Giao thức bus I2C

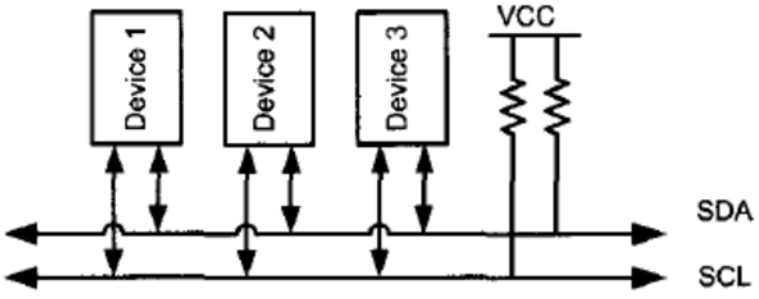
IIC (Inter-Integrated Circuit) là một giao diện bus kết nối được tích hợp vào nhiều thiết bị, chẳng hạn như cảm biến, RTC, và EEPROM. IIC cũng được gọi là I2C (I^2C) hay I bình phương C trong nhiều tài liệu kỹ thuật. Trong phần này, chúng ta sẽ xem xét các chân của bus I2C và tập trung vào các thuật ngữ và giao thức I2C.

1.1. BUS I2C

Bus I2C ban đầu được đề xuất bởi Philips, nhưng trong những năm gần đây chúng đã trở thành một tiêu chuẩn được sử dụng rộng rãi được áp dụng bởi nhiều công ty chip bán dẫn. I2C là lý tưởng để gắn các thiết bị ngoại vi tốc độ thấp vào bo mạch chủ hoặc hệ thống nhúng hoặc bất cứ nơi nào mà một giao tiếp đáng tin cậy với một khoảng cách ngắn là cần thiết. Như chúng ta sẽ thấy trong bày viết này, I2C cung cấp một giao tiếp định hướng giao tiếp với sự thừa nhận. Thiết bị I2C chỉ sử dụng 2 chân để truyền dữ liệu, thay vì 8 chân trở lên được sử dụng trong các bus truyền thống. Chúng được gọi là SCL (Serial Clock), nó đồng bộ hoá việc truyền dữ liệu giữa hai chip, và SDA (Serial Data). Việc giảm chân giao tiếp sẽ làm giảm kích thước gói và điện năng tiêu thụ đáng kể, làm cho chúng lý tưởng cho nhiều ứng dụng trong đó không gian là một mối quan tâm lớn. Hai chân này, SDA và SCK, làm cho I2C trở thành giao tiếp 2 dây. Trong nhiều ghi chú ứng dụng, bao gồm cả datasheet AVR, I2C còn được gọi là giao tiếp nối tiếp hai dây (Two-Wire Serial Interface – TWI). Trong bài viết này, chúng ta sử dụng I2C và TWI thay thế cho nhau.

1.2. ĐẶC ĐIỂM ĐƯỜNG DÂY ĐIỆN I2C

Thiết bị I2C chỉ sử dụng 2 chân mở đa chiều để truyền dữ liệu. Để triển khai I2C, chỉ cần một điện trở 4.7kΩ kéo lên cho mỗi đường bus là cần thiết (xem hình 1.1). Triển khai một dây-AND là cần thiết để triển khai các giao thức I2C. Có nghĩa là nếu một hoặc nhiều thiết bị kéo đường dây xuống mức thấp (không), đường trạng thái là không và mức của đường sẽ chỉ là 1 nếu thiết bị kéo đường dây xuống mức thấp.



Hình 1.1. Bus I2C

1.3. NÚT (NODE) I2C

Trong AVR/PIC18 có thể lên đến 120 thiết bị khác nhau có thể chia sẻ một bus I2C. Mỗi thiết bị này được gọi là Nút. Trong thuật ngữ I2C, mỗi nút có thể hoạt động như một trong hai master hoặc slave. Master là một thiết bị tạo xung cho hệ thống, nó cũng khởi tạo và huỷ việc truyền nhận. Slave là nút nhận xung và địa chị bởi master.

