I²C(TWI)와 SPI통신

출처: 한밭대학교 이재흥 교수님(허락 받음)

마이크로프로세서

HRI 연구실

김동한

Human-Robot Interaction Laboratory

KYUNG HEE UNIVERSITY

목차

시리얼 인터페이스

- 패러렐 vs 시리얼 인터페이스
  - 패러렐 인터페이스(Parallel Interface)
    - 데이터 및 어드레스가 병렬로 동시에 처리
    - 데이터의 처리 속도가 빠르다
    - SRAM등의 외부 메모리 및 고속의 주변 장치들을 연결하는데 주로 사용
    - 다수의 어드레스 신호와 데이터 신호를 사용 칩의 크기를 소형화가 어렵다.
    - ATMega128의 외부 메모리 인터페이스
  - 시리얼 인터페이스(Serial Interface)
    - 고속의 제어가 필요없는 장치들을 위해 소수의 신호를 사용
    - 어드레스와 데이터를 순차적으로 처리
    - 데이터의 처리속도가 비교적 느리다.
    - 필요한 핀수를 최소화하여 칩의 소형화에 유리하다.
    - 고속제어가 필요없는 소형 칩들에 주로 사용
    - TWI(Two Wire Serial Interface), SPI(Serial Peripheral Interface)

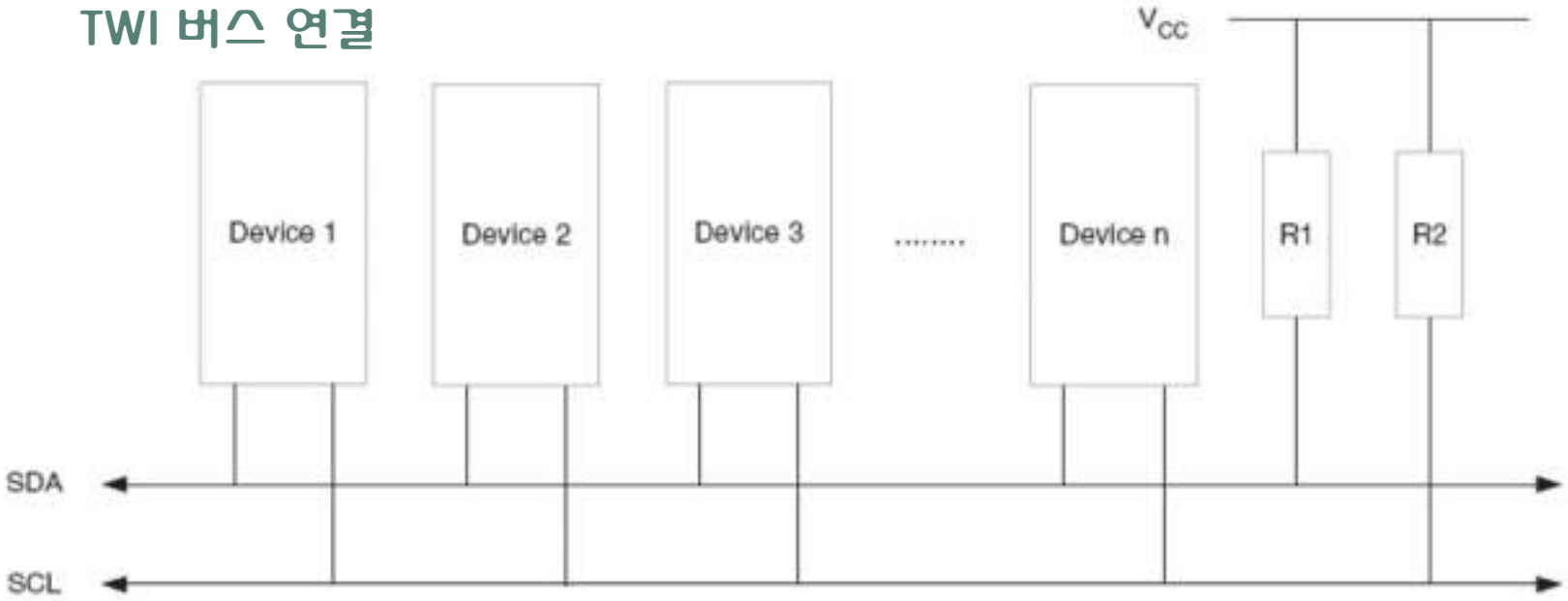
TWI(Two Wire Serial Interface)

