Dynamixel & Dynamixel SDK

2017.03.16

Contents

Dynamixel

Dynamixel

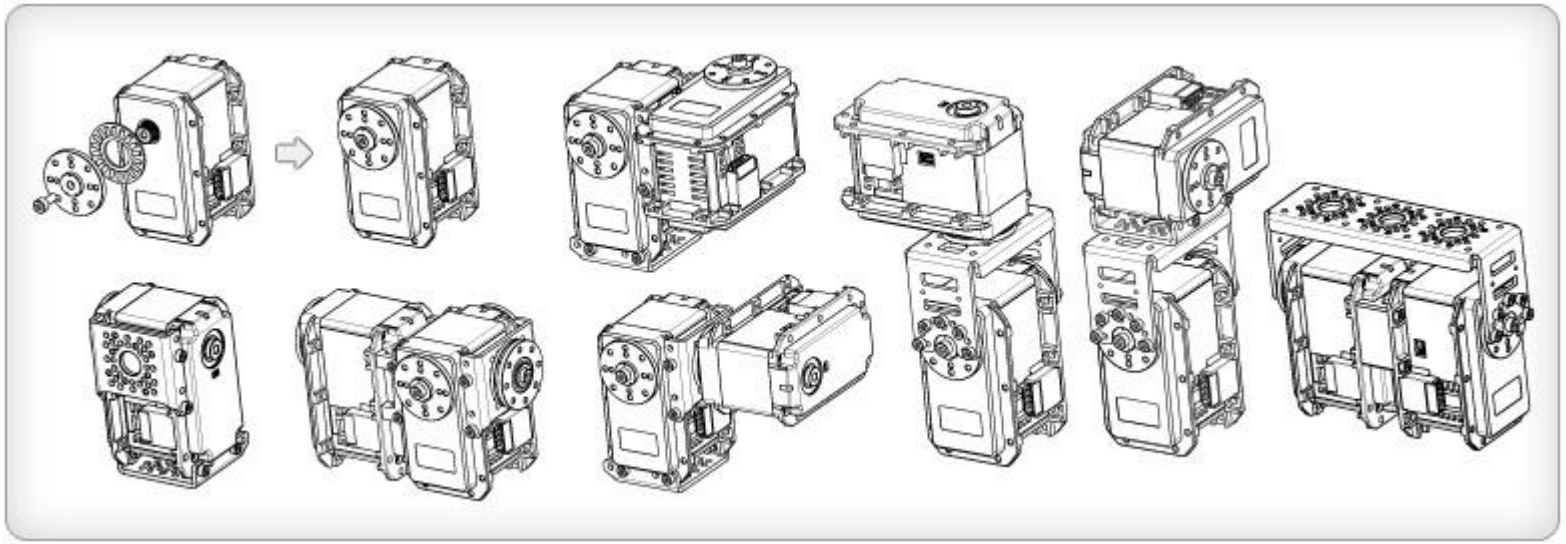
All in One Actuator to build Multi-Jointed Robots
for DIY, Educational or Research

All-in-One Modular Design

A module actuator which incorporates all the functions required for robot joints

All-around Assembling Structure

You can build diverse forms of robots by using various optional frames.



Controlled via Network

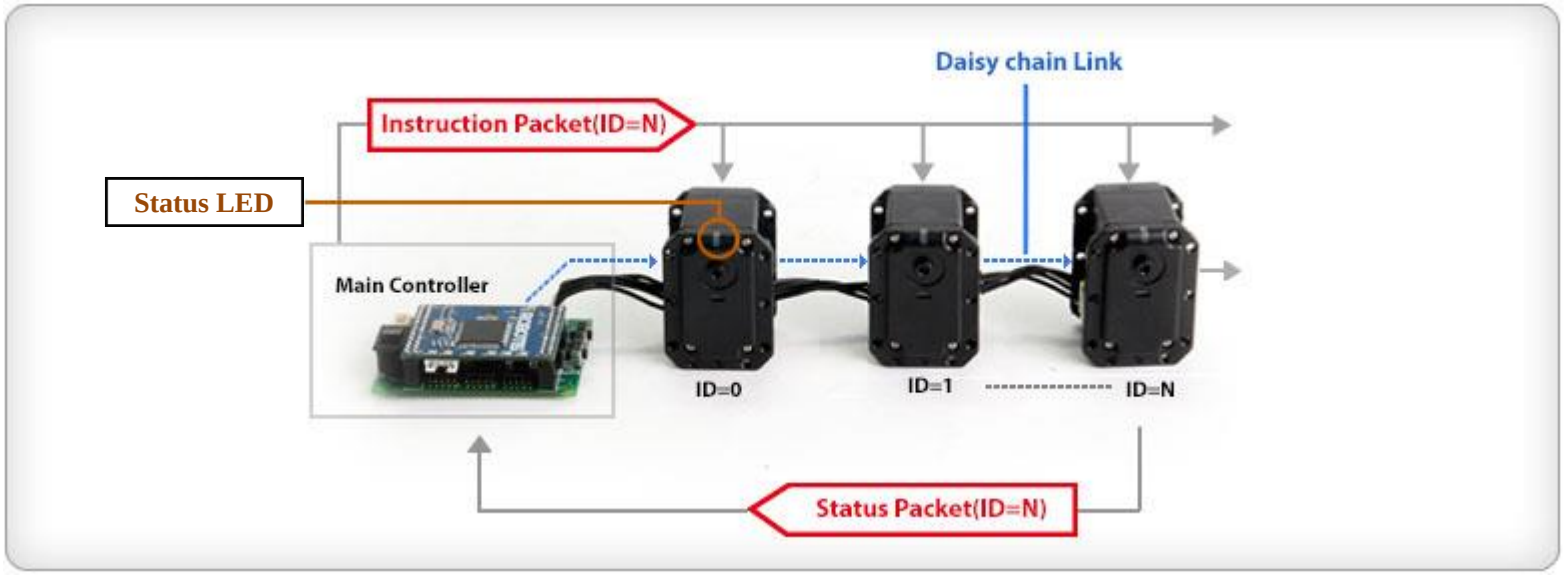
Each Dynamixels have a unique ID and can be controlled by packet communication on a BUS. They support network of RS232(TTL Level), RS485 etc. according to model.

Simplified connection structure

Wiring becomes simpler thanks to the Daisy Chain link style.

Status Display LED

LED and Shut-down (Torque-off) functions can be set regarding high temperature, overload, overvoltage, and low voltage.



Compliance Setting

Compliance margin and slope control functions available. (Depends on model)

PID Gain Control

Uses PID(Proportional, Integral, Derivative) Gains for the control. (Depends on model)

Torque Setting

Torque can be set from zero for free run state to maximum depends on the purpose.

Low-electric current/high voltage drive

Dynamixel’s high efficiency is achieved by high voltage system design, and it improves the stability of Dynamixel robot system because of the low current consumption.

Control Table

The Control Table is a structure of data implemented in the Dynamixel. Users can read a specific data from predefined address to get status of the Dynamixel with READ Instruction Packets, and modify data as well to control Dynamixels with WRITE Instruction packets.

EEPROM and RAM

Data in the RAM area is reset to the initial value whenever the power is on (Volatile). On the other hand, data written in the EEPROM area are kept even if the power is off (Non-Volatile). EEPROM area can be written only when the “Torque Enable” value is 0(Torque Off).

Address

The address is a unique value when accessing a specific data in the control table with Instruction Packet. It represents the location of data. To read from or write to the control table, user should assign the correct address in the Instruction Packet.

Access

Dynamixel has two different access properties. ‘RW’ property stands for Read-and-Write access permission, which is generally used for measuring and monitoring purpose. ‘R’ property stands for Read-Only access permission, which is generally used for acquiring Dynamixels status.

Dynamixel

- **Dynamixel Pro Control Table (example) :**

  http://support.robotis.com/en/techsupport_eng.htm#product/actuator/dynamixel_pro/dynamixelpro/control_table.htm

- **Dynamixel Selection Guide :**

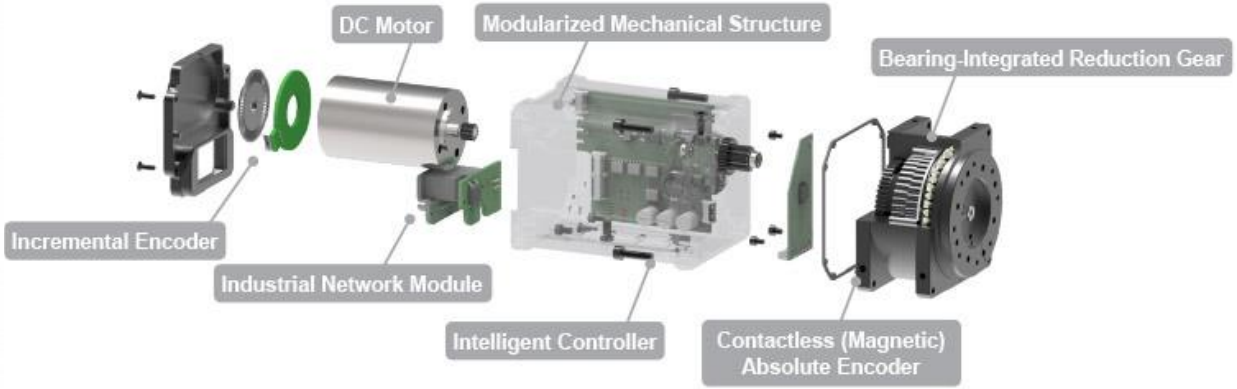
  http://en.robotis.com/index/product.php?cate_code=101310

- **Dynamixel Comparison :**

  http://support.robotis.com/en/techsupport_eng.htm#product/actuator/dynamixel_comparison.htm

Dynamixel Pro

DYNAMIXEL PRO



H 5 4 — 1 0 0 — S 5 0 0 — R

Performance
- H : High Performance
- M : Mid-Level Performance
- L : Low Cost

Dimension(mm)
- 42 : 42x42mm
- 54 : 54x54mm

Motor output(W)
- 10 : 10W
- 20 : 20W
- 30 : 30W
- 40 : 40W
- 50 : 50W
- 60 : 60W
- 100 : 100W
- 200 : 200W

Gear Type
- S : Straight

Gear Ratio
- 250 : 250 :1
- 260 : 260 :1
- 290 : 290 :1

Interface
- R : RS-485

Special features of Dynamixel Pro

- **Strong and durable reduction cycloid gearing** : Dynamixel Pro implements removable reduction cycloid gears. Small and lightweight cycloids allow a high-ratio reduction gear device. Also cycloid gears are more resistant to vibrations and impacts resulting in minimal backlash compare to conventional spur gears.
- **Variety of models** : Dynamixel Pro has various range of products to meet the customer's needs by combining motor, gear type, reduction ratio, communication protocol. Users can choose the most suitable Dynamixel Pro model necessary for the robot.
- **Variety of control algorithms** : Dynamixel Pro adopts position, velocity, and current control algorithms. Users can select these control algorithms in any combination and properly tune the robot. Dynamixel Pro provides a graph illustrating the relationship between current and torque. This feature is useful when torque control is required.
- **Precision control** : with a maximum of 502,000 units per revolution, users can control 0.0007 degrees allowing high-precision control.

Key Specifications

	Output	Dimension(mm)	Weight	Resolution	Motor
H54-200-S500-R	200W	54 x 54 x 126	855g	501,923	BLDC
H54-100-S500-R	100W	54 x 54 x 108	732g	501,923	BLDC
H42-20-S300-R	20W	42 x 42 x 84	340g	303,750	Coreless
M54-60-S250-R	60W	54 x 54 x 126	853g	251,417	BLDC
M54-40-S250-R	40W	54 x 54 x 108	710g	251,417	BLDC
M42-10-S260-R	10W	42 x 42 x 72	269g	263,187	Coreless
L54-50-S290-R	50W	54 x 54 x 108	662g	207,692	BLDC
L54-50-S500-R	50W	54 x 54 x 108	656g	361,384	BLDC
L54-30-S400-R	30W	54 x 54 x 108	612g	288,395	BLDC
L54-30-S500-R	30W	54 x 54 x 108	591g	316,384	BLDC
L42-10-S300-R	10W	42 x 42 x 72	257g	4,096	Coreless

RoboPlus Manager 2.0

RoboPlus Manager 2.0 light

RoboPlus Manager 2.0 manages Controllers and Dynamixels that comprise the robot. By connecting to the component, users can update the product to the latest firmware version and test the control table under the GUI environment.

Functions provided in RoboPlus Manager 1.0 and Dynamixel Wizard 1.0 have been integrated in RoboPlus Manager 2.0.

