1. Learning (implement APIs to OpenCM9.04)

① Digital I/O (input/output)

A. Digital output on pin 16.

Set pinMode(16, OUTPUT) in setup(); this sets pin-16 to OUTPUT.

Declare digitalWrite() to HIGH/LOW.

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

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

Pin-16 reads STATUS LED; when HIGH the LED is off; when LOW the LED is on.

```
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
}
```

Blinks in 0.1sec intervals

B. Digital input on pin 1.

Set pinMode(1, INPUT) in setup(); this sets pin-1 to INPUT.

If external pull-up needed set pinMode(1, INPUT_PULLUP); for pull-down set pinMode(1, INPUT_PULLDOWN).

digitalRead() gets HIGH/LOW value. If pin is not connected then value could be random.

```
int value = digitalRead(1); // read #1, value assigned
```

The code.

```
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. Toggle pin 1.

Switch pin 1 from high-to-low then low-to-high.

```
digitalWrite(1, HIGH); // set pin 1 to HIGH.
```

```
togglePin(1); // switches pin 1 from HIGH to LOW.
```

② Analog I/O (input/output)

Analog input pins are labeled ANALOG IN on the OpenCM9.04 silk screen..

A. Analog input pin 0.

Set pinMode(0, INPUT_ANALOG) in setup(); this sets pin-0 to INPUT_ANALOG

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

// pin-0gets analog input, value assigned.

The assigned value gets converted in a 12-bit ADC value (0~ 4095).

```
void setup(){

    pinMode(0, INPUT_ANALOG);

    SerialUSB.begin();

}

void loop(){

    int value = analogRead(0);

    SerialUSB.println(value);     // output of value.

}
```

B. Analog output (PWM) on pin 6

Set pin-6 to pinMode(6, OUTPUT) or pinMode(6, PWM).

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

Analog output as PWM. PWM’s duty cycle is set on the second value (10000).

Range is 0~ 65535.

The code.

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

In analogWrite() the second value is PWM's implementation as duty cycle.

Duty cycle = 0

Duty cycle = 512

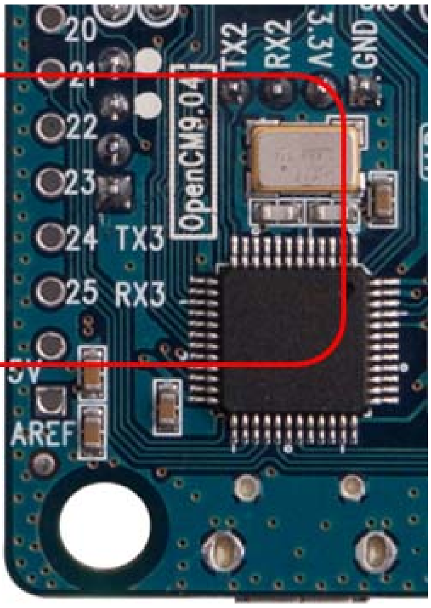
Duty cycle = 10000

Duty cycle = 30000

Duty cycle = 65535

③ Serial communicatrions

The OpenCM9.04 has a total of 3 serial devices. These are USART serial1, serial2, and serial3. Serial1 is assigned to Dynamixel comm port. Serial2 for 4-pin BT-210, BT-110 devices. To see the serial pins see reverse side of OpenCM9.04. Serial1 has TX1 and RX1. Serial2 has TX2 and RX2. Serial3 has TX3 and RX3.



<Serial3>

Serial USB device download is USB communications.

Serial USB devices are controlled via SerialUSB method.

A. Data transmission via serial device

Initialize device in and run in loop().

```
void setup(){
      Serial3.begin(57600);
}
void loop(){
      // place code here
}
```

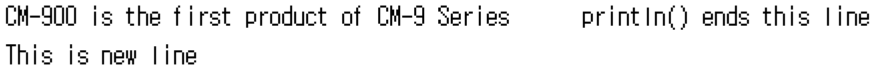
Data transmission can be outputted with print() and println(). print() has no line brakes while printIn() does.

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

"Hello World" is outputted via Serial2(TX2, RX2) device.

