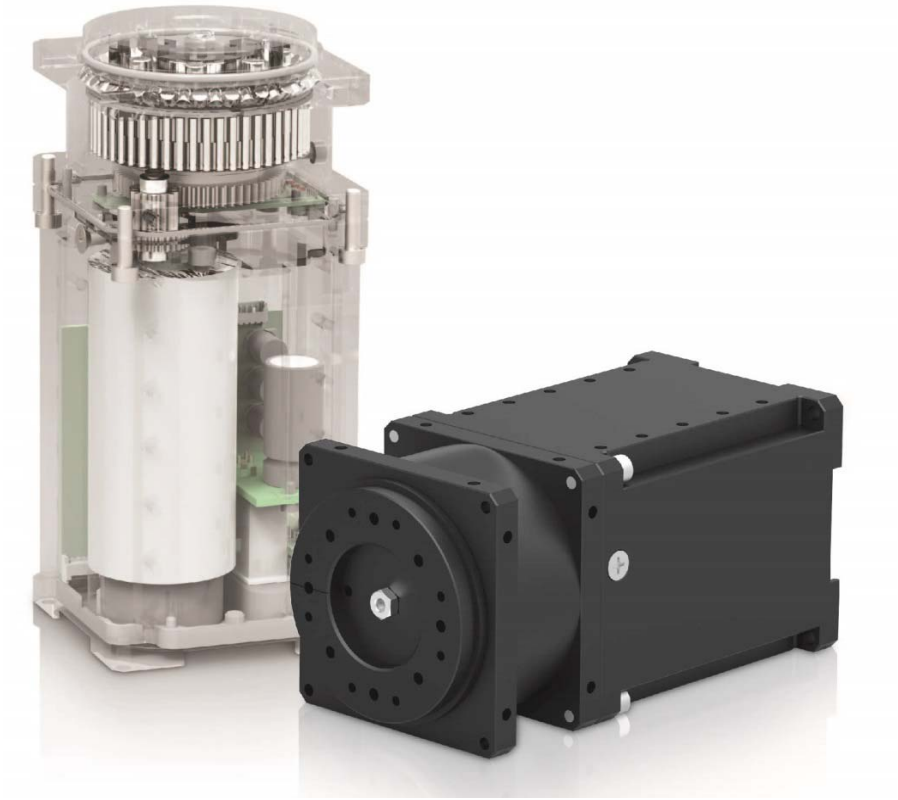
# Quick Start for Dynamixel Pro



ROBOTIS

## CONTENTS

# 1 Tutorial

This guide is written for first-time Dynamixel PRO users, but, familiar with C and C++ programming languages.

## 1.1 Roboplus

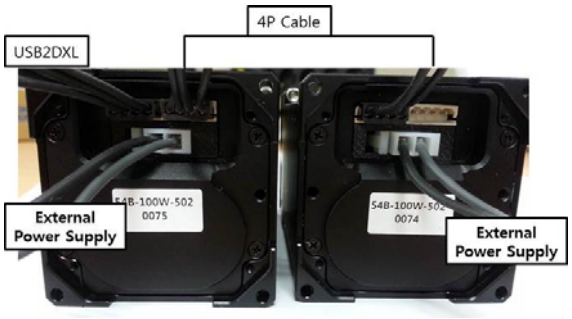
### 1.1.1 Preparation

i. RoboPlus installation

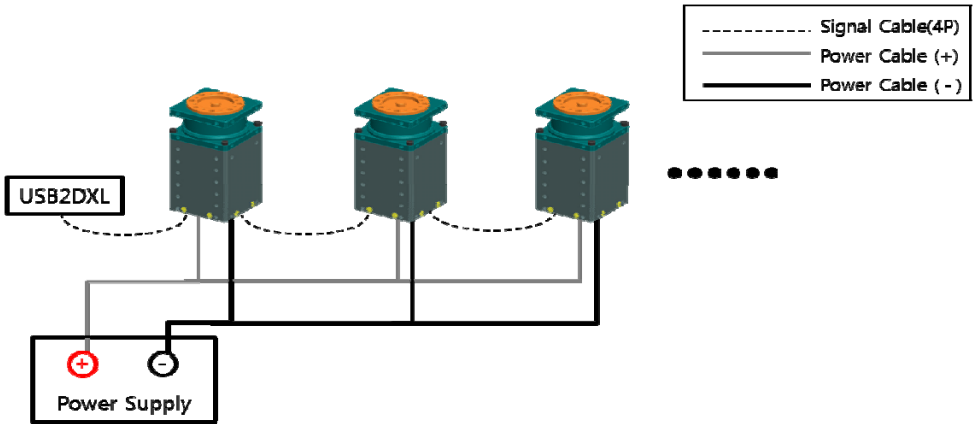
- RoboPlus is a software package that allows you to easily control and program all ROBOTIS products.
- This guide contains information regarding the use of RoboPlus's Dynamixel Wizard to test Dynamixel PRO.
- You may download the most current version of RoboPlus at http://www.robotis.com/xe/download.

ii. Wiring of Dynamixel PRO 54 series

- To operate a Dynamixel PRO, at least, a USB-to-Dynamixel dongle and 24V power supply (for high-power operations) or ROBOTIS' conventional 12V SMPS (for low-power operations) are required.
- Dynamixel PRO 54 series can be powered via external power cable or 4-pin cable; however, it is recommended to use an external power supply (via power cable) for better stability.
- As illustrated below, use a 4-pin cable to connect the USB-to-Dynamixel dongle and Dynamixel PRO.
- Connect one end of the power cable to Dynamixel PRO; the other end to the power supply.



- Please note the schematic below.




- Dynamixel PROs can be connected in series (daisy chain) with the 4-pin cables for communications (and low-power operations). However, power (high-power operations) must be supplied in parallel (individually).

iii. Wiring of Dynamixel PRO 42 series

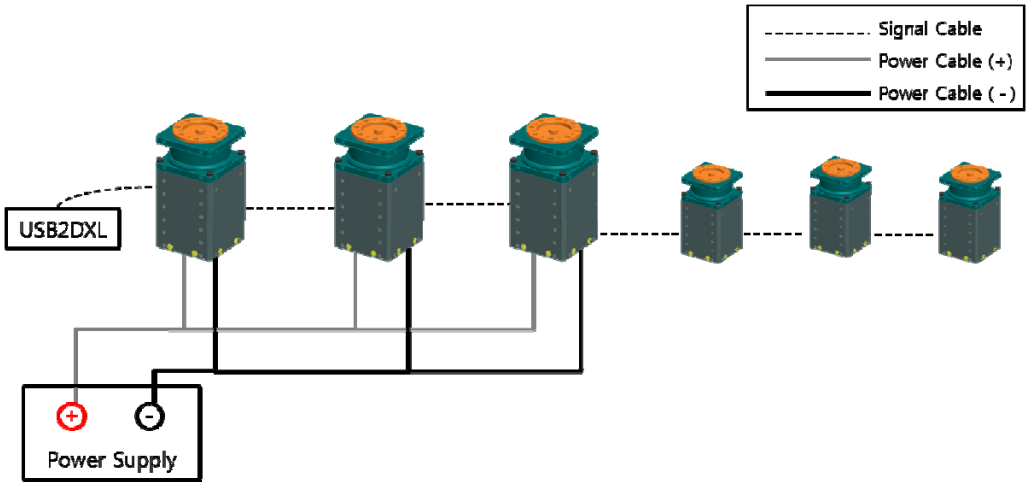
- Similar to the 54 series, the 42 series requires a 24V power supply.
- Dynamixel PRO 42 series can be powered by the 4-pin cable (low-power operations). Connecting via SMPS-to-Dynamixel is also allowed.
- A USB-to-Dynamixel dongle connects to SMPS-to-Dynamixel; then SMPS-to-Dynamixel to Dynamixel PRO(s).

iv. Wiring of Dynamixel PRO 54 & 42 series

- First, connect an USB-to-Dynamixel dongle and Dynamixel PRO via 4-pin cable.
- Second, connect a dedicated power supply to Dynamixel PRO via power cable.

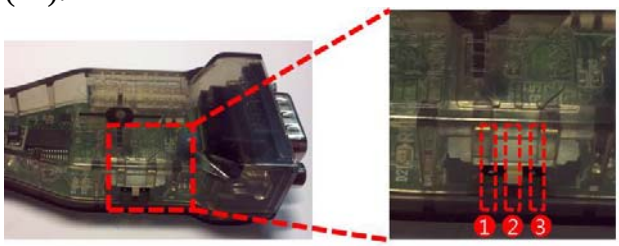
For multiple Dynamixel PROs they can be connected in series, for communications (daisy chain) and low-power operations, and parallel (individually), for high-power operations, as illustrated below.

- Please note the schematic below.



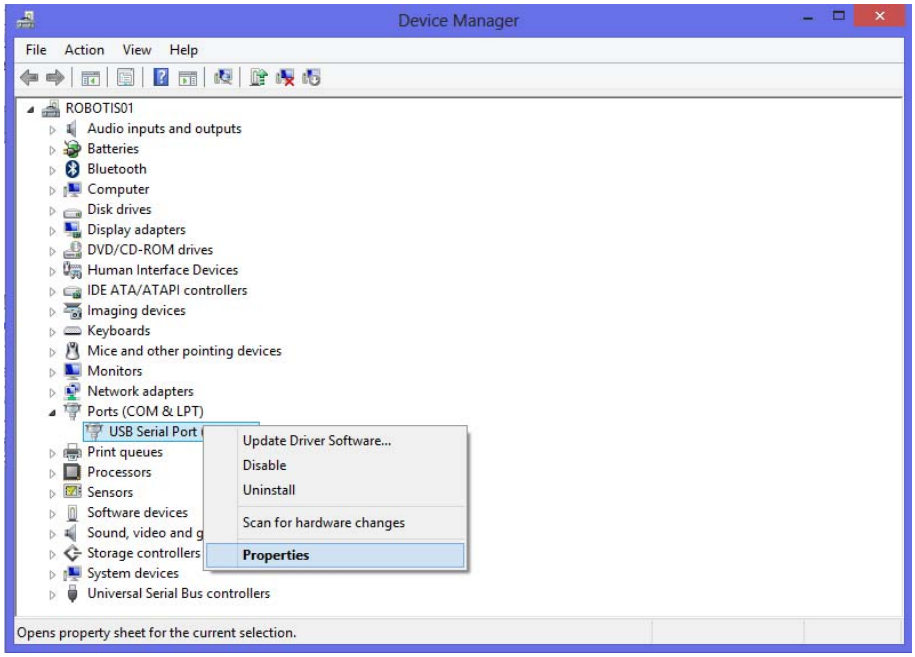

v. USB-to-Dynamixel dongle (RS485 setting)

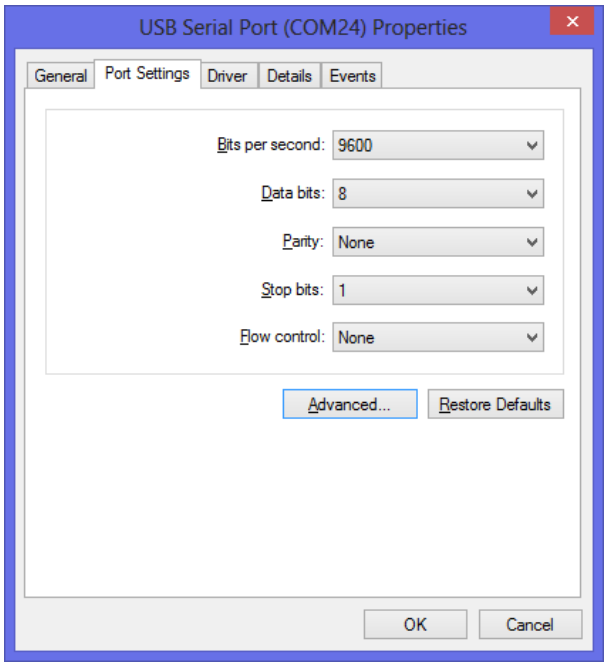
- Dynamixel PRO communicates via RS485.
- Set the switch on the left hand side of the USB-to-Dynamixel dongle to RS485 (#2).

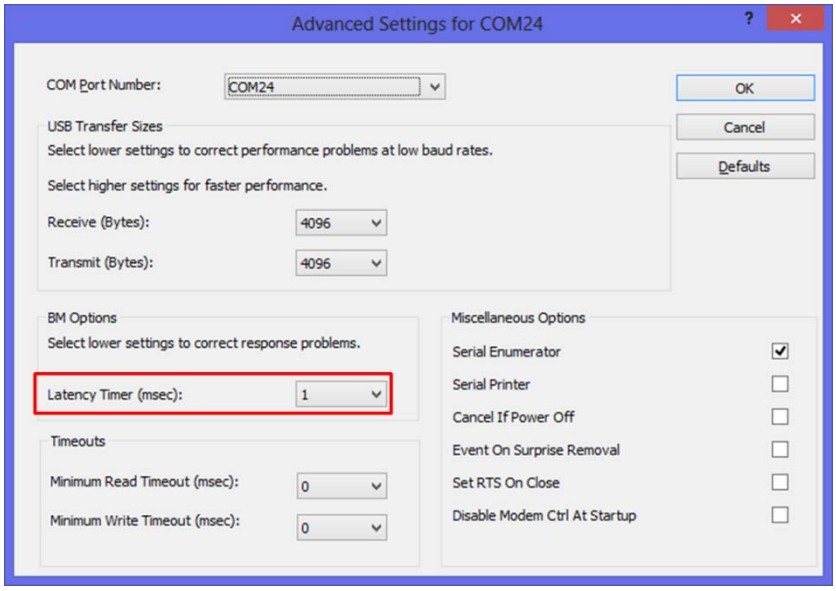



vi. COM Port Latency Time setting

- To control Dynamixel PRO using a USB-to-Dynamixel dongle, it is recommended to modify the Latency Time of the Port. Please refer to the images below to adjust the Latency Time.
- Under Windows Device Manager → Port → USB Serial Port (right mouse click)→ properties → Port Setting -> Advanced → Latency Timer (msec) → set to 1msec.


