글

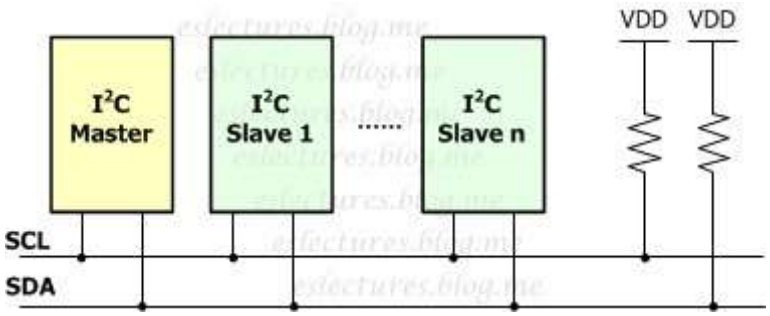
I2C(Inter Integrated Circuit)

T heory/Comn Basics

설정

(1) I2C
: I2C(Inter Integrated Circuit)버스는 마이크로프로세서와 저속 주변장치 사이의 통신을 용도로 Philips에서 개발한 규격인데 두 가닥의 선을 사용하므로 TWI(Two Wire Interface)라고도 불린다. I2C버스는 양방향 오픈 드레인 선인 SCL(Serial Clock)과 SDA(Serial Data)로 이루어져 있으며, Master-Slave 형태로 동작한다. 속도면에서는 다른 방식에 비하여 현저히 느리지만 하드웨어적으로 간단한 구성과 대화형 protocol을 만들 수 있고 하나의 버스에 많은 수의 노드를 연결 할 수 있다는 큰 장점이 있다.

(2) I2C Interface 구성
: SCL은 통신의 동기를 위한 클럭용 선이고 SDA는 데이터용 선이다. Master는 SCL로 동기를 위한 클럭을 출력하며 Slave는 SCL로 출력되는 클럭에 맞추어 SDA를 통해 데이터를 출력하거나 입력 받는다. 단지 SDA 한 선으로만 데이터를 주고 받기 때문에 I2C 버스는 반이중 (half duplex) 통신만 가능하다. SCL 선과 SDA 선은 모두 오픈 드레인므로 두 선에는 각각 풀업 저항을 연결해 주어야 한다.



모든 I2C 매스터와 슬레이브 장치들의 SCL은 서로 연결되며 SDA 또한 서로 연결된다. 모든 장치들이 SCL과 SDA를 각각 공유하고 있으므로 매스터가 슬레이브를 개별적으로 지정하기 위한 방법이 있어야 한다. I2C 버스에서 매스터가 각 슬레이브를 구분하는 수단은 슬레이브의 주소이다. 모든 I2C 슬레이브 장치들은 7 비트의 고유한 주소를 가지고 있으며 매스터는 주소로 원하는 슬레이브를 지정한다. 주소의 길이가 7 비트이므로 매스터는 최대 128 개의 슬레이브 장치들과 연결될 수 있다. 물론 슬레이브 장치들의 주소는 모두 달라야 한다.

(3) I2C 통신방식

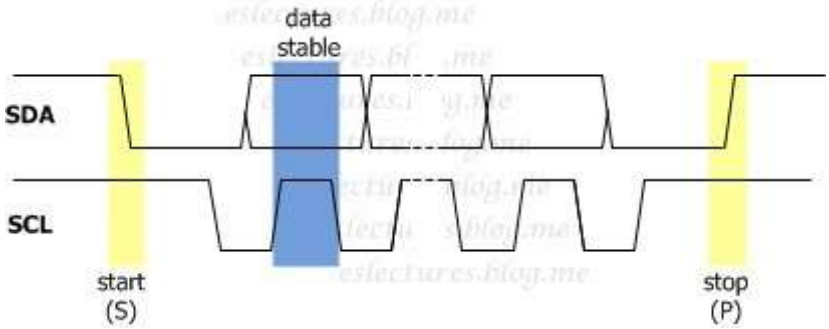
: 앞의 그림에는 매스터가 하나 밖에 없지만 I2C 버스에 두 개 이상의 매스터들이 존재할 수도 있다. 버스에 연결된 장치의 수가 아무리 많아도 한 순간에는 오직 하나의 매스터와 하나의 슬레이브만이 통신할 수 있다. 현재 통신을 진행하고 있는 매스터가 버스 소유권을 가지고 있다. 두 개 이상의 매스터가 있는 경우에 현재 버스의 소유권을 가지고 있는 매스터 및 매스터와 통신을 진행하고 있는 슬레이브 장치 이외의 나머지 장치들은 이들의 통신을 방해하지 않아야 한다.

