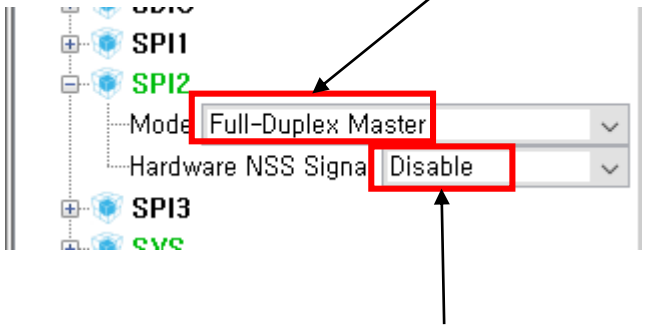
EEPROM with SPI

(stm32f407)

안상재
sangjae2015@naver.com
010-4147-2573

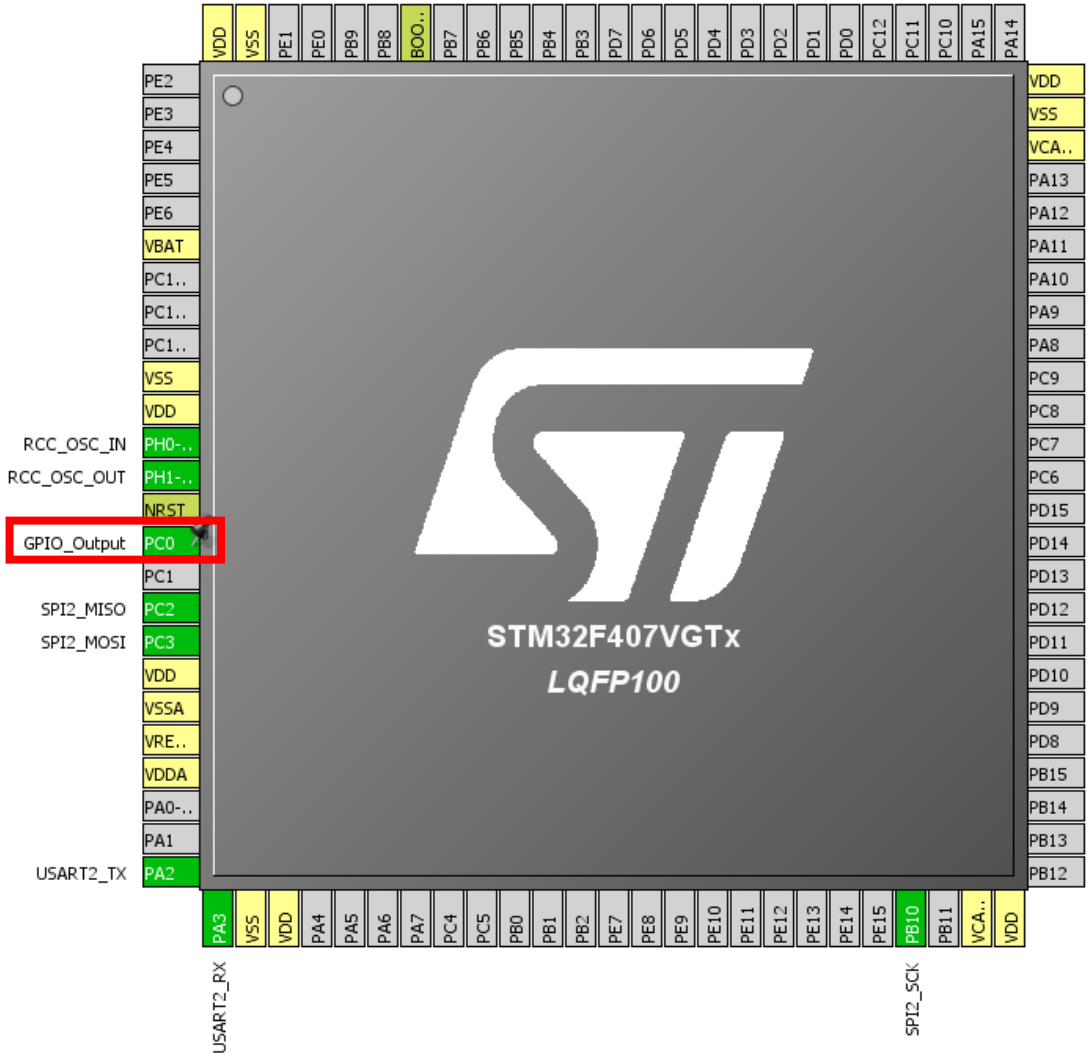
1. CubeMX 설정

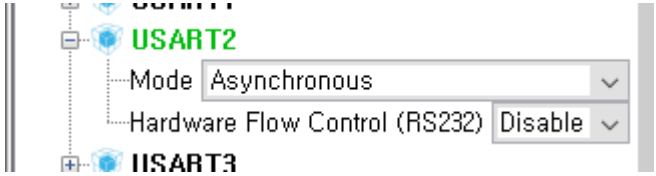
전이중 통신(송신과 수신이 동시에 일어남)