```
Serial2.print("OpenCM9.04 is the first product of OpenCM Series");
Serial2.println("     println() ends this line");
Seirla2.println("This is new line");
```

println() outputs in a new line.



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

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

Output 12 in decimal (default).

```
int abc = 128;
```

```
Seial2.print(abc);
```

Outputs abc as 128.

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

Outputs abc in hexadecimal (0x80).

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

Outputs abc in binary

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

Outputs a double data type and ends line; outputs 2 significant places

Can output declared double variables.

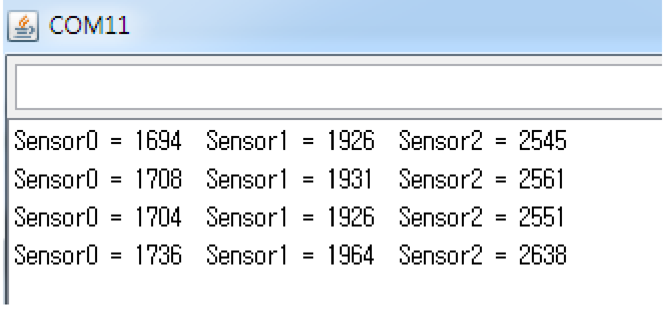
```
double   var = 1.234;
```

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

Input analog values to pin-0, pin-1, pin-2; in turn output via Serial2.

```
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);
```

sensorValue2 outputs all 3 pins one line at a time with println().



```
Sensor0 = 1694  Sensor1 = 1926  Sensor2 = 2545
Sensor0 = 1708  Sensor1 = 1931  Sensor2 = 2561
Sensor0 = 1704  Sensor1 = 1926  Sensor2 = 2551
Sensor0 = 1736  Sensor1 = 1964  Sensor2 = 2638
```

B. Receive data from serial device

Echo feature can be implemented with serial devices.

Assign temp as char type and save data from Serial2 with read(); use print() to output data for echo purposes.

```
char temp = 0;
loop(){
    if ( Serial2.available() ){
            temp = Serial2.read();
            Serial2.print(temp);
    }
}
```

The code

```
void setup(){
    Serial2.begin(57600);
}
byte temp = 0;
void loop(){
    if ( Serial2.available() ){
            temp = Serial2.read();
            Serial2.print(temp);
    }
}
```

Interrupt implementation

Interrupts from serial devices do not return values. Incoming data can be echoed with print(). This can be implemented without declaring separate prototypes.

```
void serialInterrupt(byte buffer){
    Serial2.print(buffer);
}
```

serialInterrupt() can be implemented as a pointer in setup().

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

Code with serial2 device.

```
void setup(){
    Serial2.begin(57600);
    Serial2.attachInterrupt(serialInterrupt);
}
void serialInterrupt(byte buffer){
    Serial2.print(buffer);
}
void loop(){
    //OK to leave empty here.
}
```

C. Output data with serial USB device

initialize SerialUSB device in setup(); run code in loop(). There is no need to declare baud rate value.

```
void setup(){
    SerialUSB.begin();
}
```

```
void loop(){
    //place code here
}
```

Use print() and println() for control.

```
SerialUSB.print(“CM-900 is the first product of CM-9 Series”);
SerialUSB.println(“ println() ends this line”);
SeirlaUSB.println(“This is new line”);
SerialUSB.print(12);
```

Outputs 12 in decimal (default).

```
int abc = 128;
SerialUSB.print(abc);
```

Outputs abc as 128.

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

Outputs abc in hexadecimal.

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

Outputs abc in binary

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

Output of double type; output is 3.14 (2decimal places by default).

```
double   var = 1.234;
SerialUSB.println(var);
```

Outputs var as is (with 3 decimal figures)

D. Receive data with serial USB device

Implement echo to serial USB device.

Assign temp as char type and save data from serial USB device with read(); use print() to output data for echo purposes

```
char temp = 0;

loop(){

    if ( SerialUSB.available() ){

            temp = SerialUSB.read();

            SerialUSB.print(temp);

    }

}
```

The code

```
void setup(){

    SerialUSB.begin();

}

byte temp = 0;

void loop(){

    if ( SerialUSB.available() ){

            temp = SerialUSB.read();

            SerialUSB.print(temp);

    }

}
```

Interrupt implementation

Interrupts from serial USB do not return values byte and *byte types are implemented. Incoming data can be echoed with print(). When data is written to the USB COM port is done 1byte chunks (nCount). Only index 0 of transmitted byte is necessary for echoing.

