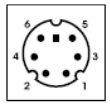
Design Examples

PS/2, UART

1. PS/2 Interface

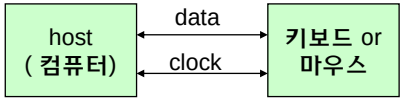
- PS/2 interface
  - IBM PS/2 개인용 컴퓨터용의 keyboard와 mouse 연결용으로 개발된 인터페이스
- pin 모양과 이름



Pin	Name	Description
1	Data	Key Data
2	N/C	Not Connect
3	GND	Ground
4	VCC	+5V DC
5	CLK	Clock
6	N/C	Not Connect

PS/2 데이터 전송

- 양방향 synchronous serial protocol 사용
  - 한 clock pulse동안 한 bit씩 전송

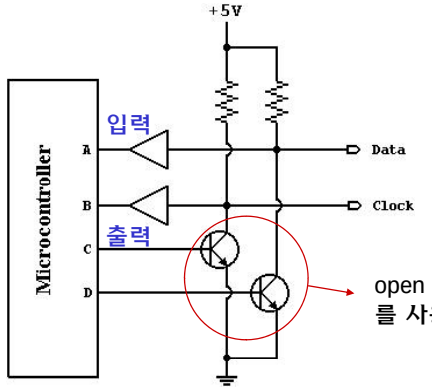


주의: PS/2 clock은 system clock과 다름

  - data와 clock은 open collector 출력 특성을 가짐
    - wired AND 동작
    - 한 라인을 양방향 전송에 사용 가능

PS/2 인터페이스 신호

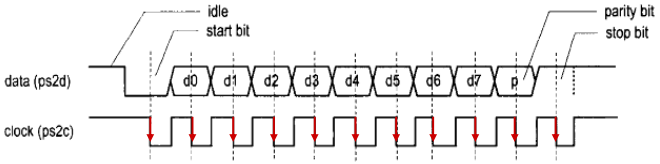
- data와 clock 신호 - 양방향



open collector 출력의 inverter 를 사용할 수 있음 (7407)

PS/2 데이터 전송

- idle 상태
  - clock=1, data=1 상태 유지
- Device to Host 전송 동작
  - device에서 clock과 data를 공급
  - clock이 high가 될 때 data가 바뀌며
    clock이 low가 될 때에는 data는 유지함(valid)
    → host에서 clock의 negative edge에서 data를 sampling하면 됨



start bit=0 odd parity stop bit=1

Scan Code

- Scan code
  - 키보드는 키보드를 누를 때와 땔 때에 키의 위치 정보인 scan code를 host로 전송

(예) 키보드 'A'

누를 때	땔 때
1C	F0 1C
(make code)	(break code)

- 일정시간 이상 누르고 있으면 make code가 계속해서 전송됨
- host에서 수신한 scan code를 저장한 후에, scan code를 ASCII코드로 변환한다.

키보드 자판의 키번호

Scan Code 표(1)

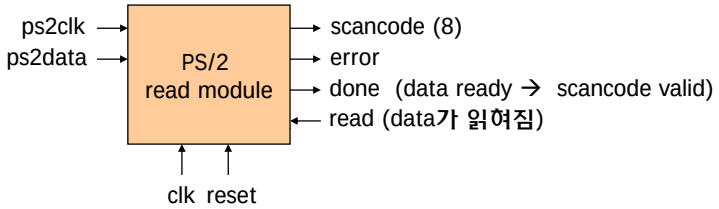
Key#	Make Code	Break Code	Key#	Make Code	Break Code
1	0E	F0 0E	31	1C	F0 1C
2	16	F0 16	32	1B	F0 1B
3	1E	F0 1E	33	23	F0 23
4	26	F0 26	34	2B	F0 2B
5	25	F0 25	35	34	F0 34
6	2E	F0 2E	36	33	F0 33
7	36	F0 36	37	3B	F0 3B
8	3D	F0 3D	38	42	F0 42
9	3E	F0 3E	39	4B	F0 4B
10	46	F0 46	40	4C	F0 4C
11	45	F0 45	41	52	F0 52
12	4E	F0 4E	43	5A	F0 5A
13	55	F0 55	44	12	F0 12
15	66	F0 66	46	1A	F0 1A
16	0D	F0 0D	47	22	F0 22

Scan code 표(2)

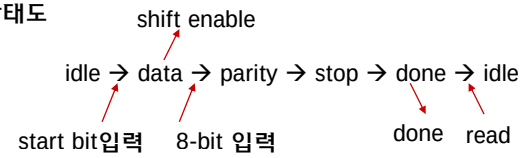
입력키	Make Code	Break Code	입력키	Make Code	Break Code
1	16	F0 16	H	33	F0 33
2	1E	F0 1E	I	43	F0 43
3	26	F0 26	J	3B	F0 3B
4	25	F0 25	K	42	F0 42
5	2E	F0 2E	L	4B	F0 4B
6	36	F0 36	M	3A	F0 3A
7	3D	F0 3D	N	31	F0 31
8	3E	F0 3E	O	44	F0 44
9	46	F0 46	P	4D	F0 4D
0	45	F0 45	Q	15	F0 15
–	4E	F0 4E	R	2D	F0 2D
=	55	F0 55	S	1B	F0 1B
A	1C	F0 1C	T	2C	F0 2C
B	32	F0 32	U	3C	F0 3C
C	21	F0 21	V	2A	F0 2A
D	23	F0 23	W	1D	F0 1D
E	24	F0 24	X	22	F0 22
F	2B	F0 2B	Y	35	F0 35
G	34	F0 34	Z	1A	F0 1A
Enter	5A	F0 5A			