RoboPlus Manager 2.0 light

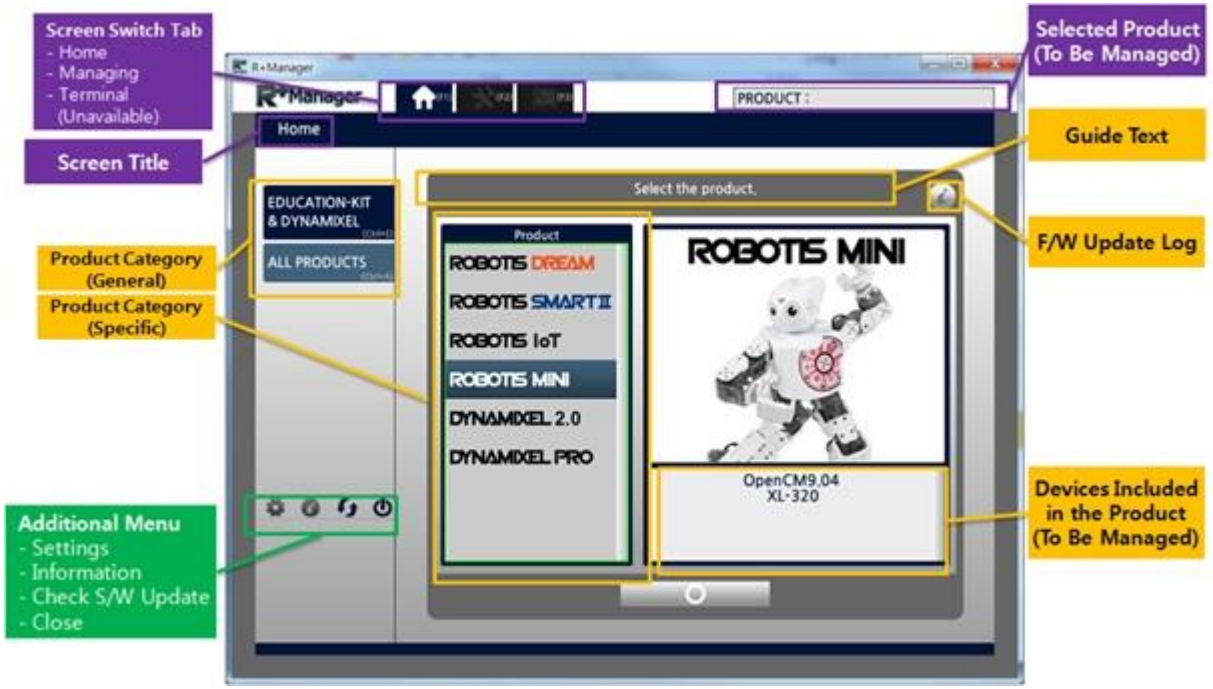
Some products are not supported on v2.0.0.

In order to use unsupported products, please use RoboPlus Manager 1.0 and Dynamixel Wizard 1.0.

- ROBOTIS DREAM / ROBOTIS SMART / ROBOTIS IoT / ROBOTIS MINI
- DYNAMIXEL Series (MX Series, X Series, PRO Series)

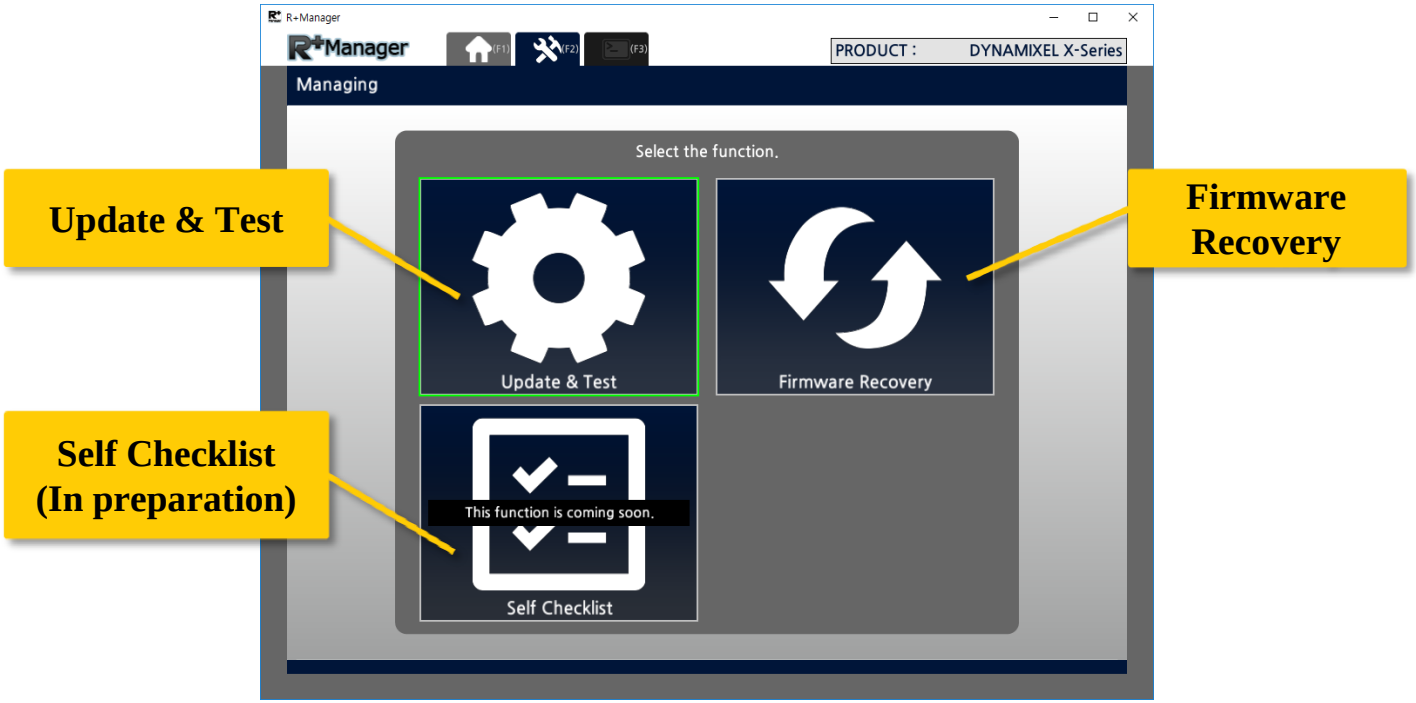
Home Screen

- Product Categories
- Supported products in the selected product series.
- Additional Menu



Managing Screen

- Update & Test
- Firmware Recovery
- Additional Menu

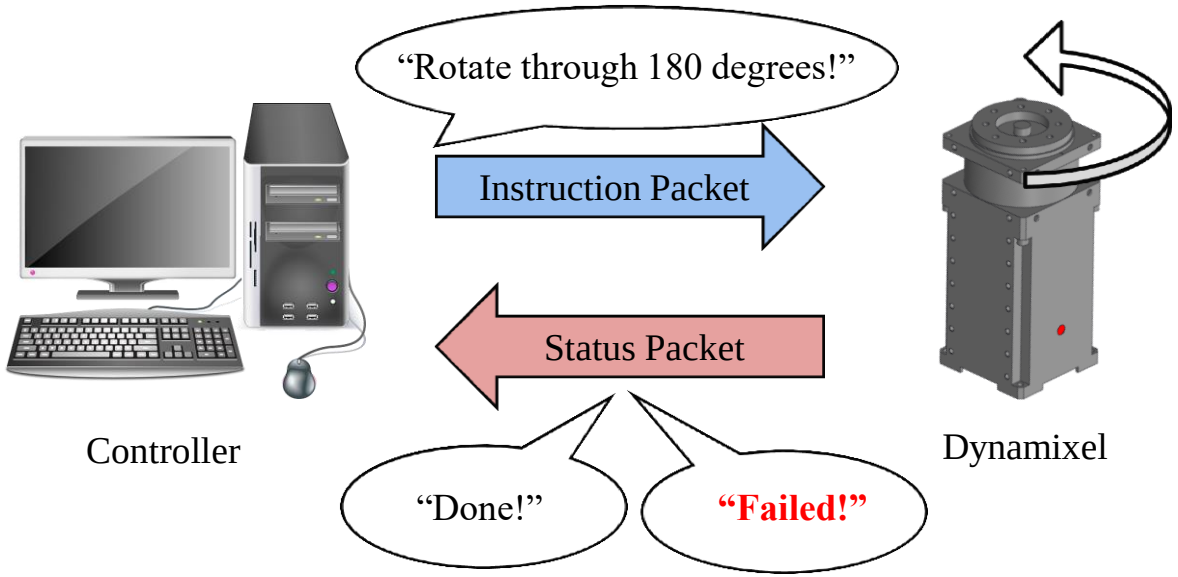


- **Download**
  - R+ Manager 2.0 light (v2.0.1) Update Release
- **Practice**
  - Updating the Firmware
  - Recovering the Firmware
  - Testing the Control Table

Dynamixel SDK

1. Protocol

Dynamixel Protocol: a communication rule needed for Dynamixel control



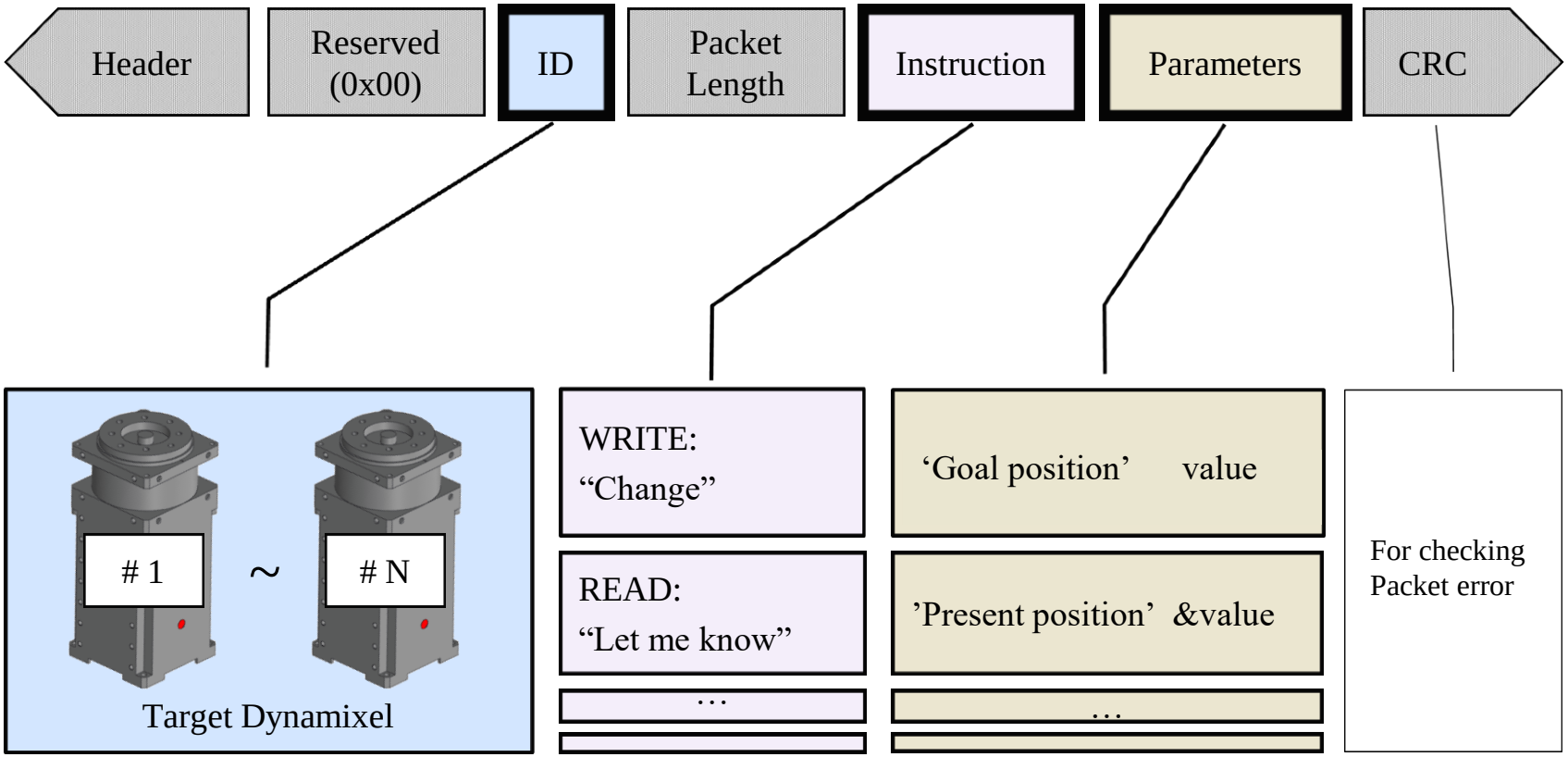
- **Protocol 1.0 : Dynamixel AX series / DX series / RX series / MX series**
- **Protocol 2.0(*) : Dynamixel PRO(**), XL, XM, XH**