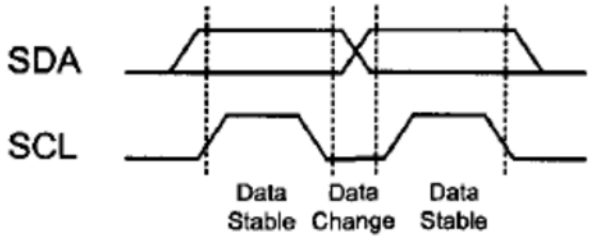
Trong I2C, cả master và slave đều có thể nhận hoặc truyền dữ liệu, vì vậy có bốn chế độ hoạt động. Đó là Master truyền (trasmitter), master nhận (receiver), slave truyền (transmitter), và slave nhận (receiver). Chú ý rằng mỗi nút có thể có một hoặc nhiều chế độ tại các thời điểm khác nhau, nhưng nó chỉ có một chế độ hoạt động cho thời gian đi. Xem ví dụ 1.1.

Ví dụ 1.1: Đưa ra một ví dụ để cho thấy các một thiết bị (nút) có thể có nhiều hơn một chế độ hoạt động.

Giải:
Nếu chúng ta kết nối AVR/PIC18 tới một EEPROM với I2C, AVR hoạt động là một master truyền để ghi vào EEPROM. AVR cũng hoạt động là master nhận để đọc từ EEPROM. Trong các phần sau, chúng ta sẽ thấy rằng một nút có thể thực hiện các hoạt động master hoặc slave tại những thời điểm khác nhau.

1.4. ĐỊNH DẠNG BIT

I2C là một giao thức đồng bộ nối tiếp (Synchronous serial), mỗi bit dữ liệu được truyền trên đường SDA được đồng bộ bởi xung (Pulse) mức cao xuống thấp (high-to-low) trên đường SCL. Theo giao thức I2C đường dữ liệu không thể thay đổi khi đường xung (Clock line) là mức cao, nó chỉ có thể thay đổi khi đường xung là mức thấp. Xem hình 1.2. điều kiện STOP và START (STOP condition, START condition) là ngoại lệ duy nhất cho quy tắc này.



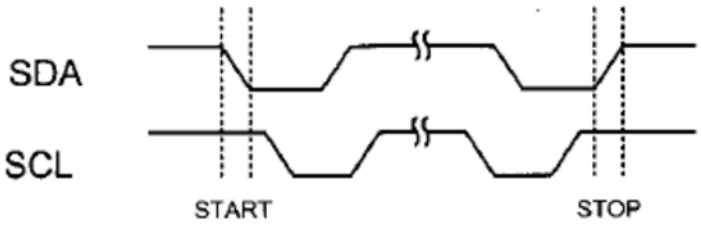
Hình 1.2. Định dạng bit I2C

1.5. ĐIỀU KIỆN START (BẮT ĐẦU) VÀ STOP (KẾT THÚC)

Như chúng ta đã đề cập ở trước, I2C là một giao thức truyền thông định hướng kết nối. Có nghĩa là mỗi lần truyền được khởi tạo bởi điều kiện START và được huỷ bởi điều kiện STOP. Hãy nhớ rằng điều kiện START và STOP được tạo bởi master.

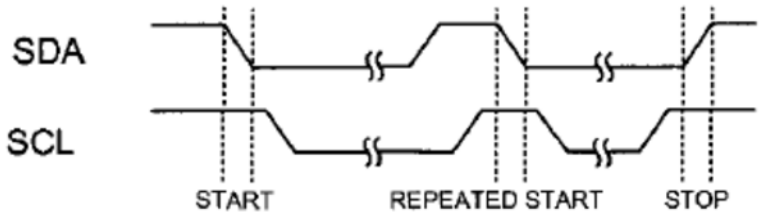
Điều kiện STOP và START phải được phân biệt với các bit địa chỉ hoặc dữ liệu. Đó là lý do tại sao chúng không tuân theo quy tắc định dạng bit mà chúng ta đã đề cập ở trước.