SS 신호를 소프트웨어적으로 인가하겠다는 뜻

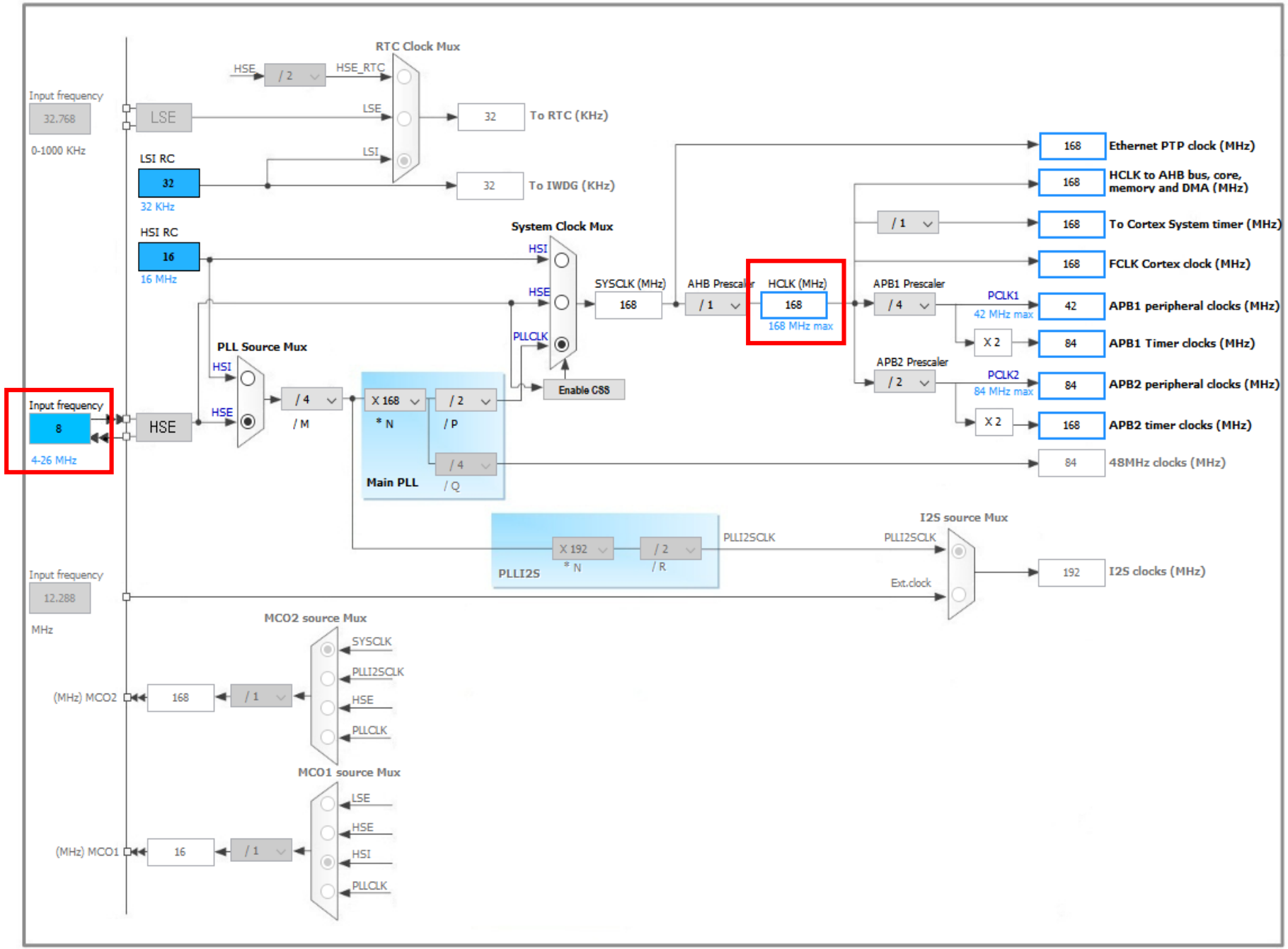
SS 신호용 GPIO핀





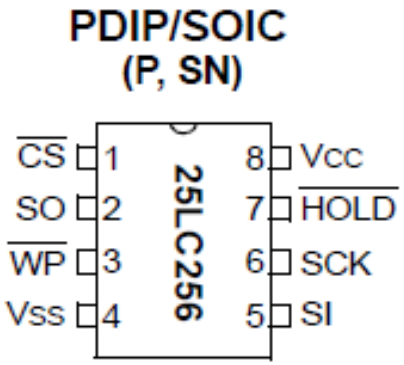
디버깅용 USART2

- Clock Configuration



2. 25LC256(EEPROM) 데이터 시트 요약

- SPI 인터페이스 제공
- 256Kbit 크기의 메모리
- 64 byte page (한번에 read/write 할 수 있는 단위)



Pin Function Table

Name	Function
$\overline{CS}$	Chip Select Input
SO	Serial Data Output
$\overline{WP}$	Write-Protect
Vss	Ground
SI	Serial Data Input
SCK	Serial Clock Input
$\overline{HOLD}$	Hold Input
Vcc	Supply Voltage

