

## 1. 배우기(Learning)

### ① 디지털 IO 입출력

#### A. 핀16번에 디지털 출력을 해봅니다.

디지털 출력을 위해서는 반드시 setup()에서 pinMode(16, OUTPUT)을 통해 16번 핀을 OUTPUT으로 설정이 필요합니다.

그리고 아래와 같이 digitalWrite() 함수로 HIGH/LOW 값을 지정합니다.

```
digitalWrite(16, HIGH); //16번핀에 HIGH를 출력합니다.
```

```
digitalWrite(16, LOW); //16번핀에 LOW를 출력합니다.
```

전체 코드를 통해 확인합니다. 16번 핀은 STATUS LED와 연결되어 있으므로 HIGH일 때는 LED가 꺼지고 LOW일 때 LED가 켜집니다.

```
void setup(){  
    pinMode(16, OUTPUT);  
}  
void loop(){  
    digitalWrite(16, HIGH);  
    delay(100); //0.1 초 지연  
    digitalWrite(16, LOW);  
    delay(100); //0.1 초 지연  
}
```

0.1초 마다 깜빡거리는 LED를 확인하실 수 있습니다.

#### B. 핀1번에 디지털 입력을 받아봅니다.

OpenCM9.04의 IO핀에 디지털 입력을 위해서는 반드시 setup()에서 pinMode(1, INPUT)을 통해 1번 핀을 INPUT으로 설정이 필요합니다.

만약 내부 풀업이 필요하다면 pinMode(1, INPUT\_PULLUP)으로 설정하시고 풀다운 회로가 필요하다면 pinMode(1, INPUT\_PULLDOWN)으로 합니다.

그리고 아래와 같이 digitalRead() 함수로 HIGH/LOW 값을 받습니다. 아무것도 연결되어 있지 않으면 Random한 값이 출력될 수 있습니다.

```
int value = digitalRead(1); // 1번핀을 읽어서 value변수에 할당함
```

전체 코드를 통해 확인합니다.

```
void setup(){  
  
    pinMode(1, INPUT);  
  
    SerialUSB.begin();  
  
}  
  
void loop(){  
    int value = digitalRead(1);  
  
    if ( value == HIGH)  
  
        SerialUSB.println("HIGH Detected!");  
  
    else  
  
        SerialUSB.println("LOW Detected!");  
  
    delay(100);  
  
}
```

### C. 핀1번의 출력을 토글 시켜봅니다.

1번 핀의 현재 출력이 HIGH이면 LOW가 되고, LOW이면 HIGH로 반전 시킵니다.

```
digitalWrite(1, HIGH); // 1번핀이 HIGH가 됩니다.
```

```
togglePin(1); // HIGH 상태였던 1번핀이 다시 LOW가 됩니다.
```

## ② 아날로그 IO 입출력

아날로그 입력은 반드시 OpenCM9.04 실크스크린의 ANALOG IN 영역에 위치한 핀번호만 아날로그 입력이 됩니다. 아날로그 출력은 TIMER를 활용한 PWM출력을 통해 아날로그 출력을 대신합니다.

### A. 핀0번에 아날로그 입력을 받습니다.

0번 핀의 아날로그 입력을 위해서는 pinMode(0, INPUT\_ANALOG)를 통해 아날로그 입력으로 핀모드 설정을 해주세요

```
int value = analogRead(0);
```

// 0번 핀에서 아날로그 입력받아서 value 변수에 할당합니다.

여기서 value에 할당되는 값은 12bit ADC 값으로 0~ 4096값이 됩니다.

전체 코드를 통해 읽은 ADC값을 출력해봅니다.

```
void setup(){
```

```
    pinMode(0, INPUT_ANALOG);
```

```
    SerialUSB.begin();
```

```
}
```

```
void loop(){
```

```
int value = analogRead(0);  
  
SerialUSB.println(value); // value값을 출력합니다.  
  
}
```

## B. 핀6번에 아날로그 출력(PWM)을 합니다.

6번 핀의 아날로그 출력은 pinMode(6, OUTPUT) 또는 pinMode(6, PWM)으로 설정이 필요합니다.

```
analogWrite(6, 10000);
```

PWM출력을 통해 아날로그 출력을 대신합니다. PWM의 Duty cycle은 두번째 인자를 통해 제어합니다. 여기서는 10000이지만 0~ 65535 범위의 값을 지정할 수 있습니다.

전체 코드를 확인합니다.

```
void setup(){  
  
    pinMode(6, OUTPUT); // 또는 pinMode(6, PWM); 동일함  
  
}  
  
void loop(){  
  
    analogWrite(6, 10000);  
  
}
```

analogWrite()의 두번째 인자를 통해 아래와 같은 다양한 Duty Cycle의 PWM을 구현할 수 있습니다.