(*) Purpose: for better accuracy of signal transmission/reception

(**) Dynamixel PRO: Dynamixel series used in Manipulator and THORMANG line up

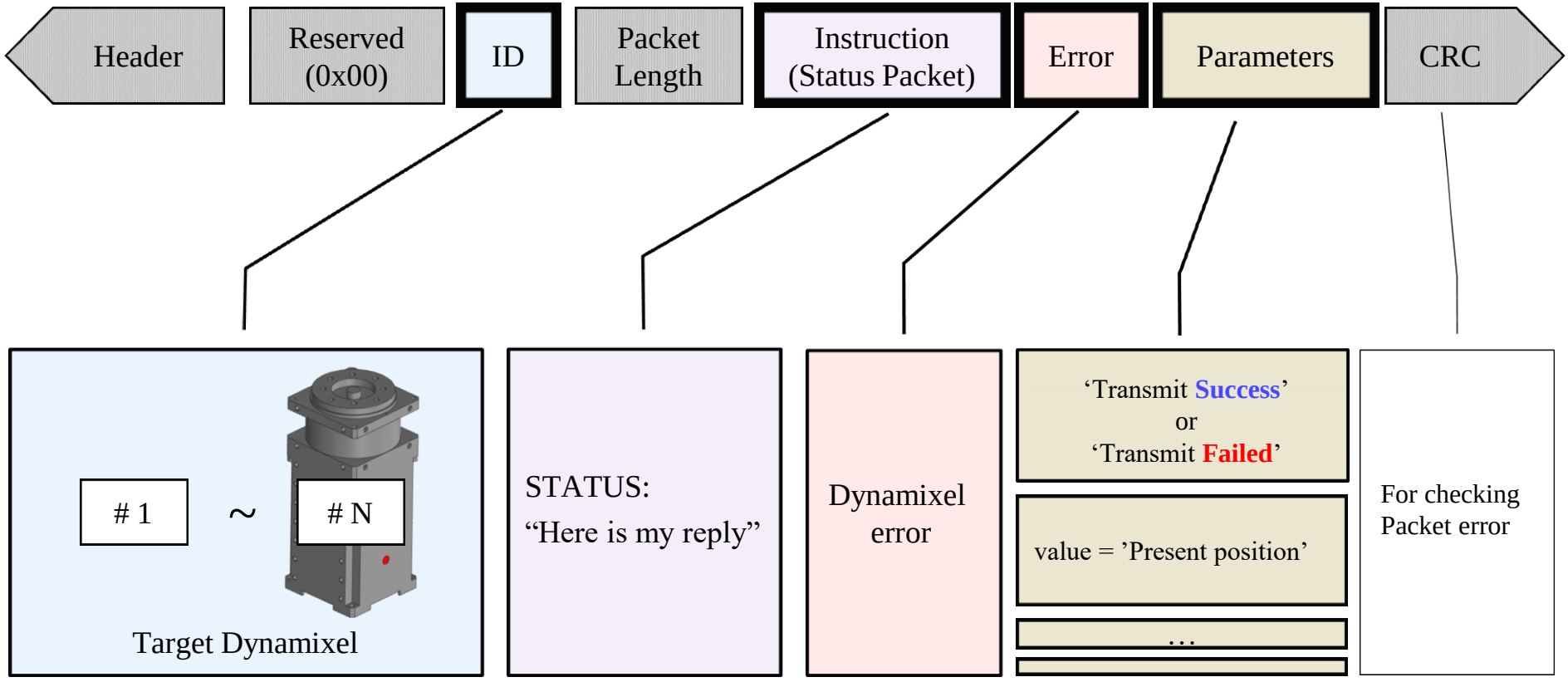
2. Packet Architecture of Protocol 2.0

Instruction Packet



2. Packet Architecture of Protocol 2.0

Status Packet



3. Instruction / Status Packet (Protocol 2.0)

http://support.robotis.com/en/product/actuator/dynamixel_pro/communication/instruction_status_packet.htm

Instruction Packet

Header			Reserved	Packet ID	Packet Length		Instruction	Parameter			16bit CRC	
0xFF	0xFF	0xFD	0x00	ID	LEN_L	LEN_H	Instruction	Parameter1	...	ParameterN	CRC_L	CRC_H

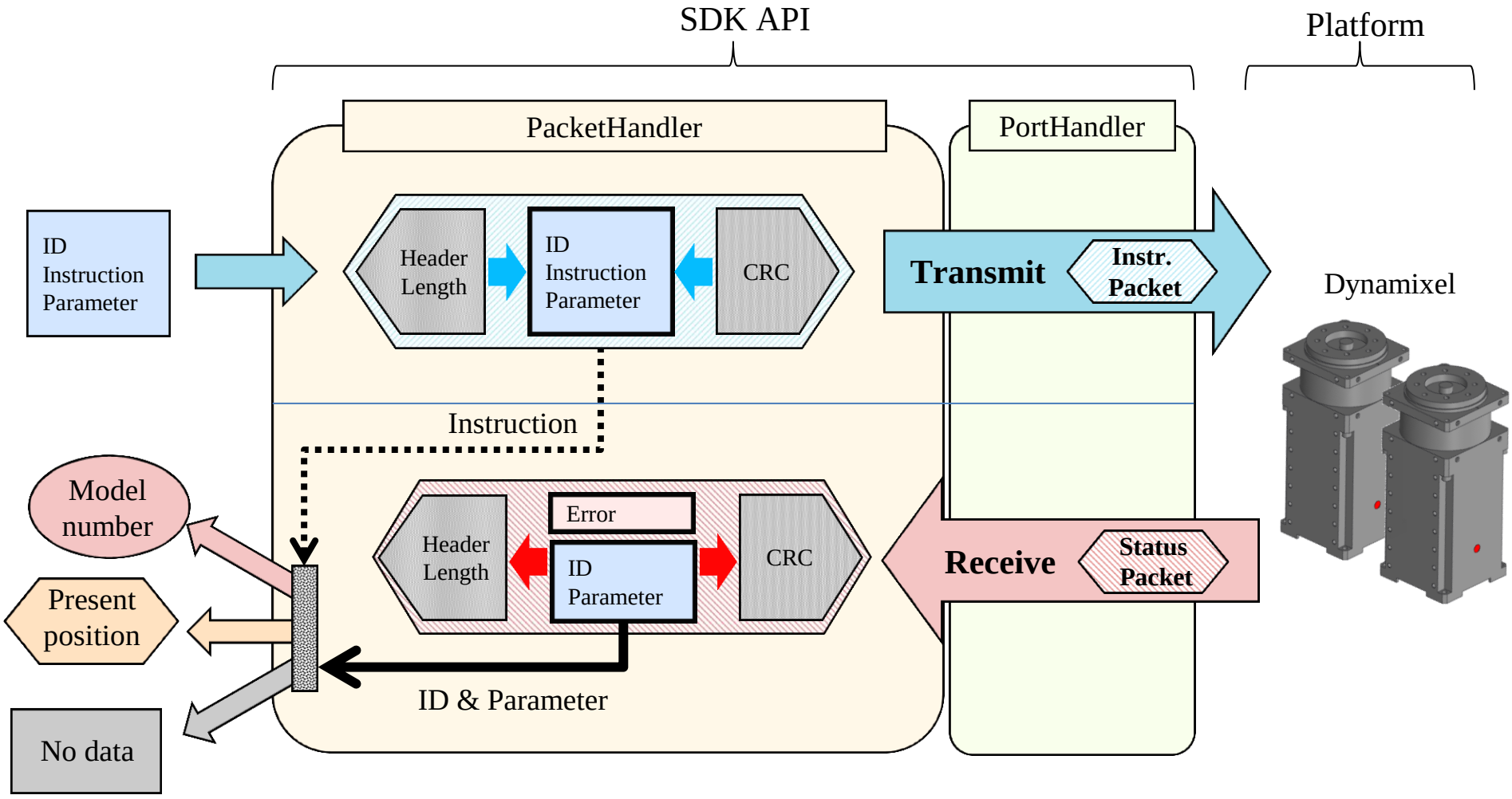
Status Packet

Header			Reserved	Packet ID	Packet Length		Instruction	Parameter			16bit CRC	
0xFF	0xFF	0xFD	0x00	ID	LEN_L	LEN_H	0x55	Param1	...	ParamN	CRC_L	CRC_H

4. Calculating CRC

http://support.robotis.com/en/product/actuator/dynamixel_pro/communication/crc.htm

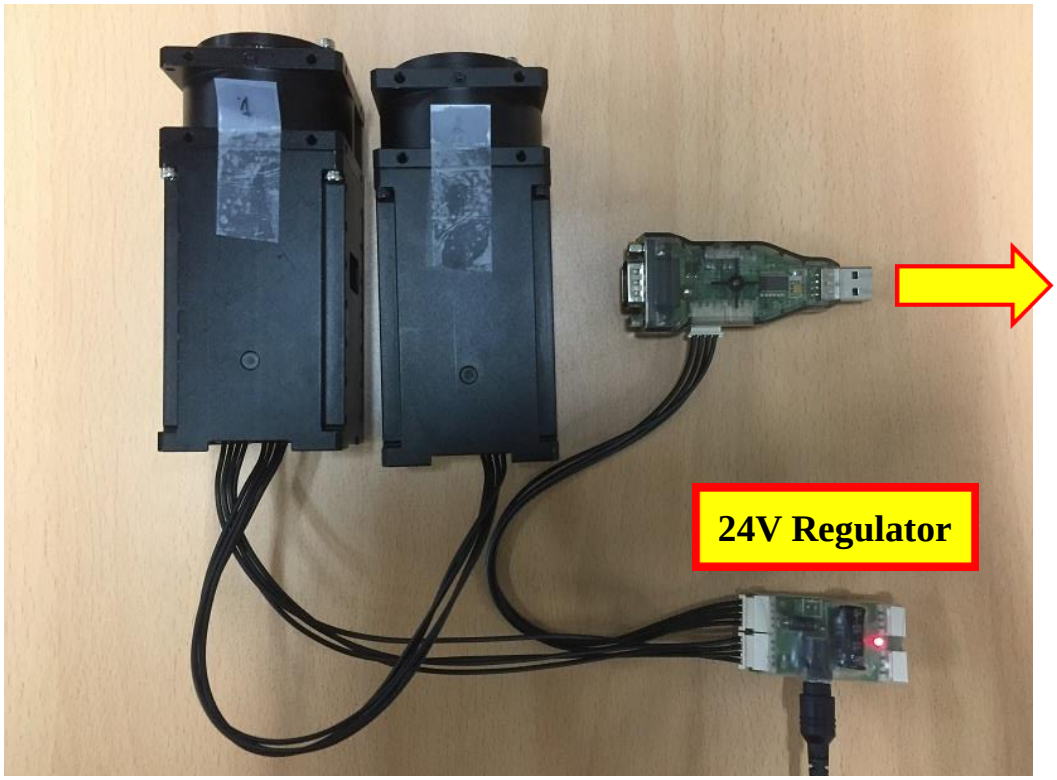
5. System Configuration Example



Practice

1. Environment Setup - H/W Configuration

A. Dynamixel setting



PC

1. Environment Setup - H/W Configuration(continue)

B. Use **R+ Manager 2.0** for Windows system.

C. Use **dxl_monitor** in the example directory for Linux system.

D. Set 1st Dynamixel

   i. ID = 1

   ii. Baudrate = 1,000,000 bps

E. Set 2nd Dynamixel

   i. ID = 2

   ii. Baudrate = 1,000,000 bps

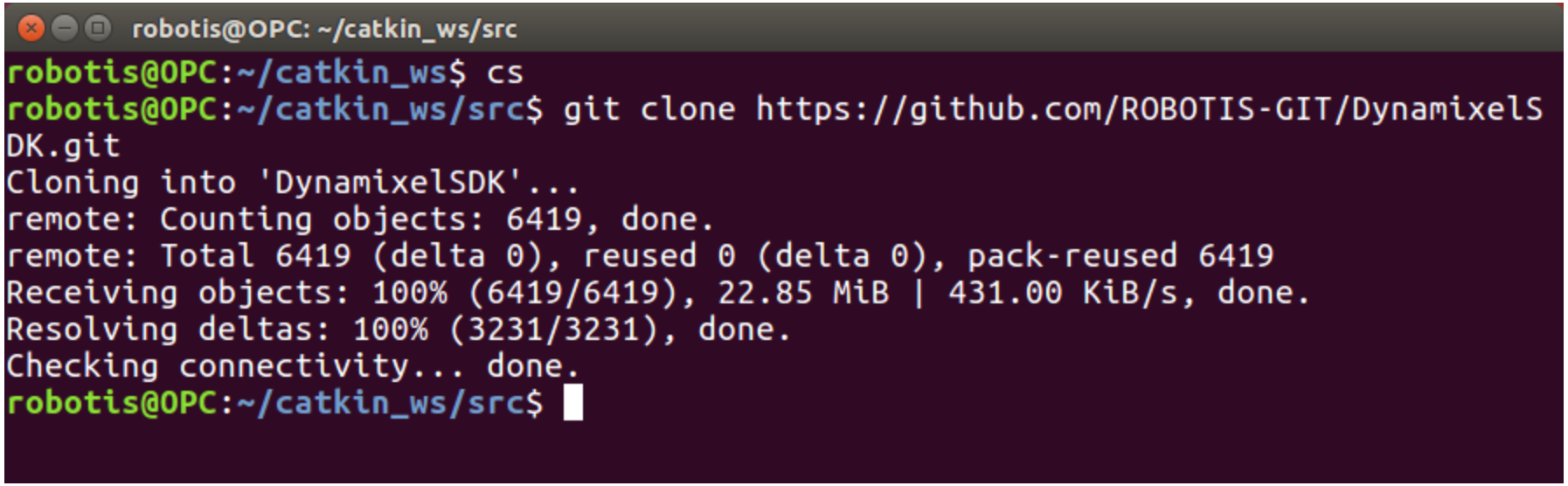
F. (Optional) Change Minimum Voltage Limit to 110 when it uses 12V SMPS

