
성 균 관 대 학 교
SIOR
로 봇 학 회

2022년 05월 01일

EMBEDDED

3 주 차

목차

- 8장 프로그램의 기본 구조
- 9장 비트 연산자
- 10장 시리얼 통신
- ~~11장 버튼 입력~~
- ~~12장 ADC~~
- Q&A

프로그램의 기본 구조

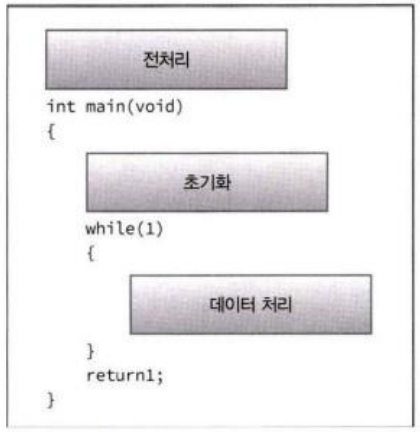


그림 8-1 C 스타일 프로그램의 구조

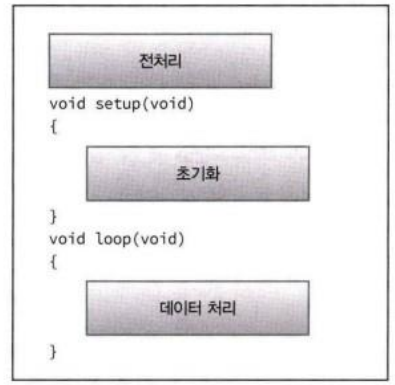


그림 8-7 아두이노 스타일 프로그램의 구조

Data Direction #Reg
Port #Reg
PIN #Reg

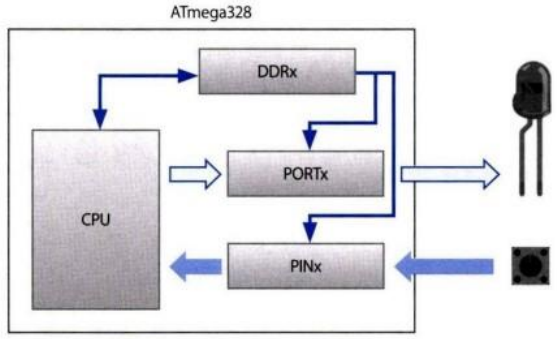


그림 8-6 데이터 입출력에 사용되는 레지스터

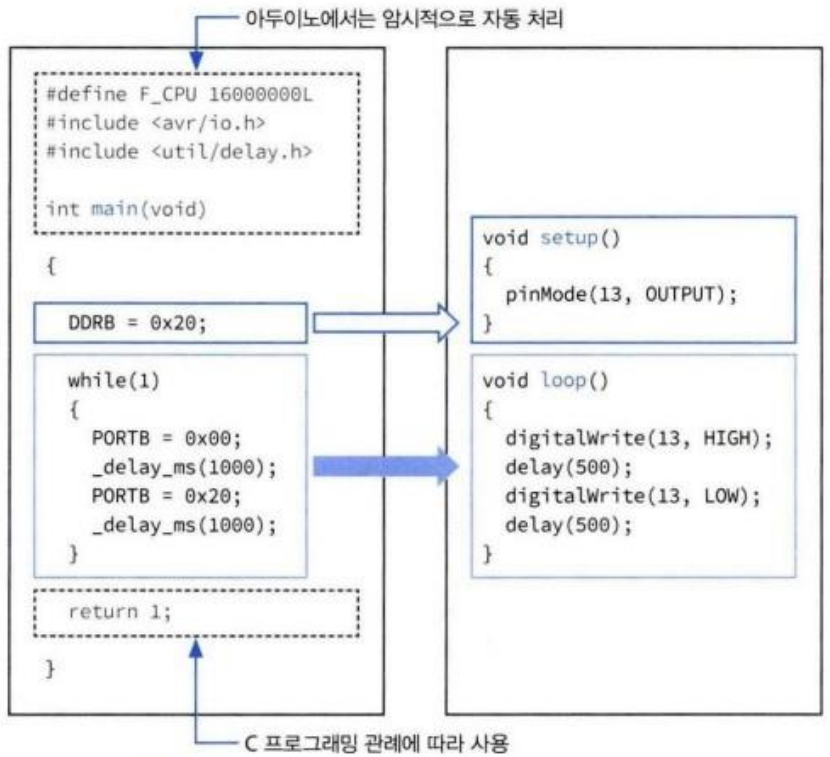


그림 8-9 C 스타일과 아두이노 스타일 프로그램의 비교

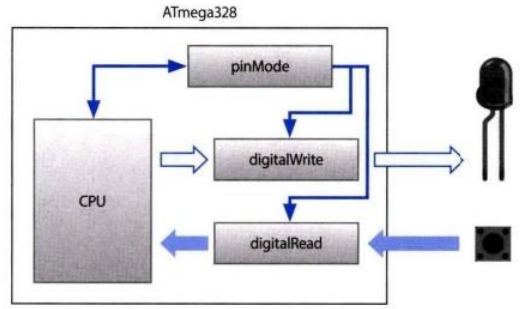


그림 8-8 데이터 입출력에 사용되는 함수