- 동작 방식

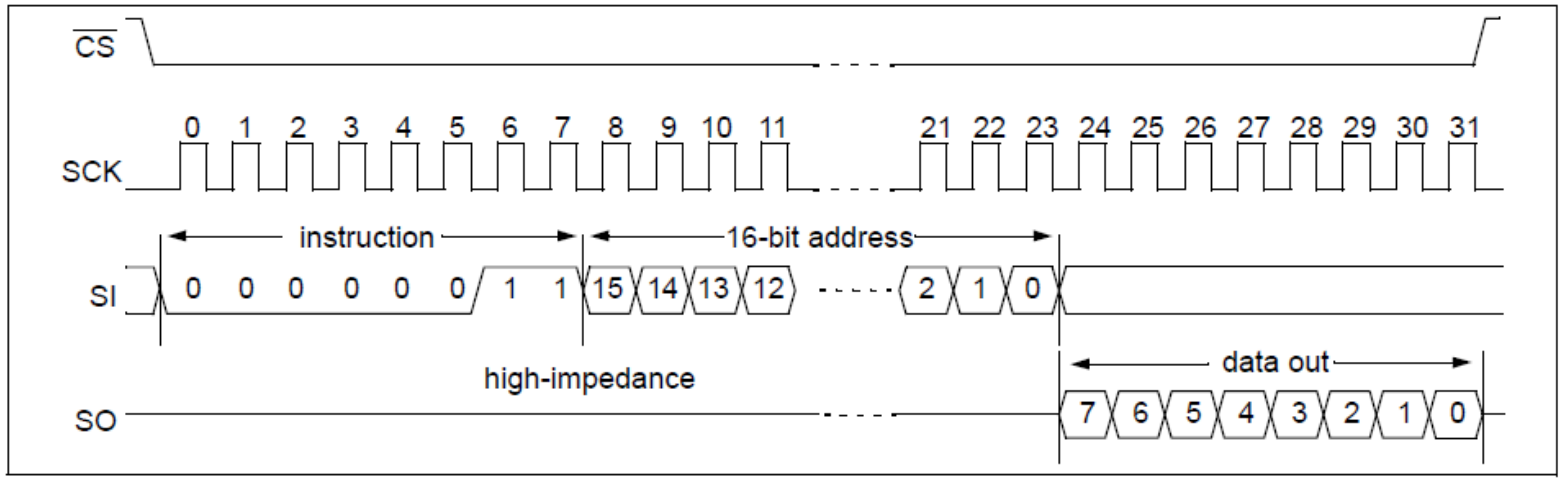
- INSTRUCTION SET

TABLE 2-1: INSTRUCTION SET

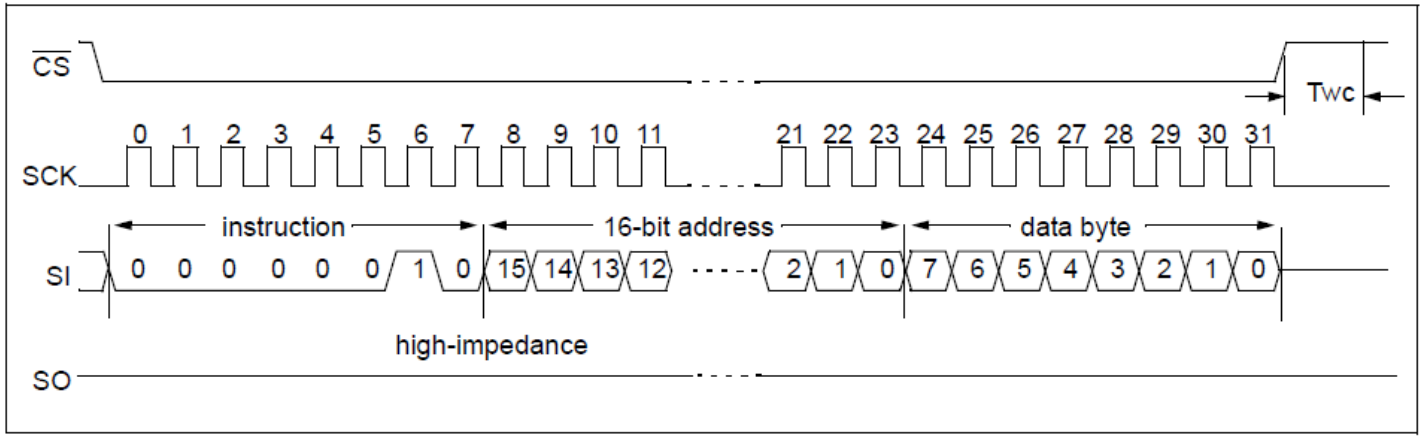
Instruction Name	Instruction Format	Description
READ	0000 0011	Read data from memory array beginning at selected address
WRITE	0000 0010	Write data to memory array beginning at selected address
WRDI	0000 0100	Reset the write enable latch (disable write operations)
WREN	0000 0110	Set the write enable latch (enable write operations)
RDSR	0000 0101	Read Status register
WRSR	0000 0001	Write Status register