Duty cycle = 0일 때

Duty cycle = 512 일 때

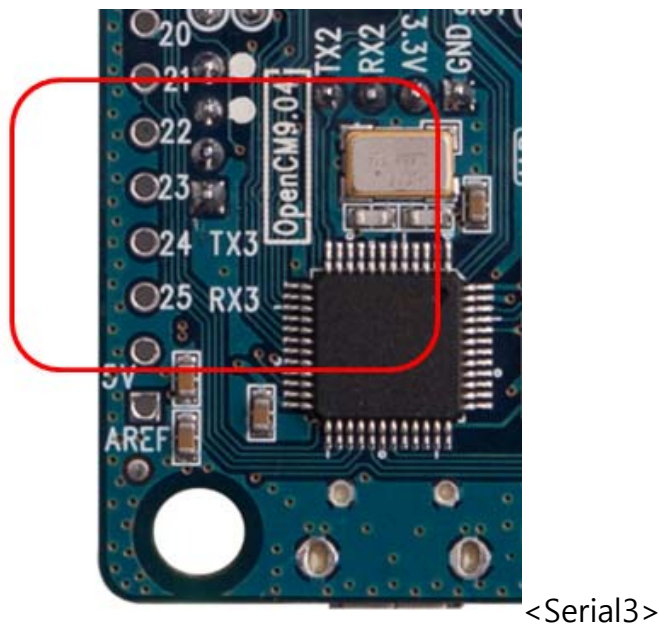
Duty cycle = 10000일 때

Duty cycle = 30000일 때

Duty cycle = 65535일 때

### ③ 시리얼 통신

OpenCM9.04는 총 3개의 시리얼 장치가 있습니다. 시리얼 장치는 USART라고 잘 알려진 기술입니다. Serial1, Serial2, Serial3이 있고 Serial1은 다이나믹셀 통신포트 전용으로 할당되어 있어서 사용상의 제한이 따릅니다. BT-210, BT-110 같은 4핀 포트를 가진 블루투스 장치를 Serial2를 통해 사용하시기를 추천합니다. Serial 번호를 확인하려면 OpenCM9.04 PCB 뒷면을 보십시오. Serial1은 TX1, RX1으로 표시되어 있고 Serial2는 TX2, RX2로 표시되어 있습니다. 마찬가지로 Serial3는 TX3, RX3입니다.



시리얼 USB 장치는 다운로드를 수행하는 USB 통신을 말합니다.

시리얼 USB 장치는 SerialUSB 클래스를 통해 제어하며 기본적인 메소드가 Serial 장치와 거의 동일합니다.

A. 시리얼 장치를 이용해서 데이터를 전송합니다.

반드시 아래와 같이 Serial3 장치에 대한 초기화를 수행한 뒤에 loop()에서 아래의 예제를 수행합니다.

```
void setup(){  
    Serial3.begin(57600);  
}  
  
void loop(){  
    //테스트 예제 코드  
}
```

데이터 전송은 print()와 println() 메소드를 통해 전송하고 print()메소드는 줄바꿈이 없는 출력을 의미하고 println()은 줄바꿈이 있는 출력을 수행합니다.

```
Serial2.print("Hello World This is OpenCM9.04");
```

"Hello World" 스트링을 Serial2(TX2, RX2)장치를 통해 출력합니다.

```
Serial2.print("OpenCM9.04 is the first product of OpenCM Series");
```

```
Serial2.println("    println() ends this line");
```

```
Serial2.println("This is new line");
```

println()은 줄바꿈을 하고 새로운 라인으로 출력을 수행합니다.

아래와 같은 출력을 확인할 수 있습니다.

```
CM-900 is the first product of CM-9 Series    println() ends this line  
This is new line
```

```
Serial2.print(12);
```

10진수 12를 출력합니다.

```
int abc = 128;
```

```
Serial2.print(abc);
```

abc변수의 값 128을 출력합니다.

```
Serial2.print(abc, 16);
```

abc변수의 값 128을 16진수 출력합니다. 0x80이 출력됩니다.

```
Serial2.print(abc, 2);
```

abc변수의 값 128을 2진수 출력을 합니다. 마찬가지로 8진수는 두 번째 인자에 8을 지정하면 되고 두 번째 인자가 없으면 기본적으로 10진수를 출력합니다.

```
Serial2.println(3.14);
```

소수점(double) 타입의 3.14를 출력하고 줄바꿈 합니다. 소수점 2자리 까지 출력합니다.

double 변수를 선언해서 출력할 수도 있습니다.

```
double var = 1.234;
```

```
Serial2.println(var);
```

핀0번, 1번, 2번에서 읽은 아날로그 입력값을 Serial2를 통해 차례로 출력해봅니다.