EMBEDDED

프로그램의 기본 구조

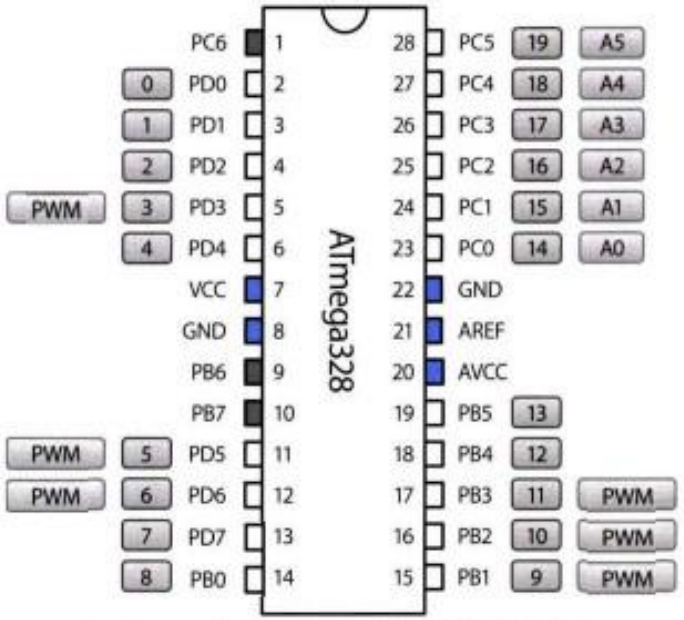


그림 4-5 아두이노 우노에서의 입출력 핀 번호

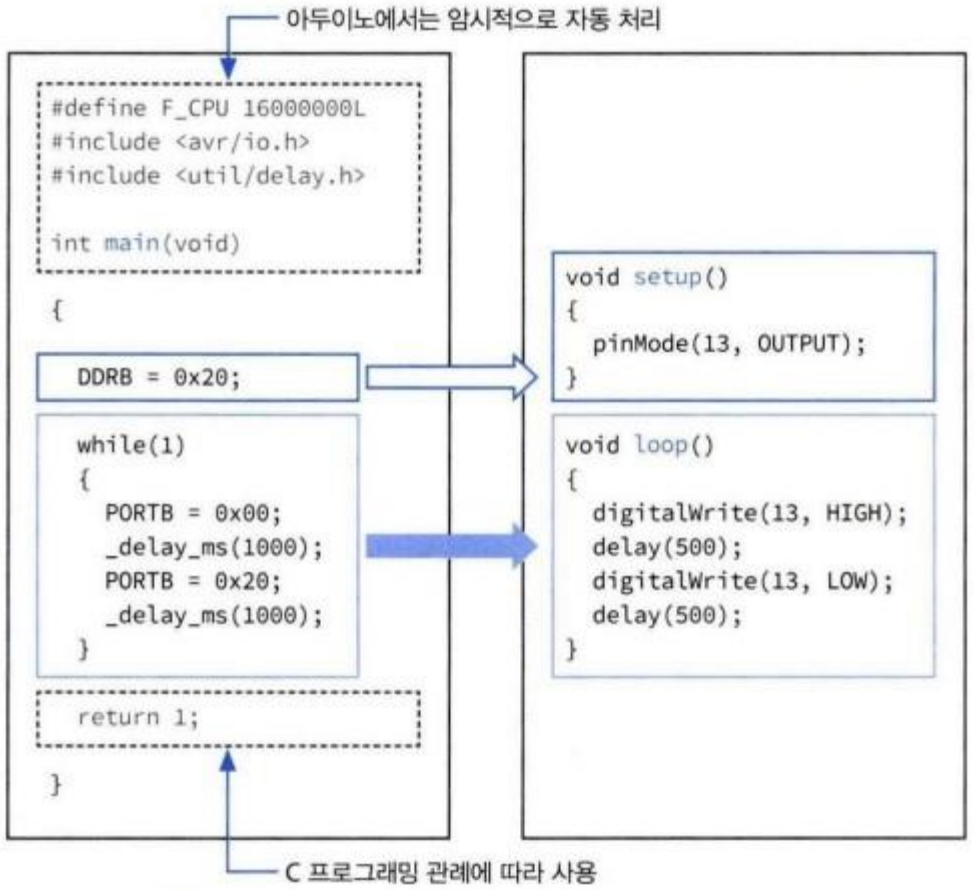


그림 8-9 C 스타일과 아두이노 스타일 프로그램의 비교

비트	7	6	5	4	3	2	1	0
비트 이름	DDx7	DDx6	DDx5	DDx4	DDx3	DDx2	DDx1	DDx0
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

(1은 출력, 0은 입력 상태)

그림 8-2 DDRx 레지스터의 구조

비트	7	6	5	4	3	2	1	0
비트 이름	PORTx7	PORTx6	PORTx5	PORTx4	PORTx3	PORTx2	PORTx1	PORTx0
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

그림 8-4 PORTx 레지스터의 구조

EMBEDDED

비트 연산자

...

0x01 << n(비트 수)