SCL과 SDA는 모두 오픈 드레인이므로 모든 장치들의 SCL과 SDA는 각각 와이어드 AND로 연결 되어 있다. 와이어드 AND 연결에서는 어느 한 장치라도 '0'을 출력하면 해당 신호의 상태는 논리 '0'이 된다. 만일 SCL이나 SDA에 연결되어 있는 어떤 장치가 논리 '0'을 출력한다면 다른 장치가 그 신호의 상태를 논리 '1'로 만들 수 있는 방법은 없다. 통신에 참여하지 않은 장치가 SCL이나 SDA로 '0'을 출력하면 매스터가 정상적으로 통신을 진행할 수 없다. 이것은 I2C 버스에 연결되어 있지만 현재 통신에 참여하지 않고 있는 장치들은 모두 자신의 출력을 플로우팅 상태로 유지해야 한다는 것을 뜻한다. (플로우팅 상태 : 신호가 low인지 high 인지 알수 없는 상태)

통신이 진행되지 않는 상황에서 모든 장치의 출력은 플로우팅 상태이므로 SCL과 SDA의 상태는 모두 논리 '1'이다. 이 상황에서 I2C 버스의 사용을 원하는 매스터는 SCL과 SDA로 시작 조건을 출력하며 버스 소유권을 주장하고 통신을 시작할 수 있다. 두 신호의 상태가 모두 논리 '1'이 아니라면 현재 다른 매스터가 버스 소유권을 가지고 통신을 진행 중에 있다는 것을 뜻하므로 그 매스터가 버스 소유권을 반납할 때까지는 다른 매스터가 새로운 통신을 시작해서는 안 된다.

(3) 안정화 구간

: 다음 그림은 I2C 프로토콜의 시작 조건, 정지 조건, 그리고 데이터 안정 구간을 보여주고 있다.

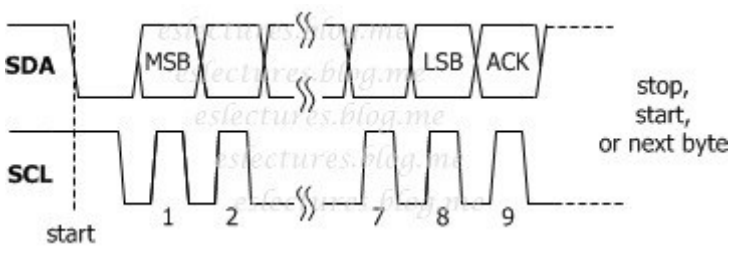


I2C 프로토콜에서 SCL이 '0'인 구간에서는 SDA의 상태 변화가 허용되지만 SCL이 '1'인 구간에서는 SDA는 안정된 논리 상태를 유지해야 한다. Master가 Slave로 데이터를 출력할 때 SCL이 '0'인 구간에서 SDA의 비트 전환을 하며 SCL이 '1'인 구간에서는 SDA의 상태를 그대로 유지한다. Slave가 데이터를 출력하고 Master가 그 데이터를 읽을 때에도 마찬가지이다. 그래서 SCL이 '1'인 구간은 데이터가 안정한 구간이다.

I2C 프로토콜에서 SCL이 '1'을 유지하고 있는 구간에서 SDA의 상태가 변하는 것은 일반 데이터 전송이 아닌 특별한 조건을 의미한다. SCL이 '1'인 동안 SDA가 '1'에서 '0'으로 바뀌는 것을 시작 조건 (start condition, S)이라고 하며 SCL이 '1'인 동안 SDA가 '0'에서 '1'로 바뀌는 것을 정지 조건 (stop condition, P)이라고 한다. I2C 통신을 원하는 Master는 SCL과 SDA가 모두 논리 '1'일 때 SDA의 상태를 '0'으로 바꾸어 시작 조건을 출력하며 다른 장치들에게 통신의 시작을 알린다. 마찬가지로 통신을 끝낼 때에는 SCL이 '1'인 동안 SDA를 '0'에서 '1'로 바꾸어 정지 조건을 출력하며 버스 소유권의 반납을 다른 장치에게 알린다.

(4) 패킷형식

: 다음 그림은 I2C 버스에서 사용되는 데이터 패킷의 형식을 보여준다. ACK을 포함한 9 비트가 I2C 규격에서 통신의 기본 단위가 된다.