- TWI(Two Wire Interface)
  - 단순하면서도 강력한 시리얼 통신 인터페이스
  - 필립스사에서 제창한 I^2C(Inter IC Bus)와 같은 방식
  - 2선을 이용해 시스템 내부에서 여러 장치들과 통신
    - TWI 프로토콜은 클록(SCL)과 데이터(SDA)만으로 양방향 버스라인 사용
  - 마스터와 슬레이브 동작을 지원하며, 다중 마스터도 가능하다
    - TWI의 7비트 어드레스는 128개의 다른 슬레이브 어드레스까지 허용
  - 버스에 연결된 모든 디바이스는 독립적인 주소를 가짐.
    - 디바이스 어드레스(Device Address) 혹은 디바이스 아이디(Device ID)라 하며, 칩의 구분을 위해 각 칩마다 고유의 디바이스 ID를 가짐.
  - TWI는 400kHz까지의 데이터 전송 속도를 가진다.

TWI(Two Wire Serial Interface)

□ TWI(Two Wire Interface)

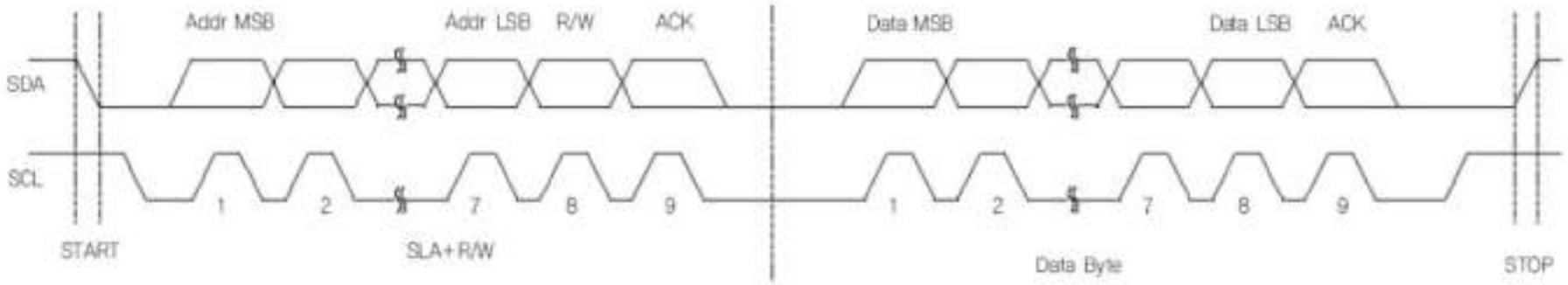
TWI 버스 연결



- 버스라인(SCL, SDA)은 풀업 저항을 통하여 +Vcc 전압으로 연결.
- 각 디바이스들은 평상시 Tri-State 상태를 유지
- 버스를 사용할 때에는 레벨 하이('1')는 Tri-State로, 레벨 Low('0')는 '0'로 출력.

TWI(Two Wire Serial Interface)

- TWI 데이터 전송형식
  - 마스터가 버스에 START 조건을 출력할 때 전송은 시작되고 STOP 조건에서 완료된다
  - START 조건, 어드레스 패킷(SLA+R/W)와 하나 또는 많은 데이터 패킷 그리고 STOP으로 구성된다.



  - ACK (인식 비트) : 현재 TWI 버스내에 마스터가 호출한 슬레이브 디바이스가 존재하고 있고, 정상적으로 어드레스를 수신했음을 알게 해줌.

TWI(Two Wire Serial Interface)

- ATMega128 TWI 레지스터
  - TWBR(TWI Bit Rate Register) :
    - TWI 비트율 레지스터
  - TWCR(TWI Control Register) :
    - TWI 제어 레지스터
  - TWSR(TWI Status Register) :
    - TWI 상태 레지스터
  - TWDR(TWI Data Register) :
    - TWI 데이터 레지스터
  - TWAR[TWI (Slave) Address Register] :
    - TWI 슬레이브 어드레스 레지스터

TWI(Two Wire Serial Interface)