2. Environment Setup - S/W Configuration

A. Source Download

i. https://github.com/ROBOTIS-GIT/DynamixelSDK/releases

ii.
```
$ cd ~/catkin_ws/src
$ sudo apt install git
$ git clone https://github.com/ROBOTIS-GIT/DynamixelSDK.git
```



2. Environment Setup - S/W Configuration(continue)

B. Compiler

i. C : GNU gcc

ii. C++ : GNU g++

C. Build Tool

i. Build-essential → make

3. Environment Setup - S/W Configuration(continue)

D. Library build

i. `$ cd ./DynamixelSDK/c++/build/linux64`

ii. To build library file,

`$ make`

```
robotis@OPC: ~/catkin_ws/src/DynamixelSDK/c++/build/linux64
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/build/linux64$ make
mkdir -p ./.objects/
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -c -I../../include -m64 -fPIC -g -c ../.
./src/dynamixel_sdk/group_bulk_read.cpp -o .objects/group_bulk_read.o
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -c -I../../include -m64 -fPIC -g -c ../.
./src/dynamixel_sdk/group_bulk_write.cpp -o .objects/group_bulk_write.o
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -c -I../../include -m64 -fPIC -g -c ../.
./src/dynamixel_sdk/group_sync_read.cpp -o .objects/group_sync_read.o
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -c -I../../include -m64 -fPIC -g -c ../.
./src/dynamixel_sdk/group_sync_write.cpp -o .objects/group_sync_write.o
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -c -I../../include -m64 -fPIC -g -c ../.
./src/dynamixel_sdk/packet_handler.cpp -o .objects/packet_handler.o
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -c -I../../include -m64 -fPIC -g -c ../.
./src/dynamixel_sdk/port_handler.cpp -o .objects/port_handler.o
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -c -I../../include -m64 -fPIC -g -c ../.
./src/dynamixel_sdk/protocol1_packet_handler.cpp -o .objects/protocol1_packet_ha
ndler.o
```

3. Environment Setup - S/W Configuration(continue)

D. Library build

iii. To make library file and copy it to the root directory,

```
$ sudo make install
```

```
robotis@OPC: ~/catkin_ws/src/DynamixelSDK/c++/build/linux64
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/build/linux64$ sudo make install
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
mkdir -p ./.objects/
g++ -shared -fPIC -m64 -o ./libdxl_x64_cpp.so ./.objects/group_bulk_read.o ./.ob
jects/group_bulk_write.o ./.objects/group_sync_read.o ./.objects/group_sync_writ
e.o ./.objects/packet_handler.o ./.objects/port_handler.o ./.objects/protocol1_p
acket_handler.o ./.objects/protocol2_packet_handler.o ./.objects/port_handler_li
nux.o -lrt
cp "./libdxl_x64_cpp.so" "/usr/local/lib/libdxl_x64_cpp.so"
ln -s "/usr/local/lib/libdxl_x64_cpp.so" "/usr/local/lib/libdxl_x64_cpp.so.2"
ln -s "/usr/local/lib/libdxl_x64_cpp.so" "/usr/local/lib/libdxl_x64_cpp.so.2.0"
ln -s "/usr/local/lib/libdxl_x64_cpp.so" "/usr/local/lib/libdxl_x64_cpp.so.2.0.0
"
cp -r ../../include/* /usr/local/include/
ldconfig
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/build/linux64$
```

3. Environment Setup - S/W Configuration(continue)

D. Library build(continue)

iv. If you need to update the library file in the root directory,
```
$ sudo make reinstall
```

v. You can see built file in
[DynamixelSDK directory]/c++/build/linux64/libdxl_x64_cpp.so

Example Lists

1. Examples for Protocol 2.0 – Dynamixel PRO

Instructions	Examples	
Ping	Ping	BroadcastPing
Read	ReadWrite	MultiPort
Write		
SyncRead	Sync ReadWrite	Indirect Address
SyncWrite		
BulkRead	Bulk ReadWrite	
BulkWrite		
Reboot	Reboot	
FactoryReset	FactoryReset	

Examples	Contents
Ping	Get DXL model number
BroadcastPing	Scan connected DXL
ReadWrite	Position Control
MultiPort	Use two ports
SyncReadWrite	Access multi DXLs
BulkReadWrite	Access multi DXL, ADDR
IndirectAddress	Use Indirect Address
Reboot	Reboot DXL
Factory	Reset all DXL settings

2. ReadWrite

```
$ cd ~catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write
$ gedit read_write.cpp
```

Change the source code according to below image and save the file.

```
#include "dynamixel_sdk.h"                                  // Uses Dynamixel SDK library

// Control table address
//#define ADDR_PRO_TORQUE_ENABLE          562                 // Control table address is different in Dynamixel model
//#define ADDR_PRO_GOAL_POSITION          596
//#define ADDR_PRO_PRESENT_POSITION       611

//**** Control Table Address for Dynamixel XM Series ******
#define ADDR_PRO_TORQUE_ENABLE          64                 // Control table address is different in Dynamixel model
#define ADDR_PRO_GOAL_POSITION          116
#define ADDR_PRO_PRESENT_POSITION       132
//**** Control Table Address for Dynamixel XM Series ******

// Protocol version
#define PROTOCOL_VERSION                2.0                 // See which protocol version is used in the Dynamixel

// Default setting
#define DXL_ID                          1                   // Dynamixel ID: 1
#define BAUDRATE                        1000000
#define DEVICENAME                      "/dev/ttyUSB0"      // Check which port is being used on your controller
                                                            // ex) Windows: "COM1"   Linux: "/dev/ttyUSB0"

#define TORQUE_ENABLE                   1                   // Value for enabling the torque
#define TORQUE_DISABLE                  0                   // Value for disabling the torque
//#define DXL_MINIMUM_POSITION_VALUE      -150000             // Dynamixel will rotate between this value
//#define DXL_MAXIMUM_POSITION_VALUE      150000              // and this value (note that the Dynamixel would not move when
the position value is out of movable range. Check e-manual about the range of the Dynamixel you use.)

//**** MIN/MAX Position Values for Dynamixel XM Series ******
#define DXL_MINIMUM_POSITION_VALUE      0                   // Dynamixel will rotate between this value
#define DXL_MAXIMUM_POSITION_VALUE      4095                // and this value (note that the Dynamixel would not move when the
position value is out of movable range. Check e-manual about the range of the Dynamixel you use.)
//**** MIN/MAX Position Values for Dynamixel XM Series ******

#define DXL_MOVING_STATUS_THRESHOLD     20                  // Dynamixel moving status threshold
```

Example Lists

2. ReadWrite

```
$ cd  linux64
$ make
```

If there is no error in the code, the executable file will be created.

```
robotis@OPC: ~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64

robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write$ cd linux64
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ make
mkdir -p .objects/
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -I../../../include -m64 -g -c ../read_write.cpp -o .objects/read_write.o
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -I../../../include -m64 -g  .objects/read_write.o -o read_write -ldxl_x64_cpp -lrt
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ ./read_write
[PortHandlerLinux::SetupPort] Error opening serial port!
Failed to open the port!
Press any key to terminate...
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ sudo chmod a+rw /dev/ttyUSB0
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
sudo: 3 incorrect password attempts
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ sudo chmod a+rw /dev/ttyUSB0
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$
```

Example Lists

2. ReadWrite

```
$ ./read_write
```

If USB port doesn't have proper permission, you'll see the following failure.

```
robotis@OPC: ~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64

robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write$ cd linux64
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ make
mkdir -p .objects/
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -I../../../include -m64 -g -c ../read_write.cpp -o .objects/read_write.o
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -I../../../include -m64 -g  .objects/read_write.o -o read_write -ldxl_x64_cpp -lrt
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ ./read_write
[PortHandlerLinux::SetupPort] Error opening serial port!
Failed to open the port!
Press any key to terminate...
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ sudo chmod a+rw /dev/ttyUSB0
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
sudo: 3 incorrect password attempts
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ sudo chmod a+rw /dev/ttyUSB0
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$
```

Example Lists

2. ReadWrite

```
$ sudo chmod a+rw /dev/ttyUSB0
```

The command will assign read and write permission on the USB port.

In the last keyword ttyUSBx, x might differ by your USB connection status.

Check which port of your USB to serial device is connected.

```
$ ls /dev/ttyUSB*
```

The USB permission has to be reassigned when the system reboots.

The following command will permanently allow you to have access on the USB by adding dialout group to the user account name you have chosen.

```
$ sudo usermod –aG dialout [user account]
```

```
Press any key to terminate
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ sudo chmod a+rw /dev/ttyUSB0
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
sudo: 3 incorrect password attempts
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ sudo chmod a+rw /dev/ttyUSB0
[sudo] password for robotis:
Sorry, try again.
[sudo] password for robotis:
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$
```

2. ReadWrite

```
$ ./read_write
```

Once you have given a permission to the USB to serial device, let's try again.

```
robotis@OPC: ~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/protocol2.0/read_write/linux64$ ./read_write
Succeeded to open the port!
Succeeded to change the baudrate!
Dynamixel has been successfully connected
Press any key to continue! (or press ESC to quit!)
[ID:001] GoalPos:000  PresPos:4094
[ID:001] GoalPos:000  PresPos:4093
[ID:001] GoalPos:000  PresPos:4092
[ID:001] GoalPos:000  PresPos:4088
[ID:001] GoalPos:000  PresPos:4084
[ID:001] GoalPos:000  PresPos:4080
[ID:001] GoalPos:000  PresPos:4073
[ID:001] GoalPos:000  PresPos:4068
[ID:001] GoalPos:000  PresPos:4063
[ID:001] GoalPos:000  PresPos:4057
[ID:001] GoalPos:000  PresPos:4051
[ID:001] GoalPos:000  PresPos:4045
[ID:001] GoalPos:000  PresPos:4039
[ID:001] GoalPos:000  PresPos:4032
[ID:001] GoalPos:000  PresPos:4026
[ID:001] GoalPos:000  PresPos:4019
[ID:001] GoalPos:000  PresPos:4013
[ID:001] GoalPos:000  PresPos:4007
[ID:001] GoalPos:000  PresPos:4001
```

Example Lists

3. dxl_monitor

```
$ cd ~catkin_ws/src/DynamixelSDK/c++/example/dxl_monitor/linux64
$ make
$ ./dxl_monitor
```

```
robotis@OPC: ~/catkin_ws/src/DynamixelSDK/c++/example/dxl_monitor/linux64
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/dxl_monitor/linux64$ make
mkdir -p .objects/
g++ -O2 -O3 -DLINUX -D_GNU_SOURCE -Wall -I../../include -m64 -g  .objects/dxl_monitor.o -o dxl_monitor -ldxl_x64_cpp -lrt
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/dxl_monitor/linux64$ ls
dxl_monitor  Makefile
robotis@OPC:~/catkin_ws/src/DynamixelSDK/c++/example/dxl_monitor/linux64$ ./dxl_monitor

********************************************************************
*                          DXL Monitor                             *
********************************************************************

Succeeded to open the port!

 - Device Name : /dev/ttyUSB0
 - Baudrate    : 1000000

[CMD]
```

Example Lists

3. dxl_monitor

help Command : display all available commands

```
robotis@OPC: ~/catkin_ws/src/DynamixelSDK/c++/example/dxl_monitor/linux64

[CMD] help

                    .---------------------------.
                    |  DXL Monitor Command List  |
                    '---------------------------'
 ========================= Common Commands =========================

 help|h|?                     :Displays help information
 baud [BAUD_RATE]             :Changes baudrate to [BAUD_RATE]
                               ex) baud 2400 (2400 bps)
                               ex) baud 1000000 (1 Mbps)
 exit                         :Exit this program
 scan                         :Outputs the current status of all Dynamixels
 ping [ID] [ID] ...           :Outputs the current status of [ID]s
 bp                           :Broadcast ping (Dynamixel Protocol 2.0 only)

 ==================== Commands for Dynamixel Protocol 1.0 ====================

 wrb1|w1 [ID] [ADDR] [VALUE] :Write byte [VALUE] to [ADDR] of [ID]
 wrw1 [ID] [ADDR] [VALUE]    :Write word [VALUE] to [ADDR] of [ID]
 rdb1 [ID] [ADDR]            :Read byte value from [ADDR] of [ID]
 rdw1 [ID] [ADDR]            :Read word value from [ADDR] of [ID]
 r1 [ID] [ADDR] [LENGTH]     :Dumps the control table of [ID]
                               ([LENGTH] bytes from [ADDR])
 reset1|rst1 [ID]            :Factory reset the Dynamixel of [ID]

 ==================== Commands for Dynamixel Protocol 2.0 ====================

 wrb2|w2 [ID] [ADDR] [VALUE] :Write byte [VALUE] to [ADDR] of [ID]
 wrw2 [ID] [ADDR] [VALUE]    :Write word [VALUE] to [ADDR] of [ID]
 wrd2 [ID] [ADDR] [VALUE]    :Write dword [VALUE] to [ADDR] of [ID]
 rdb2 [ID] [ADDR]            :Read byte value from [ADDR] of [ID]
 rdw2 [ID] [ADDR]            :Read word value from [ADDR] of [ID]
 rdd2 [ID] [ADDR]            :Read dword value from [ADDR] of [ID]
 r2 [ID] [ADDR] [LENGTH]     :Dumps the control table of [ID]
                               ([LENGTH] bytes from [ADDR])
 reboot2|rbt2 [ID]           :reboot the Dynamixel of [ID]
 reset2|rst2 [ID] [OPTION]   :Factory reset the Dynamixel of [ID]
                               OPTION: 255(All), 1(Except ID), 2(Except ID&Baud)

[CMD]
```

