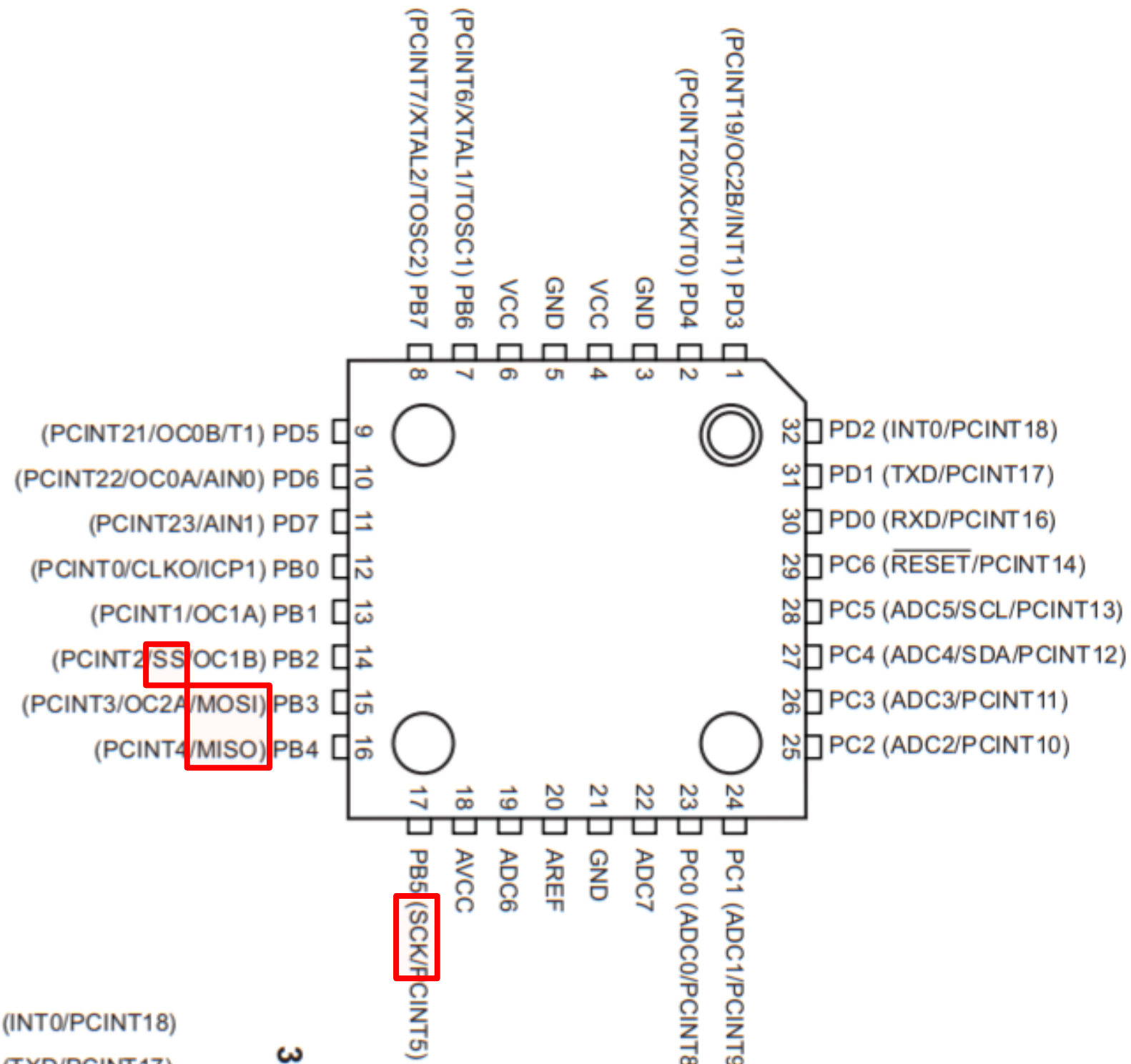
AVR HW8

임베디드스쿨1기
Lv1과정
2020. 11. 06
김인겸

2. SPI 핀



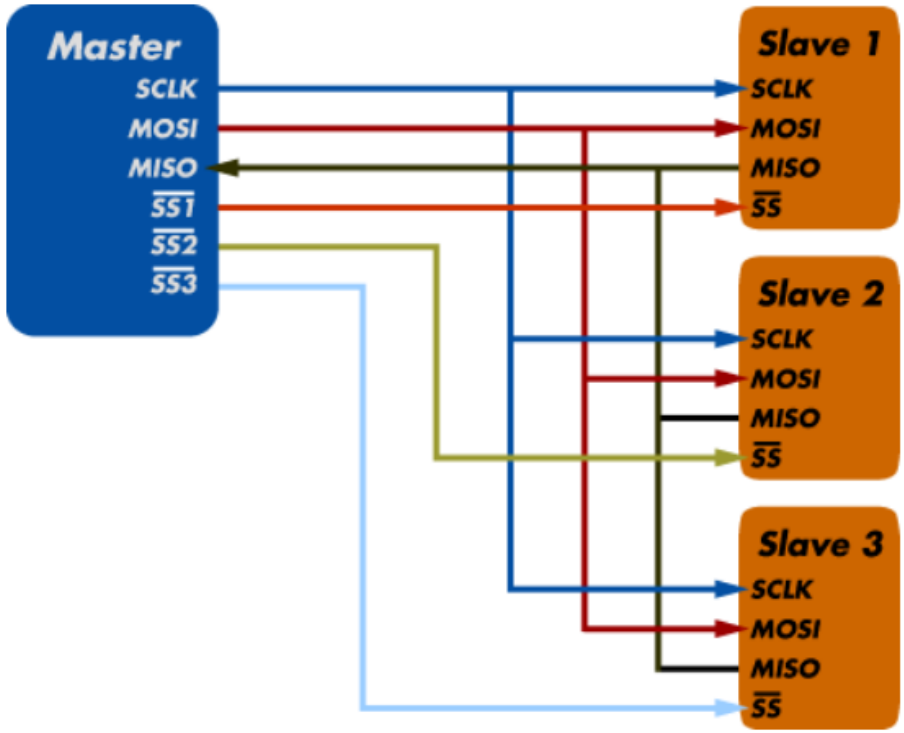
(INT0/PCINT18)

(TXD/PCINT17)

2. SPI 특징

Features

- Full-duplex, three-wire synchronous data transfer
- Master or slave operation
- LSB first or MSB first data transfer
- Seven programmable bit rates
- End of transmission interrupt flag
- Write collision flag protection
- Wake-up from idle mode
- Double speed (CK/2) master SPI mode