Device Manager
File Action View Help
ROBOTIS01
Audio inputs and outputs
Batteries
Bluetooth
Computer
Disk drives
Display adapters
DVD/CD-ROM drives
Human Interface Devices
IDE ATA/ATAPI controllers
Imaging devices
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
USB Serial Port (
Print queues
Processors
Sensors
Software devices
Sound, video and g
Storage controllers
System devices
Universal Serial Bus controllers
Update Driver Software...
Disable
Uninstall
Scan for hardware changes
Properties
Opens property sheet for the current selection.


USB Serial Port (COM24) Properties
General
Port Settings
Driver
Details
Events
Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None
Advanced...
Restore Defaults
OK
Cancel

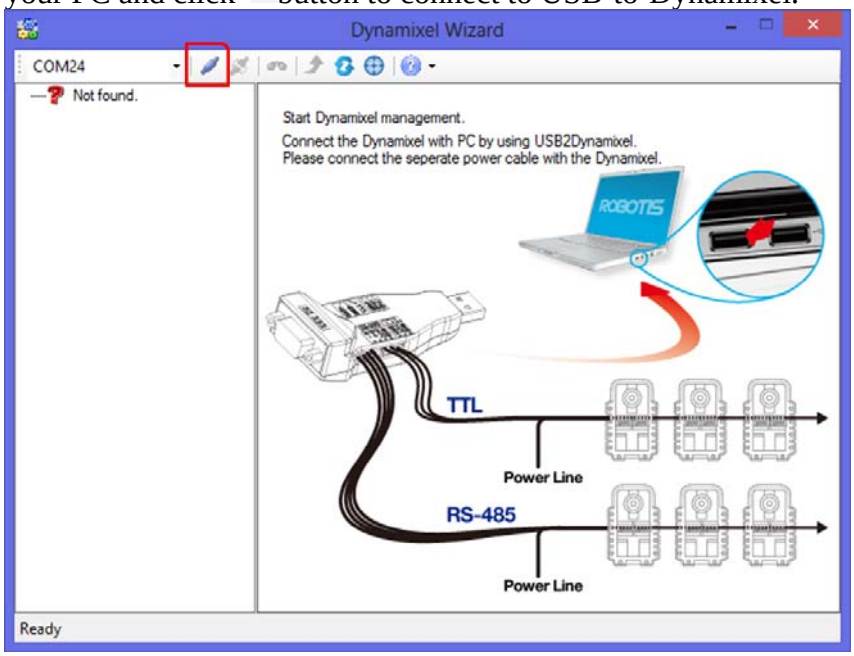



- Click “Ok” to confirm.

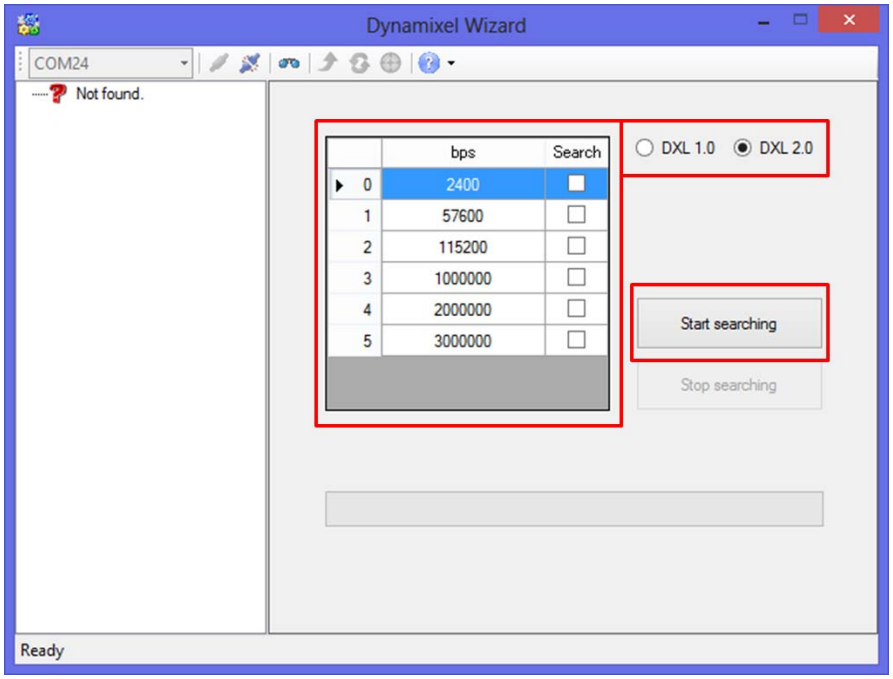
### 1.1.2 Dynamixel Wizard

#### i. Operating a Dynamixel PRO

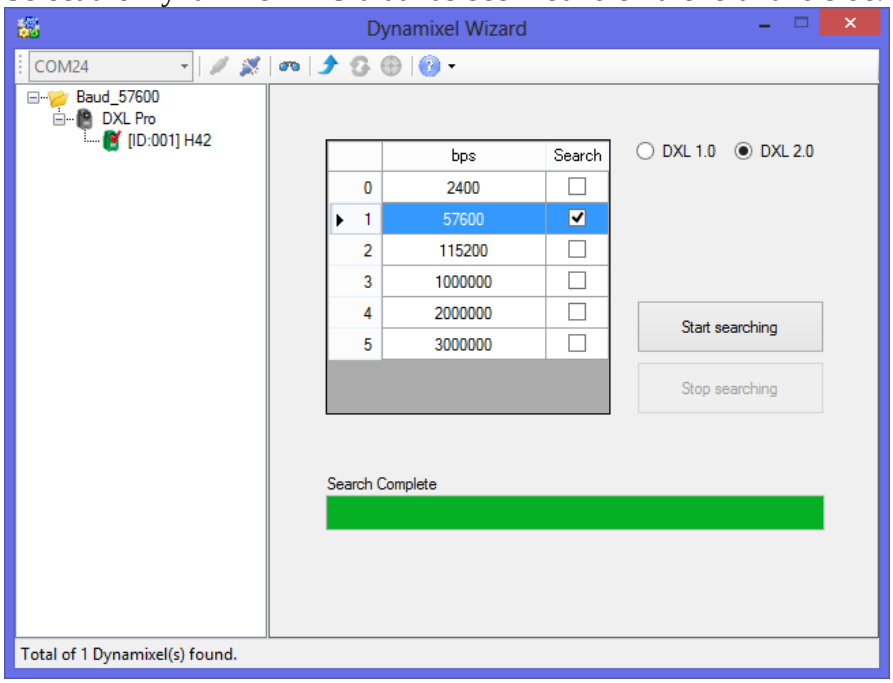
- Supply 24V power to Dynamixel PRO once the wiring is complete. For safety, finish connecting the wires before supplying the power.
- Open RoboPlus to run Dynamixel Wizard.
- Select a COM Port that refers to the correction of USB-to-Dynamixel dongle to your PC and click button to connect to USB-to-Dynamixel.

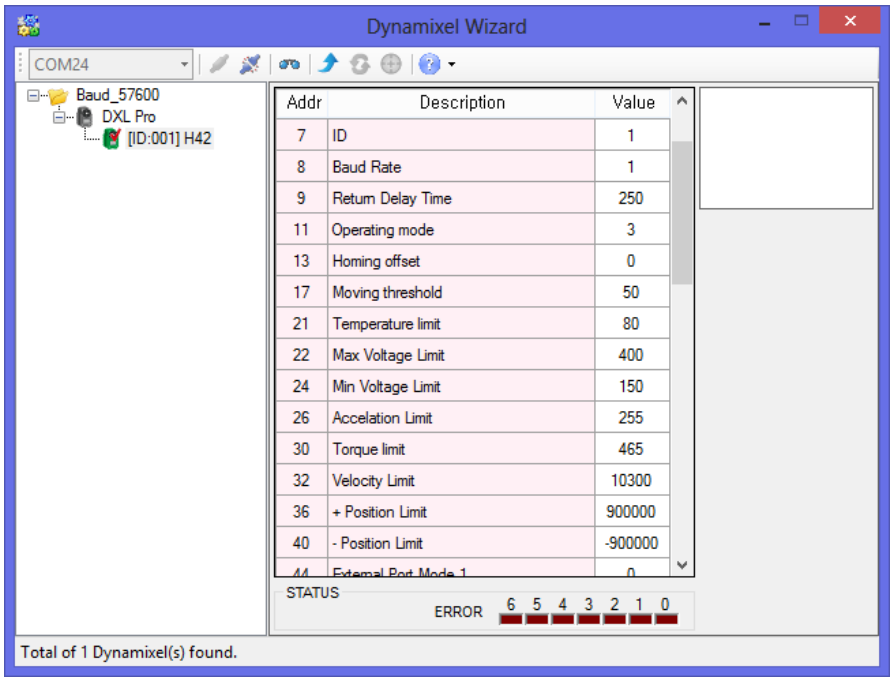



- Once the port is connected, select ‘DXL 2.0’ and ‘57600,’ then click ‘Start Searching.’ Dynamixel PRO’s Default ID setting is 1 and baud rate of 57600 bps.

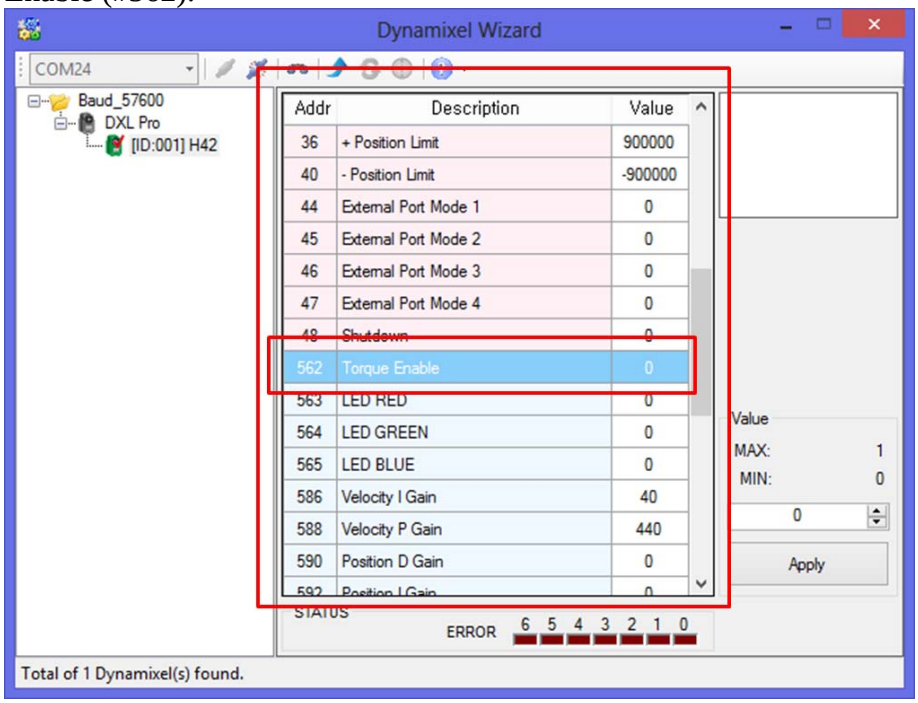



- Select the Dynamixel PRO that has been found on the left hand side.