여러 개의 print()와 println()을 이용하면 아래와 같이 보기 좋게 출력할 수 있습니다.

```
int sensorValue0=0;
```

```
int sensorValue1=0;
```

```
int sensorValue2=0;
```

```
sensorValue0 = analogRead(0);
```

```
sensorValue1 = analogRead(1);
```

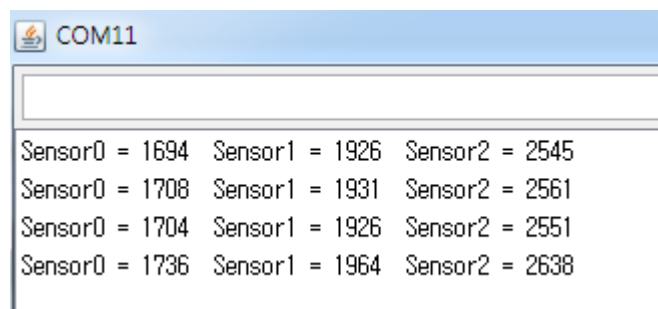
```
sensorValue2 = analogRead(2);
```

```
Serial2.print("Sensor0 = "); Serial2.print(sensorValue0);
```

```
Serial2.print("      Sensor1 = "); Serial2.print(sensorValue1);
```

```
Serial2.print("      Sensor2 = "); Serial2.println(sensorValue2);
```

마지막 sensorValue2 출력만 println()메소드를 이용해서 줄바꿈하면 3개의 아날로그 입력에 대한 깔끔한 출력을 얻을 수 있습니다.



## B. 시리얼 장치를 이용해 데이터를 받습니다.

시리얼 장치로 Echo 기능을 구현해 봅니다.

char형 변수 temp를 통해 Serial2장치에 데이터가 오면 read()메소드로 데이터를 저장하고 바로 print() 메소드로 출력하면 Echo 기능이 구현됩니다.

```
char temp = 0;
```

```
loop(){
```

```
    if ( Serial2.available() ){
```



```

        temp = Serial2.read();

        Serial2.print(temp);
    }
}

```

전체 소스는 아래와 같습니다.

```

void setup(){

    Serial2.begin(57600);

}

byte temp = 0;

void loop(){

    if ( Serial2.available() ){

        temp = Serial2.read();

        Serial2.print(temp);

    }

}

```

아래와 같이 인터럽트 방식으로 구현해 봅니다.

Serial장치의 인터럽트는 아래와 같이 반환형이 없고 byte형 인자가 하나 있는 함수를 구현합니다. 여기서 바로 print()메소드로 넘어오는 데이터 출력하면 Echo 기능이 됩니다. 따로 프로토타입 선언 없이 아무 위치에도 구현해서 사용합니다.

```

void serialInterrupt(byte buffer){

    Serial2.print(buffer);
}

```

```
}
```

이렇게 구현한 serialInterrupt()를 함수 포인터 형태로 setup() 내에서 attachInterrupt()를 통해 설정해 줍니다.

```
Serial2.attachInterrupt(serialInterrupt);
```

인터럽트를 활용한 Serial2 장치의 데이터 입력에 대한 전체 코드는 아래와 같습니다.

```
void setup(){
```

```
    Serial2.begin(57600);
```

```
    Serial2.attachInterrupt(serialInterrupt);
```

```
}
```

```
void serialInterrupt(byte buffer){
```

```
    Serial2.print(buffer);
```

```
}
```

```
void loop(){
```

```
    //코드가 없어도 됩니다.
```

```
}
```

### C. 시리얼 USB 장치를 이용해 데이터를 출력합니다.

아래와 같이 SerialUSB 장치에 대한 초기화를 수행한 뒤에 loop()에서 아래의 예제를 수행합니다. Baud rate에 대한 지정은 없습니다.

```
void setup(){
```

```
    SerialUSB.begin();
```

```
}
```

```
void loop(){
```

```
//테스트 예제 코드
```

```
}
```

Serial 장치와 사용법이 거의 같습니다. print(), println()메소드를 통해 같은 방식으로 제어 하면 됩니다.

```
SerialUSB.print("CM-900 is the first product of CM-9 Series");
```

```
SerialUSB.println(" println() ends this line");
```

```
SeirlaUSB.println("This is new line");
```

10진수 12를 출력해봅니다.

```
SerialUSB.print(12);
```

int형 변수를 통해 출력해봅니다.

```
int abc = 128;
```

```
SerialUSB.print(abc);
```

이번엔 abc 변수의 값을 16진수로 출력합니다.

```
SerialUSB.print(abc, 16);
```

abc변수의 값 128이 16진수 형태인 0x80이 출력됩니다.

```
SerialUSB.print(abc, 2);
```

abc변수의 값 128을 2진수 출력을 합니다. 마찬가지로 8진수는 두 번째 인자에 8을 지정하면 되고 두 번째 인자가 없으면 기본적으로 10진수를 출력합니다.

```
SerialUSB.println(3.14);
```

소수점(double) 타입의 3.14를 출력하고 줄바꿈 합니다. 소수점 2자리 까지 출력합니다.

double 변수를 선언해서 출력해 봅니다.

```
double var = 1.234;
```

```
SerialUSB.println(var);
```

#### D. 시리얼 USB장치를 이용해 데이터를 입력 받습니다.

시리얼 USB 장치로 Echo 기능을 구현해 봅니다.

char형 변수 temp를 통해 Serial USB장치에 데이터가 오면 read()메소드로 데이터를 저장하고 바로 print() 메소드로 출력하면 Echo 기능이 구현 됩니다

```
char temp = 0;
```

```
loop(){
```

```
    if ( SerialUSB.available() ){
```

```
        temp = SerialUSB.read();
```

```
        SerialUSB.print(temp);
```

```
    }
```

```
}
```

전체 소스는 아래와 같습니다.

```
void setup(){
```

```

SerialUSB.begin();

}

byte temp = 0;

void loop(){

    if ( SerialUSB.available() ){

        temp = SerialUSB.read();

        SerialUSB.print(temp);

    }

}

```

아래와 같이 인터럽트 방식으로 구현해 봅니다.