Example Lists

3. dxl_monitor

ex) *scan* Command : scan for connected devices

```
robotis@OPC: ~/catkin_ws/src/DynamixelSDK/c++/example/dxl_monitor/linux64

 ==================== Commands for Dynamixel Protocol 1.0 ====================

 wrb1|w1 [ID] [ADDR] [VALUE] :Write byte [VALUE] to [ADDR] of [ID]
 wrw1 [ID] [ADDR] [VALUE]    :Write word [VALUE] to [ADDR] of [ID]
 rdb1 [ID] [ADDR]            :Read byte value from [ADDR] of [ID]
 rdw1 [ID] [ADDR]            :Read word value from [ADDR] of [ID]
 r1 [ID] [ADDR] [LENGTH]     :Dumps the control table of [ID]
                               ([LENGTH] bytes from [ADDR])
 reset1|rst1 [ID]            :Factory reset the Dynamixel of [ID]

 ==================== Commands for Dynamixel Protocol 2.0 ====================

 wrb2|w2 [ID] [ADDR] [VALUE] :Write byte [VALUE] to [ADDR] of [ID]
 wrw2 [ID] [ADDR] [VALUE]    :Write word [VALUE] to [ADDR] of [ID]
 wrd2 [ID] [ADDR] [VALUE]    :Write dword [VALUE] to [ADDR] of [ID]
 rdb2 [ID] [ADDR]            :Read byte value from [ADDR] of [ID]
 rdw2 [ID] [ADDR]            :Read word value from [ADDR] of [ID]
 rdd2 [ID] [ADDR]            :Read dword value from [ADDR] of [ID]
 r2 [ID] [ADDR] [LENGTH]     :Dumps the control table of [ID]
                               ([LENGTH] bytes from [ADDR])
 reboot2|rbt2 [ID]           :reboot the Dynamixel of [ID]
 reset2|rst2 [ID] [OPTION]   :Factory reset the Dynamixel of [ID]
                               OPTION: 255(All), 1(Except ID), 2(Except ID&Baud)

[CMD] scan

Scan Dynamixel Using Protocol 1.0
........................................................................................................................
........................................................................................................................
........

Scan Dynamixel Using Protocol 2.0

 [ID:001] Model No : 01030                ... SUCCESS

 [ID:002] Model No : 01030                ... SUCCESS
........................................................................................................................
........................................................................................................................
......

[CMD]
```

1. Shared Settings

```
PortHandler *portHandler = (PortHandler*)PortHandler::GetPortHandler(DEVICE_NAME);
```
: portHandler gets methods and members of PortHandlerLinux or PortHandlerWindows, and sets the DEVICE_NAME.

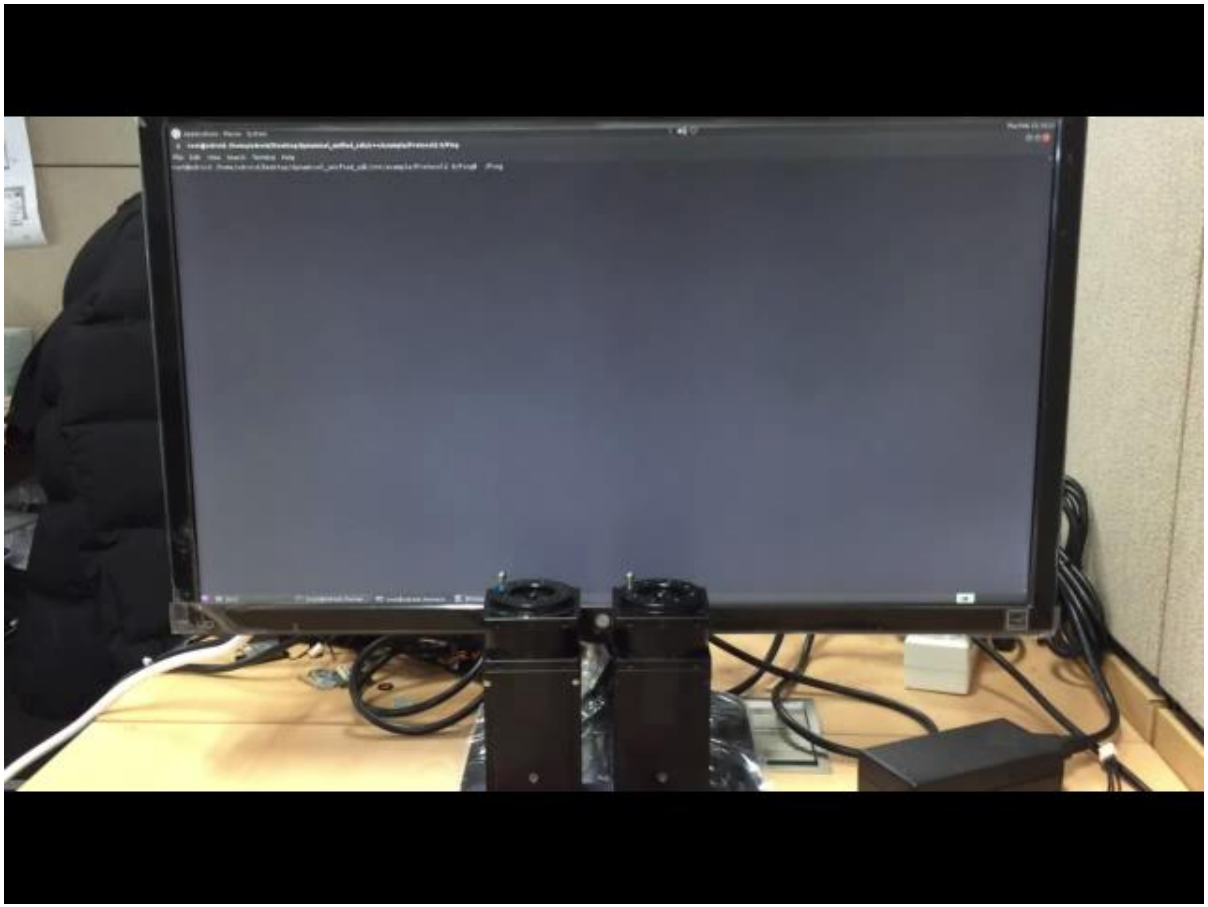
```
portHandler->OpenPort();
```
: portHandler opens and gets the handle of selected port.

```
portHandler->SetBaudRate(BAUDRATE);
```
: portHandler retries to open the selected port with selected BAUDRATE.

```
PacketHandler *packetHandler = PacketHandler::GetPacketHandler(PROTOCOL_VERSION);
```
: packetHandler gets methods and members of Protocol1PacketHandler or Protocol2PacketHandler by PROTOCOL_VERSION.

```
portHandler->ClosePort()
```
: portHandler closes selected port.

2. **Ping** : shows model number of Dynamixel

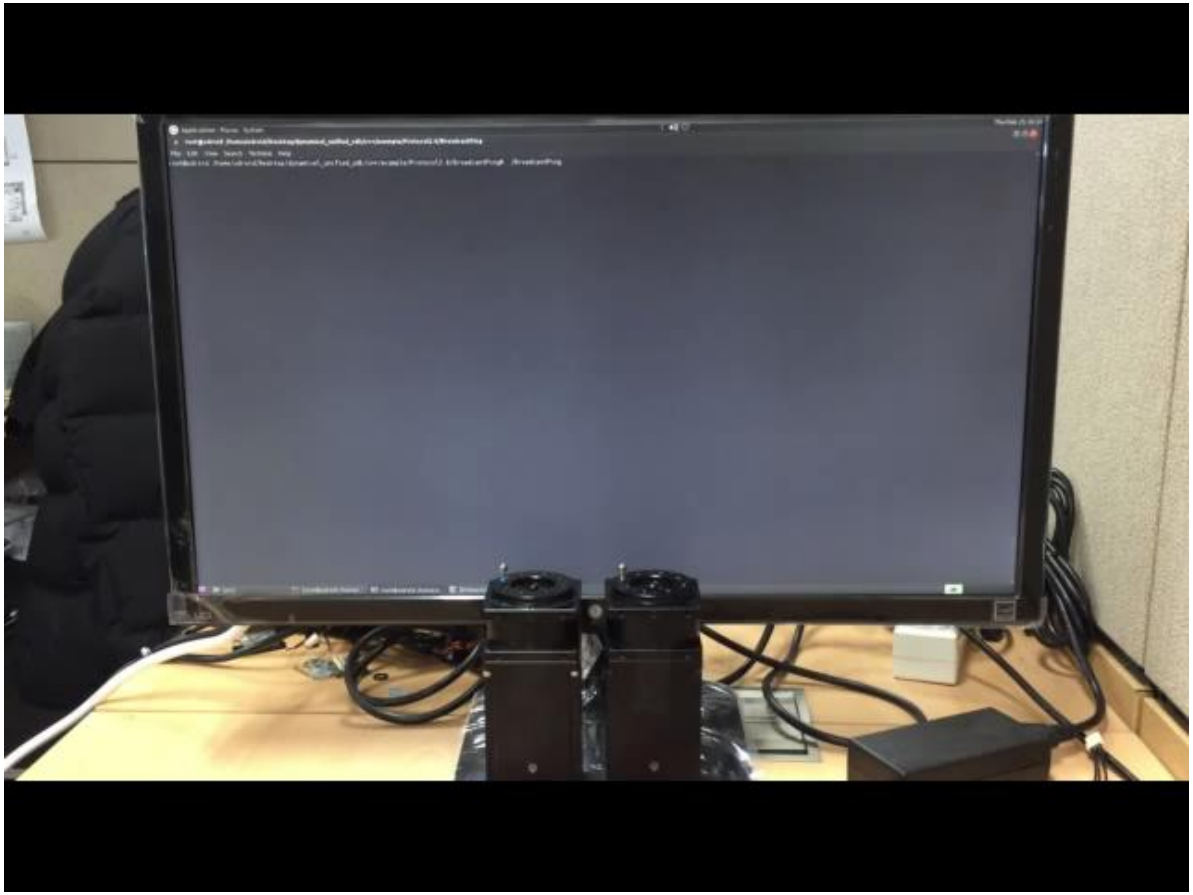


2. Ping : shows model number of Dynamixel

```
packetHandler->Ping(portHandler, DXL_ID, &dxl_model_number, &dxl_error);
```

: Try to Ping the DXL_ID and gets dxl_model_number.

3. **BroadcastPing** : shows the list of connected Dynamixels



3. BroadcastPing : shows the list of connected Dynamixels

```
packetHandler->BroadcastPing(portHandler, vec);
```

: Try to BroadcastPing all Dynamixels and gets the list of connected Dynamixels.