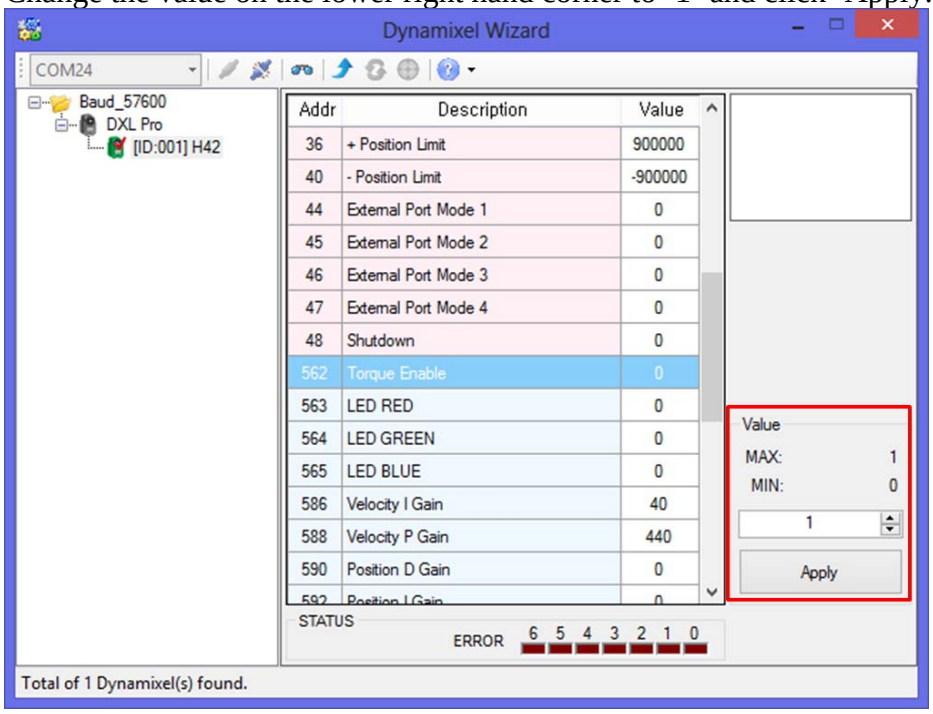




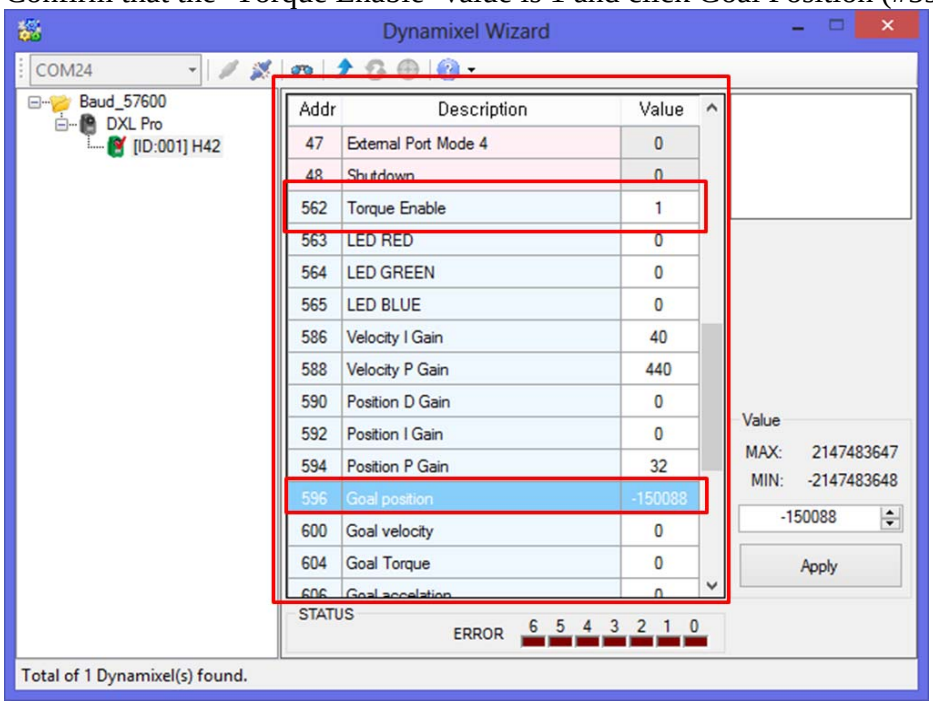

- Unlike other Dynamixel models, Dynamixel PRO is only drivable when torque is enabled. Therefore, torque must be enabled before driving Dynamixel PRO.
- Scroll down the middle table on Dynamixel PRO Wizard and locate Torque Enable (#562).



- Change the value on the lower right hand corner to ‘1’ and click ‘Apply.’

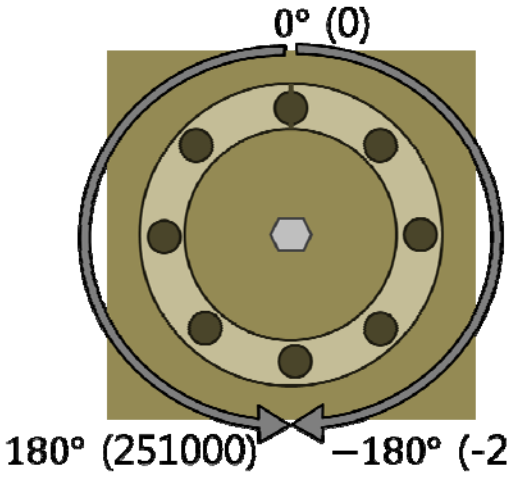


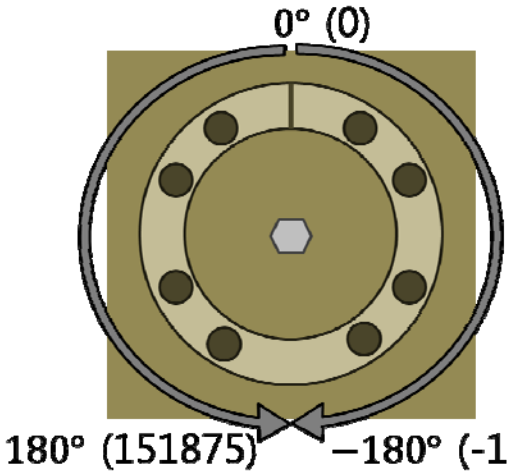

- Confirm that the ‘Torque Enable’ value is 1 and click Goal Position (#596).




- Change the value of Goal Position on the lower right hand corner with an appropriate value and click ‘Apply’. Check each Dynamixel PRO’s Min and Max Position Limits (lower and upper “soft” limits).
  **by default Dynamixel PRO 54 series: -251000 ~ 251000 and Dynamixel PRO 42 series: -151875 ~ 151875.**
- Visually verify position of Dynamixel PRO after clicking on ‘Apply.’

- If Dynamixel PRO does not move, check to see if the Torque Enable value is 1. If the value is not set to '1' change the value and try again.
- Change the Goal Position value and click 'Apply.' Visually check Dynamixel PRO's position.
- The relationship between the Goal Position value and degree of rotation is shown below.

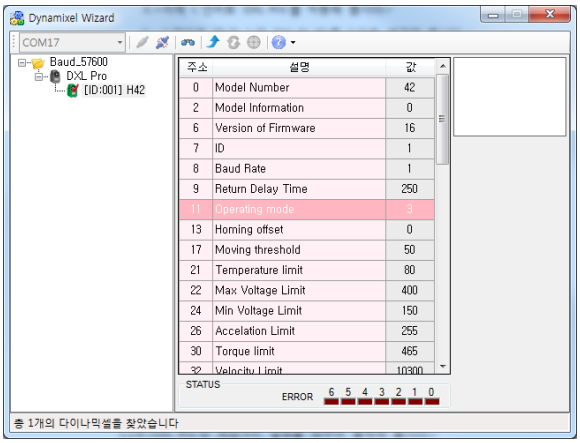
<table>
<tr><th>Model</th><th>Relationship between angle(deg) and position value</th></tr>
<tr><td>54 200W<br>54 100W</td><td>-180 ~ 180 (deg) → -251000 ~ 251000<br> 

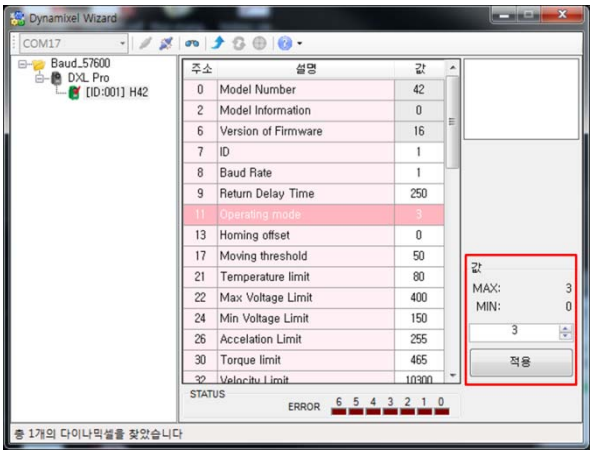
<br>$Goal\ Angle(deg) = Goal\ Position\ Value \times \frac{180°}{251000}$</td></tr>
<tr><td>42 20W</td><td>-180 ~ 180 (deg) → -151875 ~ 151875<br> 

<br>$Goal\ Angle(deg) = Goal\ Position\ Value \times \frac{180°}{151875}$</td></tr>
</table>

ii. Operating Dynamixel PRO in Wheel Mode

- There are 3 Modes in with Dynamixel PRO (Joint Mode, Wheel Mode, and Torque Mode).
- On Dynamixel Wizard, locate and click 'Operating Mode' (#11).
- When Torque Enable (#562) is turned 'On', the Operating Mode cannot be modified (left image below). The Operating Modes can be modified once the Torque Enable is turned 'Off' by setting it to 0. (right image below)



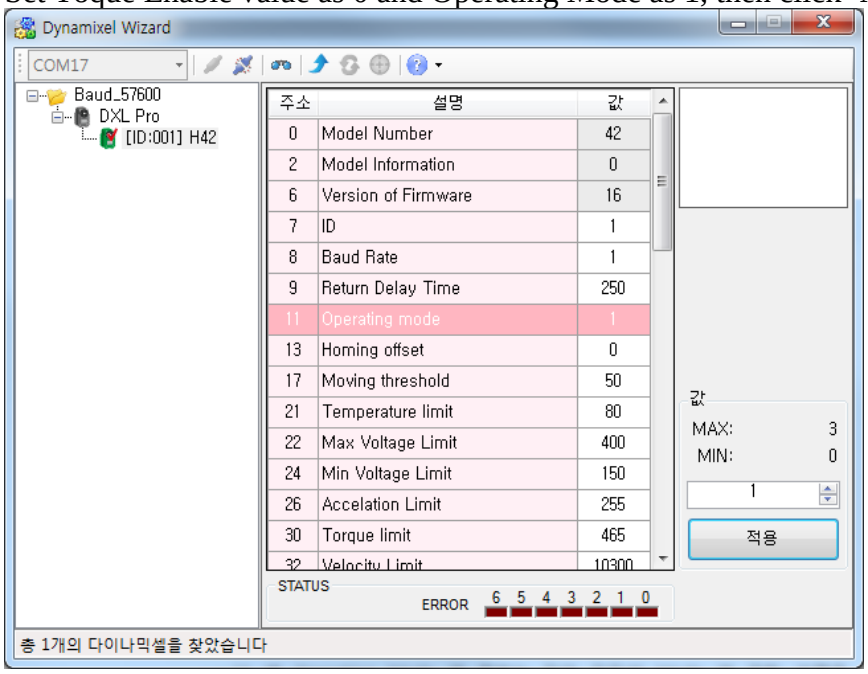




- Operating Mode (#11) settings are described below.

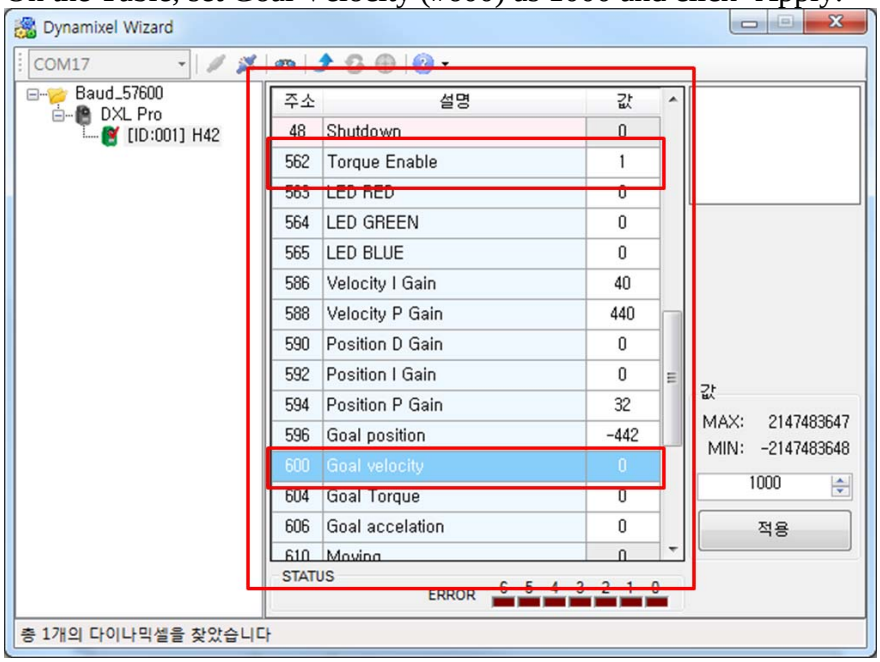
| Operating Mode | Value | Description |
|---|---|---|
| Joint Mode | 3 | Controls the velocity and position of the servo.<br>1. Can move to the desired position at a desired velocity.<br>2. Can move using self-assigned trapezoidal Velocity Profile. |
| Wheel Mode | 1 | Controls the velocity of the servo.<br>1. Can rotate the servo at a desired velocity.<br>2. Can move using self-assigned trapezoidal Velocity Profile.<br>3. Cannot control the position of the servo. |
| Torque Mode | 0 | Controls the torque of the servo.<br>1. Cannot control velocity and position.<br>2. Only controls the output torque.<br>3. Since position and velocity control are not possible, the performance is similar to Wheel Mode. |

- Note: use the reference above to set Dynamixel PRO to an appropriate mode.

- Set Toque Enable value as 0 and Operating Mode as 1, then click ‘Apply.’




- Next, set Torque Enable (#562) as 1 and click ‘Apply.’
  **※ Torque Enable value on Dynamixel PRO series MUST be 1 to be activated.**
- On the Table, set Goal Velocity (#600) as 1000 and click ‘Apply.’