Serial USB의 인터럽트는 아래와 같이 반환형이 없고 byte형 인자와 byte\* 인자가 있는 함수로 구현합니다. 여기서 바로 print() 메소드로 넘어오는 데이터를 출력하면 Echo 기능이 됩니다. PC의 터미널로 USB COM 포트에 데이터를 쓰면 1바이트 씩 전송되기 때문에 nCount =1 이고 buffer 의 index 0번 데이터만 Echo하면 충분합니다.

```

void usbInterrupt(byte nCount, byte* buffer){

    SerialUSB.print(buffer[0]);

}

```

이렇게 구현한 usbInterrupt()를 함수 포인터 형태로 setup() 내에서 attachInterrupt()를 통해 설정해 줍니다.

```

SerialUSB.attachInterrupt(usbInterrupt);

```

loop()함수는 아래와 같이 빈 함수로 두어도 괜찮습니다.

```

void loop(){

```

```
}
```

SerialUSB 장치의 인터럽트 활용한 전체 코드는 아래와 같습니다.

```
void setup(){
```

```
    SerialUSB.begin();
```

```
    SerialUSB.attachInterrupt(usbInterrupt);
```

```
}
```

```
void usbInterrupt (byte nCount, byte* buffer){
```

```
    SerialUSB.print(buffer[0]);
```

```
}
```

```
void loop(){
```

```
    //코드가 없어도 됩니다.
```

```
}
```

#### ④ 수학 함수 사용법

ROBOTIS OpenCM은 별도의 헤더파일 추가 없이 바로 삼각함수와 같은 기본 Math 라이브러리를 쓸 수 있습니다.

##### A. 기본 수학 함수 사용법

아날로그 입력을 받아서 100보다 작은 값은 받아봅니다.

```
sensorValue = min(sensorValue, 100);
```

min(a,b) 함수는 a와 b 둘중에 작은 값만 return 합니다. 따라서 100보다 큰값은 sensorValue에 할당되지 않습니다.

반대로 0보다 큰 ADC 값만 받는 코드를 만들어 봅니다.

```
sensorValue = max(sensorValue, 0);
```

max(a,b)는 a와 b 둘중에 큰 값만 return합니다. 따라서 0보다 작은 값은 return 되지 않습니다.

아날로그 입력을 받고 0~100사이의 범위의 값만 받아 봅니다.

constrain(x,a,b)를 이용하면 a~b 범위사이의 x값만 리턴합니다.

```
sensorValue = constrain(sensorValue, 0, 100);
```

서로 다른 범위의 값끼리 매핑하는 예제를 만들어 봅니다.

아날로그 입력으로 받은 값의 범위가 0~4096인데, 이 값으로 PWM출력을 한다면 1:1 매핑이 어렵습니다. 왜냐하면 PWM 출력 범위가 0~65535이기 때문입니다.

아래와 같이 map() 함수를 써서 간단히 해결할 수 있습니다.

```
sensorValue = analogRead(0); // 0번핀에서 아날로그 입력 받고
```

```
sensorValue = map(sensorValue, 0, 4095, 0, 65535);
```

```
analogWrite(8, sensorValue);
```

을 계산해 봅니다.

pow(double x, double y)함수를 이용해 간단히 구현합니다.


```
calc = pow(9, 3);
```

간단히 제곱승은 `sq(a)`라는 매크로로 해결 가능합니다.

을 계산한다면

```
calc = sq(3);
```

결과는 `calc`에 9가 할당됩니다.

 계산을 수행해봅니다.

`sqrt(double x)`를 이용해 간단히 구현합니다.

```
calc = sqrt(4); //  를 계산합니다.
```

결과는 `calc`에 2가 할당 됩니다.

## B. Sin, Cos, Tan 값을 출력해 봅니다.

`sin`, `cos`, `tan` 계산은 아래의 함수를 이용합니다.

```
double sin(double x)
```

```
double cos(double x)
```

```
double tan(double x)
```

`x`는 라디안(radian)값을 의미합니다.

라디안 3.14(180도)에 해당하는 `sin`, `cos`, `tan` 값을 구해봅니다.

```
double result=0;
```

```
result = sin(3.14); //180의 sin값 출력
```

```
result= cos(3.14); //180도의 cos값 출력
```



```
result= tan(3.14); //180도의 tan값 출력
```

## ⑤ Time 함수 사용법

프로그램 시작 이후로 지난 시간을 millisecond로 확인해 봅니다.

```
int time = millis();
```

time 변수에 프로그램 시작한 이후의 millisecond가 할당됩니다. 시간이 지남에 따라 overflow할때까지 계속 증가합니다.

참고로 millis()함수에 대한 자세한 타입은 아래와 같습니다.

```
uint32 millis(void)
```

이번에는 microsecond 단위로 시간을 체크해 봅니다.

```
time = micros();
```

프로그램 시작 이후로 지난 시간을 microsecond 단위로 반환합니다. overflow할때까지 계속 증가하고 거의 70분정도 지나면 다시 0으로 리셋 됩니다.