Điều kiện START và STOP được tạo ra bằng cách giữ mức của đường SCL cao và sau đó thay đổi mức của đường SDA. Điều kiện STAR được tạo ra bởi một sự thay đổi cao xuống thấp (high-to-low) trong đường SDA khi SCL là mức cao. Điều kiện STOP được tạo ra bởi sự thay đổi thấp lên cao (low-to-high) trong đường SDA khi SCL mức thấp. Xem hình 1.3.



Hình 1.3. Điều kiện START và STOP

Bus được xem là bận giữa mỗi cặp điều kiện START và STOP, và không có master nào khác cố gắng kiểm soát bus khi nó bận. Nếu một master, trong đó có sự kiểm soát bus, mong muốn khởi tạo một truyền nhận (transfer) mới và không muốn thoát ra khỏi bus trước khi bắt đầu một truyền nhận mới, nó là vấn đề mới thoát khỏi điều kiện START giữa cặp điều kiện START và STOP. Nó được gọi là điều kiện REPEATED START. Xem hình 1.4.



Hình 1.4. Điều kiện REPEATED START

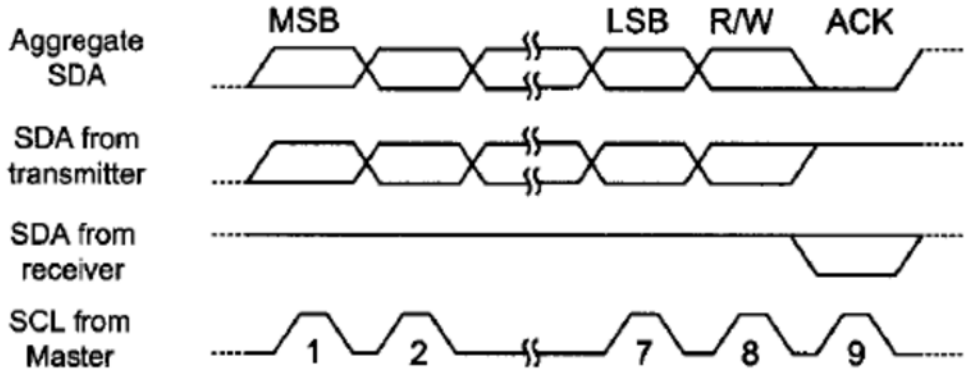
Ví dụ 1.2 cho thấy tại sao điều kiện REPEATED START là cần thiết.

Ví dụ 1.2: Đưa ra một ví dụ để trình bày khi một master phải sử dụng điều kiện REPEATED START. Điều gì sẽ xảy ra nếu master không sử dụng nó?

Giải:
Nếu chúng ta kết nối hai AVR (PIC18) (gồm AVR A và AVR B) và một EEPROM với I2C, và AVR A muốn hiển thị việc bổ sung nội dung của địa chỉ 0x34 và 0x35 của EEPROM, nó phải sử dụng điều kiện REPEATED START. Hãy xem điều gì có thể xảy ra nếu AVR A không sử dụng điều kiện REPEATED START. AVR A truyền điều kiện START, đọc nội dung địa chỉ 0x34 của EEPROM vào R16, và truyền điều kiện STOP để thoát khỏi bus. Trước khi AVR đọc nội dung của địa chỉ 0x35 vào R17, AVR B sẽ chiếm giữ bus và thay đổi nội dung các địa chỉ 0x34 và 0x35 của EEPROM. Sau đó AVR A đọc nội dung địa chỉ 0x35 vào R17, thêm nó vào R16 và hiển thị kết quả trên màn hình LCD. Kết quả trên màn hình LCD không phải là tổng giá trị cũ của địa chỉ 0x34 và 0x35 mà là tổng các giá trị mới của địa chỉ 0x34 và 0x35 của EEPROM.