```
void usbInterrupt(byte nCount, byte* buffer){
    SerialUSB.print(buffer[0]);
}
```

Implement usbInterrupt()pointer on setup() through attachInterrupt().

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

Its ok to keep loop() empty.

```
void loop(){
}
```

SerialUSB device’s interrupt code.

```
void setup(){
    SerialUSB.begin();
    SerialUSB.attachInterrupt(usbInterrupt);
}
void usbInterrupt (byte nCount, byte* buffer){
    SerialUSB.print(buffer[0]);
}
void loop(){
    //OK to leave empty here.
}
```

④ Math functions

Trigonometric functions can be implemented to ROBOTIS OpenCM without any additional header files.

A. Basic math functions

Get analog input and receive a value less than 100.

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

min(a,b) only returns values lower than 100. Anything greater than 100 sensorValue does not get assigned.

Oppositely the following return values greater than 0.

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

max(a,b) only returns values greater than 0. Anything lesser than 0 there is no return.

Receive an analog input and get values only between 0 to 100.

constrain(x,a,b) returns x (if x is between and and b).

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

When receiving converter analog values (0~4096) these are mapped 1:1. This is due to PWM having outputs (0~65535).

This can be done with map() function.

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

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

analogWrite(8, sensorValue);

Calculate (nine cube).

Simply implement pow(double x, double y) function.

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

for squares there’s a macro sq(a).

with

calc = **sq(3);**

calc returns 9.

$\sqrt{\square}$ square roots.

Simply implement sqrt(double x) function.

calc = **sqrt(4)**; calculate //$\sqrt{4}$.

calc returns 2.

B. Output Sin, Cos, Tan

Implement the following functions to obtain sin, cos, and tan.

double sin(double x)

double cos(double x)

double tan(double x)

where x is in radians.

Set a radian value of 3.14 and implement sin, cos, tan functions.

```
double result=0;
```

```
result = sin(3.14); //get sine of 180
```

```
result= cos(3.14);//get cosine of 180
```

```
result= tan(3.14); //get tangent of 180
```

⑤ Time functions

Time unit is in milliseconds.

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

The time variable returns millisecond values. Time increases until overflow.

Please refer to the millis() function type.

```
uint32 millis(void)
```

The following has time unit in microseconds.

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

time returns microsecond values. Value increases until overflow (about the 70 min mark) then it resets to 0.

Time variable outputted by SerialUSB device.

```
SerialUSB.print(“time : ”); SerialUSB.println(time);
```

Adding delay() to a blinking LED.

The CPU does nothing (remains in standby) for 1 second.

With void delay(unsigned long ms) set a value of 1000 for a delay of 1 second.

```
delay(1000);
```

for reference

1 sec = 1,000 millisecond , 1 millisecond = 1,000 microsecond

a short 500us delay.

To implement microsecond delays to the CPU implement

void delayMicroseconds(unsigned int us) function.

```
delayMicroseconds(500);
```

However, accuracy of the OpenCM9.04 CPU(STM32) is not guaranteed with regards to microsecond-type precisions.

⑥ Random numbers

Get 0~10 randomly.

long random(long max) or

long random(long min, long max).

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

there is no need to declared a minimum value; only maximum.

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

⑦ External interrupt

2. Have the LED turn on/off when pin-0 gets input signals.
3. Declare global variables and toggle flags in interrupt routines.

4. Attach interrupts with attachInterrupt().

```
volatile int state = LOW;

attachInterrupt(0, exInterrupt, CHANGE); // blink when there is a signal change
```

Implement exInterrupt() as void exInterrupt(void) type.