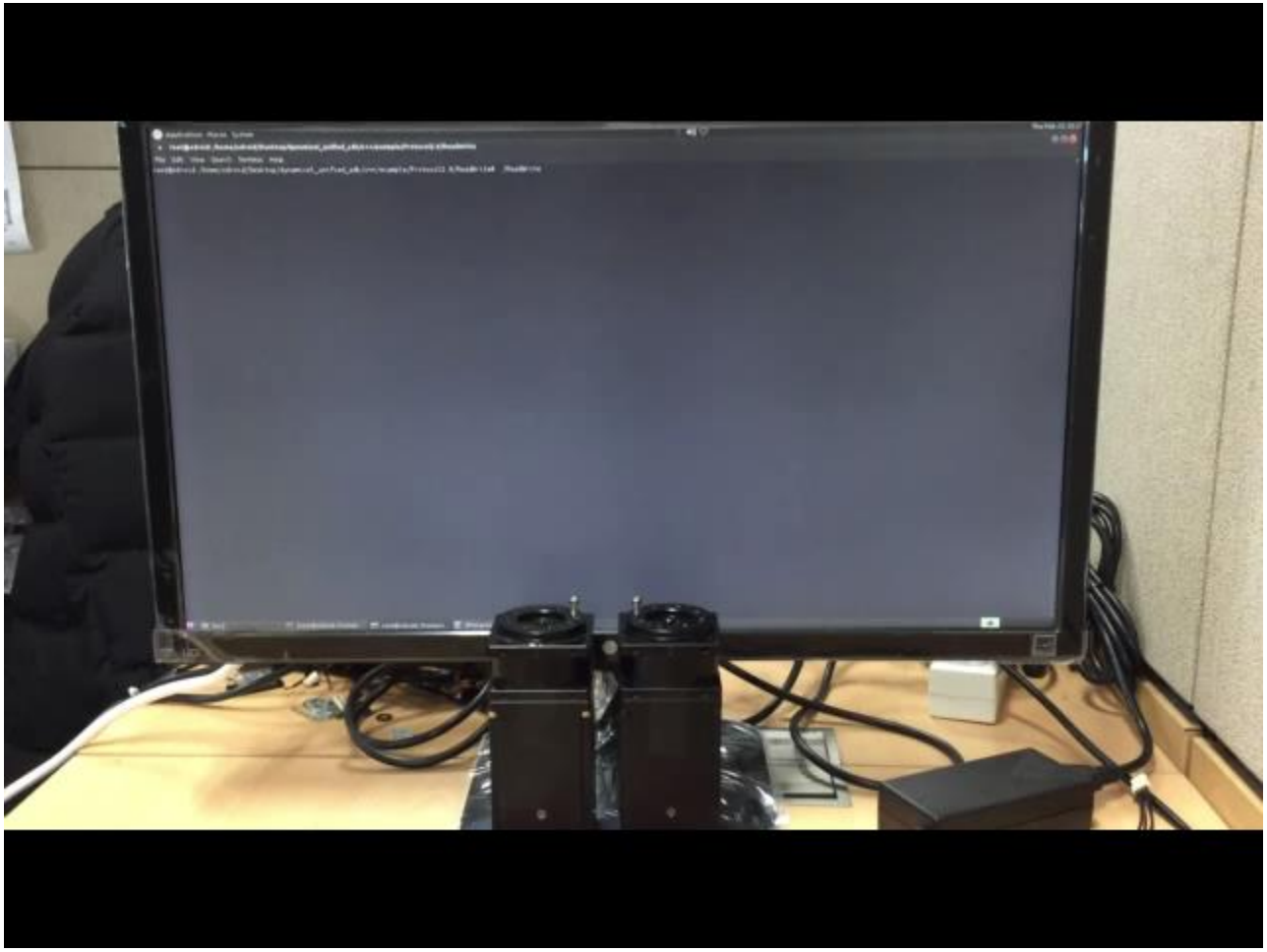
'vec' is defined as:

```
std::vector<UINT8_T> vec;
```

and use like this:

```
dxl_connected_ID = vec.at(DXL_ID);
```

4. **ReadWrite** : a Dynamixel rotates



4. ReadWrite : a Dynamixel rotates

```
packetHandler->Write1ByteTxRx(portHandler, DXL_ID, ADDR_TORQUE_ENABLE,
dxl_torque_enable, &dxl_error);
```

: packetHandler does dxl_torque_enable to DXL_ID ,ADDR_TORQUE_ENABLE, and receives dxl_error.

```
packetHandler->Write4ByteTxRx(portHandler, DXL_ID, ADDR_GOAL_POSITION,
dxl_goal_position, &dxl_error);
```

: Writes dxl_goal_position to the ADDR_GOAL_POSITION. The torque should be enabled previously.

```
packetHandler->Read4ByteTxRx(portHandler, DXL_ID, ADDR_PRESENT_POSITION,
&dxl_present_position, &dxl_error);
```

: Reads dxl_present_position from ADDR_PRESENT_POSITION. It gives the value right after this is called.

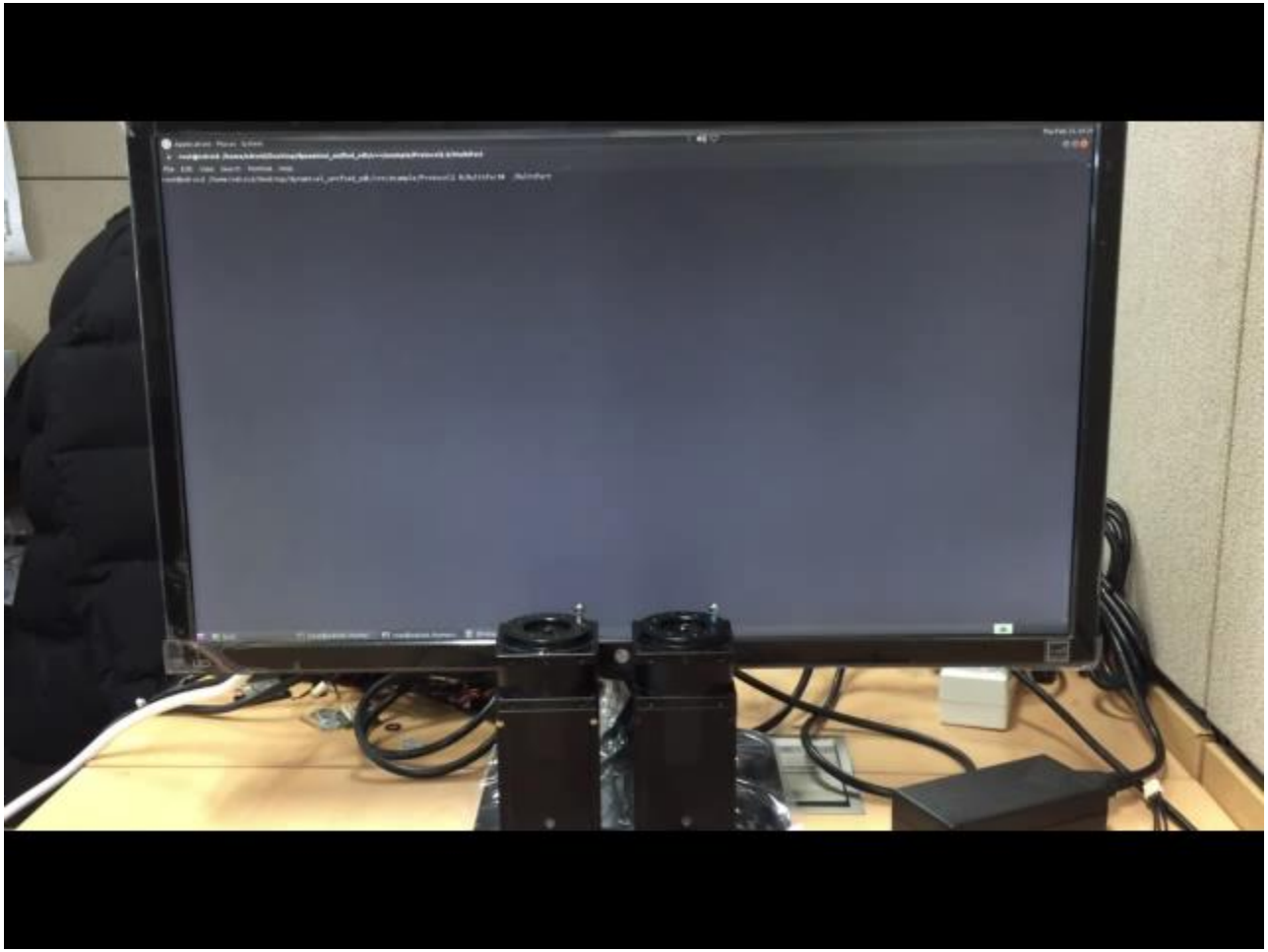
```
packetHandler->Read1ByteTxRx(portHandler, DXL_ID, ADDR_MOVING, &dxl_moving,
&dxl_error);
```

: Reads dxl_moving status. If the Dynamixel is moving, it returns 1, and if not, 0.

```
packetHandler->Write1ByteTxRx(portHandler, DXL_ID, ADDR_TORQUE_ENABLE,
dxl_torque_disable, &dxl_error);
```

: packetHandler does dxl_torque_disable.

5. **MultiPort** : two Dynamixels rotate using two ports



5. MultiPort : two Dynamixels rotate using two ports

```
packetHandler->Write4ByteTxRx(portHandler1, DXL1_ID, ADDR_GOAL_POSITION,
dxl_goal_position, &dxl_error);

packetHandler->Write4ByteTxRx(portHandler2, DXL2_ID, ADDR_GOAL_POSITION,
dxl_goal_position, &dxl_error);
```

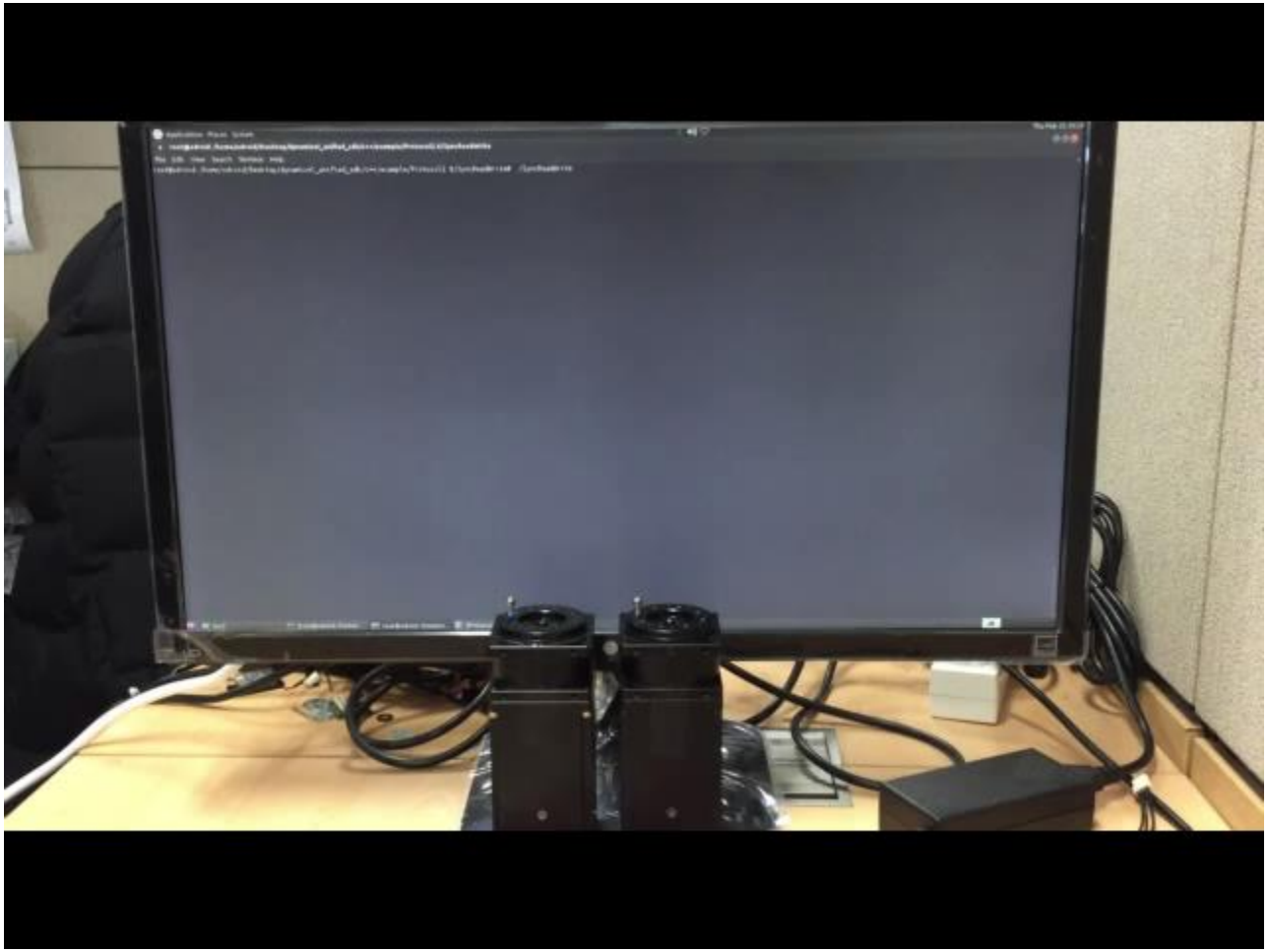
: Write dxl_goal_positions to each Dynamixels through two ports.

```
packetHandler->Read4ByteTxRx(portHandler1, DXL1_ID, ADDR_PRESENT_POSITION,
&dxl_present_position1, &dxl_error);

packetHandler->Read4ByteTxRx(portHandler2, DXL2_ID, ADDR_PRESENT_POSITION,
&dxl_present_position2, &dxl_error);
```

: Read dxl_present_position on each Dynamixels through two ports.

6. **SyncReadWrite** : writes identical data to two Dynamixels



6. SyncReadWrite : writes identical data to two Dynamixels

```
GroupSyncWrite groupSyncWrite(portHandler, packetHandler, ADDR_GOAL_POSITION,
LEN_GOAL_POSITION);
```

: initializes groupSyncWrite instance using portHandler and packetHandler.

```
param_goal_position[0] = DXL_LOBYTE(DXL_LOWORD(dxl_goal_position));
param_goal_position[1] = DXL_HIBYTE(DXL_LOWORD(dxl_goal_position));
param_goal_position[2] = DXL_LOBYTE(DXL_HIWORD(dxl_goal_position));
param_goal_position[3] = DXL_HIBYTE(DXL_HIWORD(dxl_goal_position));
```

: allocates dxl_goal_position into the byte array

```
groupSyncWrite.AddParam(DXL1_ID, param_goal_position);
groupSyncWrite.AddParam(DXL2_ID, param_goal_position);
```

: AddParameter param_goal_position to groupSyncWrite parameter storage

```
groupSyncWrite.TxPacket();
```

: Transmits the packet

```
groupSyncWrite.ClearParam();
```

: ClearParameter of groupSyncWrite parameter storage.