- READ SEQUENCE

FIGURE 2-1: READ SEQUENCE

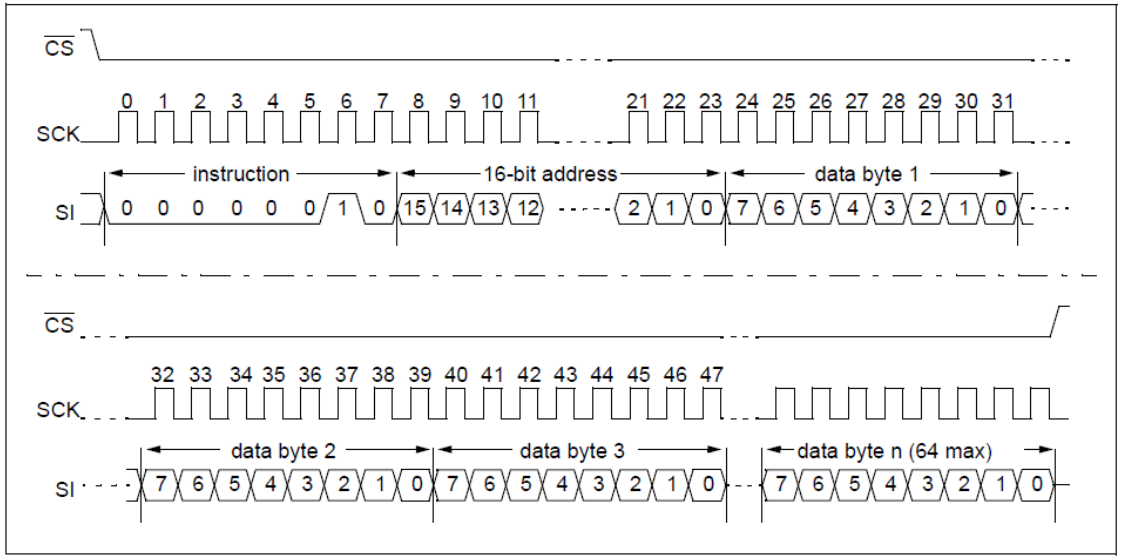


- BYTE WRITE SEQUENCE



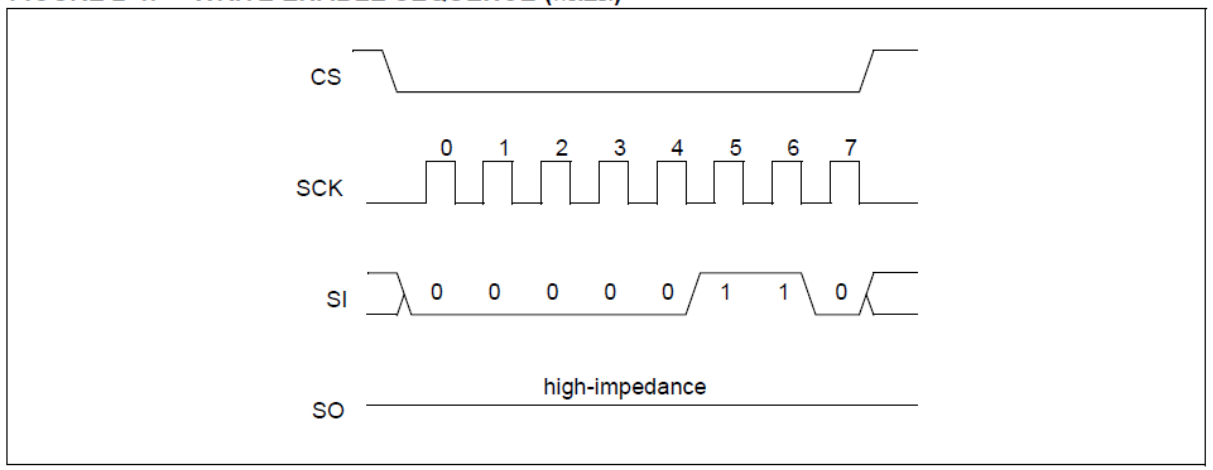