```
void exInterrupt(){
        if(state == HIGH)
                state = LOW;
        else
                state= HIGH;
}
loop(){
        digitalWrite(BOARD_LED_PIN, state);
}
```

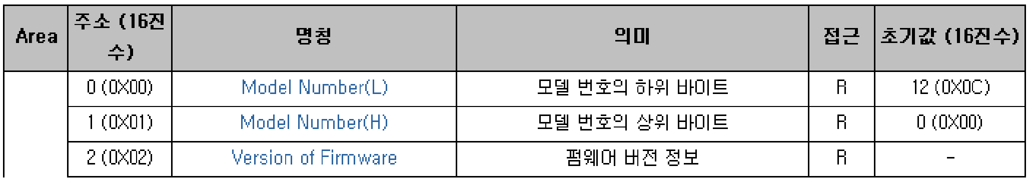
In loop() STATUS LED turns on/off based on state.

① Dynamixel

The following example is for ID=1 and baud rate set at 1Mbps [Dxl.begin(1) = 1M bps].

A. Read the AX-12Afirmware version.

The following shows the e-manual’s AX-12A control table model number and firmware addresses.



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	-

Read ID1's model number (address 0, LSBs portion) and firmware version (address 2).

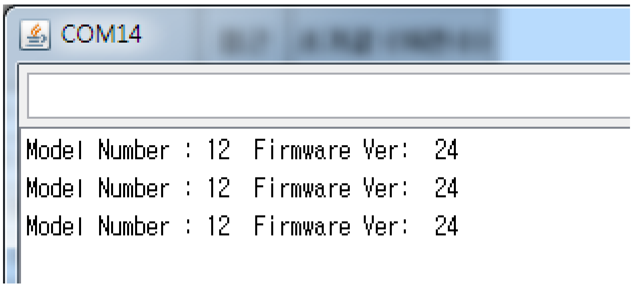
```
byte nModel = Dxl.readByte(1, 0); // 모델 번호를 읽고
```

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

the following lines are for output.

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

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

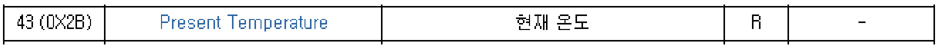


```
Model Number : 12  Firmware Ver:  24
Model Number : 12  Firmware Ver:  24
Model Number : 12  Firmware Ver:  24
```

<output>

B. Read ID 1's current temperature

The following shows the temperature address in the control table.



43 (0X2B)	Present Temperature	현재 온도	R	-

Use readByte() to get data.

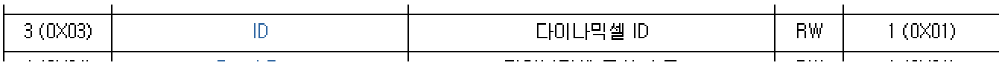
```
byte temp = Dxl.readByte(1, 43);
```

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

C. Set AX-12 to ID 2

Use readWrite() to set address #3

활용합니다.



3 (0X03)	ID	다이나믹셀 ID	RW	1 (0X01)

```
void setup(){
    Dxl.begin(1);
    delay(1000);   // add 1-sec delay.
    Dxl.writeByte(1, 3, 2);
    int CommStatus = Dxl.getResult();
    if( CommStatus == COMM_RXSUCCESS){
            SerialUSB.println("Changed Successfully!");
    }
    else{
            SerialUSB.println("Error");
    }
}
```

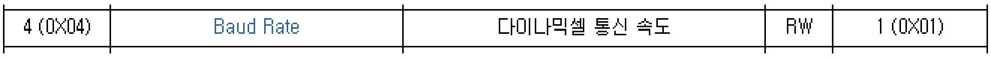
Set ID change in setup(). Always check for communications success.

ID 1 is now ID 2.

D. Change baudrate to 57600 bps

To change ID change address 4 (baud rate) via readWrite().

Refer to the index listing the baud rates; 57600 bps has a value of 34.



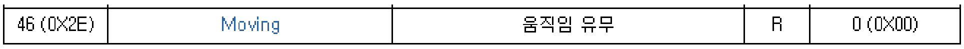
4 (0X04)	Baud Rate	다이나믹셀 통신 속도	RW	1 (0X01)