Master는 SCL과 SDA가 모두 '1'인 상태에서 SDA를 '0'으로 바꾸어, 즉 시작 조건을 출력하면서 통신의 시작을 알린다. 시작 조건 이후부터는 SCL의 상태가 '0'인 구간에서만 SDA의 논리 값이 바뀐다. 매스터가 SCL로 출력하는 클럭에 동기를 맞추어 SDA로는 데이터가 MSB부터 한 비트씩 출력된다. SCL은 항상 Master가 출력하는 신호이지만 SDA는 현재의 동작이 Master의 읽기냐 쓰기냐에 따라 매스터의 출력일 수도 슬레이브의 출력일 수도 있다.

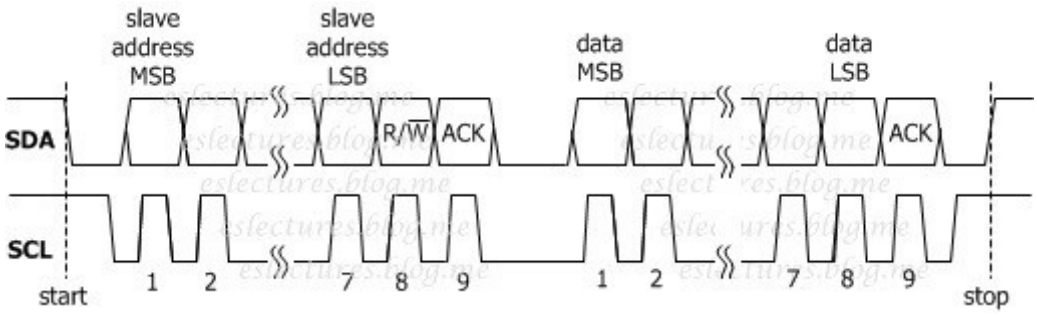
8 비트 데이터가 8 클럭 싸이클 동안 SDA로 출력되면 그 데이터를 수신한 쪽에서 9 번째 클럭에 맞추어 그 8 비트 데이터의 수신 여부를 확인해 주는 ACK 비트를 출력한다. 다시 말해서 ACK 비트는 직전의 8 비트 정보를 수신한 쪽에서 그 정보의 송신자에게 직전 8 비트 정보에 대한 수신 여부를 확인하는 값이다. ACK 비트가 '0'이면 정상 수신을 나타내며 ACK 비트가 '1'이면 (항상 그런 것은 아니지만) 정상 수신이 아님을 나타낸다. 값이 '0'인 ACK 비트와 구분하기 위하여 값이 '1'인 ACK 비트를 NACK (no acknowledgement)라고 부르기도 한다.

만일 매스터가 SDA로 8 비트 정보를 출력했다면 그 정보를 수신한 슬레이브가 9 번째 클럭에 SDA로 ACK 비트를 출력하여 매스터에게 수신 결과를 알린다. 이 때 매스터가 ACK 비트를 받으려면 9 번째 클럭 싸이클 동안 자신의 SDA 출력을 '1'로 유지하고 있어야 한다. 반대로 SDA로 출력된 8 비트 정보가 슬레이브의 출력이라면 그것을 수신한 매스터가 9 번째 클럭에서 ACK 비트를 출력한다. 이 때 매스터의 ACK 비트를 받기 위해 슬레이브는 자신의 SDA 출력을 '1'로 유지하고 있어야 한다.

8 비트 정보와 이어지는 ACK 비트의 전송이 끝난 다음에는 매스터가 정지 조건을 출력할 수도 있고 시작 조건을 다시 출력할 수도 있으며 다음 바이트 전송이 이어질 수도 있다. 다음 동작은 현재 진행 중인 동작이 무엇인가에 따라 달라진다.

(5) 주소 지정 형식

: 모든 I2C 슬레이브 장치는 7 비트의 고유한 주소를 가지고 있으며 매스터는 이 주소를 사용하여 상대 슬레이브 장치를 지정한다. 다음 그림은 매스터가 슬레이브의 주소를 지정하고 데이터를 주고 받는 방법에 대한 설명이다.