이렇게 time변수 값을 아래와 같이 SerialUSB 같은 장치로 출력해봅니다.

```
SerialUSB.print("time : "); SerialUSB.println(time);
```

이번에는 LED 깜빡이는 예제에서 많이 쓰이는 delay()에 대해 배워 봅니다.

1초동안 CPU가 아무것도 하지 않고 지연하는 방법을 알아봅니다.

void delay(unsigned long ms)를 이용해서 1000 millisecond정도 지연시키면 1초 지연이 됩니다.

```
delay(1000);
```

**참고 1 sec = 1,000 millisecond , 1 millisecond = 1,000 microsecond**

아주 미세하게 500 us 지연시키는 방법을 알아보니다.

microsecond 단위로 cpu를 지연시키기 위해서는

void delayMicroseconds(unsigned int us) 함수를 활용합니다.

```
delayMicroseconds(500);
```

**하지만 OpenCM9.04 CPU(STM32)의 클럭 정확도가 micro second단위까지 보장할 만큼 정확하지 않기 때문에 정확한 지연을 보장하지는 않습니다.**

## ⑥ 랜덤함수 사용법

0~10사이의 랜덤한 수를 발생시켜 봅니다.

long random(long max) 또는

long random(long min, long max) 를 이용합니다.

```
int ranNum = random(0, 10);
```

또는 min값 없이 max값만 지정해도 됩니다.

```
int ranNum = random(10);
```

## ⑦ 외부 인터럽트 사용법

0번핀에 입력되는 신호가 변할 때 Status LED를 켜다 끄는 코드를 만들어 봅니다.

전역변수 하나로 플래그를 만들어서 0번핀의 신호가 변할 때 마다 인터럽트 루틴에서 플래그를 토글시키는 방법을 응용합니다.

인터럽트는 attachInterrupt() 함수로 attach할 수 있습니다.

```
volatile int state = LOW;
```

```
attachInterrupt(0, exInterrupt, CHANGE); //신호가 변할 때마다 blink 실행
```

exInterrupt()를 구현합니다. void exInterrupt(void) 타입으로 구현합니다.

```
void exInterrupt(){
```

```
    if(state == HIGH)
```

```
        state = LOW;
```

```
    else
```

```
        state= HIGH;
```

```
}
```

```
loop(){
```

```
    digitalWrite(BOARD_LED_PIN, state);
```

```
}
```

state의 값에 따라 loop()에서 STATUS LED가 on/off를 반복합니다.

## ⑧ 다이나믹셀 사용법

아래의 예제에 사용된 다이나믹셀은 ID=1번, 통신 속도는 모두 Dxl.begin(1) = 1M bps로 초기화를 전제로 합니다.

A. AX-12A의 모델 넘버와 펌웨어 버전을 읽습니다.

E-manual에서 AX-12A의 Control Table에서 모델넘버와 펌웨어 버전에 해당하는 주소값은 아래와 같습니다.

Area	주소 (16진수)	명칭	의미	접근	초기값 (16진수)
	0 (0x00)	Model Number(L)	모델 번호의 하위 바이트	R	12 (0x0C)
	1 (0x01)	Model Number(H)	모델 번호의 상위 바이트	R	0 (0x00)
	2 (0x02)	Version of Firmware	펌웨어 버전 정보	R	-

ID가 1번인 AX-12A에서 모델 넘버의 하위바이트에 해당하는 0번 주소와 펌웨어 버전 정보에 해당하는 2번주소를 읽어보겠습니다. 모두 1바이트이므로 byte형 타입의 변수를 쓰면 됩니다.

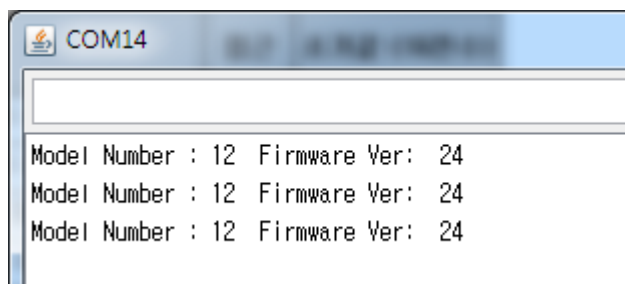
```
byte nModel = Dxl.readByte(1, 0); // 모델 번호를 읽고
```

```
byte vFirmware = Dxl.readByte(1, 2); // 펌웨어 버전을 읽습니다.
```

아래와 같이 출력합니다.

```
SerialUSB.print("Model Number : ");SerialUSB.print(nModel);
```

```
SerialUSB.print("  Firmware Ver : ");SerialUSB.println(vFirmware);
```



<출력 확인>

B. ID가 1인 AX-12A의 현재 내부 온도를 읽어봅니다.

Control Table에서 AX-12A의 내부온도에 해당하는 주소는 아래와 같습니다.