```
void setup(){
    Dxl.begin(1);
    delay(1000);   // add a 1-sec delay.
    Dxl.writeByte(1, 4, 34); // 34 = 57600 bps
    int CommStatus = Dxl.getResult();
    if( CommStatus == COMM_RXSUCCESS){
            SerialUSB.println(“Changed Successfully!”);
    }
    else{
            SerialUSB.println(“Error”);
    }
}
```

Once baud rate is changed initialize Dynamixel with Dxl.begin(34).

E. Move (rotate) ID 1

Address 46 (0x2E) of the control table deals with moving aspect.



46 (0X2E)	Moving	움직임 유무	R	0 (0X00)

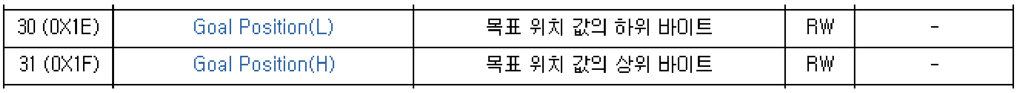
```
byte bMoving = Dxl.readByte(1, 46);
```

When ID1 moves bMoving returns 1; when not 0.

F. Move AX-12A to the 150-degree position

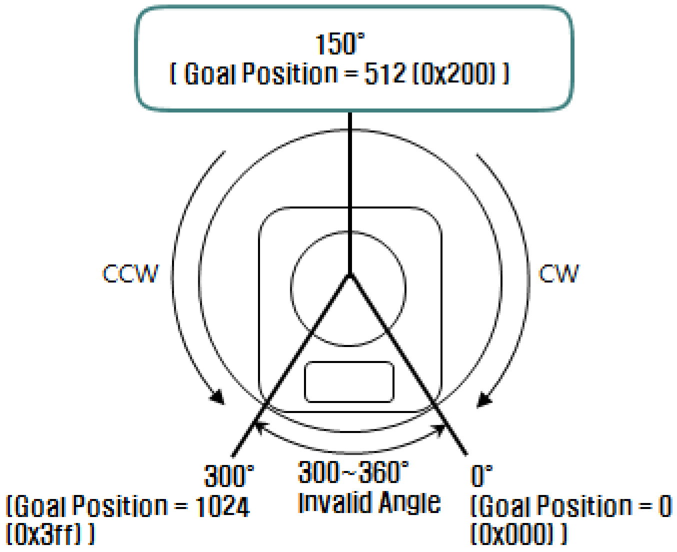
For a goal position of 150 its respective address value must be called (Goal Position L/H).

Goal position is expressed as a word (2 bytes). The table below shows both addresses needed to comprise the Goal Position word. Goal Position (L) (address 30). Use writeWord() to issue position command.



30 (0X1E)	Goal Position(L)	목표 위치 값의 하위 바이트	RW	-
31 (0X1F)	Goal Position(H)	목표 위치 값의 상위 바이트	RW	-

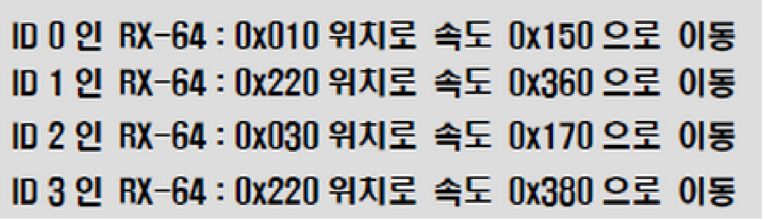
Please refer to the diagram below (also found in the e-manuals) with the position with respect to angles.



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

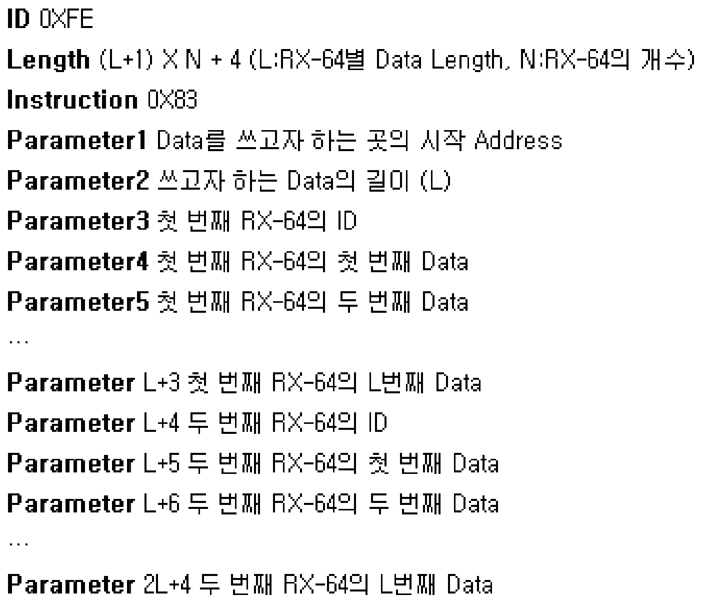
Use Dxl.getResult() to verify communications.

G. Set different speeds for several RX-64.

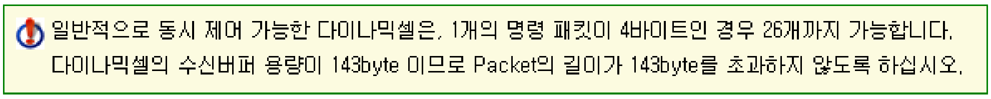


ID 0 인 RX-64 : 0x010 위치로 속도 0x150 으로 이동
ID 1 인 RX-64 : 0x220 위치로 속도 0x360 으로 이동
ID 2 인 RX-64 : 0x030 위치로 속도 0x170 으로 이동
ID 3 인 RX-64 : 0x220 위치로 속도 0x380 으로 이동

To have several Dynamixels move simultaneously issue the command syncWrite. syncWrite creates a packet then transmits it. Set the packet with setTxPacketXXXX().



ID 0XFE
Length (L+1) X 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

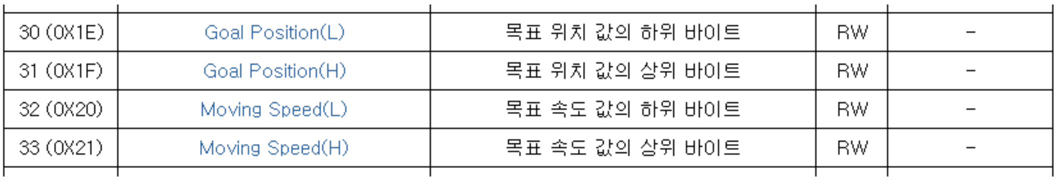


일반적으로 동시 제어 가능한 다이나믹셀은, 1개의 명령 패킷이 4바이트인 경우 26개까지 가능합니다.
다이나믹셀의 수신버퍼 용량이 143byte 이므로 Packet의 길이가 143byte를 초과하지 않도록 하십시오.

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

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

Set Goal Position and Moving Speed. Dxl.getLowByte() and Dxl.getHighByte() is explicit to high byte (MSBs).



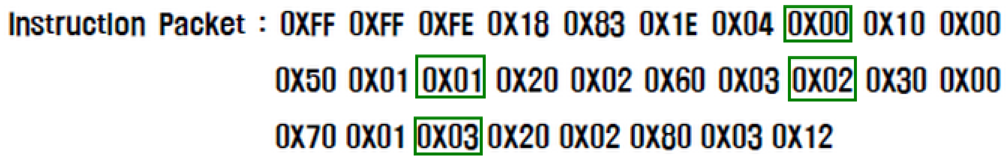
30 (0X1E)	Goal Position(L)	목표 위치 값의 하위 바이트	RW	-
31 (0X1F)	Goal Position(H)	목표 위치 값의 상위 바이트	RW	-
32 (0X20)	Moving Speed(L)	목표 속도 값의 하위 바이트	RW	-
33 (0X21)	Moving Speed(H)	목표 속도 값의 상위 바이트	RW	-

Declare position and velocity values.