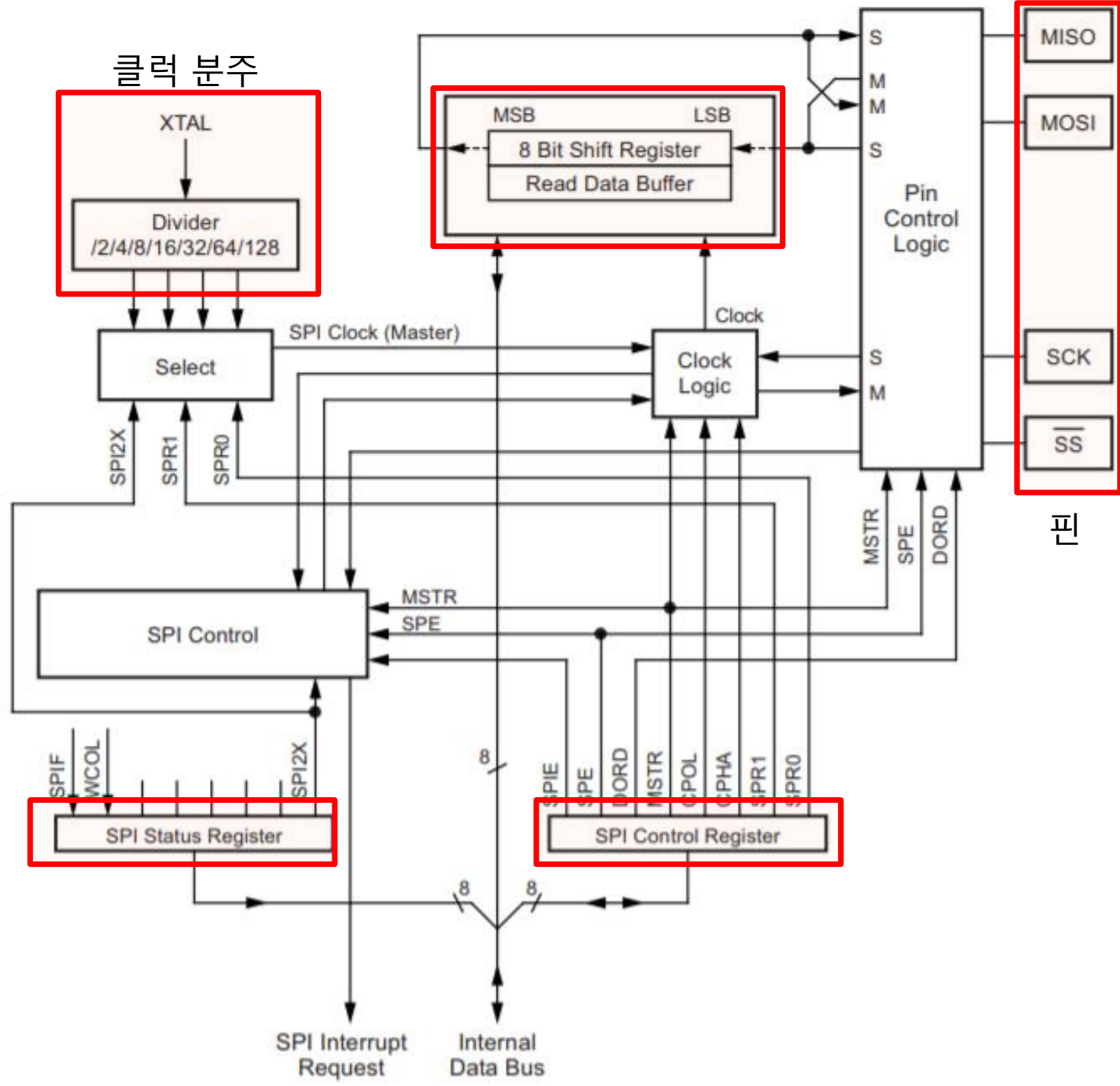


1개 Master : n개 Slave가능
SS1핀 LOW -> Slave1선택
SS2핀 LOW -> Slave2선택
SS3핀 LOW -> Slave3선택

