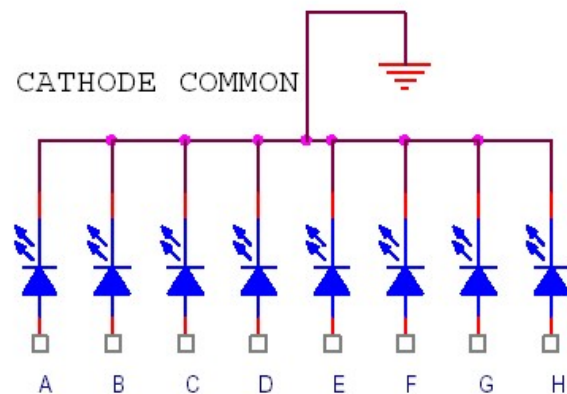
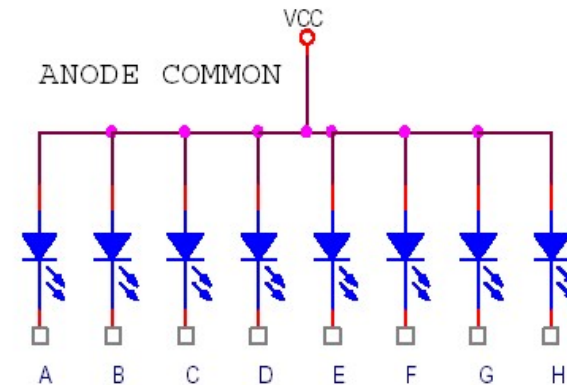
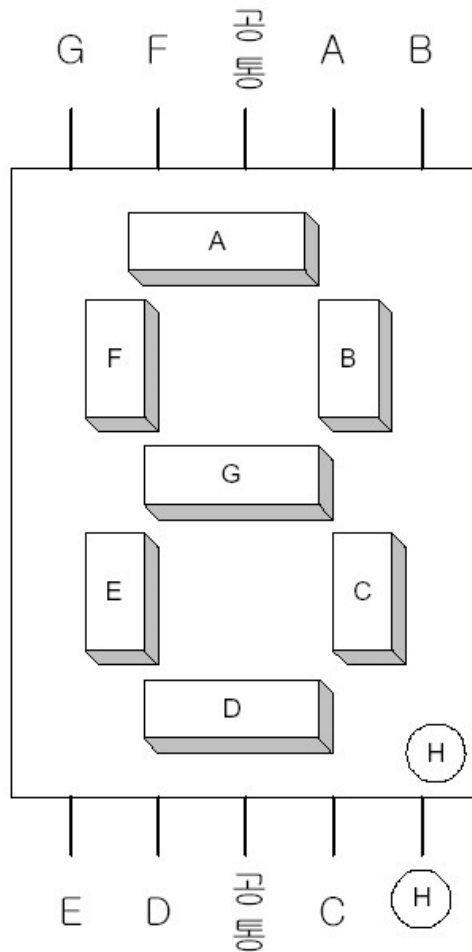
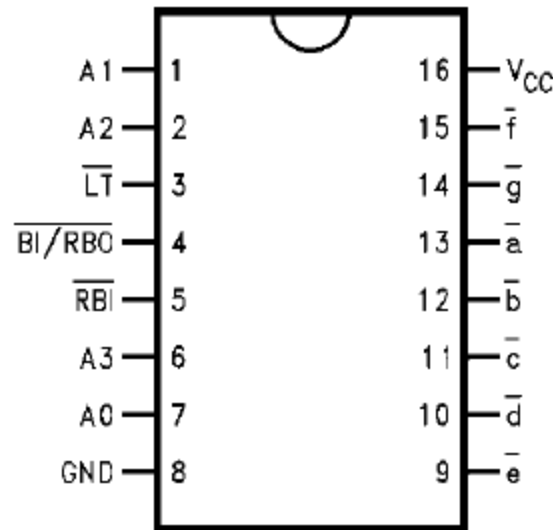