- Visually check for Dynamixel PRO rotate slowly in counter-clockwise direction.
- If Dynamixel PRO does not rotate, check if Torque Enable is set to 1.
- Set Goal Velocity (#600) to -1000 then click on ‘Apply.’ Visually check for Dynamixel PRO rotate slowly in clockwise direction.
- Set Goal Velocity as 0 and Dynamixel PRO stops moving.

- Try changing the Goal Velocity value as 3000, 10000, 15000, -3000, -10000, -15000, 0, etc… and observe the changes in rotation.
- The relationship between Goal Velocity (#600) values and rpm is as follows:

| Model | Sign of Value | Rotating Direction | RPM (minimum of 20V) |
|---|---|---|---|
| 54 200W | + | CCW | $\frac{Magnitude\ of\ Value}{Gear\ Reduction\ Ratio}$ |
| | - | CW | |
| 54 100W | + | CCW | $\frac{Magnitude\ of\ Value}{Gear\ Reduction\ Ratio}$ |
| | - | CW | |
| 42 20W | + | CCW | $\frac{Magnitude\ of\ Value}{Gear\ Reduction\ Ratio}$ |
| | - | CW | |

**※ Notice 1**

- If Torque Enable(#562) on the Table is set as 1, it means Dynamixel PRO is ready to operate. This is called the **Torque On** state.
- If Torque Enable(#562) on the Table is set as 0, it means Dynamixel PRO is unable to operate. This is called the **Torque Off** state.

**※ Notice 2**

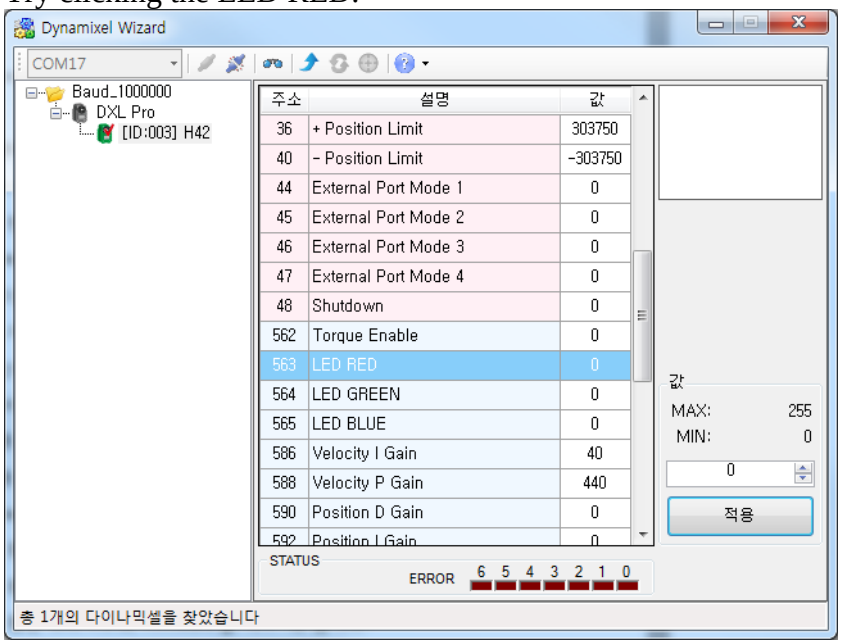
- The Table in the middle of the Dynamixel Wizard is called the Control Table.
- Control Table includes the list of functions that is used to control a Dynamixel
  - Dynamixel PRO movements and settings can be adjusted by inputting the appropriate values on the address.
- For example, on Control Table, the values on Torque Enable (#562) can be changed to 0 or 1 to turn torque. Also, appropriate values can be entered into Goal Velocity (#600) to rotate in desired velocity.

**※ Notice 3**

- The values that are 'locked' on the Control Table when the Torque Mode is on, is referred as the EEPROM domain (non-volatile).
- The values that are written on the EEPROM area are preserved even if the Dynamixel PRO is turned on/off. The value cannot be modified when the torque is on, and the torque needs to be off to modify the EEPROM area values.
- EEPROM area is indicated as pink on Dynamixel Wizard.
- The section that is indicated as blue on the Control Table, such as Goal Velocity, is called the RAM domain (volatile).
- The values on the RAM domain are lost when Dynamixel PRO is turned off. The values can be changed regardless of the torque on/off setting.

iii. LED Control of Dynamixel PRO

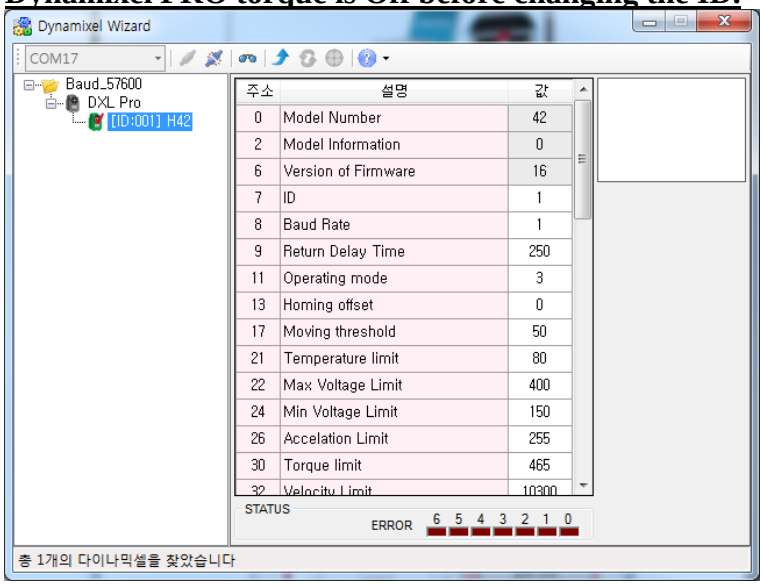
- Unlike other Dynamixels, Dynamixel PRO Dynamixel PRO has 3 color LEDs.
- Therefore, LEDs can be set to emit multiple colors.
- Try clicking the LED RED.



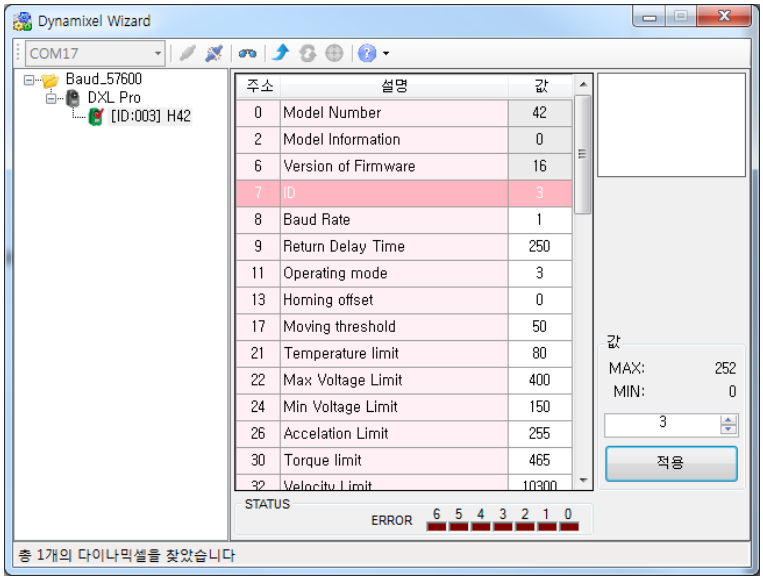

- Input 255 under LED RED value and click ‘Apply’. The red LED should turn on.
- The intensity of the LED emission can be adjusted by changing the values between 0-255.
- Try changing the values of the LED RED, LED GREEN, and LED BLUE to observe the change in colors.

iv. ID configuration

- Individual ID is required for communication when when multiple Dynamixel PROs are linked together.
- For example, a single Dynamixel PRO can be driven via its ID even when 3 Dynamixel PROs are linked together.
- A problem may occur if more than one Dynamixel with the same ID are linked together.
- #7 on the Control Table of the Dynamixel Wizard is the value for ID.
- **ID is a part of the EEPROM area on the Control Table. Be sure to check if Dynamixel PRO torque is Off before changing the ID.**



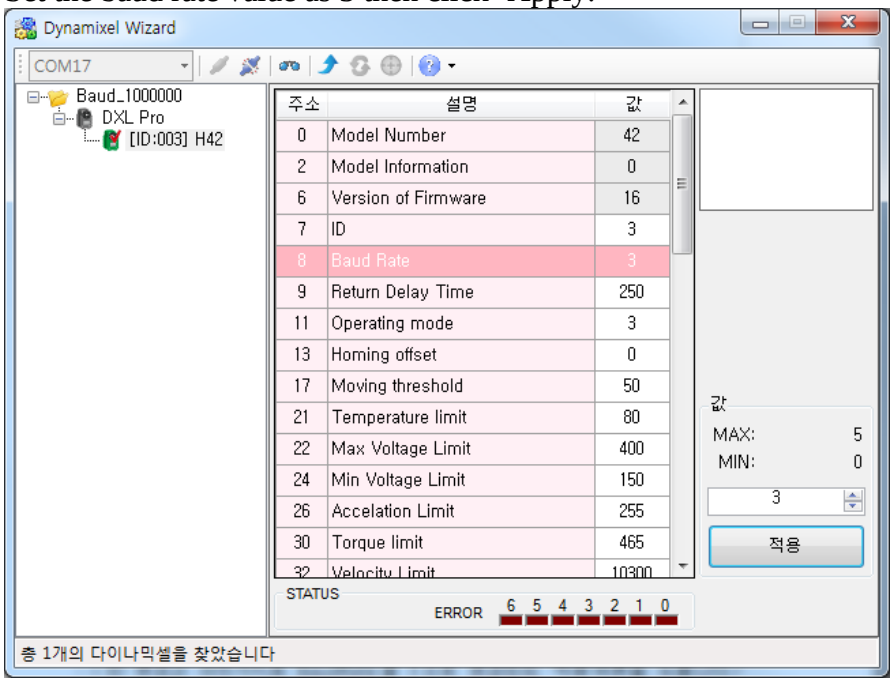

- Once changes are made to ID (#7) value to 3; click on ‘Apply.’ The change of value can be observed on the left table.




- If the Dynamixel PRO ID cannot be changed, make sure the Torque Enable is turned off.

v. Modifying the baud rate.

- #8 on the Control Table of the Dynamixel Wizard is the baud rate.
- **Baud rate is a part of the EEPROM area on the Control Table. Be sure to check if Dynamixel PRO torque is off before changing the baud rate.**
- Set the baud rate value as 3 then click 'Apply.'



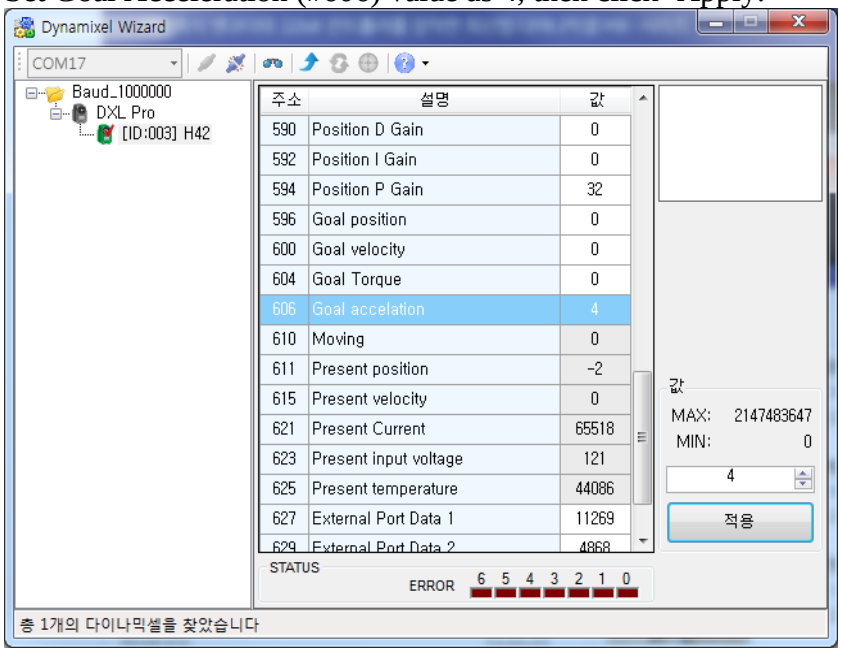

- Notice the change in baud rate to 1Mbps.
- If the baud rate cannot be modified check if Dynamixel PRO Torque Enable is turned off.
- Dynamixel PRO's baud rate and Control Table values are as follows:

| Value of Control Table | Baud Rate(bps: bit for seconds) |
|---|---|
| 0 | 2400 bps |
| 1 | 57600 bps |
| 2 | 115200 bps |
| 3 | 1 Mbps |
| 4 | 2Mbps |
| 5 | 3Mbps |
| 6 | 4Mbps |
| 7 | 4.5Mbps |
| 8 | 10.5Mbps |

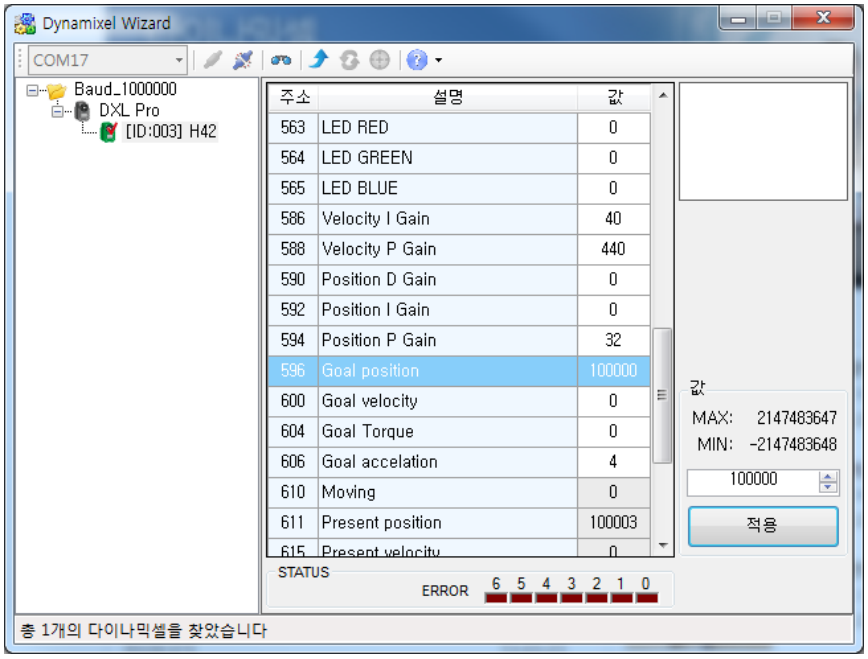
- **※ When using a USB-to-Dynamixwl dongle, Dynamixel PRO's baud rate needs to be ~~less~~ than the USB-to-Dynamixwl's supported baud rate in current use.**

vi. Accelerating Dynamixel PRO in Joint Mode

- Verify if Dynamixel PRO is set as torque off status. If not, click on the Torque Enable (#562) on Control Table and set the value to 0. Click the 'Apply' to change it to torque off status.
- Modify the Operating Mode (#11) value as 3 then click 'Apply' to set Dynamixel PRO as Joint Mode.
- Set Goal Acceleration (#606) value as 4, then click 'Apply.'