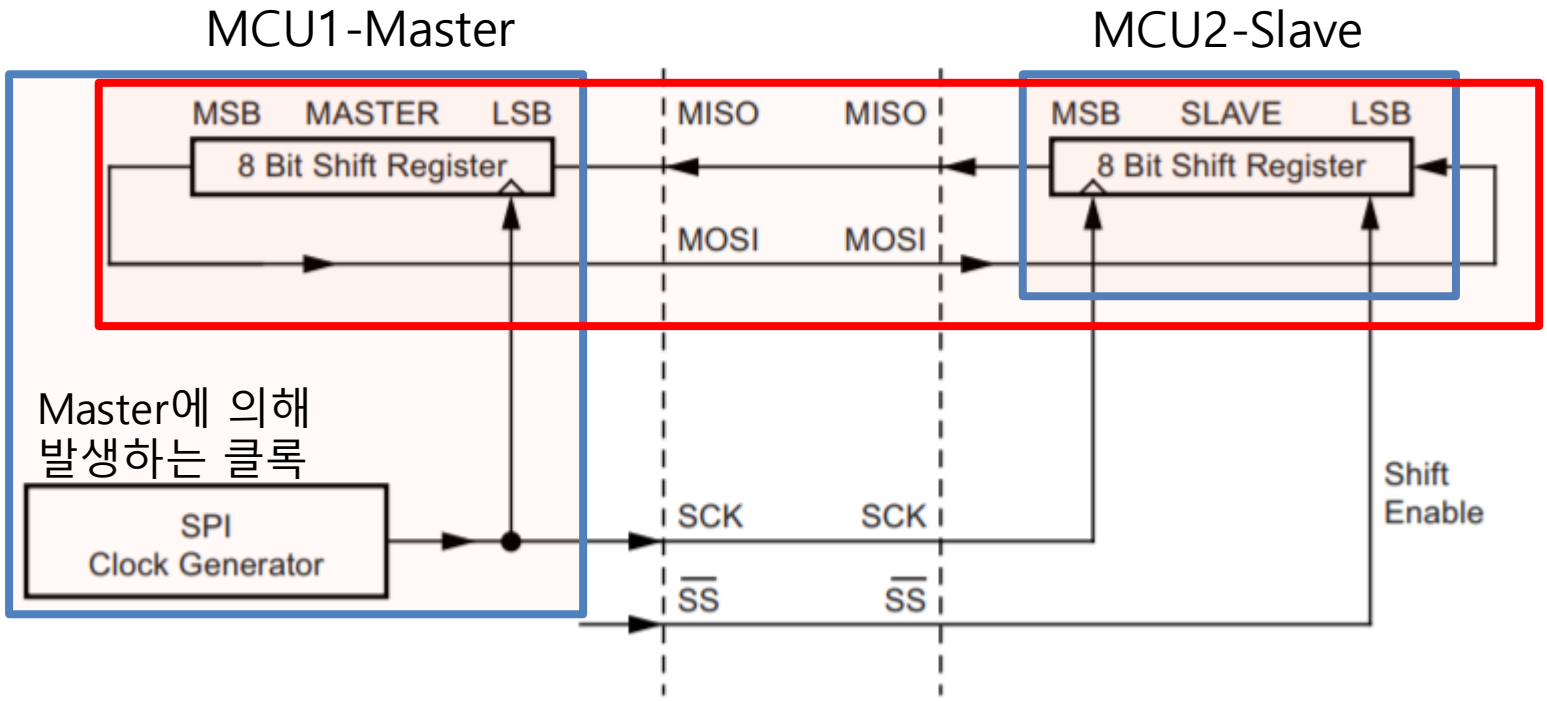
- MSB(Most Significant Bit)
: 최상위 비트

- LSB(Least Significant Bit)
: 최하위 비트

2. SPI 블록도


클럭 분주
XTAL
Divider
/2/4/8/16/32/64/128
Select
SPI Clock (Master)
MSB
LSB
8 Bit Shift Register
Read Data Buffer
Pin
Control
Logic
S
M
M
S
S
M
Clock
Clock
Logic
MISO
MOSI
SCK
SS
핀
SPI2X
SPR1
SPR0
MSTR
SPE
DORD
SPI Control
MSTR
SPE
SPIF
WCOL
SPI2X
SPI Status Register
8
SPIE
SPE
DORD
MSTR
CPOL
CPHA
SPR1
SPR0
SPI Control Register
8
8
SPI Interrupt
Request
Internal
Data Bus

2. SPI Master – Slave통신



한 바이트씩 밀어내면서
데이터를 교환

2. SPI 레지스터

SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

인터럽티 enable

SPI enable

1 : Master모드
0 : Slave모드

클럭 분주비

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Q. 000일때 16MHz/4 = 4MHz
그럼 4MHz의 클럭으로 동기통신을 하는건가요..?
보통 분주비를 몇으로 설정하나요?

2. SPI 레지스터

SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- SPI 인터럽트 플래그
전송 완료 -> 1로 세트

SPI가 Master모드로 동작할 때
이 비트를 set하면 전송속도가
2배가 됨.

Q. SPSR의 SPIF는
SPCR의 SPIE과 SREG의 I비트와는 상관없나요?

SPDR – SPI Data Register

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

2. SPI 예제 SREG의 글로벌 인터럽트 활성화하고 SPCR의 SPIE활성화 시키고 해보면 될까 싶었는데 안됩니다..

1. SS핀을 clear하는 순간 통신 시작
2. SPDR에 data를 넣는 순간 Master에서 Slave로 데이터가 송신됨