Lab3. Seven-Segment Display

Seven-Segment



74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs



Pin Descriptions

Pin Names	Description
A0–A3	BCD Inputs
$\overline{\text{RBI}}$	Ripple Blanking Input (Active LOW)
$\overline{\text{LT}}$	Lamp Test Input (Active LOW)
$\overline{\text{BI/RBO}}$	Blanking Input (Active LOW) or Ripple Blanking Output (Active LOW)
$\overline{\text{a}} - \overline{\text{g}}$	Segment Outputs (Active LOW) (Note 1)

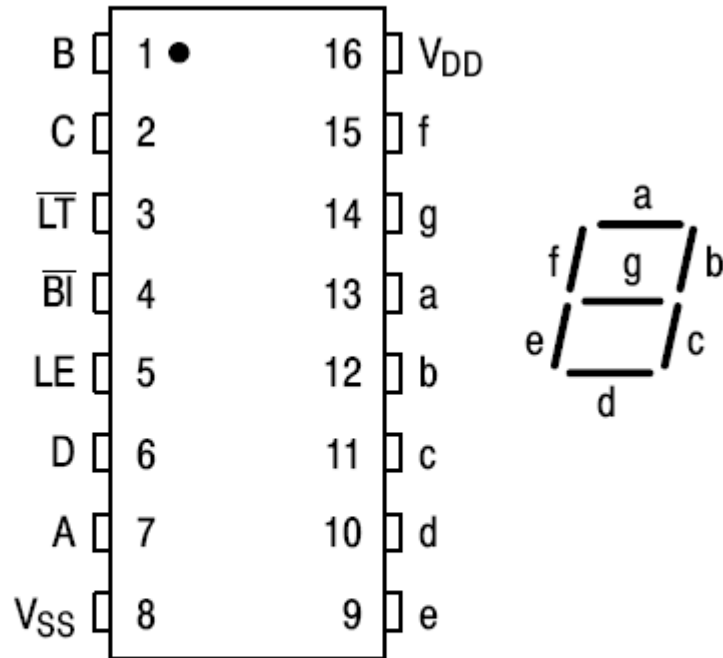
Note 1: OC—Open Collector

74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs

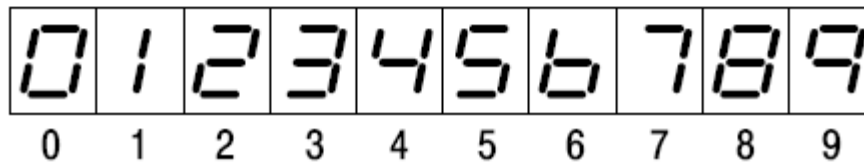
10 진수	BCD Code				Segment Output						
	A3	A2	A1	A0	/a	/b	/c	/d	/e	/f	/g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	1	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	1	1	0	0