1.6. ĐỊNH DẠNG GÓI (PACKET) TRONG I2C

Trong I2C, mỗi địa chỉ hoặc dữ liệu được truyền đi phải được đóng khung (Framed) trong một gói (Packet). Mỗi gói tin có độ dài 9-bit. 8-bit đầu tiên được đặt trên đường SDA bởi bộ truyền (Transmitter), và bit thứ 9 là một báo nhận bởi bộ nhận (Receiver) hoặc nó có thể là NACK (Not acknowledge). Xung được tạo ra bởi master, bất kể đó là bộ truyền hay nhận. Để lấy được một báo nhận, bộ truyền thoát khỏi đường SDA trong xung thứ chín để bộ nhận có thể kéo đường SDA xuống thấp để chỉ ra một ACK. Nếu bộ nhận không thể kéo đường SDA xuống thấp, nó được coi là NACK,. Xem hình 1.5.

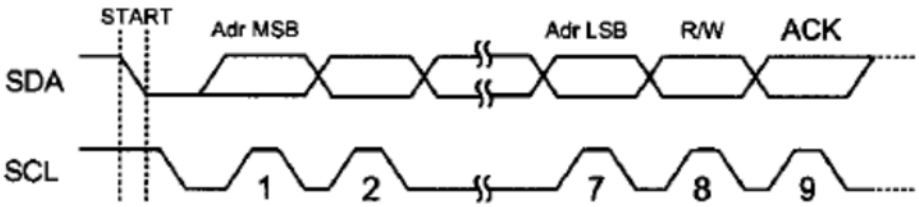


Hình 1.5. Định dạng gói trong I2C

Trong I2C, mỗi gói có thể chứa địa chỉ hoặc dữ liệu. Cũng chú ý rằng điều kiện START + địa chỉ gói + một hoặc nhiều gói + điều kiện STOP cùng nhau tạo thành một truyền dữ liệu hoàn chỉnh. Tiếp theo chúng ta se nghiên cứu địa chỉ và các định dạng gói dữ liệu và cách kết hợp chúng để làm lên một truyền nhận hoàn chỉnh.

1.6.1. Định dạng gói địa chỉ

Giống như tất cả gói khác, tất cả gói địa chỉ được truyền trên bus I2C có độ dài 9-bit. Một địa gói địa chỉ bao gồm bảy bit, một bit điều khiển READ/WRITE, và một bit báo nhận (acknowledge) (xem hình 1.6).



Hình 1.6. Định dạng gói địa chỉ trong I2C

Các bit địa chỉ được sử dụng để xử lý một thiết bị slave cụ thể trên bus. 7-bit địa chỉ cho phép địa cho phép địa chỉ master tối đa 128 slave trên bus, mặc dù địa chỉ 0000 000 được dành riêng cho cuộc gọi chung và tất cả địa chỉ định dạng 1111 xxx được dành riêng. Có nghĩa là 119 (128 – 1 – 8) thiết bị có thể được chia sẻ trên một bus I2C. Trong bus I2C các MSB của địa chỉ được truyền đi đầu tiên.

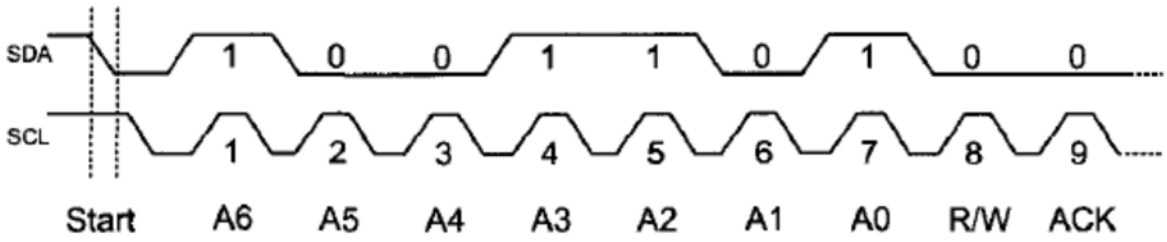
Bit thứ tám trong gói dữ liệu là bit điều khiển READ/WRITE. Nếu bit này được đặt, master sẽ đọc khung tiếp theo (dữ liệu) từ slave, ngược lại, master sẽ ghi khung (dữ liệu) tiếp theo trên bus tới slave. Khi một slave phát hiện địa chỉ của nó trên bus, nó biết rằng nó đang được địa chỉ (Addressed) và nó phải báo nhận trong chu kỳ thứ chín của SCL (ACK) bằng cách thay đổi SDA về không. Nết địa chỉ slave không sẵn sàng hoặc vì bấy kỳ lý do nào đó không muốn phục vụ master, nó để đường SDA cao trong chu kỳ xung thứ 9. Đây được coi là NACK. Trong trường hợp NACK, master có thể truyền một điều kiện STOP để huỷ quá trình truyền nhận, hoặc điều kiện REPEATED START để khởi tạo truyền nhận mới.