설계 과정

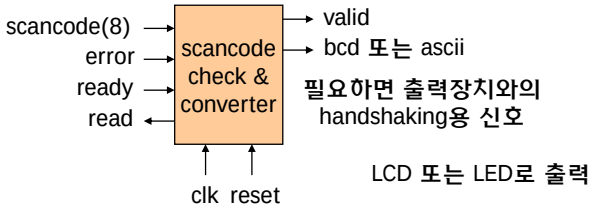
- PS/2 read module
  - 8-bit scancode를 읽어서 shift register에 저장



- 상태도

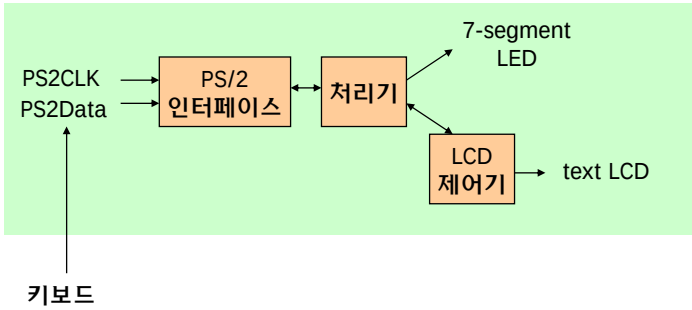


- scancode converter
  - ready가 1이 되면 scancode 저장
  - make code가 입력될 때 출력가능한 문자이면 해당되는 ascii(또는 bcd)코드를 출력
  - make code가 입력된 후에 같은 make code 또는 break code가 입력되지 않으면 error임
  - break code가 입력된 후에는 다른 make code 입력 가능



상태도: idle → make → F0 → break → idle

PS/2 시험 환경의 예



PS/2 read 모듈 설계

```
module ps2_recv(clk, reset, ps2_clk, ps2_data, read, done, error, data);
    input clk, reset;
    input ps2_clk, ps2_data;
    input read;
    output done;
    output reg error;
    output [7:0] data;
```


- ps2 clock과 data의 동기화
  - 이 신호들은 line을 통해서 전송되므로 잡음이 많아서 그대로 사용하기 곤란함
  - 시스템 clk과 동기화한 후에 사용함

```
// synchronize ps2_clk and ps2_data to host clock
always @(posedge clk) begin
    ps2c <= ps2_clk;
    ps2d <= ps2_data;
end
```

- ps2 clock의 falling edge detection
  - 동기화된 ps2 clock 신호(ps2c)도 짧은 시간동안 0,1이 반복하여 변하는 bouncing이 발생할 수 있다. → ps2c의 negative edge를 사용하는 것은 바람직하지 않다.
  - ps2c가 몇 클럭동안 계속하여 1(또는 0)일 때에 ps2 clock이 1(또는 0)인 것으로 판단함 → shift register에 연속적인 ps2c 값을 저장한 후에 이 값들이 모두 1 또는 모두 0인지 검사


```
always @(posedge clk or posedge reset) begin
    if (reset)
        ps2c_reg <= 8'b11111111;
    else  // shift reg for filtered clock
        ps2c_reg <= {ps2c, ps2c_reg[7:1]};
end

// filtered clock
always @(posedge clk or posedge reset)
    if (reset) begin
        clk_filter_prev <= 1'b1;
        clk_filter <= 1'b1;
    end
    else begin
        clk_filter_prev <= clk_filter;
        if(ps2c_reg == 8'b00000000)
            clk_filter <= 1'b0;
        else if (ps2c_reg == 8'b11111111)
            clk_filter <= 1'b1;
    end

assign falling_edge = clk_filter_prev & ~clk_filter;
```