3. Master의 MOSI핀을 통해 송신된 데이터를 Slave에서 1Byte전송이 완료될 때까지 기다림
4. 전송이 완료되면 SPDR 값에 0x20을 더해 SPDR에 덮어씀(덮어쓰는 순간 데이터가 송신됨)
5. Slave의 MISO핀을 통해 송신된 데이터를 1Byte전송이 완료될 때까지 기다리고 리턴함.

Master

```
unsigned char SPI_RxTx(unsigned char data)
{
    SPDR = data;//데이터 송신
    while(!(SPSR & (1 << SPIF)));//송신 확인!
    return SPDR;
}

void SPI_master_init(void)
{
    cbi(SREG,7);

    sbi(DDRB,2);//ss핀 출력
    sbi(DDRB,3);//MOSI핀 출력
    cbi(DDRB,4);//MISO핀 입력
    sbi(DDRB,5);//SCK핀(클럭) 출력
    PORTB = 0xFF;

    sbi(SPCR,SPIE);
    sbi(SPCR,SPE);
    sbi(SPCR,MSTR);

    sbi(SREG,7);
}
int main(void)
{
    SPI_master_init();
    UART_init();
    unsigned char data;

    while(1)
    {
        cbi(PORTB,2);//SS핀-> 0 : 통신 시작
        data = SPI_RxTx('A');
        UART_transmit(data);
        _delay_ms(1000);
    }
}
```

Slave

```
void SPI_slave_init(void)
{
    cbi(SREG,7);

    cbi(DDRB,2);    //ss핀 입력
    cbi(DDRB,3);    //MOSI핀 입력
    sbi(DDRB,4);    //MISO핀 출력
    cbi(DDRB,5);    //SCK핀(클럭) 입력
    PORTB = 0xFF;

    sbi(SPCR,SPIE);
    sbi(SPCR,SPE);
    cbi(SPCR,MSTR);

    sbi(SREG,7);
}

int main(void)
{
    SPI_slave_init();

    while(1)
    {
        while(!(SPSR & (1 << SPIF)))
        {
            SPDR = SPDR + 0x20;
        }
    }

    return 0;
}
```