```
word GoalPos[4]={0x010, 0x220, 0x030, 0x220};
word MovingSpd[4]={0x150, 0x360, 0x170, 0x380};
Dxl.setTxPacketParameter(0, 30);
Dxl.setTxPacketParameter(1, 4); // 4-bytes (2-words) data length
for( i=0; i < 4 ; i++){   // # of Dynamixels = 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);   // output current ID
    SerialUSB.print("  Goal Position : "); SerialUSB.print(GoalPos[i]);
    SerialUSB.print("  Moving Speed : "); SerialUSB.println(MovingSpd[i]);
}
Dxl.setTxPacketLength( (4+1)*4 + 4); // Packet length
```

Data length = 4, # of Dynamixels = 4

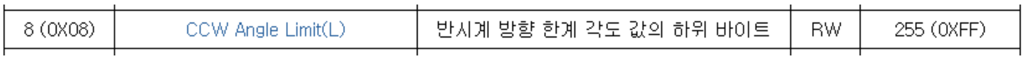
```
Dxl.txrxPacket(); // packet transmission command
int CommStatus = Dxl.getResult();
if( Dxl.getResult() == COMM_RXSUCCESS ){ // check for comm success
…
```



```
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. Limit range to 0~150degree range

Use CCW Angle Limit 0x3FF to set limit from 300 degrees to 150 degrees. Use writeByte() to send command.