MC14511B

PIN ASSIGNMENT



DISPLAY



MC14511B

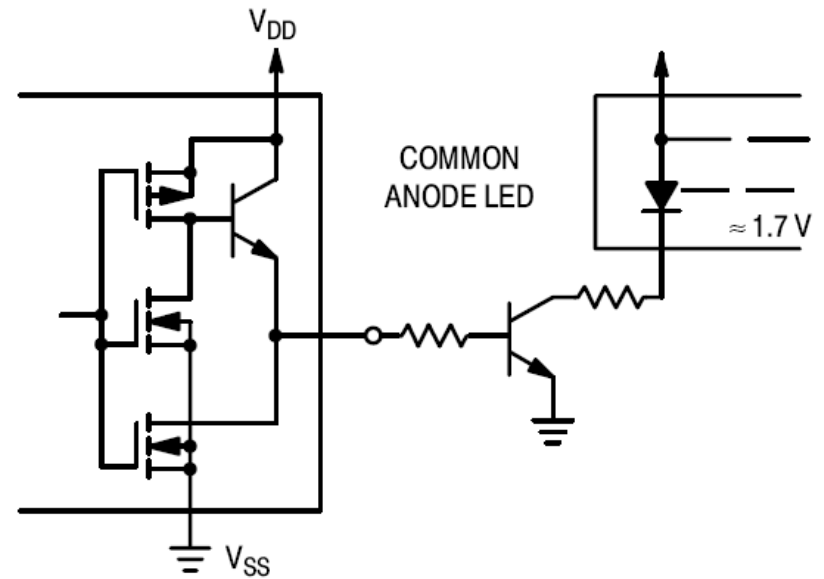
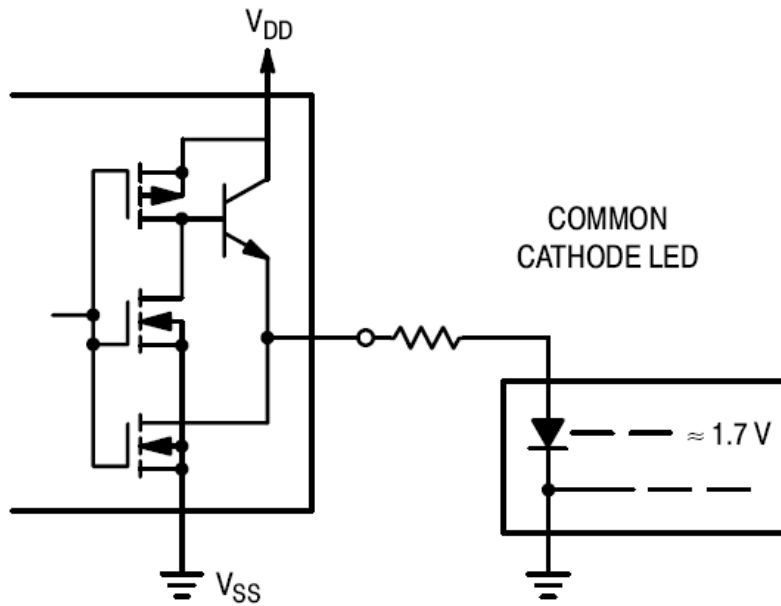
TRUTH TABLE