세팅: or 1 (x | 1) -> 1

클리어: and 0 (x & 0) -> 0

토글(반전): xor 1 (x ^ 1) -> ~x

읽기: and 1 (x & 1) -> x

EMBEDDED

시리얼통신

-데이터를 하나의 포트로 주고 받는 방식
-UART, SPI, I2C 등등

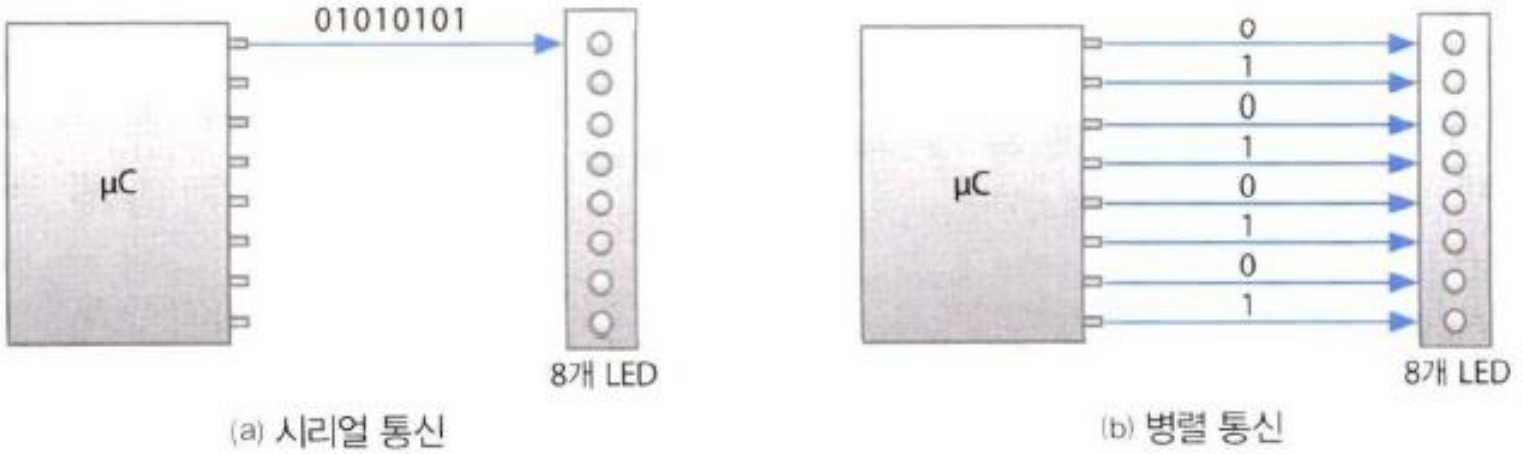


그림 10-1 시리얼 통신과 병렬 통신