43 (0x2B)	Present Temperature	현재 온도	R	-
-----------	---------------------	-------	---	---

같은 방식으로 readByte()로 한바이트 읽어 봅니다.

```
byte temp = Dxl.readByte(1, 43);
```

```
SerialUSB.print("Current Temperature : ");SerialUSB.println(temp);
```

### C. AX-12의 ID를 2로 설정합니다.

다이나믹셀 ID에 해당하는 3번 주소에 readWrite() 메소드로 활용합니다.

3 (0x03)	ID	다이나믹셀 ID	RW	1 (0x01)
----------	----	----------	----	----------

```
void setup(){
```

```
    Dxl.begin(1);
```

```
    delay(1000); // 1초정도 딜레이를 주는 것이 좋습니다.
```

```
    Dxl.writeByte(1, 3, 2);
```

```
    int CommStatus = Dxl.getResult();
```

```
    if( CommStatus == COMM_RXSUCCESS){
```

```
        SerialUSB.println("Changed Successfully!");
```

```
    }
```

```
    else{
```

```
        SerialUSB.println("Error");
```

```
    }
```

```
}
```

ID 변경은 Setup()함수에서 수행하는 것이 좋습니다. 반드시 통신 성공 여부를 확인해야 합니다.

이제 ID 1번인 다이내믹셀은 ID 2번이 되었습니다.

#### D. Baud Rate를 57600 bps로 변경합니다.

ID변경과 같은 방식으로 4번 주소에 readWrite() 메소드를 활용합니다.

57600 bps는 다이내믹셀 속도 계산을 활용하면 index 값이 34가 나옵니다.

4 (0x04)	Baud Rate	다이내믹셀 통신 속도	RW	1 (0x01)
----------	-----------	-------------	----	----------

```
void setup(){
```

```
    Dxl.begin(1);
```

```
    delay(1000); // 1초정도 딜레이를 주는 것이 좋습니다.
```

```
    Dxl.writeByte(1, 4, 34); // 34 = 57600 bps
```

```
    int CommStatus = Dxl.getResult();
```

```
    if( CommStatus == COMM_RXSUCCESS){
```

```
        SerialUSB.println("Changed Successfully!");
```

```
    }
```

```
    else{
```

```
        SerialUSB.println("Error");
```

```
    }
```

}

Baud rate가 변경되었으므로 새롭게 Dxl.begin(34)로 버스를 새롭게 초기화해야 합니다.

#### E. ID가 1인 다이내믹셀의 움직임 유무를 확인합니다.

Control Table에서 46(0x2E)값을 이용해 현재 AX-12A의 움직임 여부를 확인할 수 있습니다.

46 (0x2E)	Moving	움직임 유무	R	0 (0x00)
-----------	--------	--------	---	----------

```
byte bMoving = Dxl.readByte(1, 46);
```

ID 1번 다이내믹셀이 현재 움직이고 있다면 bMoving이라는 변수에 1이 반환되고 움직임이 없다면 0을 반환합니다.

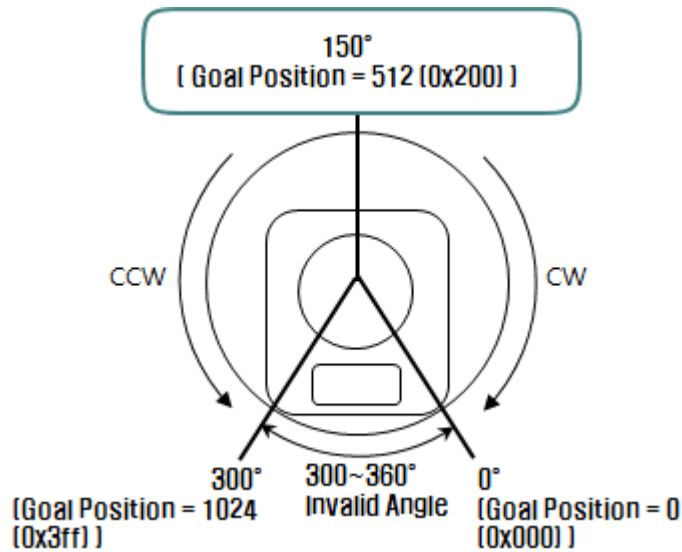
#### F. AX-12A 다이내믹셀을 150도 위치로 움직여 봅니다.

다이내믹셀을 목표 위치(150도)로 움직이려면 Goal Position에 해당하는 주소에 원하는 이동 위치를 넣어주면 됩니다.

아래와 같이 하위, 상위 2바이트로 이루어져 있으며 각각 접근하는 것보다는 하위 바이트 30(0x1E)에 writeWord() 메소드를 통해 2바이트(1워드)를 기록하는 것을 추천합니다.

30 (0x1E)	Goal Position(L)	목표 위치 값의 하위 바이트	RW	-
31 (0x1F)	Goal Position(H)	목표 위치 값의 상위 바이트	RW	-

E-manual에서 150도에 해당하는 위치는 아래의 그림과 같이 512와 매칭됨을 확인할 수 있습니다.



```
Dxl.writeWord(1, 30, 512);
```

Dxl.getResult() 함수로 통신 성공 여부를 확인하시길 바랍니다.