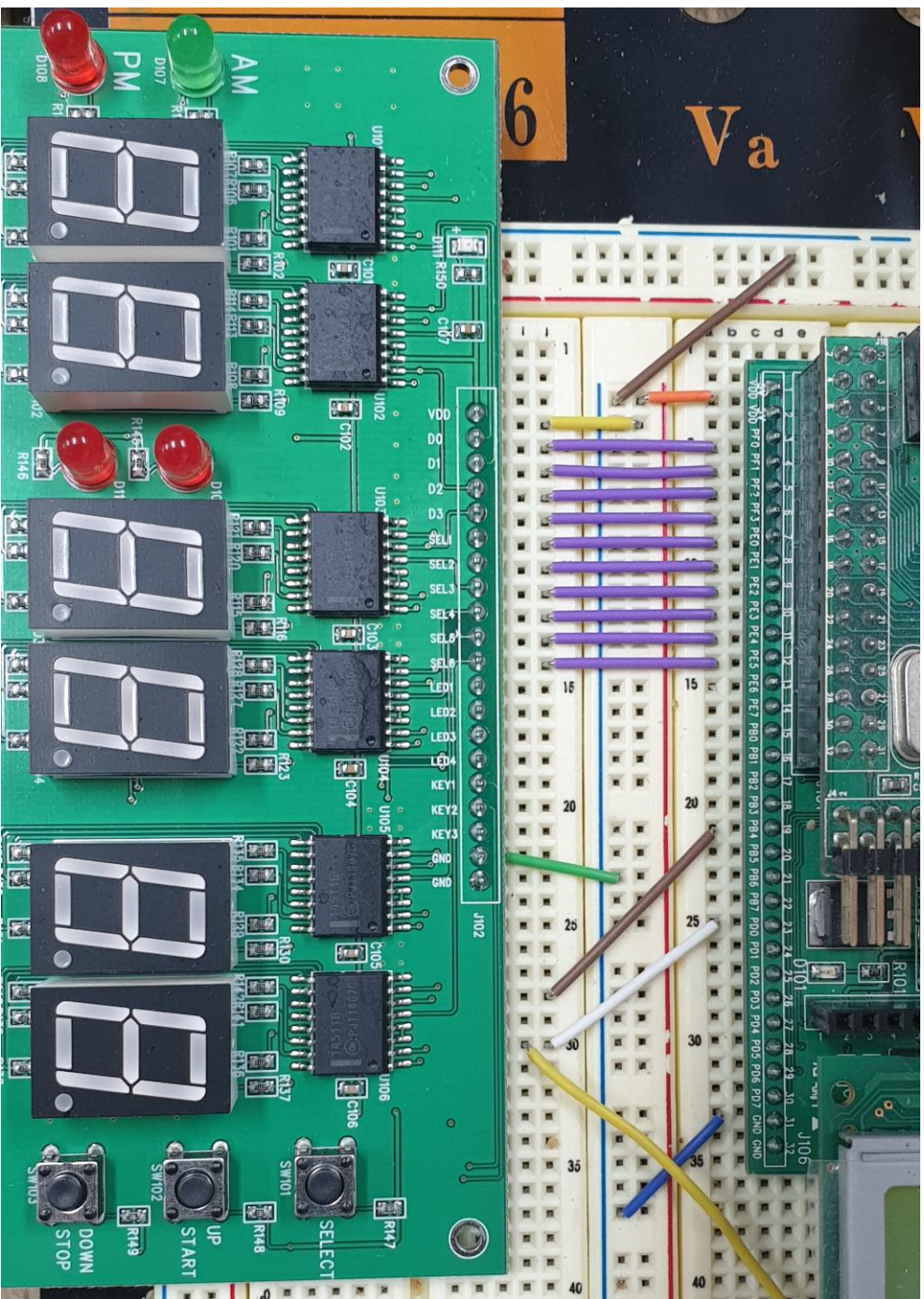
Inputs							Outputs							
LE	BI	LT	D	C	B	A	a	b	c	d	e	f	g	Display
X	X	0	X	X	X	X	1	1	1	1	1	1	1	8
X	0	1	X	X	X	X	0	0	0	0	0	0	0	Blank
0	1	1	0	0	0	0	1	1	1	1	1	1	0	0
0	1	1	0	0	0	1	0	1	1	0	0	0	0	1
0	1	1	0	0	1	0	1	1	1	1	0	0	1	2
0	1	1	0	0	1	1	1	1	1	1	0	0	1	3
0	1	1	0	1	0	0	0	1	1	0	0	1	1	4
0	1	1	0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	1	0	0	0	1	1	1	1	1	6
0	1	1	0	1	1	1	1	1	1	0	0	0	0	7
0	1	1	1	0	0	0	1	1	1	1	1	1	1	8
0	1	1	1	0	0	1	1	1	1	0	0	1	1	9
0	1	1	1	0	1	0	0	0	0	0	0	0	0	Blank
0	1	1	1	0	1	1	0	0	0	0	0	0	0	Blank
0	1	1	1	1	0	0	0	0	0	0	0	0	0	Blank
0	1	1	1	1	0	1	0	0	0	0	0	0	0	Blank
0	1	1	1	1	1	0	0	0	0	0	0	0	0	Blank
0	1	1	1	1	1	1	0	0	0	0	0	0	0	Blank
1	1	1	X	X	X	X	*							*