UART

-Baudrate: 통신 속도가 서로 일치해야 정상적으로 데이터 송수신 가능

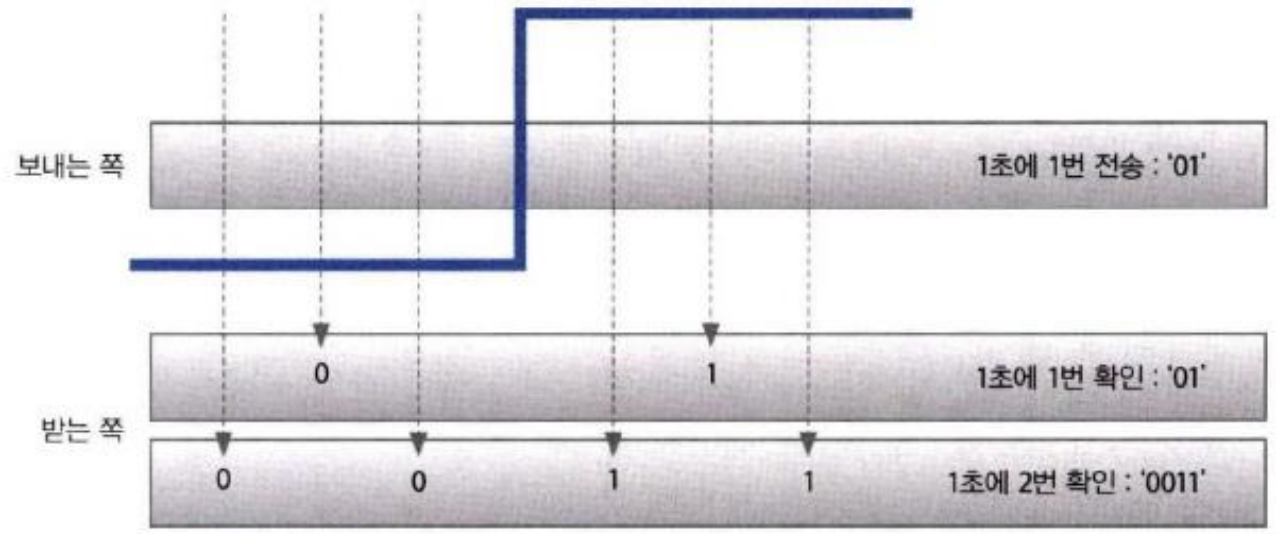


그림 10-2 송신 속도와 수신 속도 차이에 의한 전송 데이터 차이

EMBEDDED

UART

-시작비트: 평소에는 1이었다가 데이터가 전송되면 0으로 바뀐다

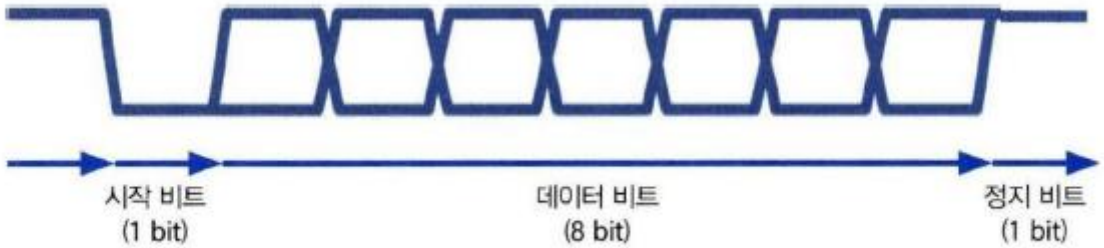