- PAGE WRITE SEQUENCE

FIGURE 2-3: PAGE WRITE SEQUENCE



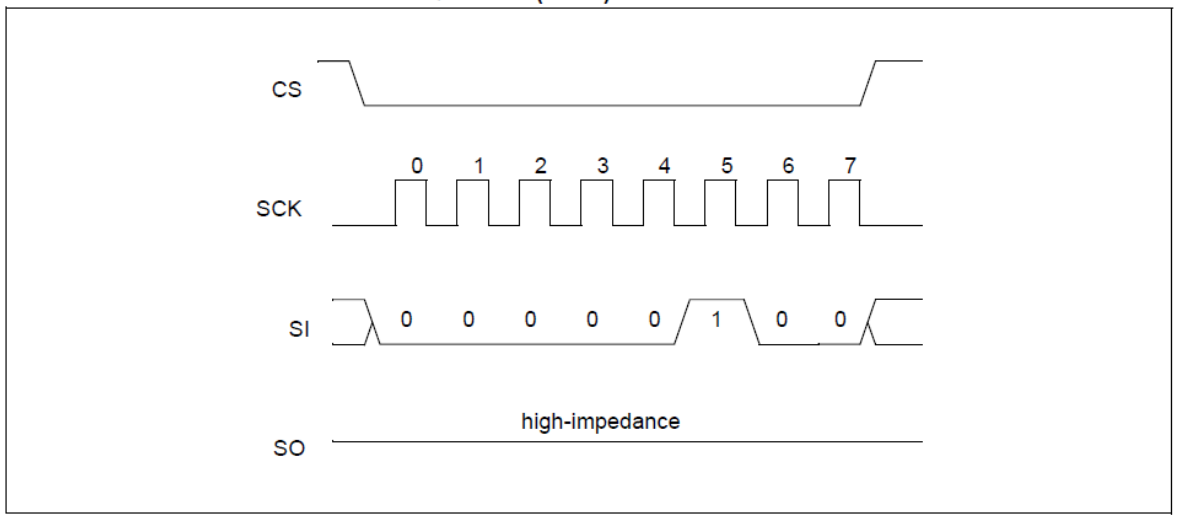
- WRITE ENABLE SEQUENCE (write 동작을 하기 전에 필요함)