Ví dụ 1.3 cho thấy là thế nào một master nói rằng nó muốn ghi vào một slave.

Ví dụ 1.3: Trình bày cách để master nói rằng nó muốn ghi vào salve với địa chỉ 1001101.

Giải:
Cách hành động sau đây sẽ được thực hiện bởi master:

1. Master sẽ xuất một xung cao xuống thấp (high-to-low) trên SDA, trong khi SCL cao để tạo ra một bit điều kiện START để bắt đầu truyền.
2. Master truyền 10011010 vào bus, Bảy bit đầu tiên (1001101) cho biết địa chỉ slave, và bit thứ tám (0) cho biết thao tác ghi và nói rằng master sẽ ghi byte (dữ liệu) tiếp theo vào slave.



Một gói địa chỉ bao gồm một địa chỉ slave và một READ được gọi là SLA+R, trong một gói địa chỉ bao gồm một địa chỉ slave và một WRITE được gọi là SLA+W.

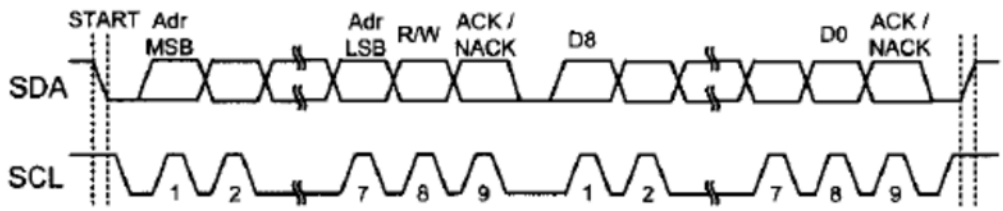
Như chúng ta đã đề cập trước đây, một địa chỉ 0000 000 được dành riêng cho cuộc gọi chung. Nghĩa là khi một master truyền địa chỉ 0000 000, tất cả các slave sẽ đáp ứng bằng cách thay đổi đường SDA về không và chờ để nhận byte dữ liệu. Điều này hữu dụng khi master muốn truyền cùng một byte dữ liệu vào tất cả slave trong hệ thống. Chú ý rằng không thể sử dụng địa chỉ cuộc gọi chung để đọc dữ liệu từ slave bởi vì không có nhiều hơn một slave có khả năng ghi vào bus tại cùng một thời điểm nhất định.

1.6.2. Định dạng gói dữ liệu

Giống như các gói khác, gói dữ liệu cũng có độ dài 9-bit. 8-bit đầu tiên là byte dữ liệu được truyền đi, và bit thứ 9 là ACK. Nếu bộ nhận đã nhận được byte cuối cùng của dữ liệu và không có nhiều dữ liệu được nhận, hoặc bộ nhận không thể nhận hoặc xử lý nhiều dữ liệu hơn, nó sẽ báo hiệu NACK bằng cách thoát khỏi đường SDA cao. Trong các gói dữ liệu, giống như các gói địa chỉ, MSB được truyền đầu tiên.

1.6.3. Kết hợp các gói địa chỉ và dữ liệu vào một truyền nhận

Trong I2C, thông thường, quá trình truyền được bắt đầu bằng điều kiện START, theo sau bởi một gói địa chỉ (SLA + R/W), một hoặc nhiều gói dữ liệu, và được kết thúc bởi một điều kiện STOP. Hình 1.7 cho thấy một truyền dữ liệu điển hình. Phải cố gắng hiểu mỗi phần tử trong hình (xem ví dụ 1.4).