- TWBR(TWI Bit Rate Register)
  - TWI 비트율 레지스터
  - 비트율 발생기에 대한 분주 요소를 선택하기 위한 레지스터

7	6	5	4	3	2	1	0
TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0

  - 비트 7~0 (TWI Bit Rate Register)
    - 비트율 발생기에 대한 분주 요소를 선택
    - 비트율 발진기는 마스터 모드에서 SCL 클록 주파수를 발생하는 주파수
  - SCL의 클럭 주파수

$$\text{SCL 주파수} = \frac{\text{CPU 클록 주파수}}{16 + 2(\text{TWBR}) \cdot 4^{\text{TWPS}}}$$

TWI(Two Wire Serial Interface)

TWCR(TWI Control Register)

TWI 제어 레지스터(TWI 동작 제어)

7	6	5	4	3	2	1	0
TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE

비트 7 : TWINT(TWI Interrupt Flag)

- TWI 인터럽트 플래그
- TWI가 현재 작업을 완료하고 응용 소프트웨어 응답을 기다릴 때, 하드웨어에 의해 세트된다.
- SREG의 I 비트와 TWCR의 TWIE 비트가 세트되면 TWI 인터럽트 벡터로 점프한다

TWI(Two Wire Serial Interface)

- TWCR(TWI Control Register)
  - 비트 6 : TWEA(TWI Enable Acknowledge Bit)
    - TWI Enable 응답 비트(ACK 펄스의 생성을 제어)
    - 이 비트에 1을 써넣은 후, 다음 조건을 만나면 ACK 펄스가 발생.
      - 디바이스에 자기 슬레이브 어드레스가 수신될 경우
      - TWAR의 TWGCE 비트가 세트인 동안 일반적인 호출이 수신될 경우
      - 데이터 바이트가 마스터 수신기 또는 슬레이브 수신기 모드로 수신될 경우
    - TWEA 비트에 0를 써넣으면 디바이스는 TWI로부터 일시적으로 끊어짐
  - 비트 5 : TWSTA(TWI START Condition Bit) :
    - TWI START 조건 비트
    - TWI 버스에서 마스터가 되고자 할때 이 비트에 1을 써넣는다.
    - START 조건이 전송되었을 때 이 비트는 소프트웨어에 의해 클리어해야 한다.

TWI(Two Wire Serial Interface)

- TWCR(TWI Control Register)
  - 비트 4 : TWSTO(TWI STOP Condition Bit) :
    - TWI STOP 조건 비트
    - 마스터 모드에서 이 비트에 1을 써넣으면 TWI 버스로 STOP 조건을 발생시킨다
    - STOP 조건이 버스에서 실행될 때 이 비트는 자동적으로 클리어된다.
  - 비트 3 : TWWC(TWI Write Collision Flag) :
    - TWI 쓰기 충돌 플래그
    - 이 플래그 비트는 TWINT가 Low일 때 TWDR(TWI Data Register)로 써넣기를 시도하면 1로 세트되고, TWINT가 High일 때 TWDR 레지스터에 써넣으면 클리어된다.

TWI(Two Wire Serial Interface)

- TWCR(TWI Control Register)
  - 비트 2 : TWEN(TWI Enable Bit)
    - TWI Enable 비트
    - 이 비트는 TWI 동작을 Enable시키고 TWI 인터페이스를 활성화한다.
    - 이 비트에 1을 써넣으면 TWI가 Enbale되어 SCL과 SDA 핀을 제어하게 되고, 0을 써넣으면 TWI는 오프되고 모든 TWI 전송은 어떤 행위에 관계없이 종료된다.
  - 비트 0 : TWIE(TWI Interrupt Enable)
    - TWI 인터럽트 Enable 비트
    - 이 비트에 1이고, SREG의 I 비트가 1로 세트되면 TWINT가 High인 동안 TWI 인터럽트가 활성화된다.