그림 10-4 UART의 데이터 전송

-연결은 TX와 RX가 서로 교차하도록

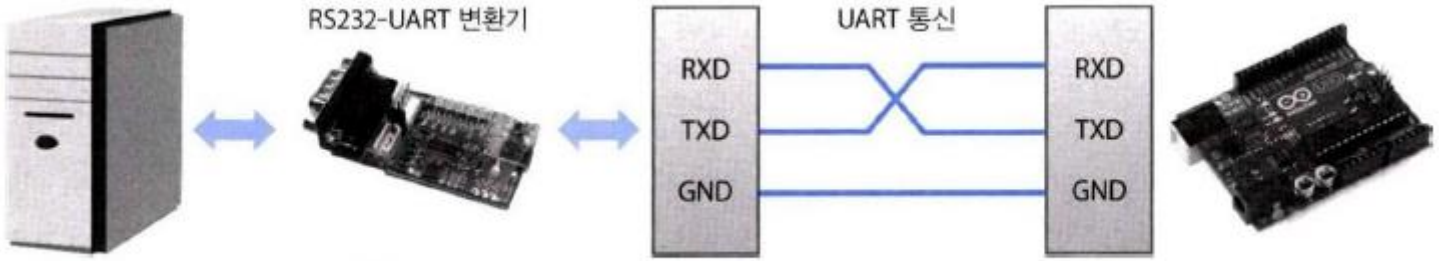
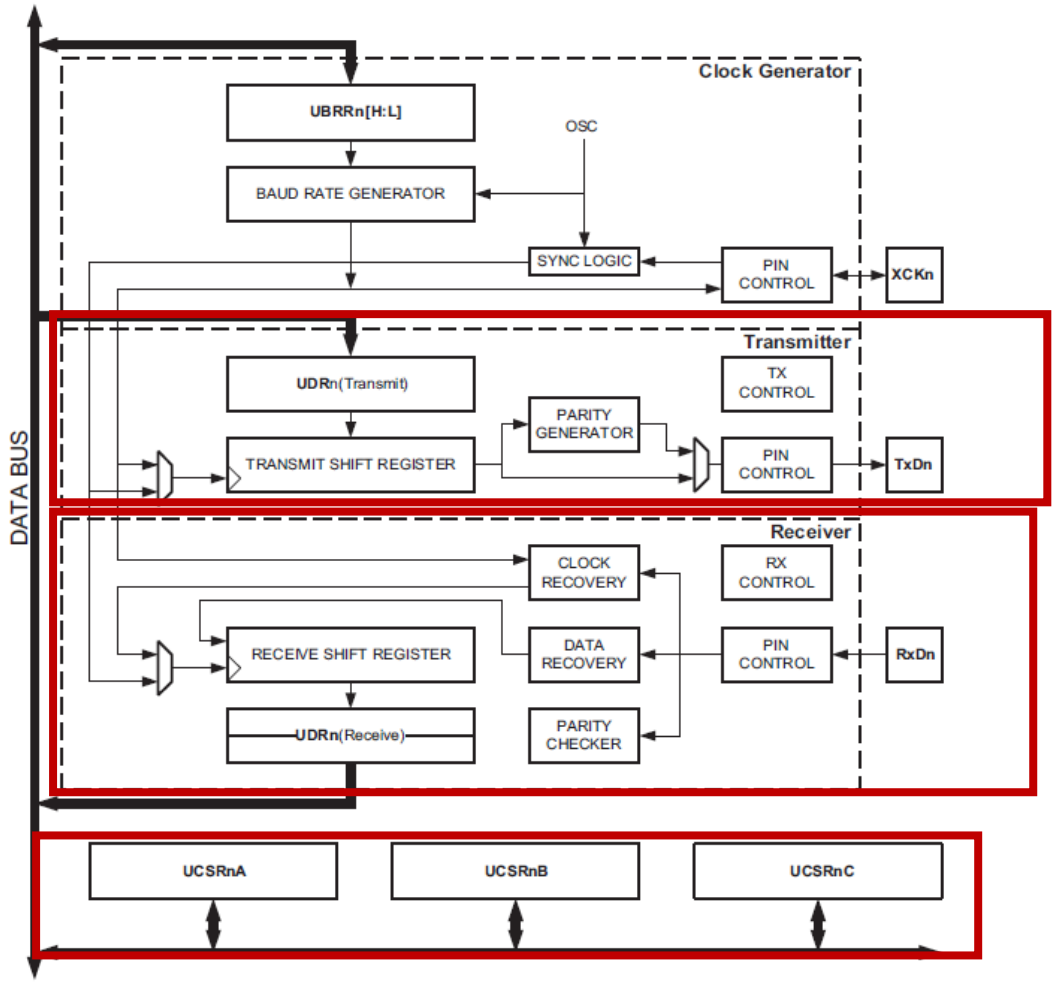


그림 10-5 컴퓨터와 아두이노 우노 보드의 시리얼 연결 1[33]

EMBEDDED

UART

USART Block Diagram[1]