FIGURE 2-4: WRITE ENABLE SEQUENCE (WREN)



- WRITE DISABLE SEQUENCE (write 동작을 금지함)

FIGURE 2-5: WRITE DISABLE SEQUENCE (WRDI)



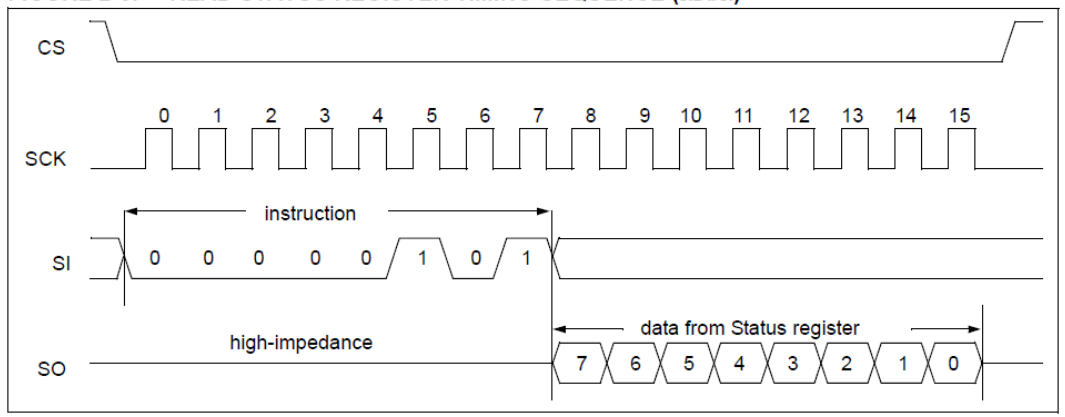
- 25LC256의 STATUS REGISTER (자세한 설명은 데이터 시트 참조)

TABLE 2-2: STATUS REGISTER

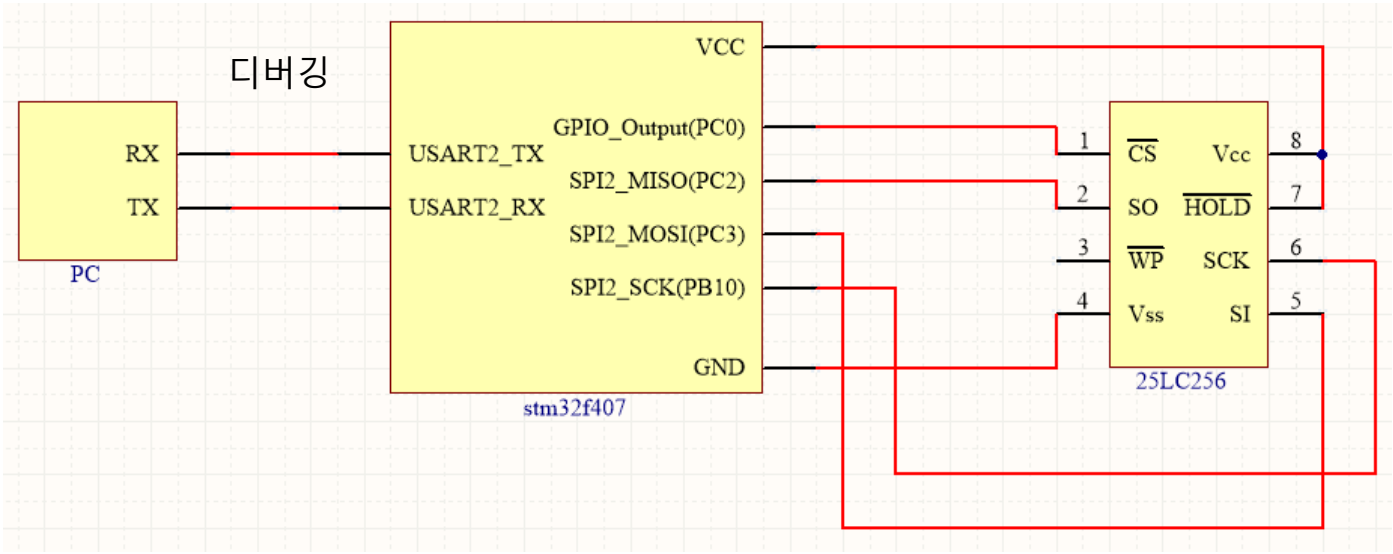
7	6	5	4	3	2	1	0
W/R	–	–	–	W/R	W/R	R	R
WPEN	X	X	X	BP1	BP0	WEL	WIP
W/R = writable/readable. R = read-only.							