TWI(Two Wire Serial Interface)

- TWSR(TWI Status Register)
  - TWI 상태 레지스터
  - TWI의 상태와 프리스케일러 값을 보여주는 레지스터

7	6	5	4	3	2	1	0
TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0

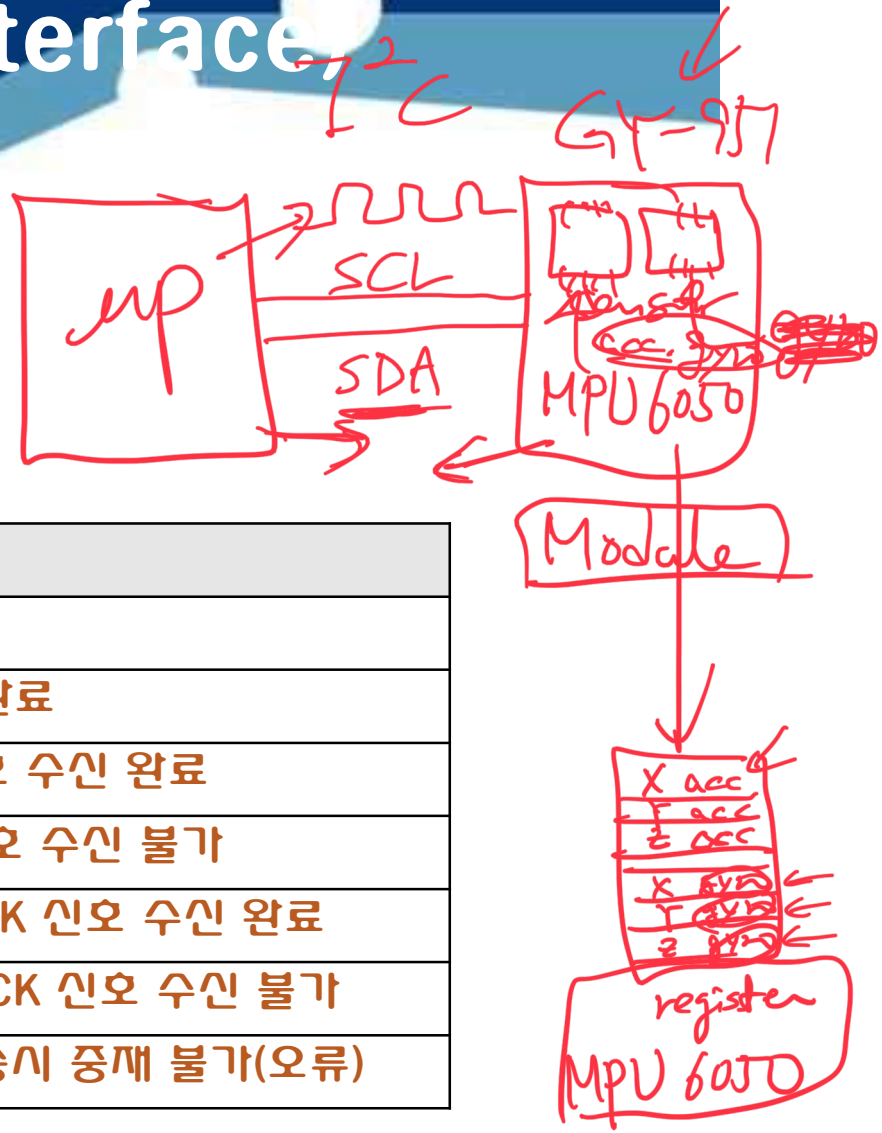
  - 비트 7~3 : TWS(TWI Status) : TWI 상태
    - TWI의 상태를 알려주는 비트

TWI(Two Wire Serial Interface)

□ TWSR(TWI Status Register)

Master Transmitter Mode에서의 Status Code

Status Code(Hex)	설명
08	Start 코드 전송 완료
10	Repeated START 코드 전송 완료
18	SLA+W 전송완료 및 ACK 신호 수신 완료
20	SLA+W 전송 완료 및 ACK 신호 수신 불가
28	데이터 바이트 전송완료 및 ACK 신호 수신 완료
30	데이터 바이트 전송 완료 및 ACK 신호 수신 불가
38	SLA+W 나 데이터 바이트 전송시 중재 불가(오류)



TWI(Two Wire Serial Interface)

□ TWSR(TWI Status Register)

Master Receiver Mode 에서의 Status Code

Status Code(Hex)	설명
08	Start 코드 전송 완료
10	Repeated START 코드 전송 완료
38	SLA+R 나 NOT ACK 비트 중재 불가(오류)
40	SLA+R 전송완료 및 ACK 신호 수신 완료
48	SLA+R 전송 완료 및 NOT ACK 신호 수신
50	데이터 바이트 수신완료 및 ACK 신호 반송 완료
58	데이터 바이트 수신완료 및 NOT ACK 신호 반송