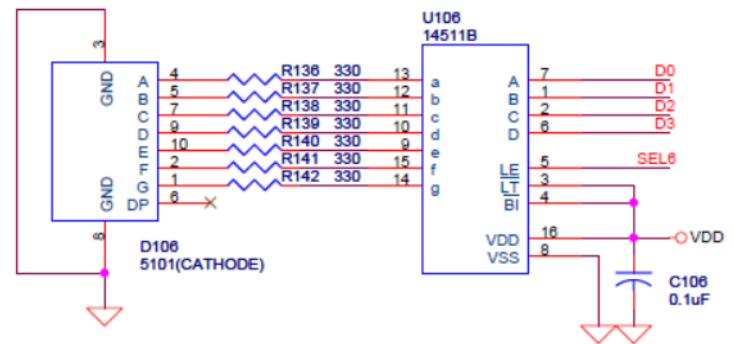
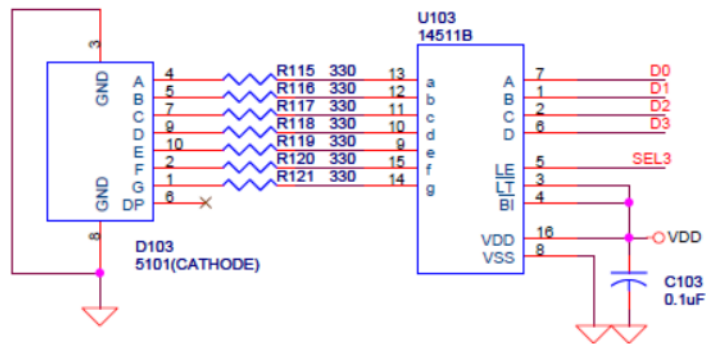
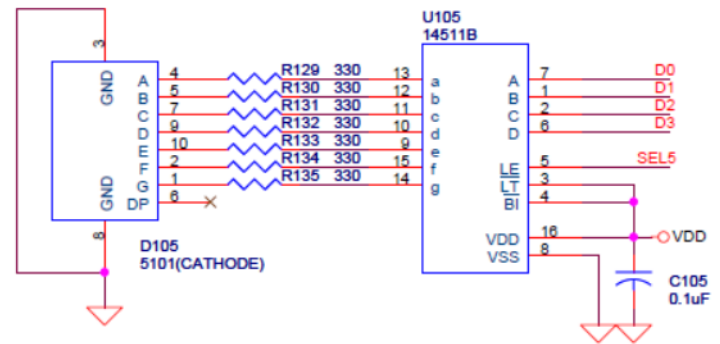
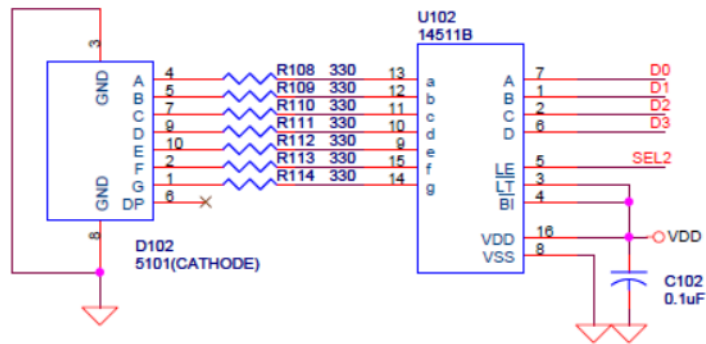
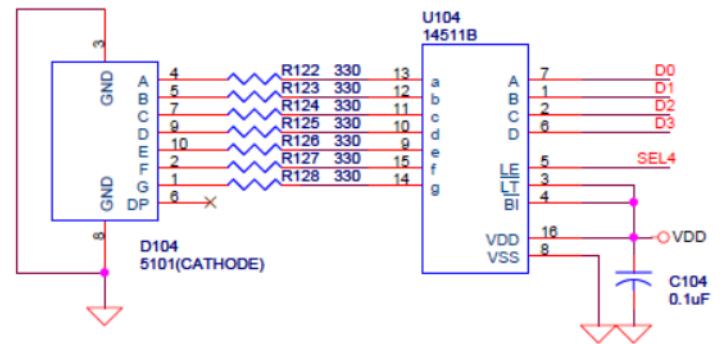
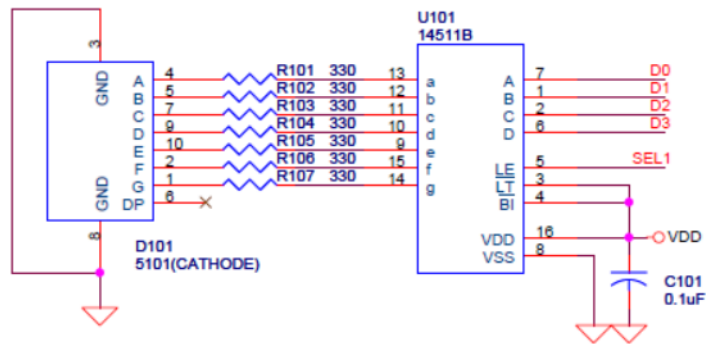
X = Don't Care