G. 각의 RX-64에 대해서 각각 다음의 위치와 속도로 이동합니다.

**ID 0 인 RX-64 : 0x010 위치로 속도 0x150 으로 이동**  
**ID 1 인 RX-64 : 0x220 위치로 속도 0x360 으로 이동**  
**ID 2 인 RX-64 : 0x030 위치로 속도 0x170 으로 이동**  
**ID 3 인 RX-64 : 0x220 위치로 속도 0x380 으로 이동**

여러 개의 다이나믹셀을 한번에 움직이기 위해서는 Syncwrite방법으로 제어할 수 있습니다. Syncwrite는 아래의 규칙으로 직접 Packet을 만들고 동시에 전송합니다. Packet은 setTxPacketXXXX() 메소드를 이용해서 만들어 봅니다.



**ID** 0xFE

**Length** (L+1) × N + 4 (L:RX-64별 Data Length, N:RX-64의 개수)

**Instruction** 0x83

**Parameter1** Data를 쓰고자 하는 곳의 시작 Address

**Parameter2** 쓰고자 하는 Data의 길이 (L)

**Parameter3** 첫 번째 RX-64의 ID

**Parameter4** 첫 번째 RX-64의 첫 번째 Data

**Parameter5** 첫 번째 RX-64의 두 번째 Data

...

**Parameter** L+3 첫 번째 RX-64의 L번째 Data

**Parameter** L+4 두 번째 RX-64의 ID

**Parameter** L+5 두 번째 RX-64의 첫 번째 Data

**Parameter** L+6 두 번째 RX-64의 두 번째 Data

...

**Parameter** 2L+4 두 번째 RX-64의 L번째 Data



일반적으로 동시 제어 가능한 다이나믹셀은, 1개의 명령 패킷이 4바이트인 경우 26개까지 가능합니다.  
다이나믹셀의 수신버퍼 용량이 143byte 이므로 Packet의 길이가 143byte를 초과하지 않도록 하십시오.

```
Dxl.setTxPacketId(BROADCAST_ID);
```

```
Dxl.setTxPacketInstruction(INST_SYNC_WRITE);
```

Goal Position과 Moving Speed에 대한 컨트롤 테이블은 연속해서 있지만 writeWord()와는 다르게 반드시 Dxl.getLowByte(), Dxl.getHighByte() 메소드를 통해 하위, 상위 바이트를 명시합니다.

30 (0x1E)	Goal Position(L)	목표 위치 값의 하위 바이트	RW	-
31 (0x1F)	Goal Position(H)	목표 위치 값의 상위 바이트	RW	-
32 (0x20)	Moving Speed(L)	목표 속도 값의 하위 바이트	RW	-
33 (0x21)	Moving Speed(H)	목표 속도 값의 상위 바이트	RW	-

먼저 위치와 속도 데이터에 대한 배열을 선언합니다.

```
word GoalPos[4]={0x010, 0x220, 0x030, 0x220};
```

```
word MovingSpd[4]={0x150, 0x360, 0x170, 0x380};
```

```
Dxl.setTxPacketParameter(0, 30);
```

```
Dxl.setTxPacketParameter(1, 4); //4바이트(2워드) 데이터 길이
```

```
for( i=0; i < 4 ; i++){ // 다이나믹셀 개수 = 4
```

```
    Dxl.setTxPacketParameter(2+5*i, i);
```

```
    Dxl.setTxPacketParameter(2+5*i+1, Dxl.getLowByte(GoalPos[i]));
```

```
    Dxl.setTxPacketParameter(2+5*i+2, Dxl.getHighByte(GoalPos[i]));
```

```
    Dxl.setTxPacketParameter(2+5*i+3, Dxl.getLowByte(MovingSpd[i]));
```

```
    Dxl.setTxPacketParameter(2+5*i+4, Dxl.getHighByte(MovingSpd[i]));
```

```
    SerialUSB.print("ID : "); SerialUSB.print(i); // 현재 ID 출력
```

```
    SerialUSB.print(" Goal Position : "); SerialUSB.print(GoalPos[i]);
```

```
    SerialUSB.print(" Moving Speed : "); SerialUSB.println(MovingSpd[i]);
```

```
}
```

```
Dxl.setTxPacketLength( (4+1)*4 + 4); // Packet 전체 길이(계산식 참조)
```

데이터 길이 = 4, 다이나믹셀 개수 = 4

```
Dxl.txrxPacket(); // 최종 Packet 전송 명령
```

```
int CommStatus = Dxl.getResult();
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // 통신 성공 여부 체크
```

```
...
```

```
Instruction Packet : 0xFF 0xFF 0xFE 0x18 0x83 0x1E 0x04 0x00 0x10 0x00  
                    0x50 0x01 0x01 0x20 0x02 0x60 0x03 0x02 0x30 0x00  
                    0x70 0x01 0x03 0x20 0x02 0x80 0x03 0x12
```

H. 동작각을 0~150도로 제한합니다.

CCW Angle Limit이 0x3FF일 경우 300도 이므로 150도에 해당하는

0x200을 writeByte() 메소드를 이용해 전송합니다.