TWI(Two Wire Serial Interface)

- TWSR(TWI Status Register)
  - 비트 1~0 : TWPS(TWI Prescaler Bit) :
    - TWI 프리스케일러 비트
    - 비트율 프리스케일러를 제어하기 위한 비트

TWI Prescaler 설정표

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

TWI(Two Wire Serial Interface)

- TWDR(TWI Data Register) :
  - TWI 데이터 레지스터
    - 전송 모드에서 TWDR은 전송될 다음 바이트를 포함
    - 수신 모드에서는 수신된 마지막 바이트를 포함

7	6	5	4	3	2	1	0
TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0

  - 비트 7~0 : TWD(TWI Data)
    - TWI 데이터
    - 전송되는 다음 데이터 바이트 또는 2줄 직렬 버스에서 수신된 마지막 데이터 바이트를 포함.

TWI(Two Wire Serial Interface)

- TWAR[TWI (Slave) Address Register]
  - TWI 슬레이브 어드레스 레지스터
  - 슬레이브 모드로 사용될 때 슬레이브 어드레스를 저장하는 레지스터

7	6	5	4	3	2	1	0
TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE

  - 비트 7~1 : TWA[TWI(Slave) Address]
    - TWI 슬레이브 어드레스
    - 이 일곱 비트는 TWI 유닛의 슬레이브 어드레스를 나타낸다.
  - 비트 0 : TWGCE(TWI General Call Recognition Enable Bit)
    - TWI 일반호출 인식 Enable비트
    - 이 비트가 세트되면 TWI 버스에 주어진 일반호출의 인식을 Enable한다.

TWI(Two Wire Serial Interface)

TWI 동작

- AVR TWI는 바이트 단위로 동작이 이루어지고, 인터럽트를 기본으로 한다.
- 인터럽트는 바이트의 수신이나 START 조건의 전송처럼 모든 버스의 이벤트 뒤에 발생된다
- SREG의 I 비트와 함께 TWCR의 TWIE(TWI Interrupt Enable) 비트가 1로 세트되면 TWINT 플래그가 '1'이 될 때 인터럽트가 발생한다
- TWIE 비트가 클리어되면, 응용 소프트웨어에서 TWI 버스에서의 동작 상태를 알기 위해 TWINT 플래그를 정기적으로 조사(poll)해야 한다.

TWI(Two Wire Serial Interface)

□ TWI를 동작시키는 방법

1.TWCR의 TWINT, TWSTA, TWEN 비트를 세팅하여 START Condition을 내보낸다.

2. TWCR의 START Condition이 정상적으로 출력되어 TWINT 플래그가 세팅되길 기다린다.

3. TWSR을 체크하여 START 상태인지 확인한다. 아니면 오류이다.

4. SLA+W 를 TWDR 레지스터에 넣는다. TWCR의 TWINT 플래그를 클리어 시킨다.

5. SLA+W가 전송되고 ACK비트가 정상적으로 도착하여 TWCR의 TWINT가 세팅되길 기다린다.

6. TWSR을 체크하여 MT_SLA_ACK 상태인지 확인한다. 아니면 오류이다.

7. Data를 TWDR 레지스터에 넣는다. TWCR의 TWINT 플래그를 클리어 시킨다.

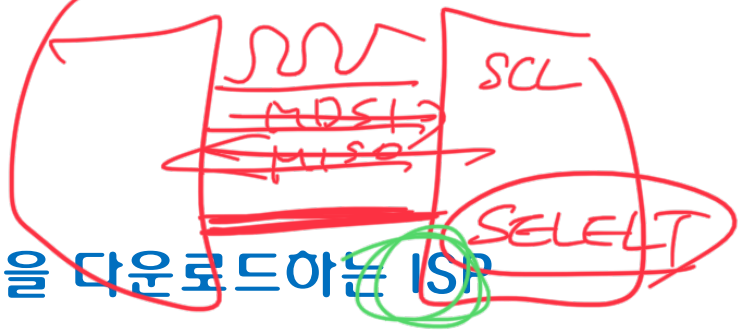
8. Data가 전송되고 ACK비트가 정상적으로 도착하여 TWCR의 TWINT가 세팅되길 기다린다.