8 (0X08)	CCW Angle Limit(L)	반시계 방향 한계 각도 값의 하위 바이트	RW	255 (0XFF)

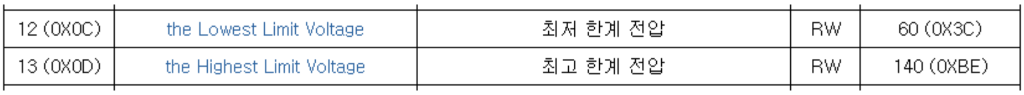
```
Dxl.writeByte(1, 8, 0x200);
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // check for comm success
```

```
...
```

I. Set input voltage between 10V ~ 17V

10V value is 100(0x64) and 17V is 170(0xAA). Use writeByte() to set commandThe address value from the control table is 12(0x0C) LSBs and 13(0x0D) MSBs.



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);
```

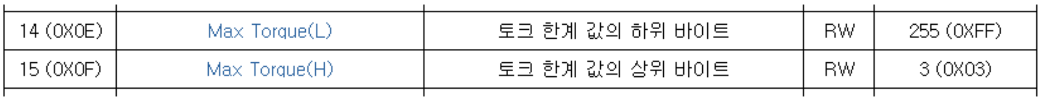
```
Dxl.writeByte(1, 13, 170);
```

```
if( Dxl.getResult() == COMM_RXSUCCESS ){ // check for comm succes
```

```
...
```

J. Set torque to 50% of max

Set a max Torque (0x3FF) to 50% (0x1FF). Max Torque's LSBs address is 14(0x0E). Use writeByte() to send command.



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 ){ // check for comm sucess
…
```

Turn power off then turn it back on to have new torque take place.

K. Set position of 180-degrees at 57RPM

Declare:

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

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

```
Dxl.writeWord(1, 32, 512);   // set velocity at 57 RPM
Dxl.writeWord(1, 30, 512);     // set position of 180-degrees
if( Dxl.getResult() == COMM_RXSUCCESS ){ // check for comm success
…
```

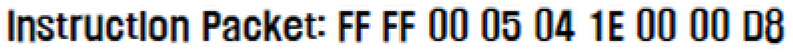
L. Set ID 0 position of 0 and ID 1 of 300 (both must operate simultaneously)

Use Syncwrite and setTxPacketXXX() to create a packet with INST_REG_WRITE and INST_ACTION. For reference 0 degrees is 0 (0x000) and 300 degrees is 1023 (0x3FF).

ID=0, Instruction = INST_REG_WRITE, Address = 30(0x1E), Data = 0

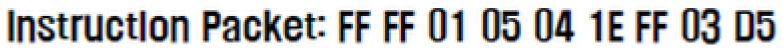
ID=1, Instruction = INST_REG_WRITE, Address = 30(0x1E), Data = 1023

```
Dxl.setTxPacketId(0); // set explicit control of ID 0
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);   //total data length = data length + 3

Dxl.txrxPacket();

if( Dxl.getResult() == COMM_RXSUCCESS ){ // check for comm success

...
```


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

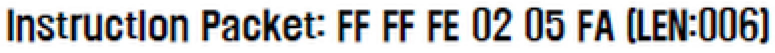
Second Dynamixel packet transmission

```
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 ){ // check for comm success

...
```


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

While ID0 and ID1 are pending INST_ACTION packet is transmitted to run instructions.

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

Dxl.setTxPacketLength(2);

Dxl.txrxPacket();

if( Dxl.getResult() == COMM_RXSUCCESS ){ // check for comm success

…
```



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

Check for communications success every time a packet is sent.