8 (0x08)	CCW Angle Limit(L)	반시계 방향 한계 각도 값의 하위 바이트	RW	255 (0xFF)
----------	--------------------	------------------------	----	------------

```
Dxl.writeByte(1, 8, 0x200);
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // 통신 성공 여부 체크
```

```
...
```

#### I. 동작전압을 10V ~ 17V로 설정합니다.

10V의 데이터는 100(0x64), 17V는 170(0xAA) 이므로 각각 writeByte() 메소드를 통해 하한, 상한을 기록합니다. 컨트롤 테이블 주소는 각각 하한 범위 전압 = 12(0x0C), 상한 범위 전압 = 13(0x0D) 입니다.

12 (0x0C)	the Lowest Limit Voltage	최저 한계 전압	RW	60 (0x3C)
13 (0x0D)	the Highest Limit Voltage	최고 한계 전압	RW	140 (0xBE)

```
Dxl.writeByte(1, 12, 100);
```

```
Dxl.writeByte(1, 13, 170);
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // 통신 성공 여부 체크
```

```
...
```

#### J. 토크를 최대값의 50%만 발휘하도록 합니다.

MAX Torque 값을 최대값인 0x3FF의 50%인 0x1FF로 설정합니다. Max Torque 하위 바이트 주소 14(0x0E)에 writeByte()이용해 데이터를 씁니다.

14 (0x0E)	Max Torque(L)	토크 한계 값의 하위 바이트	RW	255 (0xFF)
15 (0x0F)	Max Torque(H)	토크 한계 값의 상위 바이트	RW	3 (0x03)

```
Dxl.writeByte(1, 14, 0x1FF);
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // 통신 성공 여부 체크
```

...

전원을 off 한 뒤 다시 전원을 공급해야만 Max Torque가 변경됩니다.

K. 57RPM의 속도로 Position 180도에 위치시킵니다.

Moving Speed( Address 32(0x20) ) = 512(0x200)

Goal Position( Address 30(0x1E) ) = 512 (0x200)

로 설정합니다. 아래와 같이 word단위 데이터로 접근합니다.

```
Dxl.writeWord(1, 32, 512); // 57 RPM의 속도 설정
```

```
Dxl.writeWord(1, 30, 512); // 180도 위치로 이동
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // 통신 성공 여부 체크
```

...

L. ID가 0인 AX-12는 위치0도에 ID가 1인 AX-12는 위치 300도에 위치시킵니다.(단 두 개의 AX-12 를 똑같은 시점에 출발)

Syncwrite와 마찬가지로 직접 setTxPacketXXX() 메소드를 활용해서 Packet을 만듭니다. 이 경우 INST\_REG\_WRITE와 INST\_ACTION을 이용해서 Packet을 만들어 보니다. 참고로 위치 0도는 0에 해당하고 300도는 1023(0x3FF)에 해당합니다.

ID=0, Instruction = INST\_REG\_WRITE, Address = 30(0x1E), Data = 0

ID=1, Instruction = INST\_REG\_WRITE, Address = 30(0x1E), Data = 1023

```
Dxl.setTxPacketId(0); // 0번 ID 제어를 명시합니다.
```

```
Dxl.setTxPacketInstruction(INST_REG_WRITE);
```

```
Dxl.setTxPacketParameter(0, 30); // Goal Position Address
```

```
Dxl.setTxPacketParameter(1, Dxl.getLowByte(0)); // Low Byte
```

```
Dxl.setTxPacketParameter(2, Dxl.getHighByte(0)); // High Byte
```

```
Dxl.setTxPacketLength(5); //전체 데이터 길이 = 데이터 길이 + 3
```

```
Dxl.txrxPacket();
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // 통신 성공 여부 체크
```

```
...
```

**Instruction Packet: FF FF 00 05 04 1E 00 00 D8**

두 번째 다이내믹셀에 대한 패킷 전송

```
Dxl.setTxPacketId(1);
```

```
Dxl.setTxPacketInstruction(INST_REG_WRITE);
```

```
Dxl.setTxPacketParameter(0, 30); // Goal Position Address
```

```
Dxl.setTxPacketParameter(1,Dxl.getLowByte(1023)); //Low Byte
```

```
Dxl.setTxPacketParameter(2, Dxl.getHighByte(1023)); //High Byte
```

```
Dxl.setPacketLength(5);
```

```
Dxl.txrxPacket();
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // 통신 성공 여부 체크
```

```
...
```

**Instruction Packet: FF FF 01 05 04 1E FF 03 D5**

0번과 1번 다이내믹셀의 레지스터에 대기중인 Instruction을 바로 실행하려면 INST\_ACTION Packet을 전송합니다.

```
Dxl.setTxPacketId(BROADCAST_ID);
```

```
Dxl.setTxPacketInstruction(INST_ACTION);
```

```
Dxl.setTxPacketLength(2);
```

```
Dxl.txrxPacket();
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // 통신 성공 여부 체크
```

```
...
```

**Instruction Packet: FF FF FE 02 05 FA (LEN:006)**

각각 Packet을 만들어서 전송할 때마다 통신 성공 여부를 체크하는 것이 좋습니다.