EMBEDDED

실습1: 에코 백

```
#define F_CPU 16000000L
#include <avr/io.h>
#include <util/delay.h>

void UART_INIT(void) {
    UCSR0A |= _BV(U2X0);                    // 2배속 모드

    UBRR0H = 0x00;                          // 통신 속도(보율) 설정
    UBRR0L = 207;

    // 비동기, 8비트 데이터, 패리티 없음, 1비트 정지 비트 모드
    UCSR0C |= 0x06;

    UCSR0B |= _BV(RXEN0);                   // 송수신 가능
    UCSR0B |= _BV(TXEN0);
}

unsigned char UART_receive(void)
{
    while( !(UCSR0A & (1<<RXC0)) );         // 데이터 수신 대기
    return UDR0;
}
```

```
void UART_transmit(unsigned char data)
{
    while( !(UCSR0A & (1<<UDRE0)) );        // 송신 가능 대기
    UDR0 = data;                            // 데이터 전송
}

int main(int argc, char *argv[])
{
    unsigned char data;

    UART_INIT();                            // UART 통신 초기화
    while (1) {
        data = UART_receive();              // 데이터 수신
        UART_transmit(data);                // 수신된 문자를 에코 백
    }

    return 0;
}
```

레지스터

- ATMEGA328의 USART통신과 관련된 레지스터들
- UCSR0A, UCSR0B, UCSR0C : USART장치 설정및 장치 상태와 관련
- UBRR0H, UBRR0L : Baudrate 설정
- UDR0 : 입출력 버퍼
- 자세한 설명은 데이터시트 “20.USART0 -> Register description

EMBEDDED

레지스터

-UCSR0A

비트	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
읽기/쓰기	R	R/W	R	R	R	R	R/W	R/W
초깃값	0	0	1	0	0	0	0	0

그림 10-13 UCSR0A 레지스터의 구조

7	RXC0	Receive Complete: 수신 버퍼(UDR0)에 읽지 않은 문자가 있을 때는 1이 되고 버퍼가 비어 있을 때는 0이 된다. UCSR0B 레지스터의 RXCIE0 비트와 함께 사용되어 수신 완료 인터럽트를 발생시킬 수 있다
5	UDRE0	USART Data Register Empty: 송신 버퍼(UDR0)가 비어 있어 데이터를 받을 준비가 되어 있을 때 1이 된다. UCSR0B 레지스터의 UDRIE0 비트와 함께 사용되어 송신 데이터 레지스터 준비 완료 인터럽트를 발생시킬 수 있다.
1	U2X0	USART Double Transmission Speed: 비동기 전송 모드에서만 사용되며, 2배속 모드이면 1, 1배속 모드이면 0의 값을 가진다.

```
while( !(UCSR0A & (1<<UDRE0)) );        // 송신 가능 대기
```

```
while( !(UCSR0A & (1<<UDRE0)) );
```

```
UCSR0A |= _BV(U2X0);                    // 2배속 모드
```

레지스터

-UCSR0B

비트	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

그림 10-14 UCSR0B 레지스터의 구조

4	RXEN0	RX Enable: UART 수신기의 수신 기능을 활성화한다.
3	TXEN0	TX Enable: UART 송신기의 송신 기능을 활성화한다.
2	UCSZ02	USART Character Size: UCSR0C 레지스터와 함께 전송 데이터의 비트 수를 결정한다.

```
UCSR0B |= _BV(RXEN0);
UCSR0B |= _BV(TXEN0);
```
// 송수신 가능

EMBEDDED

레지스터

- UCSR0C

비트	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	1	1	0

그림 10-15 UCSR0C 레지스터의 구조

표 10-3 통신 모드

UMSEL01	UMSEL00	모드
0	0	비동기 USART
0	1	동기 USART
1	0	-
1	1	마스터 SPI 모드

표 10-4 패리티 비트 모드

UPM01	UPM00	패리티 비트
0	0	사용 안 함
0	1	-
1	0	짝수 패리티
1	1	홀수 패리티

표 10-5 데이터 비트

UCSR0B 레지스터	UCSR0C 레지스터		데이터 비트 수
UCSZ02	UCSZ01	UCSZ00	
0	0	0	5
0	0	1	6
0	1	0	7
0	1	1	8
1	0	0	-
1	0	1	-
1	1	0	-
1	1	1	9

EMBEDDED

레지스터

-UBRROH, UBRROL

비트	7	6	5	4	3	2	1	0
UBRR0H	-	-	-	-	UBRR0H [11:8]			
UBRR0L	UBRR0L [7:0]							
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