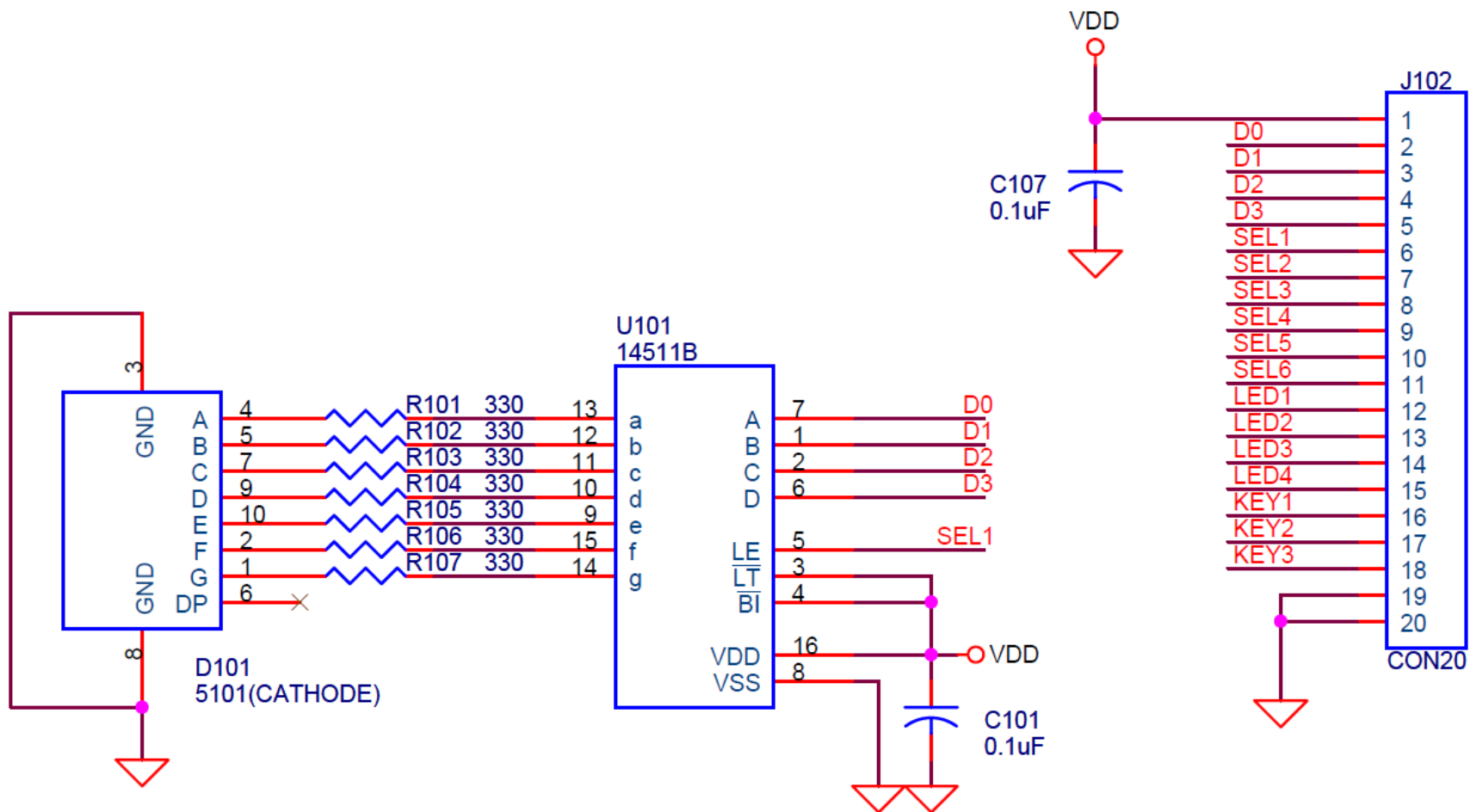
*Depends upon the BCD code previously applied when LE = 0