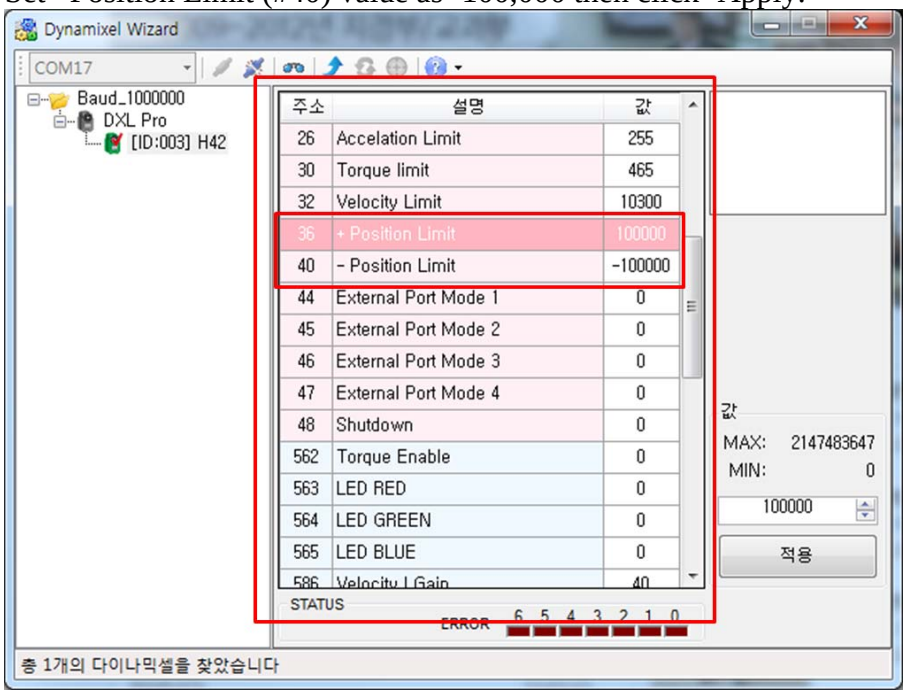

- Set Torque Enable (#562) value as 1 and click 'Apply' to change Dynamixel PRO to 'Torque On' status.
- Set the Goal Position (#596) value to 100,000 or -100,000, then click 'Apply.' Note the difference in the movement of Dynamixel PRO from the time the Goal Acceleration was set to 0.




- If Dynamixel PRO cannot be driven check if Torque Enable is turned on.
- Also, if the Goal Acceleration movement does not differ from the input value, check if Goal Acceleration value is set to 4.
- If the Goal Acceleration value is not zero Dynamixel PRO will create a ‘Trapezoidal Velocity Profile’ (2.2 reference).
- Input random variables into Goal Acceleration and Goal Position to observe the different responses of Dynamixel PRO.

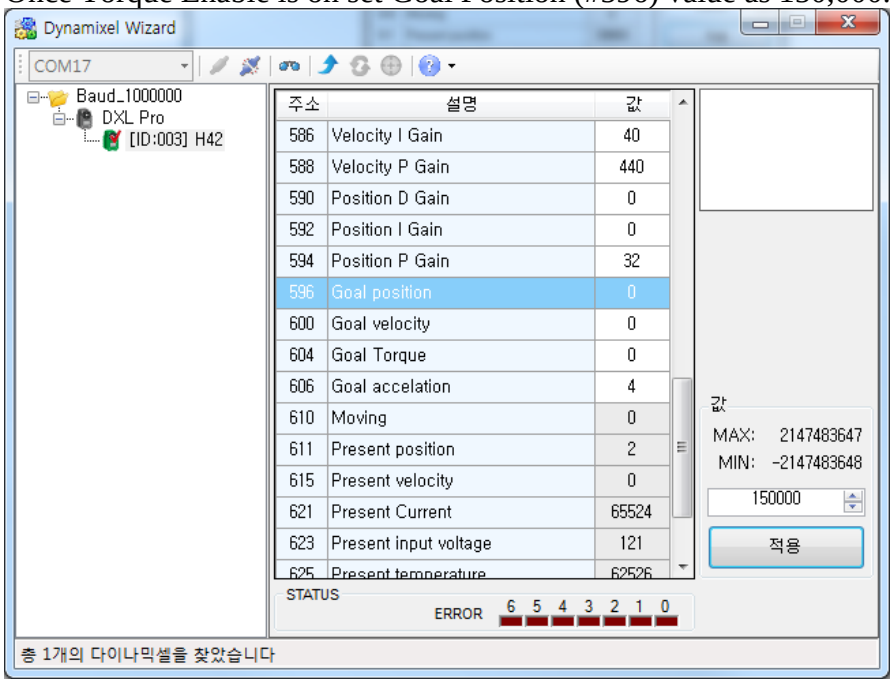
vii. Limiting the range of motion of Dynamixel PRO

- Set Dynamixel PRO to torque off status.
- Set +Position Limit (#36) value as 100,000 then click ‘Apply.’
- Set –Position Limit (#40) value as -100,000 then click ‘Apply.’




- **<u>On the Control Table, +Position Limit (#36) and –Position Limit (#40) are a part of the EEPROM area.</u>**
- **<u>Therefore, if the values cannot be changed check if the Torque Enable is turned On.</u>**
- If +Position Limitd and –Position Limit values can be modified it means Torque Enable is on.

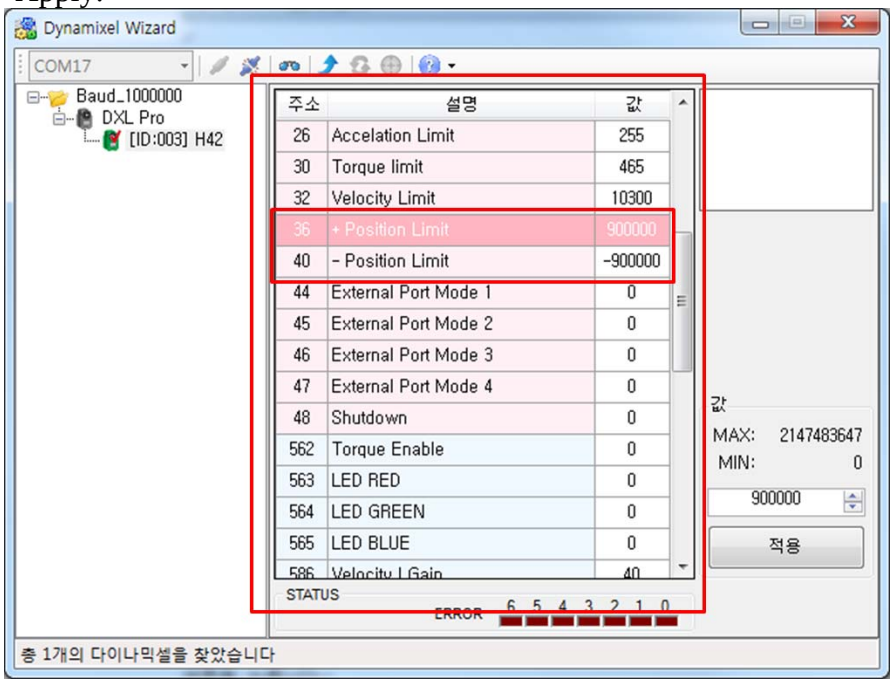
- Once Torque Enable is on set Goal Position (#596) value as 150,000.




- Even though the Dynamixel PRO is in a torque on state, there is no response becauase the Position Limit value exceeds the set limit.
- If Dynamixel PRO is operational, check the +Position Limit value, and if the value is not 100,000, turn torque off. Input the +Position Limit value as 100,000 and click ‘Apply;’ then turn torqueoOn.
- This time, set the Goal Position value as 100,000. Observe movement to position value ‘100,000.’
- Input the Goal Position value as -150,000 (there should be no movement). Try inputting -100,000 instead (Dynamixel PRO will move).
- Input random Goal Position values and click on ‘Apply.’ Position values will move within the “soft” limits.

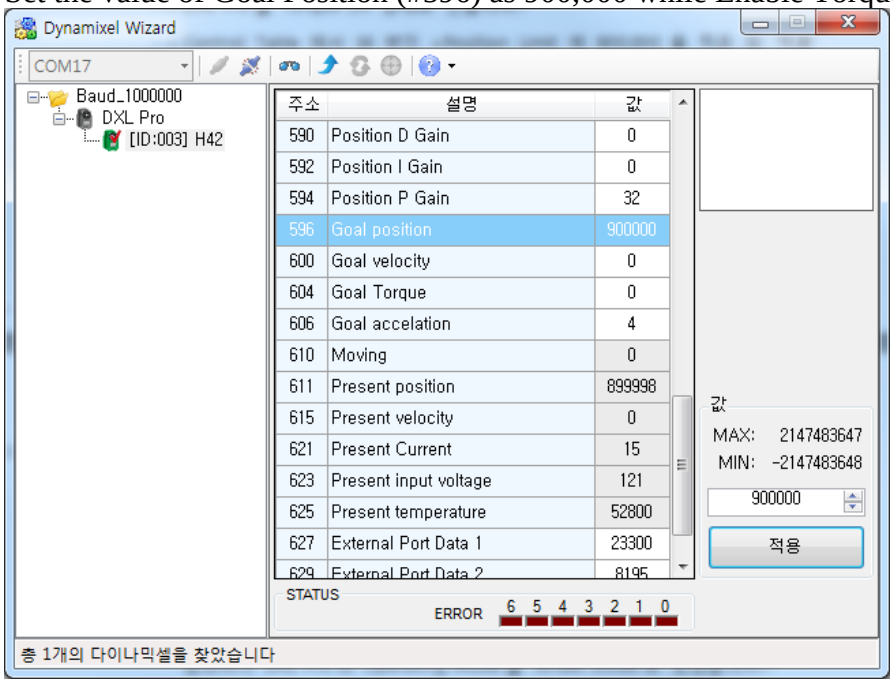
viii. Extending the range of motion of Dynamixel PRO

- +Position Limit and –Position Limit restrict or extend Dynamixel PRO's operation range.
- Turn off Torque Enable by changing the value to 0.
- On the Control Table, set +Position Limit (#36) value as 900,000 then click 'Apply.'
- On the Control Table, set –Position Limit (#40) value as -900,000 then click 'Apply.'




- **<u>+Position Limit(#36) and –Position Limit(#40) on the Control Table are a part of EEPROM area.</u>**
- **<u>If the values cannot be changed, check if Torque Enable is on.</u>**
- Change the +Position and –Position Limit values, and then turn Torque Enable on.

- Set the value of Goal Position (#596) as 900,000 while Enable Torque is on.