먼저 Master는 시작 조건에 이어서 자신이 원하는 Slave의 7 비트 주소를 출력한다. 버스에 연결되어 있는 모든 Slave 장치들은 SDA 선을 계속 감시하면서 Master가 출력한 주소가 자신의 주소와 일치하는지 여부를 검사한다. 만일 Master가 출력한 주소가 어떤 Slave의 주소와 같으면 그 Slave는 ACK 비트에 '0'을 출력하여 Master에게 응답한다. Master가 출력한 주소를 가진 Slave가 없으면 아무도 ACK 비트로 '0'을 출력하지 않으므로 ACK 비트의 상태는 '1'을 유지하여 Master는 NACK을 받게 된다.

Slave 주소 7 비트 다음에 오는 8 번째 비트는 다음 동작이 Master의 읽기인지 쓰기인지를 가리킨다. Master가 이 비트로 0을 출력했다면 이것은 Master 쓰기임을 뜻하므로 Slave 주소 다음의 1 바이트 데이터는 Master의 출력이다. 이 데이터는 앞서 주소가 지정된 Slave를 위한 것이므로 그 Slave가 이 값을 읽어 들인다. 반면에 Slave 주소 7 비트 다음에 오는 8 번째 비트가 1 이면 이것은 다음 동작이 Master의 읽기라는 것을 의미한다. 그러므로 Master에 의해 주소가 지정된 Slave는 ACK을 응답한 후 이어서 Master에게 보낼 데이터를 출력한다.

주의할 것은 Slave 주소는 7 비트이지만 패킷의 상위 7 비트로 정렬되고 마지막에 R/W 비트가 LSB 위치에 추가된다는 것이다. 그렇기 때문에 Slave의 주소를 왼쪽으로 한 비트 쉬프트 시킨 후의 8 비트 값을 Slave의 주소라고 생각하는 것이 더 편할 수도 있다. 예를 들면 DS1037이라는 RTC 칩의 I2C 주소는 1101000b으로 고정되어 있다. 즉 DS1037의 I2C 주소는 0x68이다. DS1037에 데이터를 기록해야 할 때 Master는 I2C 버스로 시작 조건을 출력한 다음 11010000b을 출력한다. DS1037의 데이터를 읽고자 한다면 Master는 I2C 버스로 시작 조건을 출력한 다음 11010001b을 출력한다. 그러므로 8 비트 값인 0xd0을 DS1037의 쓰기 주소, 0xd1을 DS1037의 읽기 주소로 생각할 수도 있다.

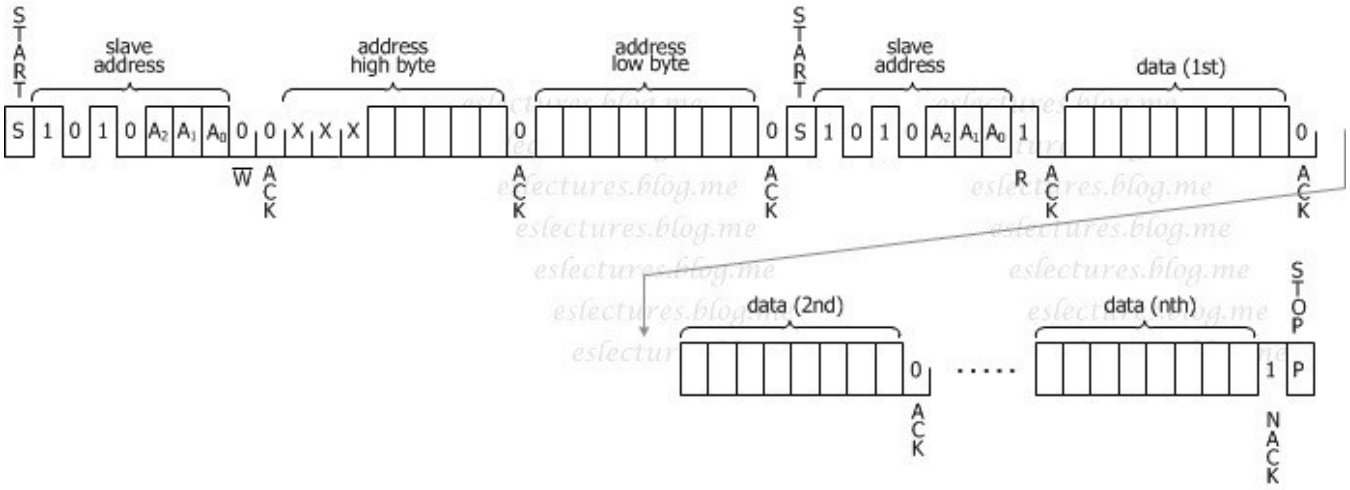
I2C 버스 규격은 I2C 버스의 전기적 특성 및 타이밍과 함께 앞에서 설명한 시작 조건, 정지 조건, 주소 지정 방법 등에 대해서만 규정하고 있다. 구체적인 통신 방법, 즉 Master가 Slave의 주소를 출력한 다음 정지 비트를 출력할 때까지

주고 받는 데이터의 바이트 수, Slave의 내부 레지스터 주소 지정 방법 등은 Slave 장치에 따라 달라진다. 매우 다양한 종류의 I2C Slave 장치가 있으니 여러 가지 다양한 방법이 있을 수 있다.