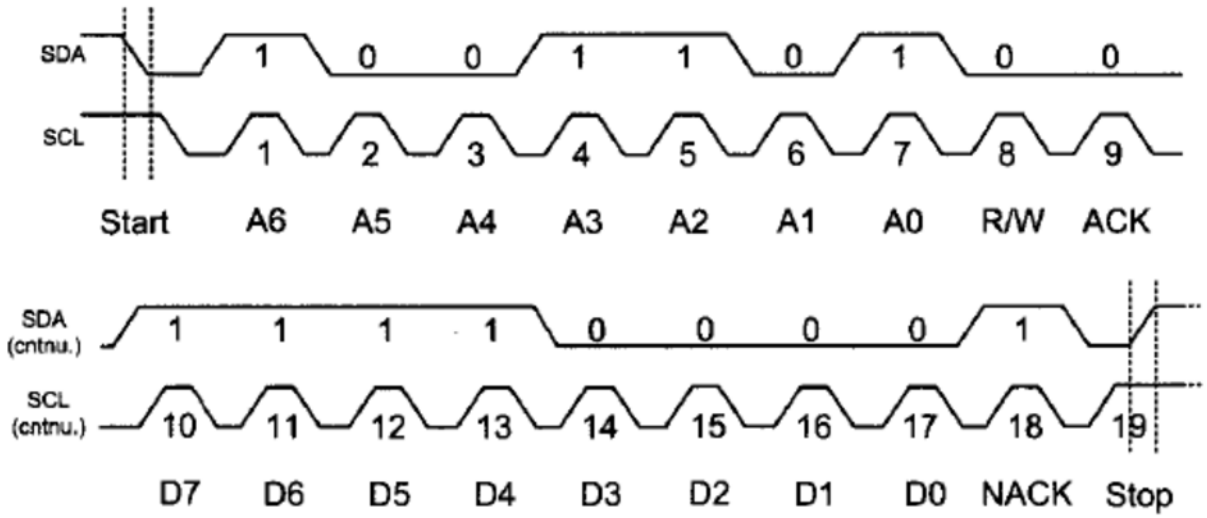
Hình 1.7. Truyền dữ liệu điển hình

Ví dụ 1.4: Trình bày cách một master ghi giá trị 11110000 vào một slave với địa chỉ

1001101.

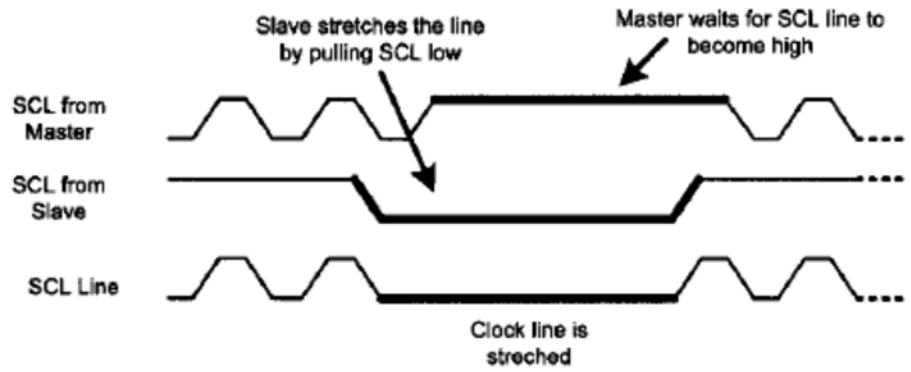
Giải:
Các hành động sau đây được thực hiện bởi master:

1. Master xuất một xung cao xuống thấp (high-to-low) trên SDA trong khi, SCL cao để tạo ra một điều kiện START để bắt đầu truyền.
2. Master truyền 10011010 vào bus. Bảy bit đầu tiên (1001101) cho biết địa chỉ của slave và bit thứ tám (0) cho biết bắt đầu hoạt động ghi rằng master sẽ ghi byte (dữ liệu) tiếp theo vào slave.
3. Slave kéo đường SDA thấp để báo hiệu một ACK nói rằng nó đã sẵn sàng để nhận byte dữ liệu.
4. Sau khi nhận được ACK, master sẽ truyền byte dữ liệu (11110000) trên đường SDA (MSB đầu tiên).
5. Khi thiết bị slave nhận dữ liệu, nó rời khỏi đường SDA cao để báo hiệu NACK. Điều này thông báo cho master rằng slave đã nhận được byte dữ liệu cuối cùng và không cần thêm bất kỳ dữ liệu nào.
6. Sau khi nhận được NACK, master sẽ biết rằng không có thêm nào dữ liệu được truyền đi. Master thay đổi đường SDA khi đường SCL cao để truyền điều kiện STOP và thoát khỏi bus.



1.7. KÉO DÀI XUNG (CLOCK STRETCHING)

Một trong những tính năng của giao thức I2C là kéo dài xung. Nó là một loại kiểm soát lưu lượng. Nếu một thiết bị slave được địa chỉ không sẵn sàng để xử lý nhiều dữ liệu hơn nó sẽ kéo dàu xung bằng cách giữ đường xung (SLC) thấp sau khi nhận (hoặc gửi) một bit của dữ liệu. Vì vậy master sẽ không có khả năng nâng đường xung (vì thiết bị có dây-ANDed) và sẽ đợi cho đến khi slave thoát khỏi đường SCL để cho thấy nó đã sẵng sàng để truyền bit tiếp theo. Xem hình 1.8.



Hình 1.8. Kéo dài xung

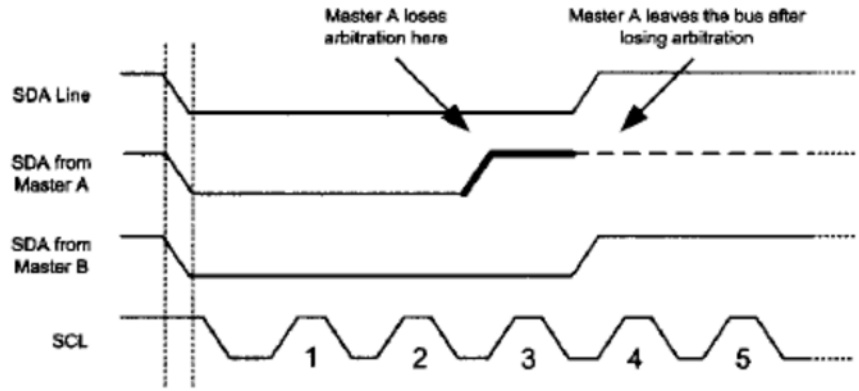
1.8. SỰ PHÂN XỬ (ARBITRATION)

Giao thức I2C hỗ trợ một thệ thống bus đa master. Điều này không có nghĩa là nhiều hơn một master có thể sử dụng bus tại cùng một thời điểm. Thay vào đó, mỗi master sẽ chờ đợi để truyền hiện tại kết thúc và sau đó bắt đầu sử dụng bus. Nhưng có có thể hai hoặc nhiều master khởi tạo một truyền tại cùng một thời điểm, Trong trường hợp này sự phân xử xảy ra.

Mỗi bộ truyền phải kiểm tra mức của bus và so sánh nó với mức mong đợi, nếu nó không khớp, bộ truyền đó đã mất sự phân xử, và sẽ chuyển ssang chế độ slave. Trong trường hợp của sự phân xử, master chiến thắng sẽ tiếp tục công việc của nó. Chú ý rằng cả bus bị hỏng cũng không bị mất dữ liệu. Xem ví dụ 1.5.

Ví dụ 1.5: Hai master, A và B, bắt đầu vào cùng một thời điểm. Điều gì sẽ xảy ra nếu master A muốn ghi vào slave 0010 000 và master B cũng muốn ghi vào slave 0001 111?

Giải:
Master sẽ thua trong sự phân xử trong lần xung thứ ba nởi vì đường SDA khác với đầu ra của master. Tại thời điểm xung thứ ba. Master A chuyển sang chế độ slave và rời khỏi bus sau khi thua sự phân xử.