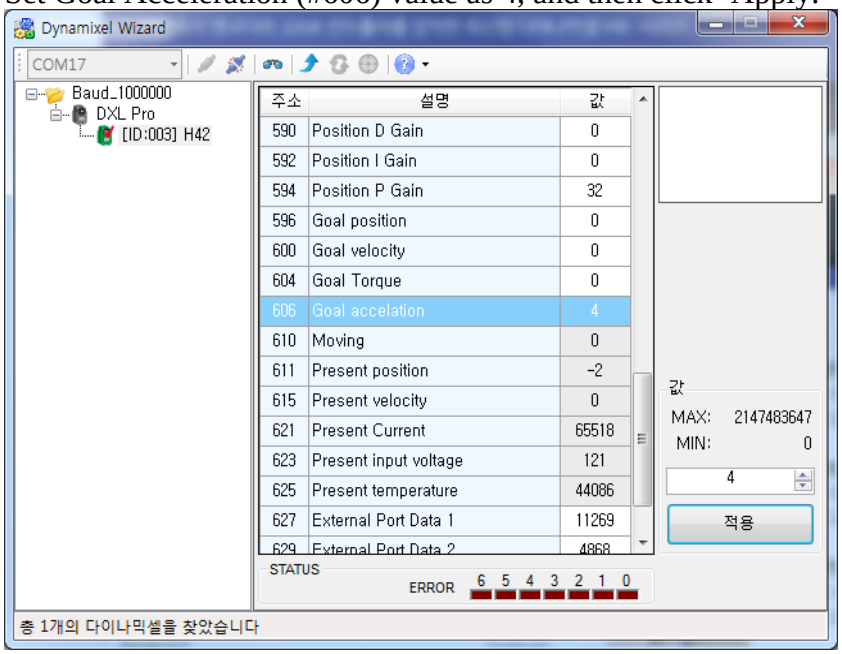



- Dynamixel PRO rotates more than 360 degrees to reach its goal position.
- If Dynamixel PRO is not responsive check the +Position Limit value. If the value is not 900,000 turn Torque Enable off. Change the +Position Limit value to 900,000 and click on ‘Apply.’ Turn Torque Enable back on.
- Set Goal Position as -900,000. Dynamixel PRO will rotate a couple of times in a couter-clockwise direction and stop at the designated position.
- Input random values into the Goal Position and click ‘Apply.’ Verify that Dynamixel PRO reaches the Goal Position.
- Please refer to 1.1.2.i**.** ‘Relationship between angle (degrees) and position value.’

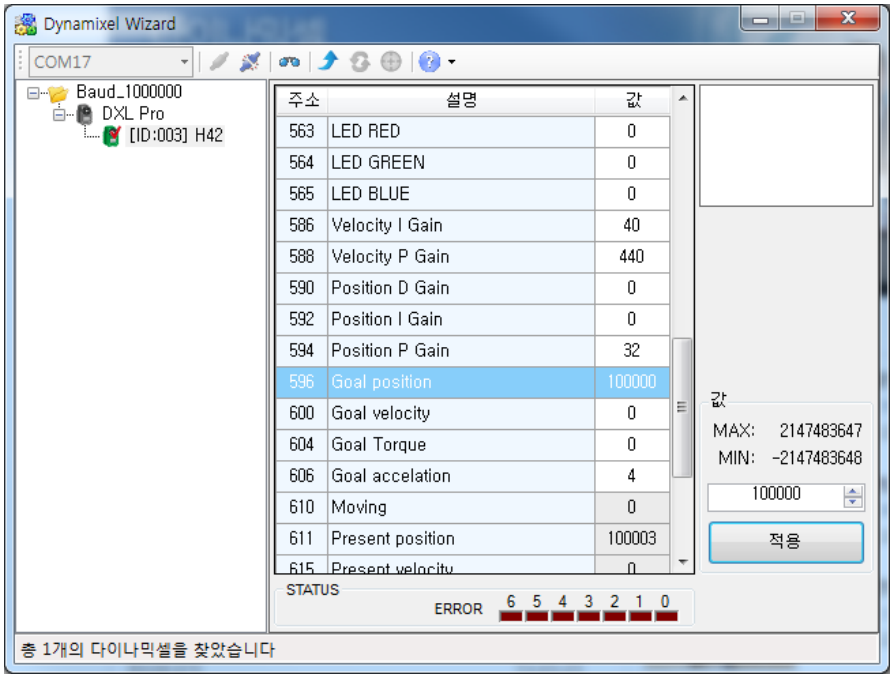
ix. Accelerating Dynamixel PRO in Wheel Mode

x. Refer to '1.1.2. ii 'Operating Dynamixel PRO in Wheel Mode'

- Set Goal Acceleration (#606) value as 4, and then click 'Apply.'




- Set Torque Enable (#562) as 1, and click 'Apply' to set Dynamixel PRO to torque on status.
- Set the Goal Velocity (#600) value to 5,000 or -5,000 and click 'Apply.' Compare the movement to the time when the Goal Acceleration value was set to 0.

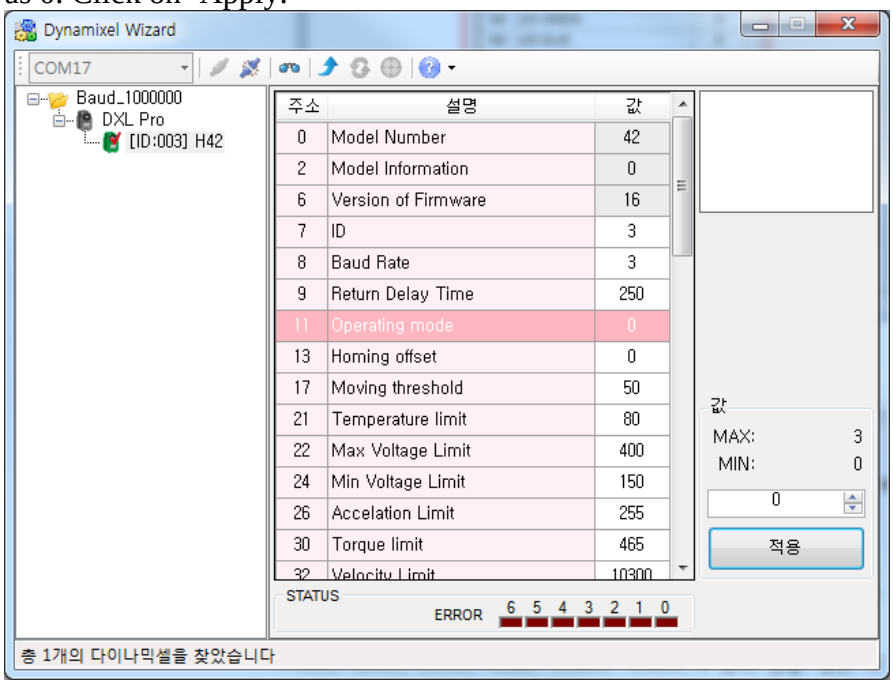



- If Dynamixel PRO does not move check it the Enable Torque value is 1.
- If the movement is no different than when the Goal Acceleration value was 0 check if the Goal Acceleration value is set to 4.

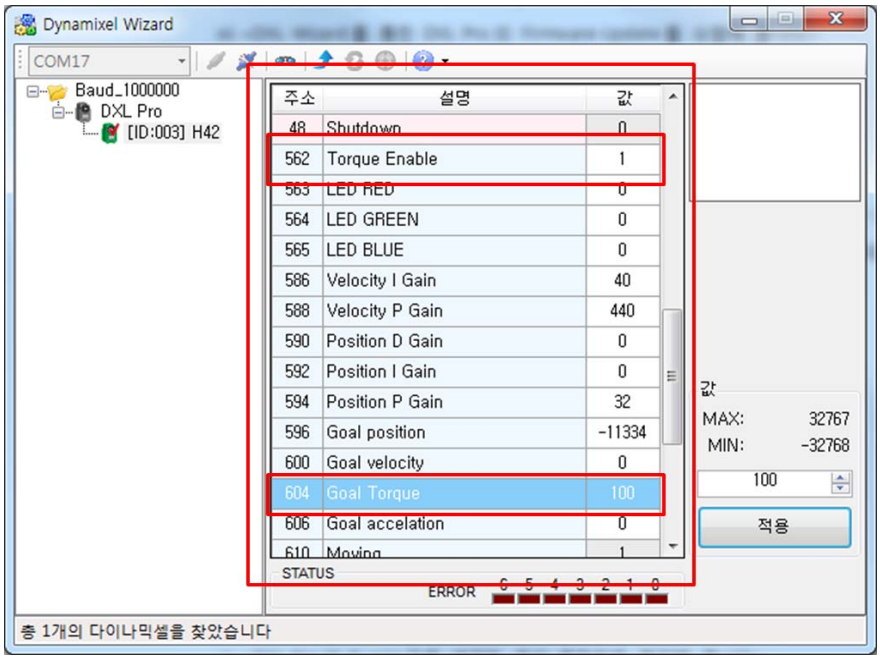
- If the Goal Acceleration value is not zero Dynamixel PRO will create a ‘Trapezoidal Velocity Profile’ (2.2 reference).
- Input random values into Goal Acceleration and Goal Position to observe the different responses of Dynamixel PRO.

xi. Torque Mode of Dynamixel PRO

- Set Dynamixel PRO to torque off condition and input the Operating Mode value as 0. Click on ‘Apply.’




- Enable the torque by turning Torque Enable on.
- Set Goal Torque (#604) value as 100, and click ‘Apply’.

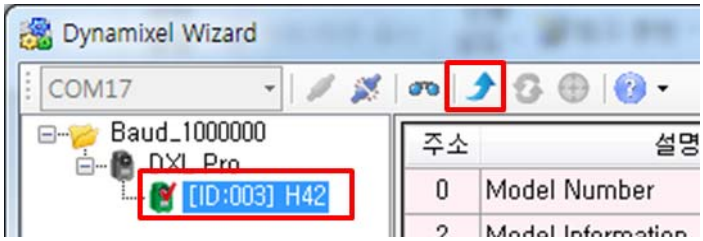



- Note the rotation of the Dynamixel PRO accelerating in the counter-clockwise direction.
- If Dynamixel PRO is not moving check if Dynamixel PRO is in torque off condition.

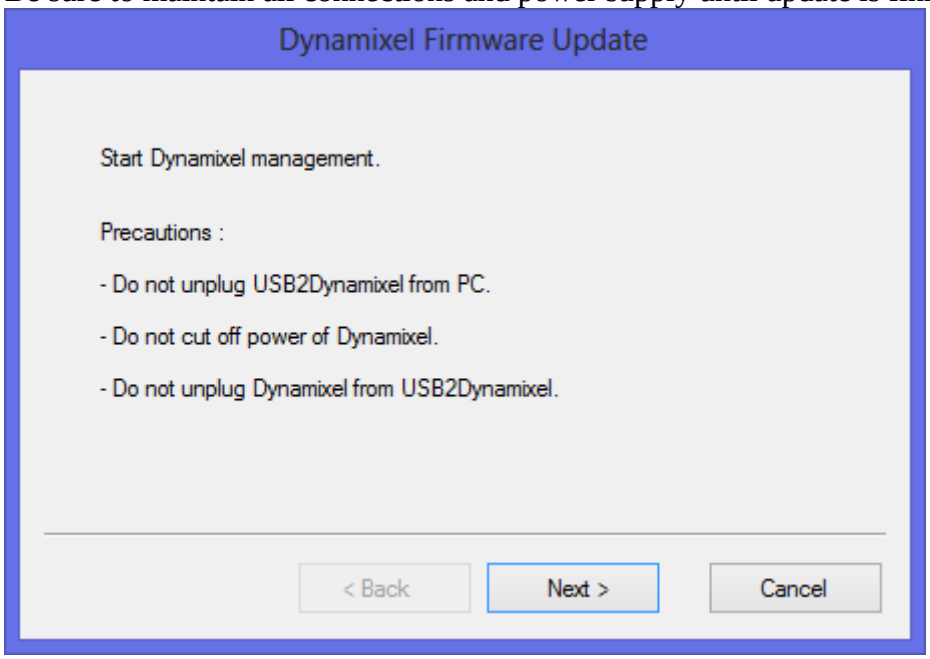
- Set the Goal Torque (#604) as -100 and click on ‘Apply.’
- Note the rotation of Dynamixel PRO accelerating in the clockwise direction.
- Set the Goal Velocity as 0 to stop rotation.
- Try inputing 30, 100, 400, -400, -100, -30, 0, etc… values into Goal Velocity and click on ‘Apply.’ Observe how the Dynamixel PRO responds.
- Goal Torque (#604) value controls the flow of current into Dynamixel PRO.
- The relationship between the Goal Torque value and the current is shown below. Please refer to Dynamixel PRO datasheet to see how current relates to torque.

| Model | Relationship between goal torque and current |
|---|---|
| 54 | $Current\,(mA) = goal\ torque\ vlaue \times \frac{33000}{2048}$ |
| 42 | $Current\,(mA) = goal\ torque\ vlaue \times \frac{30000}{2048}$ |

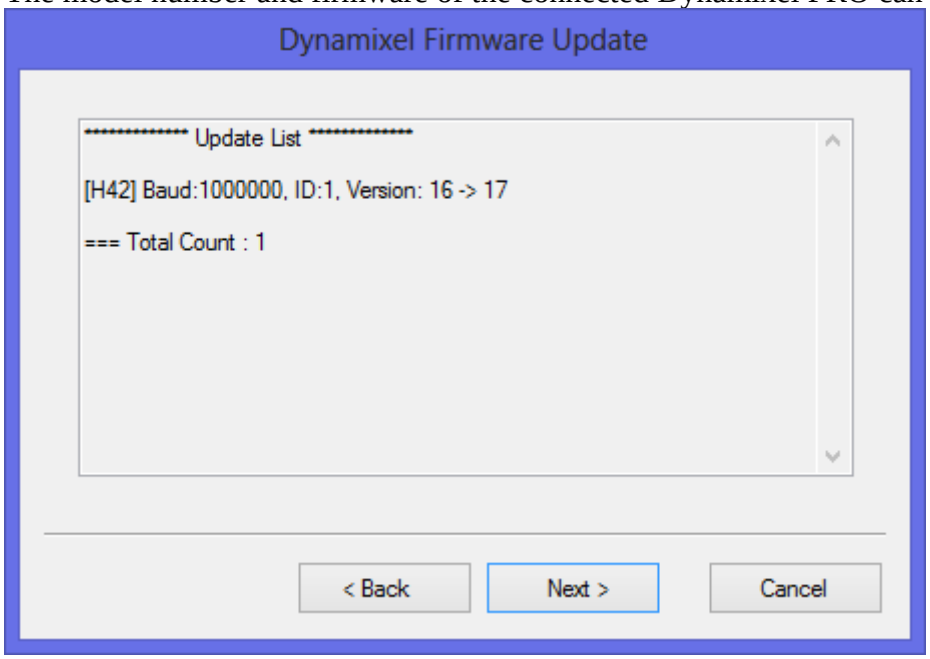
xii. Update Dynamixel PRO's firmware

- Firmware refers to code installed into Dynamixel PRO. This code is responsibe for controlling Dynamixel PRO.
- Dynamixel Wizard connects to the internet and checks for the latest firmware
- When a new firmware for Dynamixel PRO is detected a red checkmark appears on the Dynamixel PRO icon (see illustrations below).
- 