9. TWSR을 체크하여 MT_DATA_ACK 상태인지 확인한다. 아니면 오류이다

10. TWCR TWINT, TWSTO, TWEN 비트를 세팅하여 STOP Condition을 전송한다.

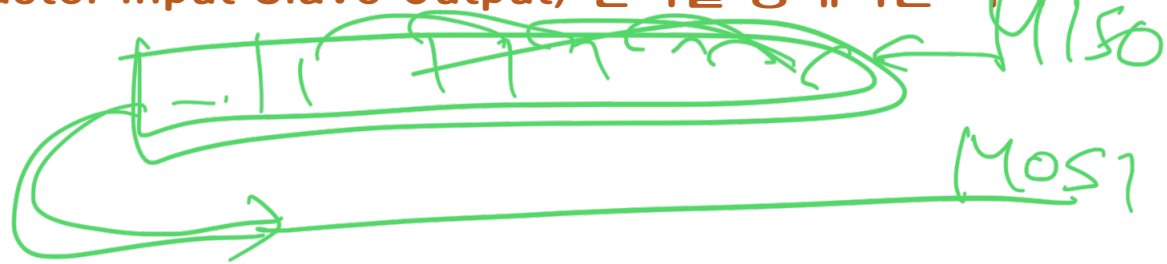
SPI(Serial Peripheral Interface)

- SPI(Serial Peripheral Interface)
  - 직렬 주변 장치 인터페이스(모토로라사에서 제창한 방식)
  - AVR과 주변 장치 디바이스간에, 또는 여러 AVR 디바이스들간에 고속 동기 데이터를 전송하기 위한 인터페이스.
  - ATmega128에서는 SS(PB0, pin 10), SCK(PB1, pin 11), MOSI(PB2, pin 12), MISO(PB3, pin 13) 4선을 이용.
  - 전이중(Full-Duplex) 통신 방식으로, 3개의 신호선을 이용하는 동기 데이터 전송방식을 따름.
  - 마스터와 슬레이브 방식으로 동작
  - 7가지의 비트율 세팅
  - SPI는 ATmega128의 플래쉬롬에 프로그램을 다운로드하는 ISP 기능 제공



SPI(Serial Peripheral Interface)