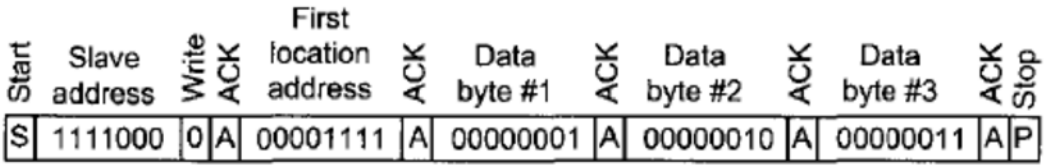
1.9. GHI KHỐI NHIỀU BYTE

Chế độ ghi khối là một phương tiện hiệu quả để nạp các vị trí liên tiếp. Nó được hỗ trợ trong I2C, SPI và nhiều giao thức nối tiếp khác. Trong chế độ khối, chúng ta cung cấp địa chỉ của vị trí đầu tiên, theo sau đó là byte dữ liệu cho vị trí đó. Rồi sau đó, các byte liên tiếp được ghi vào các bị trí bộ nhớ liên tiếp. Trong chế độ này, thiết bị I2C

tăng dần vị trí địa chỉ miễn là điều kiện STOP không được phát hiện. Các bước sau được sử dụng để gửi (ghi) nhiều byte dữ liệu trong chế độ khối cho các thiết bị I2C.

1. Tạo một điều kiện START.
2. Truyền địa chỉ slave theo sau là không (cho ghi).
3. Truyền địa chỉ của vị trí đầu tiên.
4. Truyền dữ liệu cho vị trí đầu tiên và từ đó trổ đi, chỉ cần cung cấp liên tiếp byte dữ liệu đặt vào trong các vị trí bộ nhớ liên tiếp.
5. Tạo một điều kiện STOP.

Hình 1.9 trình bày cách ghi 0x01, 0x02, và 0x03 đến ba vị trí liên tiếp bắt đầu từ vị trí 00001111 của slave 1111000.



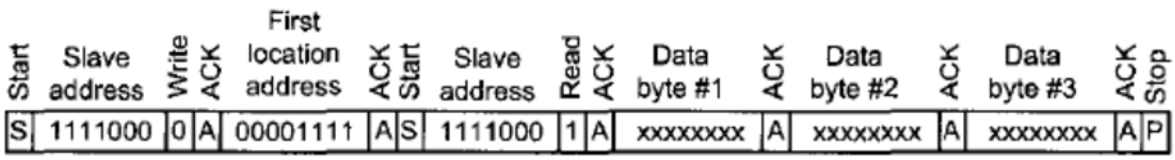
Hình 1.9. Ghi khối nhiều byte

1.10. ĐỌC KHỐI NHIỀU BYTE

Chế độ đọc khối là một cách hiệu quả để mang ra các nội dung liên tục của các vị trí. Trong chế độ khối, chúng ta chỉ cần cung cấp địa chỉ của vị trí đầu tiên. Từ đó trở đi, nội dung được mang ra từ các vị trí bộ nhớ liên tiếp. Trong chế độ này, thiết bị I2C tăng dần vị trí địa chỉ miễn là điều kiện STOP không được phát hiện. Các bước sau được sử dụng để lấy (đọc) nhiều byte dữ liệu sử dụng chế độ khối cho thiết bị I2C.

1. Tạo một điều kiện START.
2. Truyền địa chỉ slave theo sau là không (cho ghi địa chỉ).
3. Truyền địa chỉ của vị trí đầu tiên.
4. Tạo một điều kiện START (REPEATED START).
5. Truyền địa chỉ slave theo sau là một (cho đọc).
6. Đọc dữ liệu từ vị trí đầu tiên và từ đó trở đi, mang rội dung ra từ vị trí bộ nhớ liên tiếp.
7. Tạo một điều kiện STOP.

Hình 1.10 trình bày cách đọc ba vị trí liên tiếp bắt đầu từ vị trí 00001111 của master.



Hình 1.10. Đọc khối nhiều byte