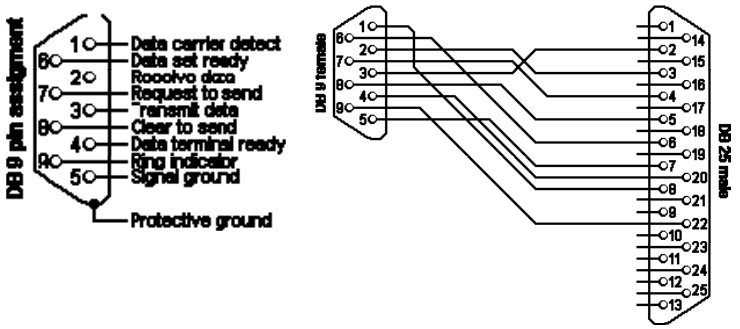


```
// state machine
always @(posedge clk or posedge reset)
    if (reset) begin
        state <= S0;
        count <= 0;
        data <= 8'b11111111;
    end
    else
        case (state)
         S0: if (falling_edge) begin    // start bit
                data <= 8'b10101010;
                parity <= 1'b0;
                if (~ps2d) state <= S1;
                else state <= S_ERR;
            end
         S1: if (falling_edge) begin    // 8-bit data
                data <= {ps2d, data[7:1]};  // shift right
                parity <= parity ^ ps2d;
                if (count < 7) count <= count + 1'b1;
                else begin count <= 0; state <= S2; end
            end
```

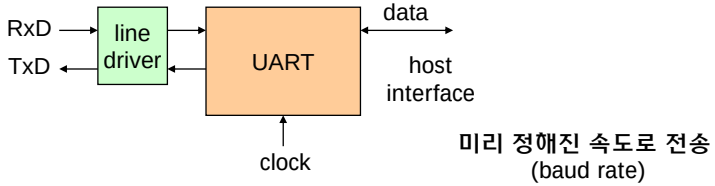
```
            S2: if (falling_edge) begin    // parity bit
                   if (parity ^ ps2d) state <= S3;
                   else state <= S_ERR;
                end
            S3: if (falling_edge) begin    // stop bit
                   if (ps2d) begin state <= S4; end
                   else state <= S_ERR;
                end
            S4: if (read) state <= S0;
            S_ERR: begin
                   error <= 1'b1;
                   if (read) state <= S0;
                end
             default: state <= S0;
           endcase

assign done = (state==S4);
```


2. UART 설계

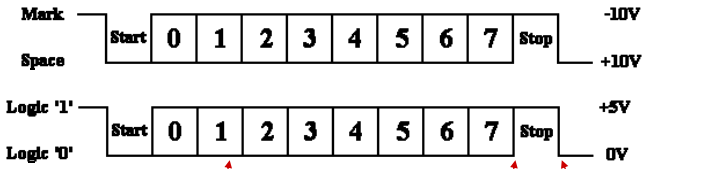
- UART (Universal Asynchronous Receiver and Transmitter)
  - RS232C 비동기 통신을 위한 인터페이스 제어기
  - 25-pin DB 또는 9-pin DB connector 사용하여 외부와 연결
  - 모뎀 제어 신호와 handshaking 신호도 포함됨



RS232C 비동기 통신



미리 정해진 속도로 전송
(baud rate)



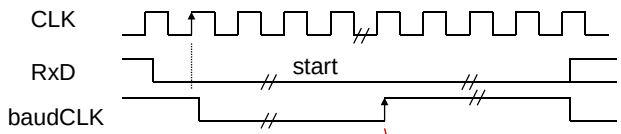
data는 8-bit 또는 7-bit 사용

parity bit가 추가될 수 있음

stop bit도 추가될 수 있음

설계과정

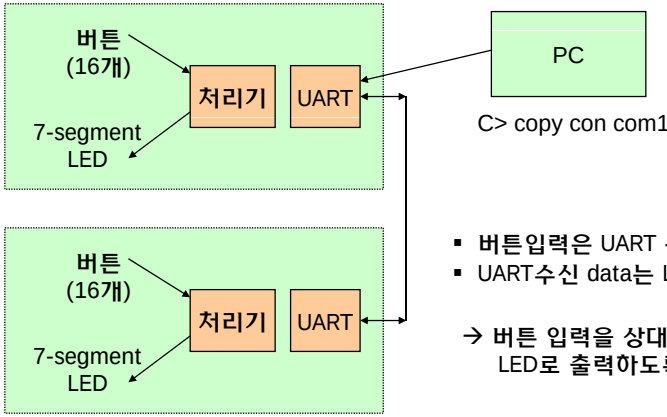
- baud rate clock 발생기
  - RxD에서 start 신호가 들어오면 baud rate 속도의 clock을 발생시켜서 수신 data의 sampling에 사용함
  - 전송할 때에는 baud rate 속도의 clock에 동기시켜서 bit 전송
  - baud rate는 레지스터에 값을 지정할 수 있도록 하여 조정 가능함



이 지점에서 sampling

- data receiver
  - baud rate clock에 동기시켜서 RxD 값을 shift register에 저장함
  - data를 모두 수신하면 status register에 표시를 하고 host에 알려주어서 수신된 data를 읽도록 함
- data transmitter
  - baud rate clock에 동기시켜서 shift register에 저장된 출력 data를 한번에 한 bit씩 shift 시키면서 TxD로 내보냄
  - data를 모두 송신하면 status register에 표시를 하고 host에 알려주어서 새로운 data을 쓸 수 있도록 함

UART 시험 환경의 예



- 버튼입력은 UART 송신
- UART수신 data는 LED 출력

→ 버튼 입력을 상대편의 LED로 출력하도록 함