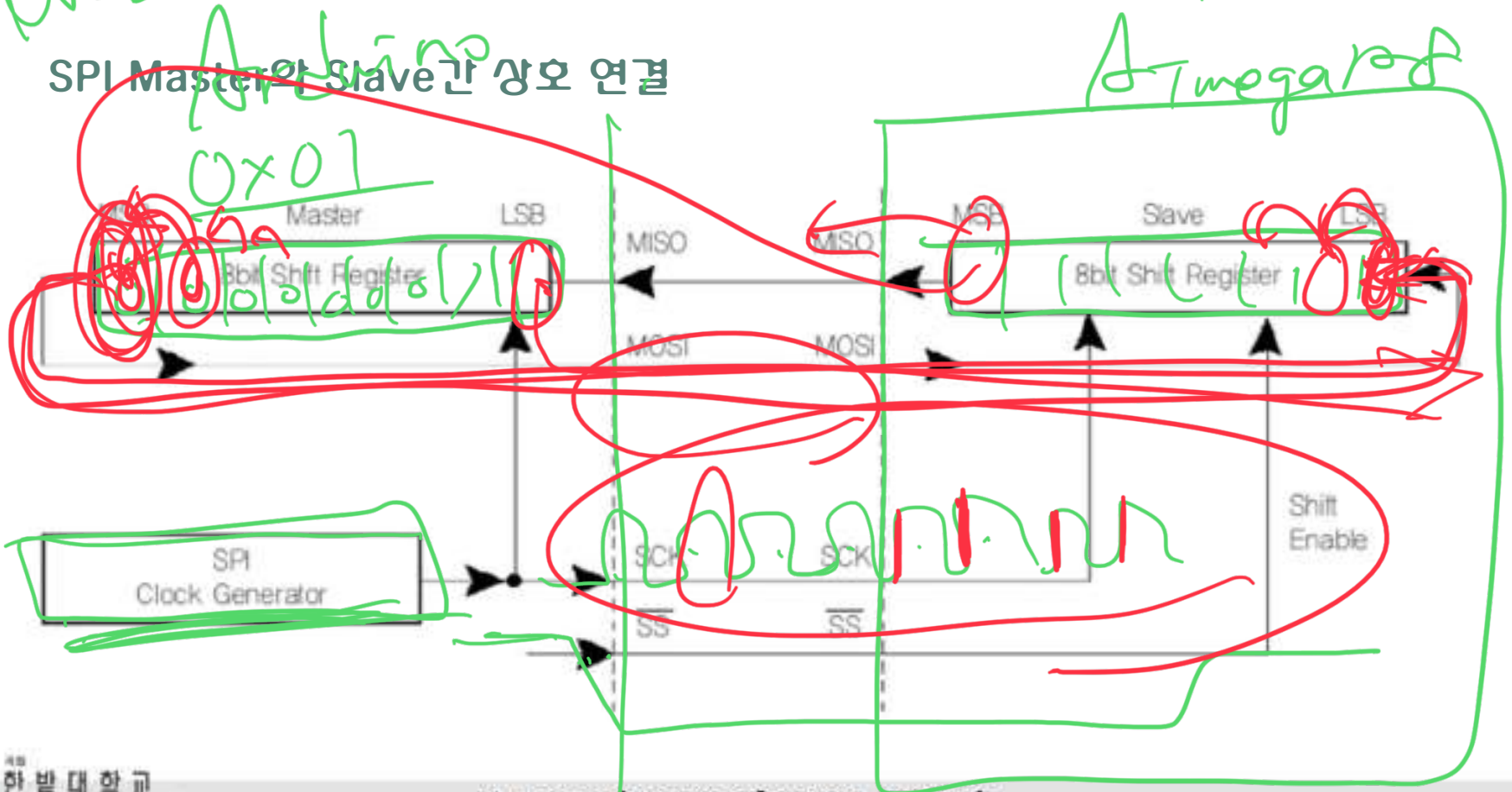
- SPI(Serial Peripheral Interface) 동작
  - SPI는 반드시 1개의 마스터와 1개의 슬레이브 사이에서만 동작
  - 마스터가 슬레이브에게 데이터를 보낼 때
    - 여러 슬레이브에서 원하는 슬레이브에게 SS (Slave Select) 신호를 0 레벨로 출력하여 선택
    - 클록 신호를 발생하여 SCK(Serial Clock)을 통해 출력
    - 송신할 데이터를 시프트 레지스터에 데이터를 준비하여 MOSI(Master Output Slave Input) 단자로 출력한다.
    - 동시에 MISO (Master Input Slave Output) 단자를 통해서는 더미 데이터가 입력



SPI(Serial Peripheral Interface)

SPI(Serial Peripheral Interface) 동작

SPI Master와 Slave간 상호 연결



SPI(Serial Peripheral Interface)

- SPI(Serial Peripheral Interface) 레지스터
  - SPCR(SPI Control Register)
    - SPI 제어 레지스터
  - SPSR(SPI Status Register)
    - SPI 상태 레지스터
  - SPDR(SPI Data Register)
    - SPI 데이터 레지스터

SPI(Serial Peripheral Interface)

- SPCR(SPI Control Register)
  - SPI 제어 레지스터 (SPI 동작을 제어하기 위한 레지스터)
  - 인터럽트 Enable, 동작 모드 설정, Clock 모드 설정들에 사용

7	6	5	4	3	2	1	0
SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

  - 비트 7 (SPIE:SPi Interrupt Enable)
    - 1로 세트하면 SPI 전송완료 인터럽트 개별 Enable
  - 비트 6 (SPE: SPi Enable)
    - 1로 세트하면 SPI 직렬통신을 허용

SPI(Serial Peripheral Interface)

- SPCR(SPI Control Register)
  - 비트 5 (DORD:Data ORDer)
    - 1로 설정하면 LSB부터 전송하고 0으로 하면 MSB부터 전송한다
  - 비트 4 (MSTR:Master/Slave Select)
    - 1로 설정하면 마스터로 동작하고 0으로 하면 슬레이브로 동작한다.
  - 비트 3 (CPOL:Clock POLarity)
    - 데이터 샘플링 동작이 수행되는 SCK 클록의 극성을 설정한다
    - 디폴트값인 0이면 Leading Edge의 경우는 상승 에지로, Trailing Edge의 경우에는 하강에지로 선정된다.
    - CPOL=1이면 Leading Edge의 경우는 하강 에지로, Trailing Edge의 경우에는 상승에지로 선정된다.