- READ STATUS REGISTER

FIGURE 2-6: READ STATUS REGISTER TIMING SEQUENCE (RDSR)



3. 회로도



Chip을 항상 enable 시키기 위해 HOLD를 Vcc에 묶어놈.

4. 소스 코드

```
void SPI2_Comm(uint8_t* T_buffer, uint8_t* R_buffer, unsigned int num)
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, RESET);
    HAL_SPI_TransmitReceive(&hspi2, T_buffer, R_buffer, num, 100);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, SET);
}
```

- SPI 통신으로 num 갯수만큼의 데이터를 송수신하는 함수
- CS 핀을 LOW로 한 후에 SPI통신 송수신, 그 후에 CS 핀 HIGH

```
uint8_t T_buffer[] = {0x00, 0x00, 0x00, 0x00};
uint8_t R_buffer[] = {0x00, 0x00, 0x00, 0x00};
uint16_t EEPROM_addr = 0x5AD6;
uint8_t instruction[] = {0x03, 0x02, 0x04, 0x06, 0x05, 0x01};
```

- 송신과 수신 데이터를 저장할 변수 배열 선언
- 데이터를 write할 메모리 주소 : 0x5AD6
- instruction set를 배열에 저장함

```
uint8_t buf1[100] = {0};
uint8_t buf2[100] = {0};
uint8_t buf3[100] = {0};
```

디버깅을 하기위한 변수 선언

```
T_buffer[0] = instruction[3];   // 0x06 : set the write enable latch
SPI2_Comm(T_buffer, R_buffer, 1);
```

- 메모리에 데이터를 write를 하기위해 write enable 신호를 write함

```
T_buffer[0] = instruction[4];   // 0x05 : read status register
SPI2_Comm(T_buffer, R_buffer, 2);

HAL_Delay(100);

sprintf(buf1, "saved value1 = 0x%x\r\n", R_buffer[1]);
HAL_UART_Transmit(&huart2, buf1, 100, 10);  // status register : 0x02 => write 가능 상태
```

- 현재 메모리에 write가 가능한 상태인지 확인하기 위해 status registe를 read 함.
- 0x02가 read되면 write 가능한 상태

```
T_buffer[0] = instruction[1];   // 0x02 : write date to memory array beginning at selected address
T_buffer[1] = EEPROM_addr >> 8;
T_buffer[2] = EEPROM_addr & 0x00FF;
T_buffer[3] = 0xaa;
SPI2_Comm(T_buffer, R_buffer, 4);
HAL_Delay(100);

T_buffer[0] = instruction[4];   // 0x05 : read status register
SPI2_Comm(T_buffer, R_buffer, 2);
sprintf(buf2, "saved value2 = 0x%x\r\n", R_buffer[1]);  // status register : 0x00 => write 끝난 상태
HAL_UART_Transmit(&huart2, buf2, 100, 10);
HAL_Delay(100);
```

- 미리 지정한 주소에 0xaa 데이터를 write함.
- write 상태가 완료되었는지 확인하기 위해 status register 를 read 함.
- 0x00 이 read되면 write 상태가 완료되었음.

```
T_buffer[0] = instruction[0];   // 0x03 : read
T_buffer[1] = EEPROM_addr >> 8; // high byte
T_buffer[2] = EEPROM_addr & 0x00FF; // low byte
T_buffer[3] = 0x00;

SPI2_Comm(T_buffer, R_buffer, 4);
sprintf(buf3, "saved value3 = 0x%x\r\n", R_buffer[3]);
HAL_UART_Transmit(&huart2, buf3, 100, 10);
```

- 데이터를 write한 주소에 들어있는 값을 확인하기 위해 read함.

- 결과 화면(터미널 창)

```
saved value1 = 0x2
saved value2 = 0x0
saved value3 = 0xaa
saved value3 = 0xaa
saved value3 = 0xaa
saved value3 = 0xaa
saved value3 = 0xaa
saved value3 = 0xaa
saved value3 = 0xaa
saved value3 = 0xaa
```

- 실험 모습