대부분의 마이크로컨트롤러는 I2C 인터페이스를 지원한다. 그렇기 때문에 UART를 사용할 때와 비슷하게 I2C 관련 몇 개의 레지스터를 설정한 후 레지스터 읽기/쓰기를 통해 간단하게 I2C 통신을 실행할 수 있다. 마이크로컨트롤러가 I2C 인터페이스를 지원하지 않으면 프로그램을 통해 비교적 쉽게 I2C 프로토콜을 구현할 수 있다. 송?수신 타이밍을 정확하게 맞추어야 하는 비동기 통신과 달리 Master가 출력하는 클럭에 맞추어 모든 동작이 이루어지므로 타이밍에 관련된 어려움이 특별히 없기 때문이다.

(6) I2C 통신의 예

: 직렬 EEPROM, ADC, DAC, RTC 등을 포함한 다양한 매우 장치들이 I2C 인터페이스를 지원하고 있다. I2C 슬레이브 장치와 통신을 하려면 그 슬레이브의 데이터쉬트를 참고하여 구체적인 통신 방법을 이해해야 한다. 다음 그림은 24LC64라는 64 Kbit 직렬 EEPROM의 내부 데이터를 읽기 위한 여러 방법 중 하나이다.



24LC64의 주소는 1010A2A1A0b로 정해져 있다. 앞 4 비트는 1010로 고정되어 있지만 뒤의 3 비트 A2A1A0는 사용자가 임의로 바꿀 수 있다. 주소의 마지막 3 비트를 사용자가 조절할 수 있도록 24LC64는 이 3 비트에 각각 대응되는 3 개의 입력 핀 A2, A1, A0을 가지고 있다. 주소의 일부를 사용자가 정하도록 만든 것은 24LC64에 여러 가지 다른 주소를 부여할 수 있도록 하기 위한 것이다. 만일 24LC64의 주소가 완전히 고정되어 있다면 I2C Master가 구동할 수 있는 24LC64는 1 개뿐이다.

위의 그림은 24LC64의 저장된 값을 특정 주소부터 연속으로 읽는 방법이다. 맨 먼저 Master는 시작 조건을 출력한 다음 24LC64의 7 비트 주소를 출력하고 8 번째 비트로 '0'을 출력하여 다음 동작이 쓰기라는 것을 알린다. Master가 지정한 주소와 동일한 주소를 가진 24LC64가 있다면 그 IC는 ACK을 출력하여 Master에게 응답한다. ACK을 받은 Master는 이어서 24LC64의 저장 공간 중에서 자신이 원하는 주소를 MSB부터 연속해서 출력한다. 24LC64의 저장 용량은 64 Kbits, 즉 8 K 바이트이므로 내부 주소 지정을 위해서는 13 비트의 주소 정보가 필요하다. 그러므로 Master는 두 번에 걸쳐 주소를 출력한다. 매 바이트를 받을 때마다 24LC64는 Master에게 ACK을 보낸다.

24LC64 내부의 주소를 출력한 후 ACK을 받으면 Master는 시작 조건과 Slave 주소를 다시 한 번 더 출력한다. 그런데 이번에는 Master가 24LC64의 데이터를 읽을 차례이므로 7 비트 Slave 주소 다음에 오는 8 번째 비트는 '1'로 설정한다. 24LC64가 다시 Master에게 ACK으로 응답하면 그 다음부터 Master는 24LC64가 출력하는 데이터를 연속해서 읽는다. 24LC64의 출력을 1 바이트 받을 때마다 Master는 ACK을 출력하여 이를 24LC64에게 알린다. ACK을 받은 24LC64는 내부에서 주소를 1 증가시키고 새로운 주소의 데이터를 읽어서 다음 SCL 클럭에 맞추어 출력하게 된다. 데이터를 연속해서 수신한 Master는 자신이 원하는 마지막 데이터를 받으면 이번에는 NACK을 출력하고 이어 정지 조건을 출력하여 통신을 끝낸다.