MC14511B

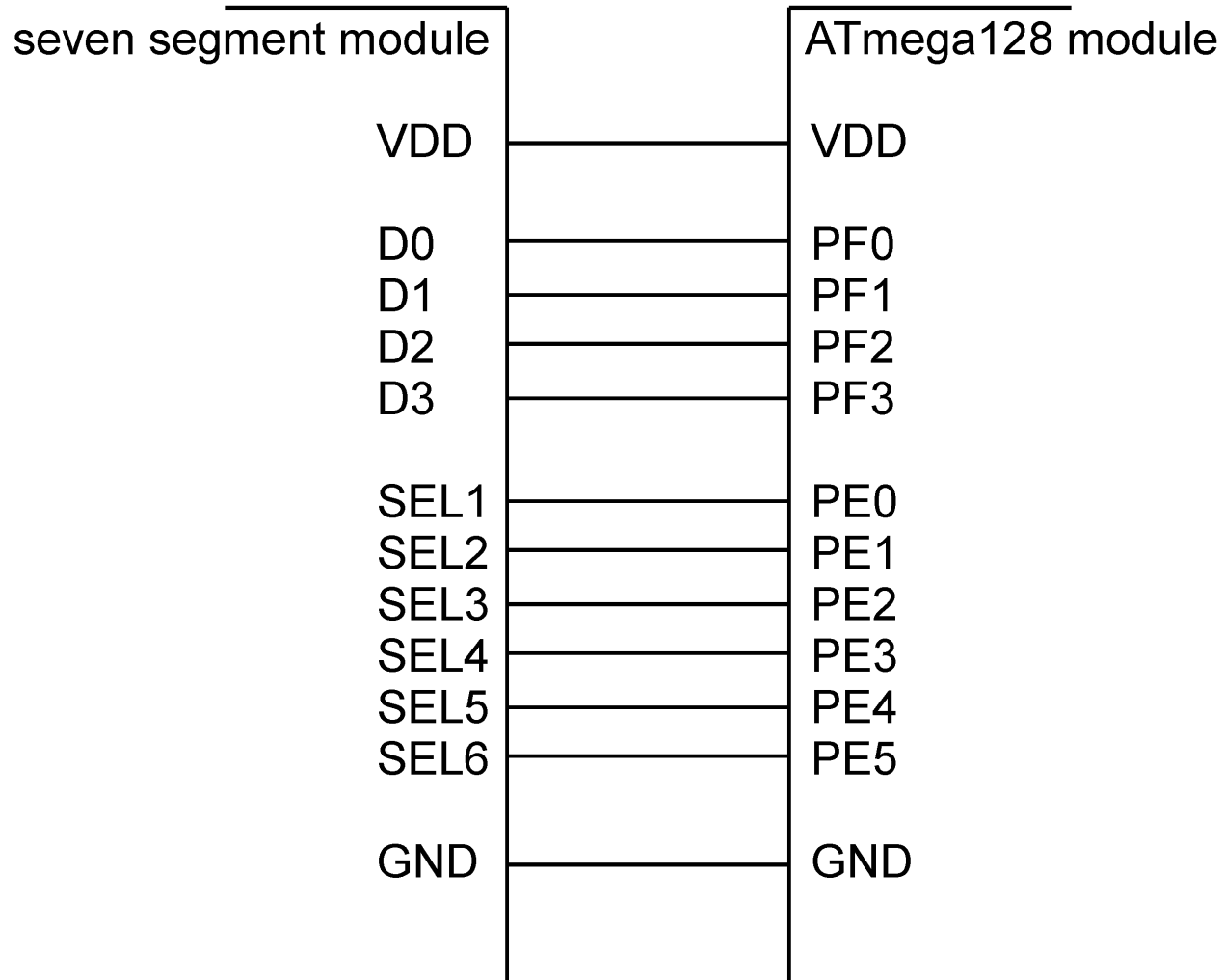








Wiring diagram



seven_segment.c(1)

```
DDRE = 0xff;
```

```
DDRF = 0xff;
```

```
while(1)
```

```
{
```

```
    PORTE = 0xfe;
```

```
    PORTF = 1;
```

```
    PORTE = 0xfd;
```

```
    PORTF = 2;
```

```
    PORTE = 0xfb;
```

```
    PORTF = 3;
```

```
    PORTE = 0xf7;
```

```
    PORTF = 4;
```

seven_segment.c(2)

```
PORTE =0xef;
```

```
PORTF = 5;
```

```
PORTE =0xdf;
```

```
PORTF = 6;
```

```
delay_ms(1000);
```

```
PORTE =0xfe;
```

```
PORTF = 7;
```

```
PORTE =0xfd;
```

```
PORTF = 8;
```

```
PORTE =0xfb;
```

```
PORTF = 9;
```

seven_segment.c(3)

```
PORTE =0xf7;
```

```
PORTF = 0;
```

```
PORTE =0xef;
```

```
PORTF = 1;
```

```
PORTE =0xdf;
```

```
PORTF = 2;
```

```
delay_ms(1000);
```

```
}