표 10-6 UBRR0 레지스터 값 계산식

동작 모드	U2X0	보율 계산식	UBRR0 계산식
비동기 1배속	0	$\frac{f_{osc}}{16 \cdot (UBRR0 + 1)}$	$\frac{f_{osc}}{16 \cdot BAUD} - 1$
비동기 2배속	1	$\frac{f_{osc}}{8 \cdot (UBRR0 + 1)}$	$\frac{f_{osc}}{8 \cdot BAUD} - 1$
동기	-	$\frac{f_{osc}}{2 \cdot (UBRR0 + 1)}$	$\frac{f_{osc}}{2 \cdot BAUD} - 1$

EMBEDDED

레지스터

-UDR0 : 입출력 버퍼
-수신한 값과 송신할 값이 저장된다.

비트	7	6	5	4	3	2	1	0
	RXB [7:0]							
	TXB [7:0]							
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

그림 10-18 UDR0 레지스터의 구조

실습1: 에코 백

_BV = PORTB |= (1 << PB6);

```
#define F_CPU 16000000L
#include <avr/io.h>
#include <util/delay.h>

void UART_INIT(void) {
    UCSR0A |= _BV(U2X0);                    // 2배속 모드

    UBRR0H = 0x00;                          // 통신 속도(보율) 설정
    UBRR0L = 207;

    // 비동기, 8비트 데이터, 패리티 없음, 1비트 정지 비트 모드
    UCSR0C |= 0x06;

    UCSR0B |= _BV(RXEN0);                   // 송수신 가능
    UCSR0B |= _BV(TXEN0);
}

unsigned char UART_receive(void)
{
    while( !(UCSR0A & (1<<RXC0)) );         // 데이터 수신 대기
    return UDR0;
}
```

```
void UART_transmit(unsigned char data)
{
    while( !(UCSR0A & (1<<UDRE0)) );        // 송신 가능 대기
    UDR0 = data;                            // 데이터 전송
}

int main(int argc, char *argv[])
{
    unsigned char data;

    UART_INIT();                            // UART 통신 초기화
    while (1) {
        data = UART_receive();              // 데이터 수신
        UART_transmit(data);                // 수신된 문자를 에코 백
    }

    return 0;
}
```

실습2: UP & DOWN(문자열 수신)

1) 프로젝트에 UART.h, UART.c 파일 추가 후 라이브러리로 사용 (#include “UART.h”)

```
void UART_printString(char *str)          // 문자열 송신
{
    for(int i = 0; str[i]; i++)           // '\0' 문자를 만날 때까지 반복
        UART_transmit(str[i]);            // 바이트 단위 출력
}

void UART_print8bitNumber(uint8_t no)     // 숫자를 문자열로 변환하여 송신. 8비트
{
    char numString[4] = "0";
    int i, index = 0;

    if(no > 0){                           // 문자열 변환
        for(i = 0; no != 0 ; i++)
        {
            numString[i] = no % 10 + '0';
            no = no / 10;
        }
        numString[i] = '\0';
        index = i - 1;
    }

    for(i = index; i >= 0; i--)           // 변환된 문자열 출력
        UART_transmit(numString[i]);
}
```

실습2: UP & DOWN(문자열 수신)

CR, LF 끄기!!!!!!!!!!!!!!!!

```
while(1)
{
    data = UART_receive();              // 데이터 수신
    if(data == TERMINATOR){             // 종료 문자를 수신한 경우
        buffer[index] = '\0';
        process_data = 1;               // 수신 문자열 처리 지시
    }
    else{
        buffer[index] = data;           // 수신 버퍼에 저장
        index++;
    }

    if(process_data == 1){                      // 문자열 처리
        if(strcmp(buffer, "DOWN") == 0){        // 카운터 감소
            counter--;
            UART_printString("Current Counter Value : ");
            UART_print16bitNumber(counter);
            UART_printString("\n");
        }
        else if(strcmp(buffer, "UP") == 0){     // 카운터 증가
            counter++;
            UART_printString("Current Counter Value : ");
            UART_print16bitNumber(counter);
            UART_printString("\n");
        }
        else{                                   // 잘못된 명령어
            UART_printString("** Unknown Command **");
            UART_printString("\n");
        }
        index = 0;
        process_data = 0;
```

성 균 관 대 학 교

Thank You

로 봇 동 아 리