6. SyncReadWrite : writes identical data to two Dynamixels

```
GroupSyncRead groupSyncRead(portHandler, packetHandler, ADDR_PRESENT_POSITION,
LEN_PRESENT_POSITION);
```
: initializes groupSyncRead instance.

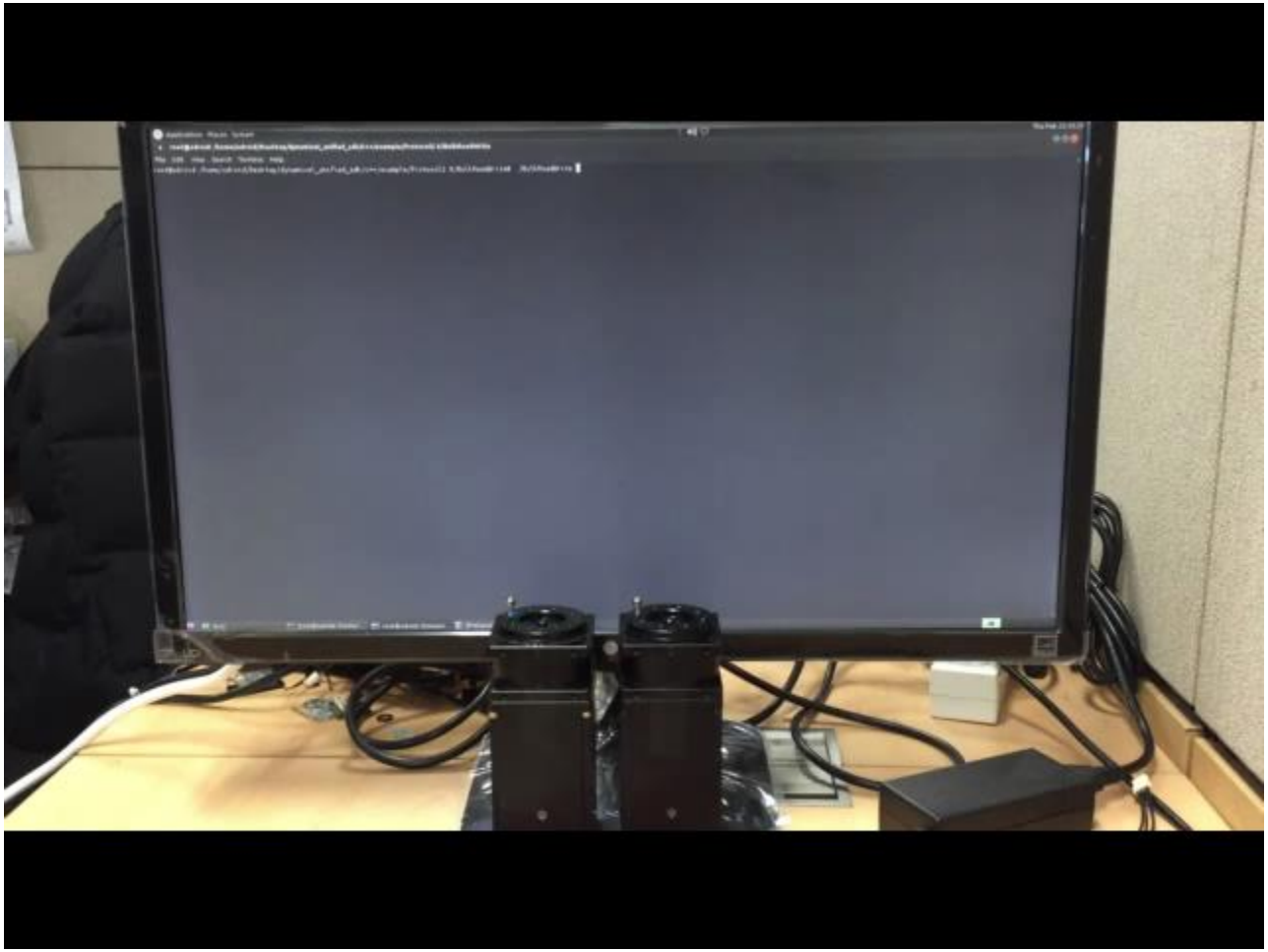
```
groupSyncRead.AddParam(DXL1_ID);
groupSyncRead.AddParam(DXL2_ID);
```
: AddParameter storage for Dynamixel's present position value

```
groupSyncRead.TxRxPacket();
```
: Transmits and receives the packet.

```
groupSyncRead.GetData(DXL1_ID, ADDR_PRESENT_POSITION, &dxl_present_position);
groupSyncRead.GetData(DXL2_ID, ADDR_PRESENT_POSITION, &dxl_present_position);
```
: GetData of dxl_present_positions

```
groupSyncRead.ClearParam();
```
: ClearParameter of groupSyncRead parameter storage.

7. **BulkReadWrite** : writes different data to two Dynamixels



7. BulkReadWrite : writes different data to two Dynamixels

```
GroupBulkWrite groupBulkWrite(portHandler, packetHandler);
```
: initializes groupBulkWrite instance.

```
param_goal_position[0] = DXL_LOBYTE(DXL_LOWORD(dxl_goal_position));
param_goal_position[1] = DXL_HIBYTE(DXL_LOWORD(dxl_goal_position));
param_goal_position[2] = DXL_LOBYTE(DXL_HIWORD(dxl_goal_position));
param_goal_position[3] = DXL_HIBYTE(DXL_HIWORD(dxl_goal_position));
```
: allocates dxl_goal_position into the byte array

```
groupBulkWrite.AddParam(DXL1_ID , ADDR_GOAL_POSITION, LEN_GOAL_POSITION,
param_goal_position);
groupBulkWrite.AddParam(DXL2_ID , ADDR_LED_RED, LEN_LED_RED, led_value);
```
: AddParameter param_goal_position and led_value to groupBulkWrite parameter storage

```
groupBulkWrite.TxPacket();
```
: Transmits the packet

```
groupBulkWrite.ClearParam();
```
: ClearParameter of groupBulkWrite parameter storage.

7. BulkReadWrite : writes different data to two Dynamixels

```
GroupBulkRead groupBulkRead(portHandler, packetHandler);
```
: initializes groupBulkRead instance.

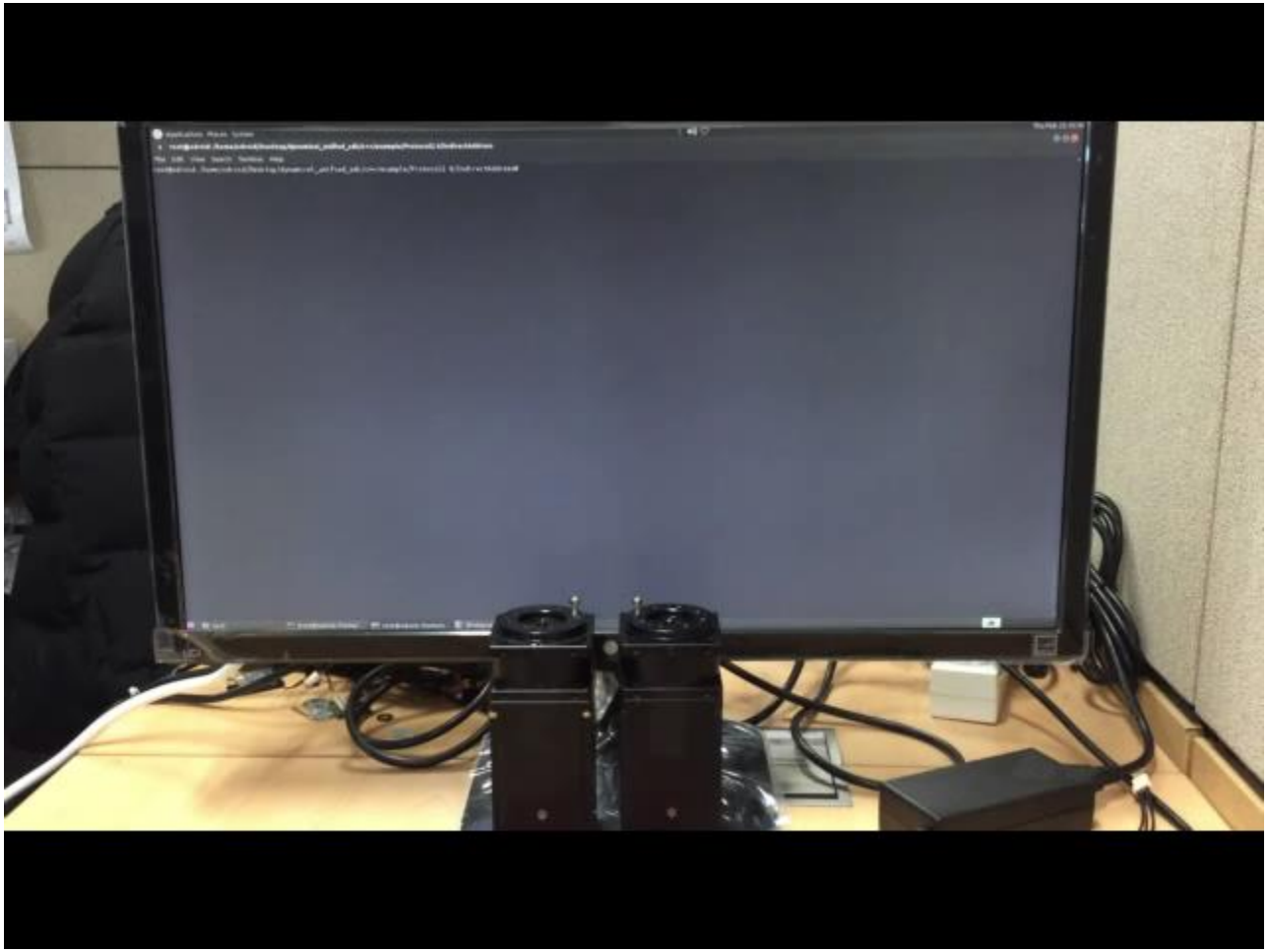
```
groupBulkRead.AddParam(DXL1_ID, ADDR_PRESENT_POSITION, LEN_PRESENT_POSITION);
groupBulkRead.AddParam(DXL2_ID, ADDR_LED_RED, LEN_LED_RED);
```
: Add parameter storage for Dynamixel's present position and LED value

```
groupBulkRead.TxRxPacket();
```
: Transmits and receives the packet.

```
groupBulkRead.GetData(DXL1_ID, ADDR_PRESENT_POSITION, &dxl_present_position);
groupBulkRead.GetData(DXL2_ID, ADDR_LED_RED, &dxl_led_value_read);
```
: GetData dxl_present_position or dxl_led_value_read from either of the Dynamixels

```
groupBulkRead.ClearParam();
```
: ClearParameter of groupBulkRead parameter storage.

8. **IndirectAddress** : writes data using indirect address



8. IndirectAddress : writes data using indirect address

```
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_WRITE + 0, ADDR_GOAL_POS + 0, &dxl_error);
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_WRITE + 2, ADDR_GOAL_POS + 1, &dxl_error);
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_WRITE + 4, ADDR_GOAL_POS + 2, &dxl_error);
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_WRITE + 6, ADDR_GOAL_POS + 3, &dxl_error);
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_WRITE + 8, ADDR_LED_RED, &dxl_error);
```

: makes INDIRECTDATA replace 4 Byte GOAL_POSITION and 1 Byte LED_RED.

```
GroupSyncWrite groupSyncWrite(portHandler, packetHandler, ADDR_INDIRECTDATA_WRITE,
LEN_INDIRECTDATA_WRITE);
```

: initializes groupSyncWrite instance and points out the INDIRECTADDRESS.

```
param_indirect_data_write[0] = DXL_LOBYTE(DXL_LOWORD(dxl_goal_position));
param_indirect_data_write[1] = DXL_HIBYTE(DXL_LOWORD(dxl_goal_position));
param_indirect_data_write[2] = DXL_LOBYTE(DXL_HIWORD(dxl_goal_position));
param_indirect_data_write[3] = DXL_HIBYTE(DXL_HIWORD(dxl_goal_position));
param_indirect_data_write[4] = dxl_led_value;
```

: allocates dxl_goal_position and dxl_led_value into the byte array

```
groupSyncWrite.AddParam(DXL_ID, param_indirect_data_write);
```

: AddParameter storage for the datas

8. IndirectAddress : writes data using indirect address

```
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_READ + 0, ADDR_PRES_POS + 0, &dxl_error);
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_READ + 2, ADDR_PRES_POS + 1, &dxl_error);
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_READ + 4, ADDR_PRES_POS + 2, &dxl_error);
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_READ + 6, ADDR_PRES_POS + 3, &dxl_error);
packetHandler->Write2ByteTxRx(portHandler, DXL_ID, ADDR_INDIRECTADDR_READ + 8, ADDR_MOVING, &dxl_error);
```

: makes INDIRECTDATA replace 4 Byte PRESENT_POSITION and 1 Byte MOVING.

```
GroupSyncRead groupSyncRead(portHandler, packetHandler, ADDR_INDIRECTADDR_READ,
LEN_INDIRECTDATA_READ);
```

: initializes groupSyncRead instance and points out the INDIRECTADDRESS.