위의 예는 I2C 통신을 사용하는 24LC64를 사용하기 위한 많은 방법들 중 하나일 뿐이다. I2C 인터페이스를 가진 장치마다 사용법이 다르기도 하거니와 한 장치에서도 명령의 종류에 따라 다양한 방법이 있을 수 있다.

(7) I2C의 장단점

○ I2C 버스의 가장 큰 장점으로 Slave 장치의 수에 관계 없이 두 가닥의 선만으로 Master와 Slave 사이의 양방향 통신이 가

능하다는 것을 들 수 있다. 두 개 이상의 Master를 사용할 수 있다는 사실은 I2C 버스가 가진 또 다른 장점이다.

○ 반면에 I2C 버스의 양방향 특성은 오픈 드레인 구조에 기반을 두고 있으므로 근본적으로 동시 양방향 통신이 불가능하고 통신 속도를 높이기 어렵다. 표준 모드에서 I2C 통신의 최대 속도는 100 kHz이며 고속 모드에서는 최고 400 kHz까지 가능하다. 다른 동기 통신 프로토콜, 예를 들어 뒤에서 설명할 SPI와 비교하면 매우 낮은 속도이므로 I2C 버스는 저속 주변장치와의 데이터 통신이나 주변 장치의 동작 모드 설정 등의 용도로 많이 쓰인다. 동작 모드 설정은 사용 빈도가 높지 않기 때문에 낮은 속도가 별 문제가 되지 않는다.

○ I2C 인터페이스를 사용하는 제품의 수는 무척 많은 반면 Slave의 주소는 7 비트로 그 길이가 제한되어 있어서 다수의 Slave 장치를 사용할 때 Slave 주소가 충돌할 가능성이 있다. 이런 문제에 대한 해결책으로 Slave 주소의 하위 2 ~ 3 비트를 사용자가 임의로 설정할 수 있는 제품들이 많기는 하지만 주소가 7 비트로 제한되어 있어서 근본적인 해결책이 될 수는 없다.

I2C_Bus_specification(ver2.1).pdf

※ 출처 : http://eslectures.blog.me/80137860924

29 구독하기

'T heory > Comn Basics' 카테고리의 다른 글

USB(Universal Serial Bus) (0)	2011.10.18
UART(Universal Asynchronous Receiver & Transmitter) (1)	2011.10.17
I2C(Inter Integrated Circuit) (18)	2011.10.17
GPIO(General Purpose I/O) (2) (0)	2011.10.17
SPI(Serial Peripheral Interface Bus) (2)	2011.10.17
I2S(Integrated Interchip Sound) (3)	2011.10.17

TRACKBACK :0 COMMENT :18
by Latte 하늘이도우사 2011. 10. 17. 17:38

댓글

- 뽀개기 2013.01.30 17:09 ADDR EDIT/DEL REPLY
  - Latte 하늘이도우사 2013.01.30 18:12 신고 EDIT/DEL
- 쩡도 2013.02.28 05:41 ADDR EDIT/DEL REPLY
- DK 2013.11.28 15:34 ADDR EDIT/DEL REPLY
- 안녕 2013.12.24 10:42 ADDR EDIT/DEL REPLY
  - Latte 하늘이도우사 2013.12.24 16:08 신고 EDIT/DEL
  - 안녕 2013.12.26 10:17 EDIT/DEL
  - Latte 하늘이도우사 2013.12.26 13:10 신고 EDIT/DEL
- 신선노름 2014.10.11 18:22 ADDR EDIT/DEL REPLY
  - Latte 하늘이도우사 2014.10.13 09:06 신고 EDIT/DEL
- wannab 2015.07.20 17:06 ADDR EDIT/DEL REPLY
- ㅎㅎ 2015.10.21 14:18 ADDR EDIT/DEL REPLY
- banan 2015.11.17 13:18 ADDR EDIT/DEL REPLY
- hardwareman 2016.04.04 11:38 ADDR EDIT/DEL REPLY
- AVRFREAK 2016.04.29 14:38 ADDR EDIT/DEL REPLY

- 데자와 2016.06.01 14:01 ADDR EDIT/DEL REPLY
- 트런들 2016.08.22 14:35 ADDR EDIT/DEL REPLY
- 면봉 2018.08.19 23:29 ADDR EDIT/DEL REPLY

name password homepage

http://

비밀글

comment

댓글 남기기