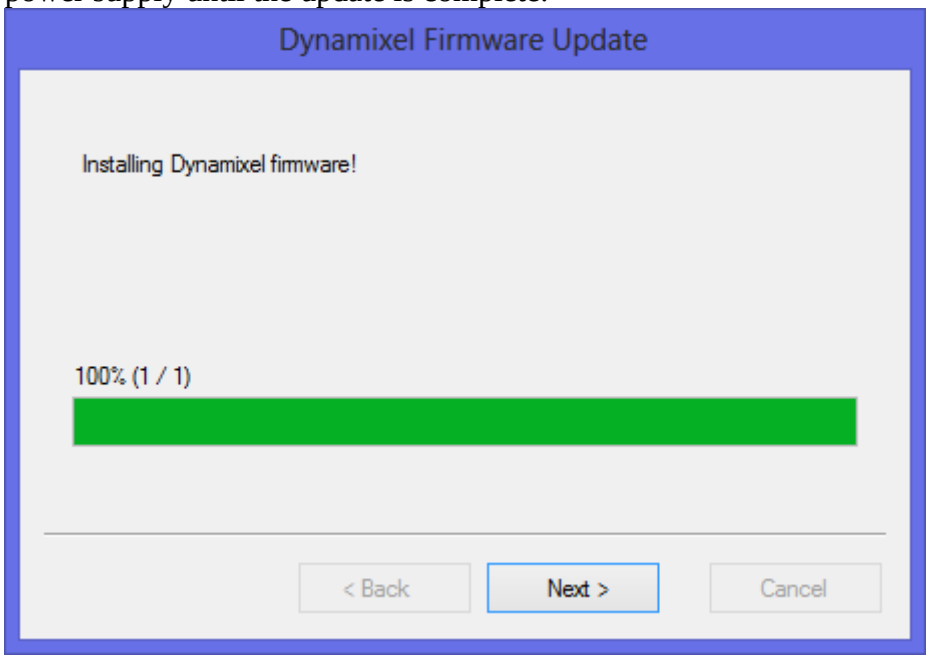
- **Only one Dynamixel PRO must be connected at a time during firmware update.**
- For firmware update select the correct Dynamixel PRO device and click the Firmware update button.
- Begin firmware update by following the instructions.
- Be sure to maintain all connections and power supply until update is finished.



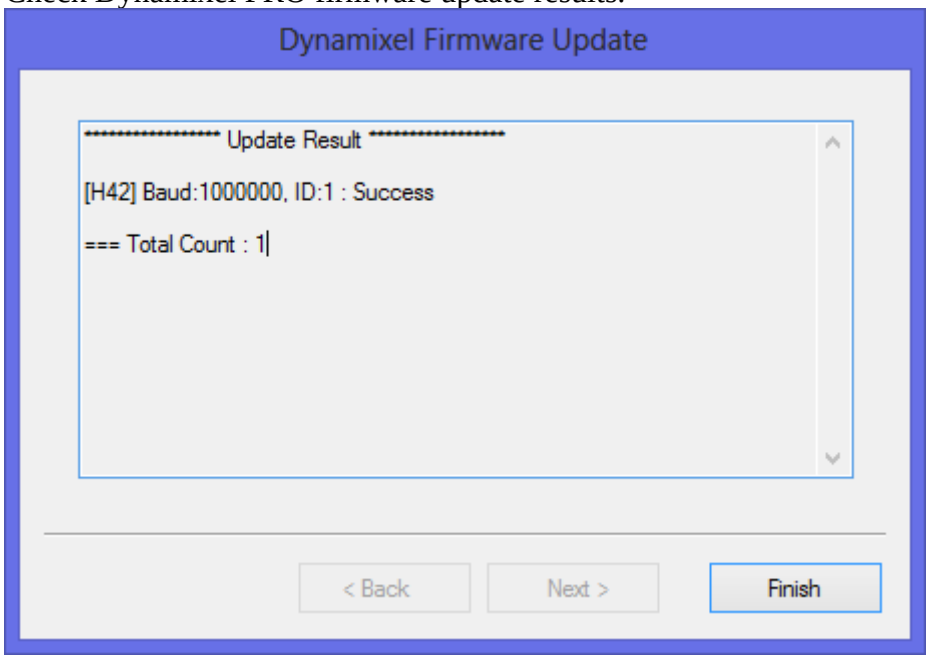
- The model number and firmware of the connected Dynamixel PRO can be verified.




- Click on 'Next' to begin firmware update. Be sure to maintain all connections and power supply until the update is complete.



- Check Dynamixel PRO firmware update results.

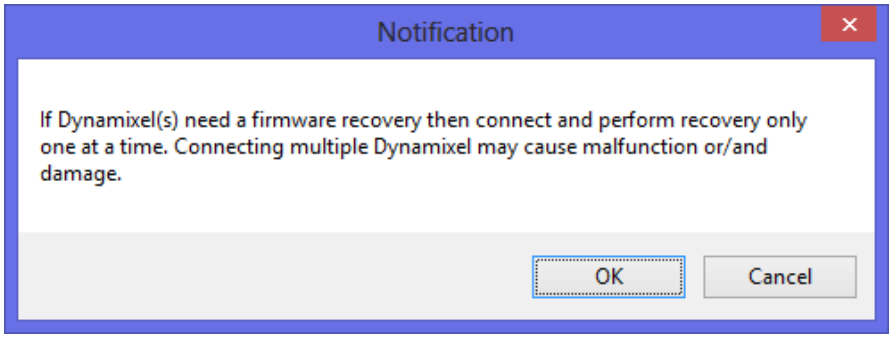


xiii. Dynamixel PRO Firmware Recovery with Dynamixel Wizard.

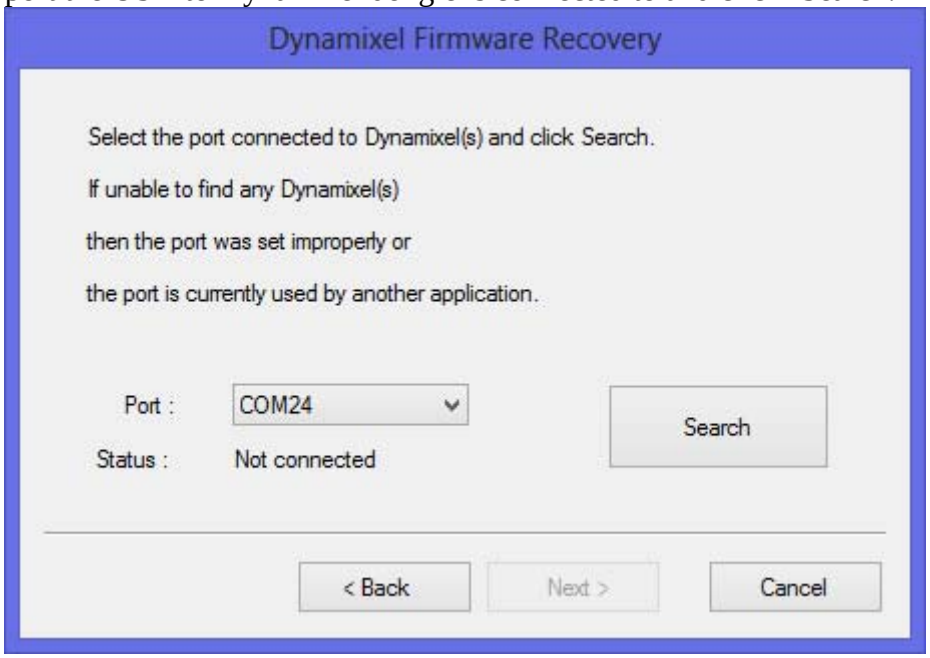
- Dynamixel Wizard can be used to recover the Dynamixel PRO's firmware if there is an issue.
- **※ When firmware is recovered, all the settings are set to default; therefore, the ID and baud rate must be checked. Be sure to have the USB-to-Dynamixel dongle properly set to the appropriate connector (RS-485).**
- **※ When recoverying the firmware, Dynamixel PRO must be connected one at a time. Dynamixel PRO may malfunction if two or more Dynamixel PROs are daisy chained when recoverying the firmware.**
- Begin Dynamixel PRO firmware recovery by clicking on the firmware recovery icon from the toolbar.
- Select thet COM port number of the dongle to begin firmware recovery.


- Please read the instructions and begin Dynamixel PRO firmware recovery.

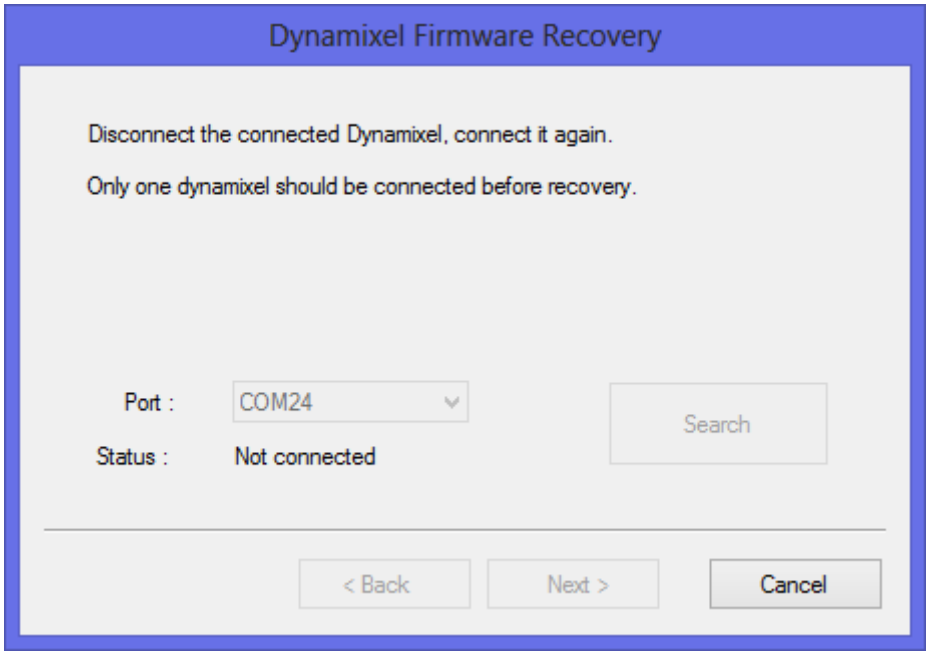

- Only one Dynamixel PRO must be linked to before initiating the Dynamixel firmware recovery.



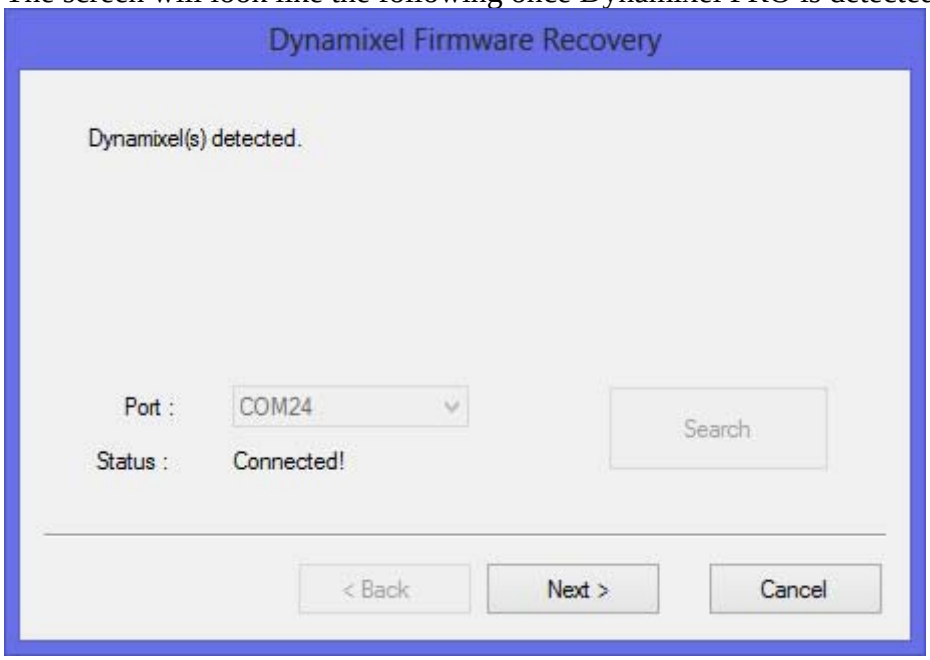

- Dynamixel PRO cannot be searched automatically because the firmware is not recognized. Therefore, you must set the Dynamixel PRO connected port manually. Since Dynamixel PRO cannot be recognized if the port is in use, finish other programs, and then continue the procedure.
- Select USB-to-Dynamixel-connected port and press "Search" button. Select the port the USB-to-Dynamixel dongle is connected to and click ‘Search.’