```

```
}
```

Exercise

- 먼저, 다음과 같은 함수를 완성하고 시험해 본다.
`void display_digits(unsigned int count, unsigned int number) { }`
- 위 함수는 2개의 입력 변수 `count`, `number` 를 가지며 각각은 0~99 까지의 숫자이다.
- 입력 변수 `number`의 숫자는 `seven segment`의 가장 우측의 2자리에 숫자를 나타낸다.
- 입력 변수 `count`의 숫자는 `seven segment`의 가운데 2자리에 숫자를 나타낸다.
- `Seven segment`의 가장 왼쪽 2자리는 항상 0을 나타낸다.

Exercise

- 앞의 함수를 이용하여 아래 동작을 구현한다.
- 처음 시작 시 7-segment는 000000에서 정지하고, 키 입력을 기다린다.
- Key를 누르는 순간 200msec마다 1씩 증가하는 2자리 숫자를 7-segment의 가장 우측 2자리에 나타낸다.
- Key를 떼는 순간에는 아무 변화 없다.
- Key를 다시 누르는 순간 7-segment의 가장 우측 2자리 숫자가 증가하는 것을 멈춘다.
- Key를 다시 누르는 순간 7-segment의 가장 우측 2자리 숫자가 증가하는 것을 다시 시작한다.
- 이와 같은 동작이 반복된다.

Exercise

- 증가하는 숫자 값이 99에 도달하면 다시 0으로 돌아간다.
- **Seven segment** 의 가운데 2자리에는 키 스위치를 누른 횟수를 나타낸다. 즉 스위치를 한번 누를 때마다 1씩 증가한다. 이 숫자도 99에 도달하면 다시 0으로 돌아간다.