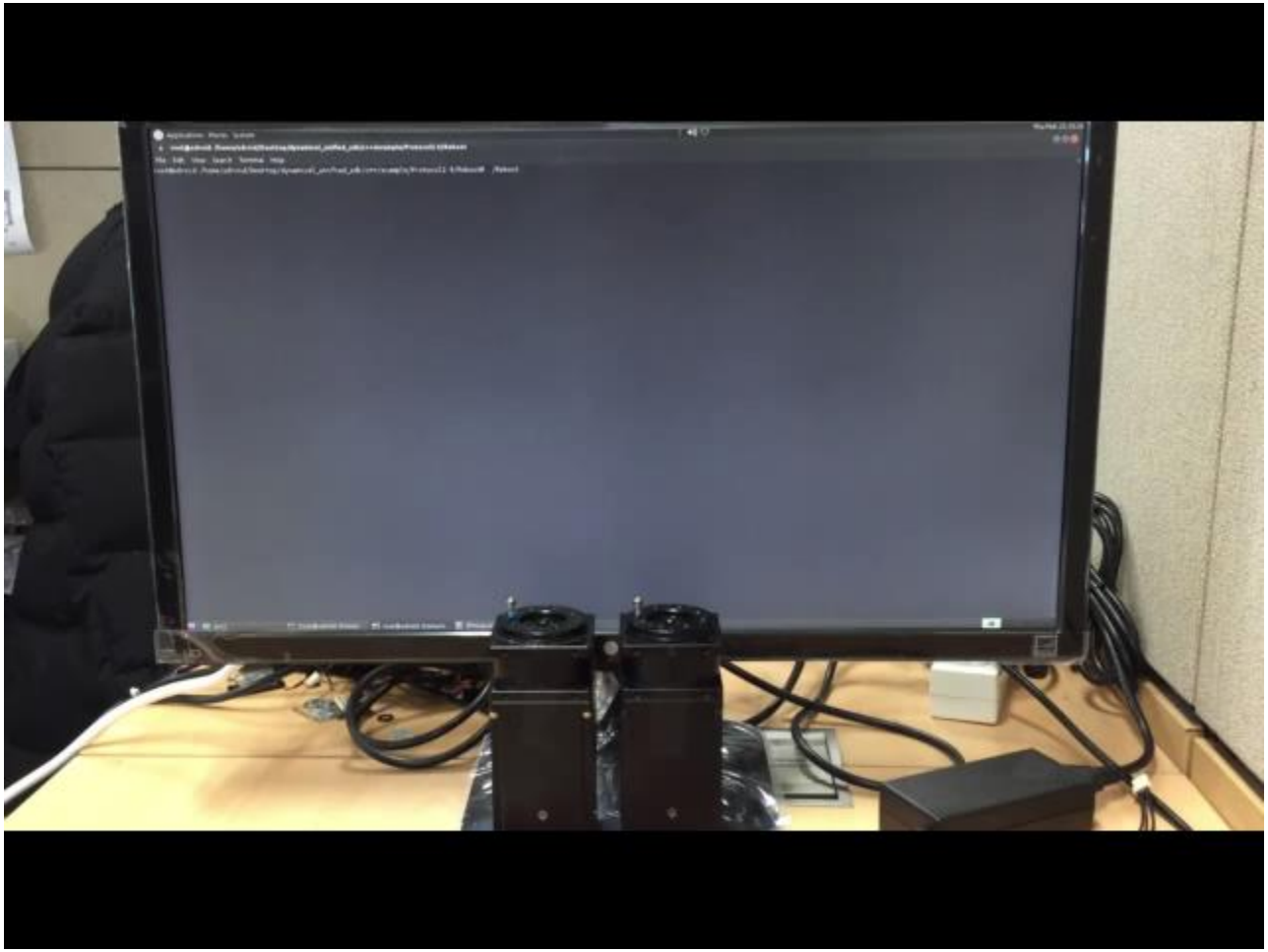
```
groupSyncRead.AddParam(DXL_ID);
```

: AddParameter storage for Dynamixel's present position and LED value

```
groupSyncRead.GetData(DXL_ID, ADDR_INDIRECTDATA_READ, &dxl_present_position);
groupSyncRead.GetData(DXL_ID, ADDR_INDIRECTDATA_READ + 4, &dxl_moving);
```

: GetData of dxl_present_position and dxl_moving from the INDIRECTDATA

9. Reboot : Reboots Dynamixels



9. Reboot : Reboots Dynamixels

```
packetHandler->Reboot(portHandler, DXL_ID, &dxl_error);
```

: Try to Reboot. DXL Green LED will flicker while it reboots

10. FactoryReset : Resets Dynamixel with default values

```
packetHandler->FactoryReset(portHandler, DXL_ID, OPERATION_MODE, &dxl_error);
```

: Try to FactoryReset. This will reset Dynamixel settings to default values

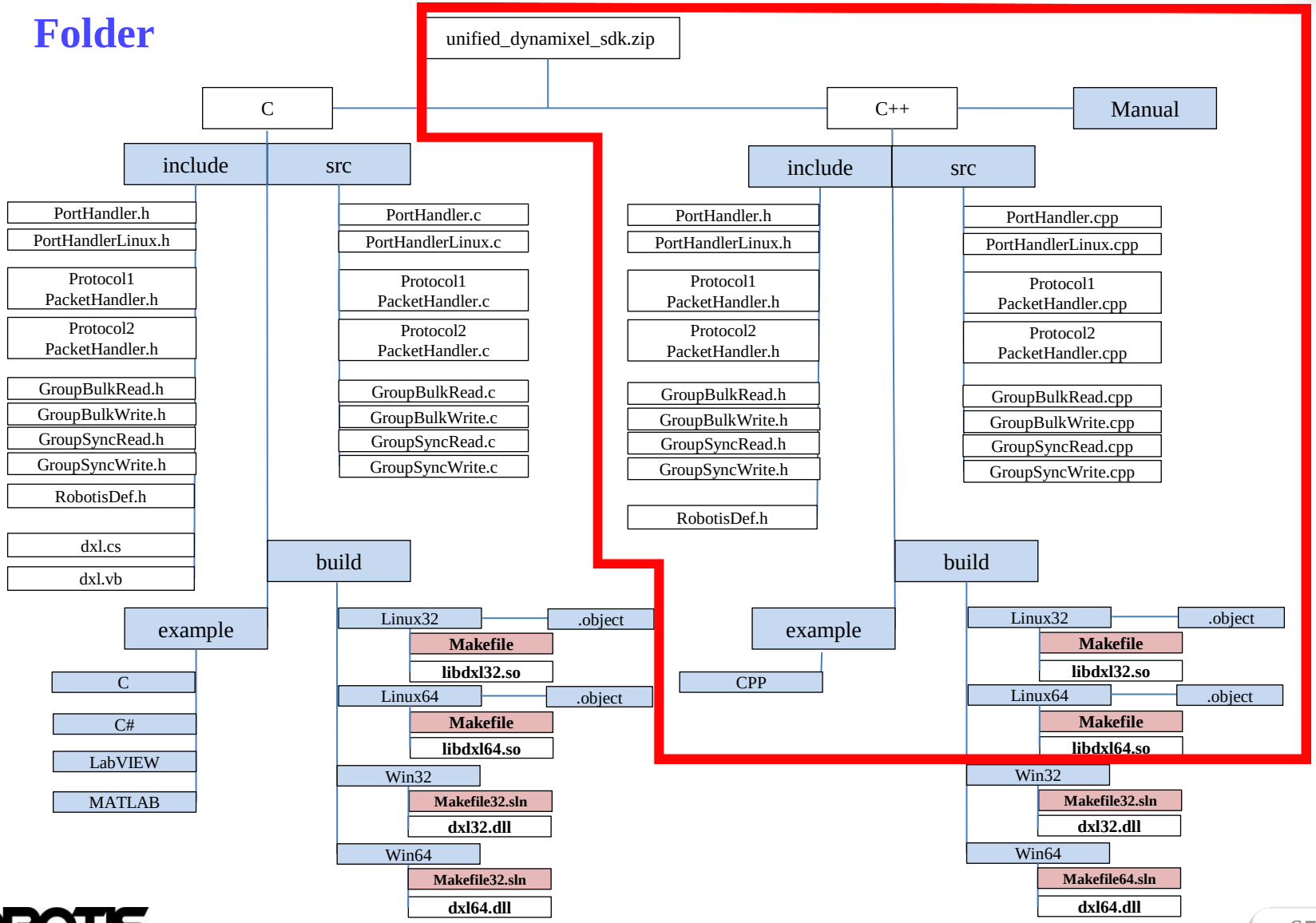
FactoryReset has three OPERATION_MODE :
0xFF : reset all values
0x01 : reset all values except ID
0x02 : reset all values except ID and baudrate

(In this example, Hardware Error will be occurred when 12V SMPS is used for Dynamixel Pro)

SDK Structure
(API Reference)

SDK Structure(API Reference)

1. Folder



2. Class

`PortHandler`	handles port and devices
`PortHandlerLinux`	handles port and devices with Linux
`PortHandlerWindows`	handles port and devices with Windows
`PacketHandler`	handles packets
`Protocol1PacketHandler`	handles PROTOCOL 1.0 packets
`Protocol2PacketHandler`	handles PROTOCOL 2.0 packets
`GroupSyncRead`	handles Dynamixel group for SyncRead
`GroupBulkRead`	handles Dynamixel group for BulkRead
`GroupSyncWrite`	handles Dynamixel group for SyncWead
`GroupBulkWrite`	handles Dynamixel group for BulkWrite

3. Methods

- **PortHandler**
- **PortHandlerLinux**
- **PortHandlerWindows**
- PacketHandler
- Protocol1PacketHandler
- Protocol2PacketHandler
- GroupSyncRead
- GroupBulkRead
- GroupSyncWrite
- GroupBulkWrite

`OpenPort()`	opens the port
`ClosePort()`	closes the port
`ClearPort()`	clears the port
`SetPortName()`	sets device name
`GetPortName()`	gets device name
`SetBaudRate()`	sets port baudrate
`GetBaudRate()`	gets port baudrate
`GetBytesAvailable()`	how many data can be read
`ReadPort()`	reads port packet buffer
`WritePort()`	writes port packet buffer
`SetPacketTimeout()`	sets time to decide communication result
`IsPacketTimeout()`	checks communication result

3. Methods

- PortHandler
- PortHandlerLinux
- PortHandlerWindows
- **PacketHandler**
- **Protocol1PacketHandler**
- **Protocol2PacketHandler**
- GroupSyncRead
- GroupBulkRead
- GroupSyncWrite
- GroupBulkWrite

`TxPacket()`	trasmits the packet
`RxPacket()`	receives the packet
`TxRxPacket()`	transmits & receives the packet
`Ping()`	ping
`BroadcastPing`	ping all connected Dynamixels
`Action()`	commands 'Run' the Regwritten
`RegWrite()`	writes the packets wait for the 'Action' command
`Reboot()`	Reboots Dynamixel
`FactoryReset()`	resets all Dynamixel settings
`Read functions`	functions for Reading
`Write functions`	functions for Writing
`SyncRead functions`	functions for SyncRead
`SyncWrite functions`	functions for SyncWrite
`BulkRead functions`	functions for BulkRead
`BulkWrite functions`	functions for BulkWrite

3. Methods

- PortHandler
- PortHandlerLinux
- PortHandlerWindows
- PacketHandler
- Protocol1PacketHandler
- Protocol2PacketHandler
- **GroupSyncRead**
- **GroupBulkRead**
- GroupSyncWrite
- GroupBulkWrite

`AddParam()`	adds parameter to the packet
`RemoveParam()`	removes parameter to the packet
`ClearParam()`	clears parameter to the packet
`TxPacket()`	transmits the packet
`RxPacket()`	receives the packet
`TxRxPacket()`	transmits & receives the packet
`GetData()`	gets read data

3. Methods

- PortHandler
- PortHandlerLinux
- PortHandlerWindows
- PacketHandler
- Protocol1PacketHandler
- Protocol2PacketHandler
- GroupSyncRead
- GroupBulkRead
- **GroupSyncWrite**
- **GroupBulkWrite**

`AddParam()`	adds parameter to the packet
`ChangeParam()`	removes parameter to the packet
`ChangeParam()`	changes parameter of the packet
`ClearParam()`	clears parameter of the packet
`TxPacket()`	transmits the packet

More Information

For more details, please visit ROBOTIS support page

: support.robotis.com/en/