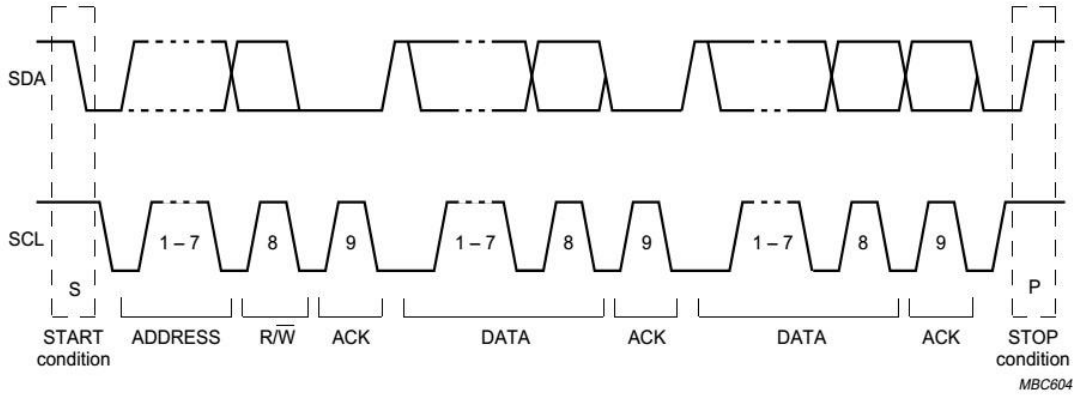


Mỗi một thiết bị ngoại vi sẽ có một địa chỉ riêng do nhà sản xuất ra nó quy định. Địa chỉ đó có thể là cố định hay thay đổi. Riêng bit điều khiển hướng sẽ quy định chiều truyền dữ liệu. Nếu bit này bằng "0" có nghĩa là byte dữ liệu tiếp theo sau sẽ được truyền từ chủ đến tớ, còn ngược lại nếu bằng "1" 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.

Hiện nay có thể đánh địa chỉ các thiết bị trên bus I2C dưới dạng 10 bit địa chỉ. Việc thực hiện đánh dấu địa chỉ theo khung 10 bit được thực hiện nếu sau lệnh START ta gửi chuỗi 11110 (số nhị phân) ra đường SDA.

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/W trong byte đầu tiên (byte địa chỉ) được truyền đi.



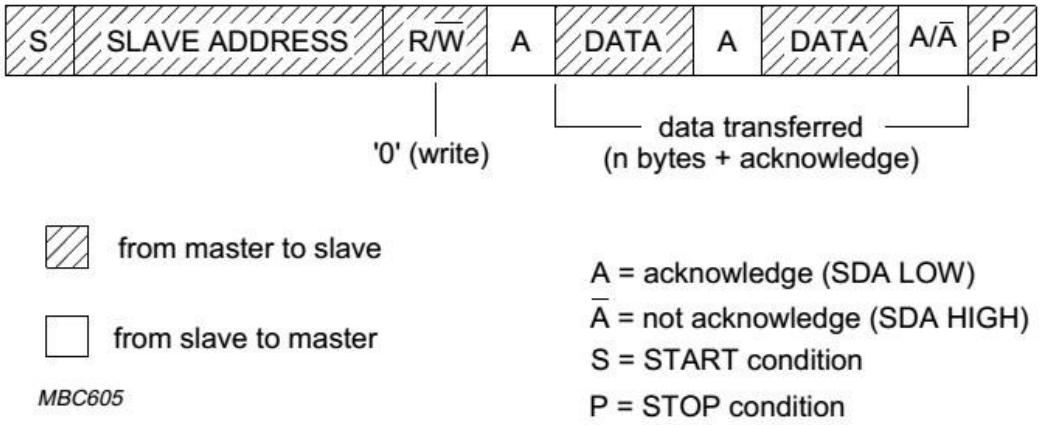
5. Truyền dữ liệu từ chủ đến tớ (ghi dữ liệu): Thiết bị chủ khi muốn ghi dữ liệu đến con tớ, quá trình thực hiện là:

- 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/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.



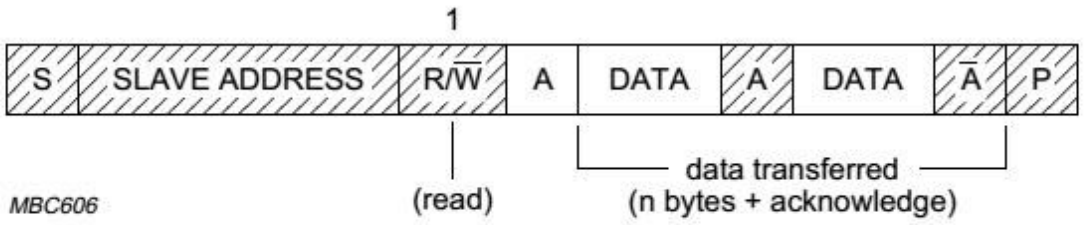
6. Truyền dữ liệu từ tớ đến chủ (đọc dữ liệu): Thiết bị chủ muốn đọc dữ liệu từ thiết bị tớ, quá trình thực hiện như sau:

- Khi bus rỗi, thiết bị chủ tạo xung START, báo hiệu bắt đầu giao tiếp

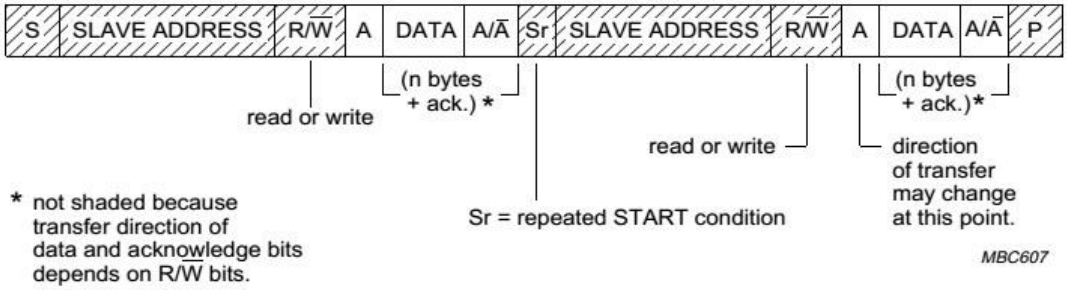
- Thiết bị chủ gửi địa chỉ của thiết bị tớ càn giao tiếp cùng với bit R/W = 1 và đợi xung ACK từ phía thiết bị tớ

- Sau xung ACK từ con tớ, 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.



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 MasterSlave là chế độ cơ bản trong một bus I2C, toàn bộ bus được quản lý bởi một master duy nhát. Trong chế độ này sẽ không xảy ra tình trạng xung đột bus hay mất đồng bộ xung clock vì chỉ có một con chủ (master) duy nhất có thể tạo xung clock.

Chế độ Multi Master

Trên bus I2C có thể có nhiều hơn một con chủ điều khiển bus. Khi đó bus I2C sẽ hoạt động ở chế độ Multi Master.

Cấu hình chuẩn giao tiếp I2C trong trình biên dịch CCS

1.#USE I2C (***options***)

Cú pháp:	#USE I2C (***options***)	
Các phần tử:	**MASTER**	Thiết lập ở chế độ master (master mode)
	MULTI_MASTER	Thiết lập ở chế độ multi_master (multi_master mode)
	SLAVE	Thiết lập ở chế độ slave mode
	SCL=pin	Chỉ định chân SCL (pin là bit địa chỉ)
	SDA=pin	Chỉ định chân SDA (pin là bit địa chỉ)
	ADDRESS=nn	Chỉ định địa chỉ slave mode
	FAST	Sử dụng tốc độ truyền nhanh (Fast)
	FAST=nnnnnn	Thiết lập tốc độ là nnnnnn hz
	SLOW	Sử dụng tốc độ truyền chậm.
	RESTART_WDT	Khởi động WDT trong khi chờ thực thị I2C_READ
	FORCE_HW	Bắt buộc sử dụng phần cứng khi giao tiếp (Khối module phần cứng I2C)
	FORCE_SW	Bắt buộc sử dụng phần mềm khi giao tiếp (Tự tạo mã giả lập bằng phần mềm do CCS tự sinh code)
	NOFLOAT_HIGH	Does not allow signals to float high, signals are driven from low to high
	SMBUS	Bus được sử dụng không phải là I2C bus, nhưng gần giống.
	STREAM=id	Gán một tên nhận dạng đến cổng I2C này. Tên nhận dạng này có thể sử dụng trong các hàm giống i2c_read hoặc i2c_write.
	NO_STRETCH	Do not allow clock streaching
	MASK=nn	Thiết lập một mặt nạ địa chỉ cho các phần có hỗ trợ Ví dụ: #use I2C(slave,sda=PIN_C4,scl=PIN_C3, address=0xa0, FORCE_HW, mask=0xDF)
	I2C1	Thay vì sử dụng SCL= và SDA= đây thiết lập chân giao tiếp tương ứng với module I2C thứ nhất.