SPI(Serial Peripheral Interface)

- SPCR(SPI Control Register)
  - 비트 2 (CPHA:Clock PHAse)
    - 데이터 샘플링 동작이 수행되는 SCK 클록의 위상을 설정
    - 디폴트값인 0이면 Leading Edge의 경우는 샘플링이 되고, Trailing Edge의 경우에는 셋업된다.
    - CPHA=1이면 반대로 된다.
  - 비트 1~0 (SPR1~0:SPi clock Rate select 1~0)
    - SPSR의 비트0(SPI2X비트)과 함께 SCK 클럭 신호의 주파수 분주비를 설정
    - SPSR의 비트0이 0인 상태에서 00이면 시스템 클록의 4분주, 01이면 16분주, 10이면 64분주, 11이면 128분주로 설정된다.
    - SPSR의 비트0이 1인 상태에서는 주파수를 두배로 하여 분주비를 반감시키는데 00이면 시스템 클록의 2분주, 01이면 8분주, 10이면 32분주, 11이면 64분주로 설정된다.

SPI(Serial Peripheral Interface)

- SPSR(SPI Status Register)
  - SPI 상태 레지스터(SPI의 동작 상태를 나타내는 레지스터)
  - 인터럽트 플래그와 쓰기 충돌 플래그등이 있고, SCK 클럭 주파수를 2배로 설정하는데에도 사용

7	6	5	4	3	2	1	0
SPIF	WCOL	–	–	–	–	–	SPI2X

SPI(Serial Peripheral Interface)

- SPSR(SPI Status Register)
  - 비트 7 (SPIF:SPi Interrupt Flag)
    - 전송이 완료되면 1로 세트되면서 인터럽트가 발생된다
    - 마스터로 설정하고 SS 핀이 입력으로 설정되어 0레벨이 입력되면 SPCR의 비트4(MSTR)는 자동으로 클리어되며 슬레이브 모드로 되고 SPIF가 세트되면서 인터럽트가 발생된다.
  - 비트 6 (WCOL:Write COLision flag)
    - SPI를 통해 데이터를 전송하고 있는 동안에 SPDR레지스터를 기록하려고 하면 '1'로 세트된다.
    - SPSR을 읽고 SPDR에 접근하는 경우에 SPIF와 함께 클리어된다.
  - 비트 0(SPI2X:SPI Double speed)
    - 마스터로 동작할 때 SCK클록 신호의 주파수를 2배로 설정한다.

SPI(Serial Peripheral Interface)

- SPDR(SPI Data Register)
  - SPI 데이터 레지스터
  - SPI의 데이터 전송에 사용되는 레지스터

7	6	5	4	3	2	1	0
SPDR7	SPDR6	SPDR5	SPDR4	SPDR3	SPDR2	SPDR1	SPDR0

SPI(Serial Peripheral Interface)

- SPI 제어 프로그램
  - SPI 마스터 모드로 SPI를 초기화하고, 데이터를 전송하는 프로그램

```
void SPI_MasterInit(void)
{
    /* MOSI와 SCK 핀을 출력으로, 다른 핀들을 입력으로 선언 */
    DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
    /* SPI Enable, Matser 모드, clock rate fck/16  */
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}
void SPI_MasterTransmit(char cData)
{
    /* 전송 시작 */
    SPDR = cData;
    /* 전송완료까지 대기 */
    while(!(SPSR & (1<<SPIF))) ;
}
```

SPI(Serial Peripheral Interface)

- SPI 제어 프로그램
  - SPI 슬레이브 모드로 SPI를 초기화하고, 데이터를 수신하는 프로그램

```
void SPI_SlaveInit(void)
{
    /* MISO핀을 출력으로, 나머지는 모두 입력으로 선언 */
    DDR_SPI = (1<<DD_MISO);
    /* SPI Enable*/
    SPCR = (1<<SPE);
}
char SPI_SlaveReceive(void)
{
    /* 수신 완료시까지 대기 */
    while(!(SPSR & (1<<SPIF)));
    /* Data Register 값을 리턴 */
    return SPDR;
}
```