	I2C2	Thay vì sử dụng SCL= và SDA= đây thiết lập chân giao tiếp tương ứng với module I2C thứ hai
	NOINIT	Không thực hiện khởi tạo khối ngoại vi I2C, chỉ khởi tạo khi khi sử dụng lệnh I2C_INIT() khi chạy chương trình.
	Có một vài chíp cho phép	
	DATA_HOLD	No ACK is sent until I2C_READ is called for data bytes (slave only)
	ADDRESS_HOLD	No ACK is sent until I2C_read is called for the address byte (slave only)
	SDA_HOLD	Min of 300ns holdtime on SDA a from SCL goes low
Mục đích:	CCS cung cấp hỗ trợ giao tiếp I2C dựa trên phần cứng và trên phần mềm (đối với các vi điều khiển không hỗ trợ module giao tiếp i2C bằng phần cứng), Tham khảo datasheet đối với dòng vi điều khiển mà bạn muốn lập trình, không phải tất cả Pic đều hỗ trợ module phần cứng đối với giao tiếp I2C. Thư viện I2C chứa đựng các hàm để thực hiện giao tiếp. #USE I2C ảnh hưởng đến các hàm I2C_START, I2C_STOP, I2C_READ, I2C_WRITE and I2C_POLL đến khi #USE I2C tiếp theo được khai báo. Các hàm được tạo bằng phần mềm ngoại trừ khi bạn khai báo FORCE_HW.	
Ví dụ:	#use I2C(master, sda=PIN_B0, scl=PIN_B1) #use I2C(slave,sda=PIN_C4,scl=PIN_C3, address=0xa0, FORCE_HW) #use I2C(master, scl=PIN_B0, sda=PIN_B1, fast=450000) //Thiết lập tốc độ là 450 KBSP	

Code:
#use I2C(slave,sda=PIN_C4,scl=PIN_C3, address=0xa0, FORCE_HW, mask=0xDF)

Với khai báo như trên chip chấp nhận cả địa chỉ 0xA0, và địa chỉ 0x80, và tạo ra ngắt trên mỗi địa chỉ. Thiết bị tớ (slave) sẽ phải đọc byte địa chỉ và xác định địa chỉ mà nó phản hồi đến.

Mặt nạ địa chỉ hoạt động như bên dưới:

0b10100000 địa chỉ
0b11011111 mặt nạ

0b10x00000 địa chỉ được chấp nhận

2.i2c_poll( )

Hàm này trả về True nếu trong bộ đệm nhận có byte dữ liệu. Nếu hàm này trả về true . Ta gọi hàm I2C_read() sẽ trả về 1 byte dữ liệu đã nhận được.

Ví dụ:

```
if(i2c-poll())

buffer [index]=i2c-read();// đọc dữ liệu
```

3. i2c_speed( )

Cú pháp:

i2c_speed (baud)

i2c_speed (stream, baud)

Trong đó:

baud: là số bit trên giây

stream:tên nhận dạng đến cổng I2C được thiết lập trong phần #use I2C

Trả về: Hàm này không có tham số trả về

Chức năng:

Hàm này sẽ thay đổi tốc độ I2C khi chạy chương trình. Nó chỉ làm việc khi đang sử dụng module i2c bằng phần cứng.

Ví dụ:

I2C_Speed (400000);

4. i2c_start( )

Cú pháp

i2c_start()

i2c_start(stream)

i2c_start(stream, restart)

Trong đó:

stream:tên nhận dạng đến cổng I2C được thiết lập trong phần #use I2C

Tạo một điều kiện start khi thiết bị đóng vai trò marter mode

Ví dụ:

```
i2c_start();

i2c_write(0xa0);     // Địa chỉ của thiết bị muốn giao tiếp

i2c_write(address);  // Viết dữ liệu đến thiết bị

i2c_start();         // Restart

i2c_write(0xa1);     // Thay đổi hướng truyền

data=i2c_read(0);    // Nhận dữ liệu từ thiết bị

i2c_stop();
```

5. i2c_stop( )

Hàm này tạo một điều kiện stop khi thiết bị là master

6. i2c_slaveaddr( )

Hàm này thiết lập địa chỉ khi thiết bị là slave

7. i2c_read( )

Cú pháp:

data = i2c_read();

data = i2c_read(ack);

data = i2c_read(stream, ack);

Trong đó:

ack là tùy chọn, mặc định là 1

ack=0 : không có chỉ thị ack

ack=1: Có chỉ thị ack

ack=2: Chỉ sử dụng với slave

stream:tên nhận dạng đến cổng I2C được thiết lập trong phần #use I2C

Ví dụ:

```
1
2
3
4
5
6
7
8
9
10
11
```

```
i2c_start();

i2c_write(0xa1);

data1 = i2c_read(TRUE);

data2 = i2c_read(FALSE);

i2c_stop();

<span style="font-size:20px"><span style="font-family:Times New Roman,Times,serif">i2c_write( )</span></span>
```

Cú pháp:

i2c_write (***data***)

i2c_write (stream, *data*)

data: dữ liệu 8 bit

Stream: tên định danh cổng I2C được định nghĩa trong phần #use I2C

Tham số trả về: hàm này trả về bit ACK.

0 nghĩa là ACK

1 Nghĩa là Non-ACK

2: nghĩa là có xung đột khi sử dụng ở chế độ đa chủ.

Hàm này sẽ không trả về một ACK nếu sử dụng trong chế độ slave

Gửi 1 byte quan bus I2C. Trong chế độ master hàm này sẽ tạo xung clock với dữ liệu và trong chế độ slave nó sẽ chò xung clock từ master. Hàm này không cung cấp chức năng timeout. Hàm này trả về bit ACK.