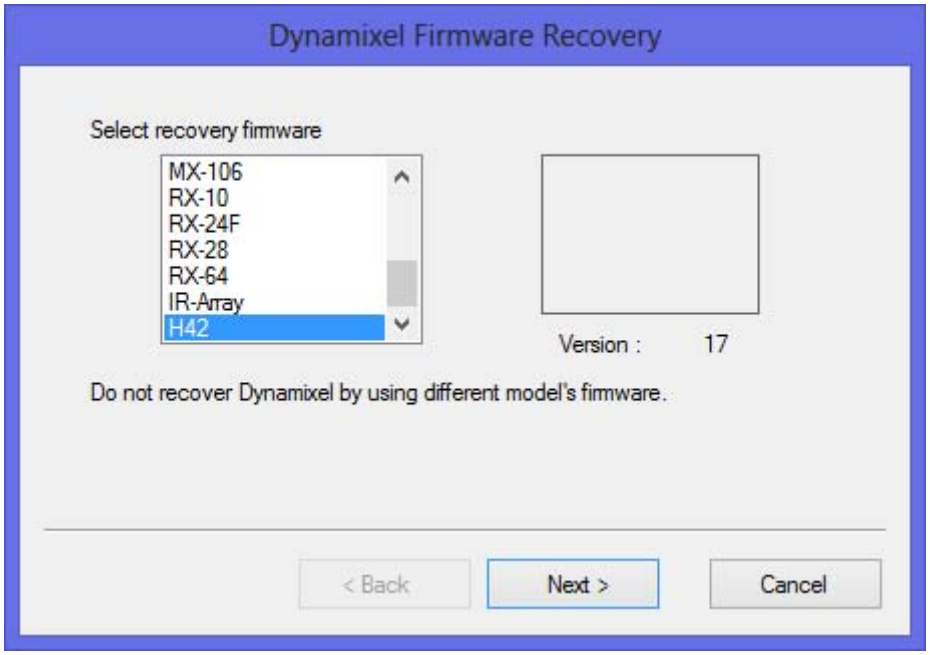

- Turn off Dynamixel PRO then turn it back on to let the program detect Dynamixel PRO.



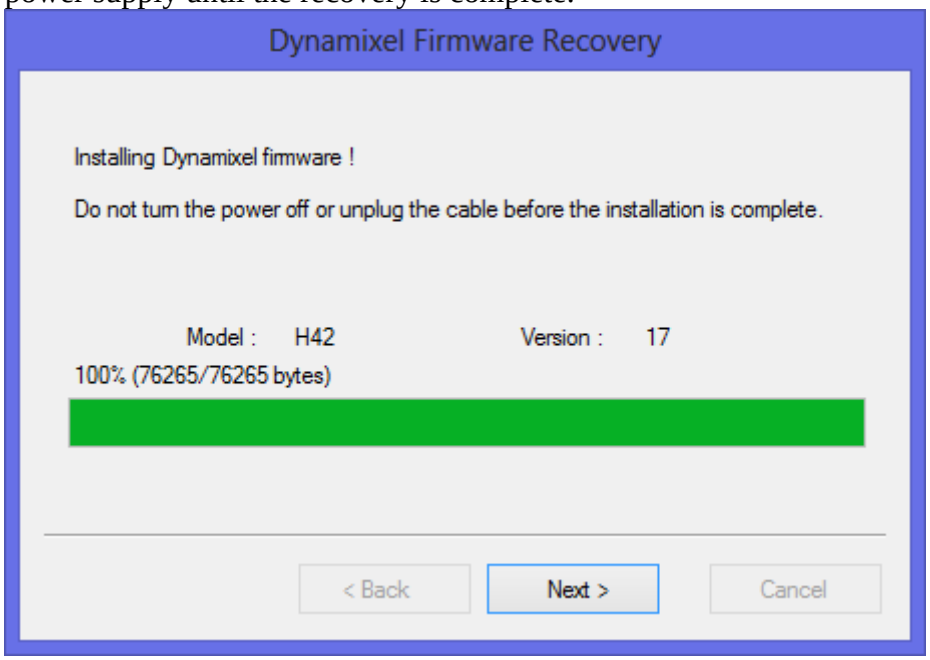

- If the Dynamixel PRO is not detected turn Dynamixel PRO off and on again.
- The screen will look like the following once Dynamixel PRO is detected.



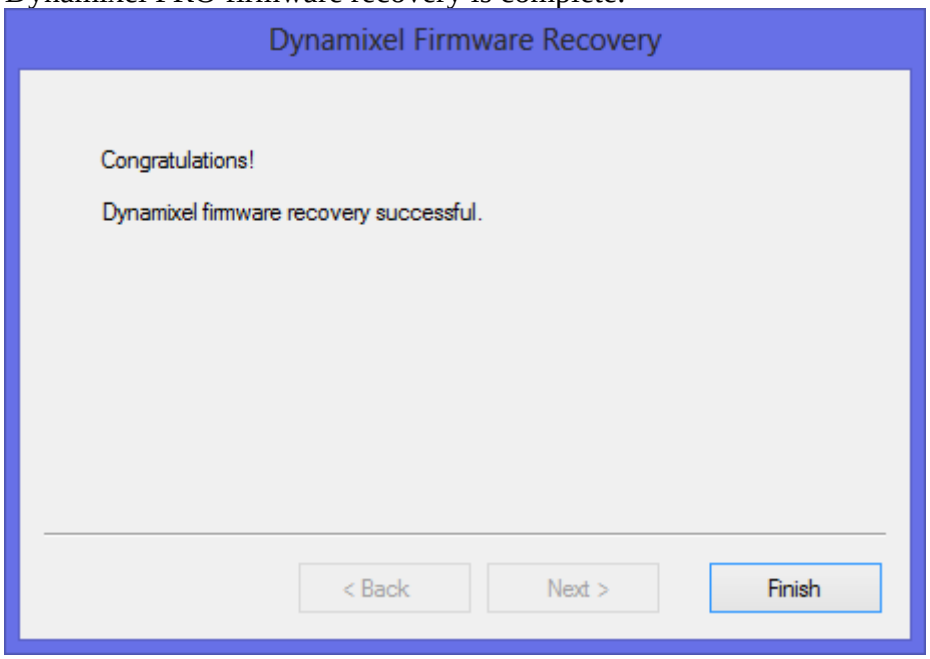

- Once Dynamixel PRO is detected downloadable information will appear. In Dynamixel PRO firmware recovery mode select the correct Dynamixel PRO version. Dynamixel PRO may malfunction if an incorrect firmware is downloaded.




- Click on 'Next' to start firmware recovery. Be sure to maintain all connections and power supply until the recovery is complete.



- Dynamixel PRO firmware recovery is complete.

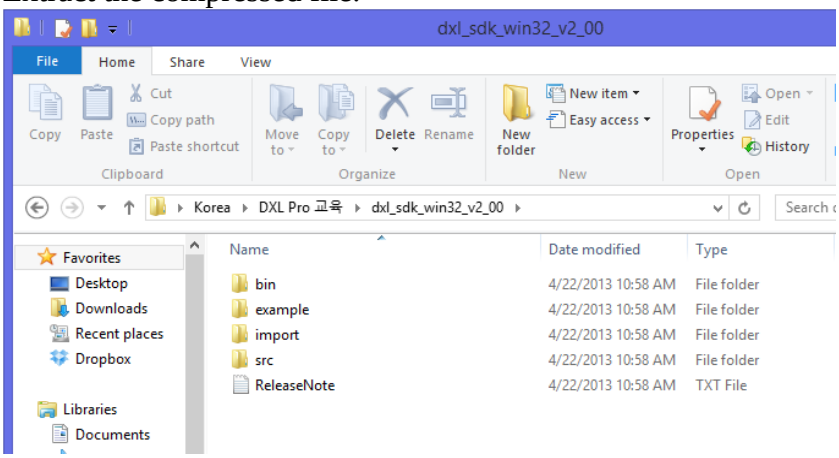


## 1.2 Visual Studio 2010

### 1.2.1 Preparation

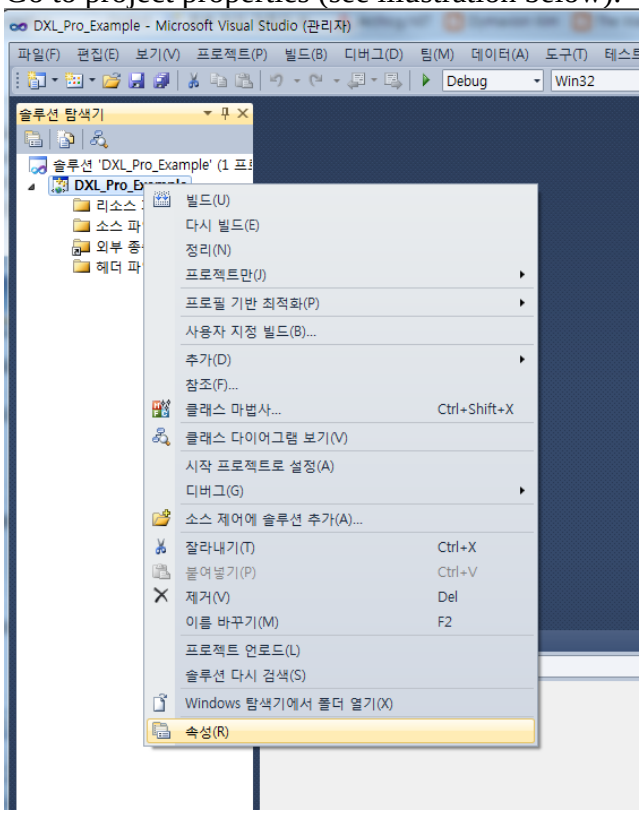
i. Setting the development envrionment.

- For the following tutorial, prepare one Dynamixel PRO with an ID 1 and baud rate of 57600 bps.
- To familiarize the terminology used in the tutorial please go over the previous chapters.
- Find the Dynamixel SDK 2.0 for Windows with the included USB memory dongle.
- Extract the compressed file.

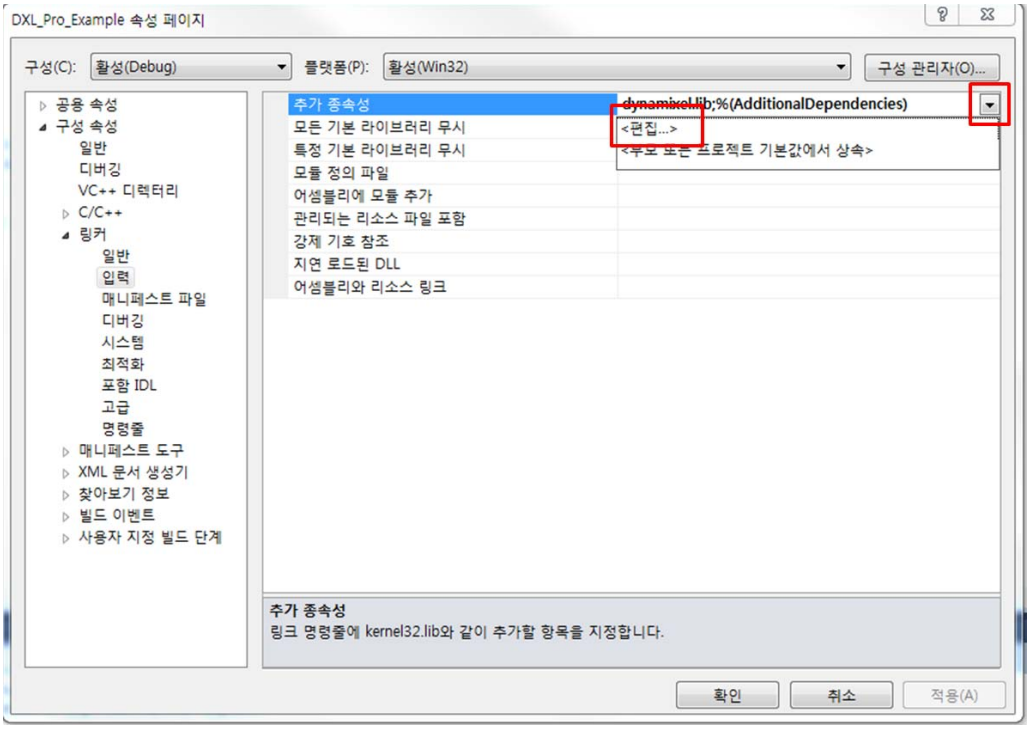



- Dynamixel SDK for Windows contains the following:
  **`/bin`** : DLL for Windows.
  **`/import`** : header and library files are located. (lib, h)
  **`/src`** : Contains DLL source code.
  **`/example`** : examples of various examples commonly used to control Dynamixel PRO.
- Properly set settings to use Dynamixel SDK in Visual Studio 2010.
- Create ‘New Project’ on Visual Studio 2010.
- In the ‘Project Folder’ you made, copy the following files are found inside the extracted SDK file: **`bin/dynamixel.dll`**, **`/import/dynamixel.h`**, **`/import/dynamixel.lib`**

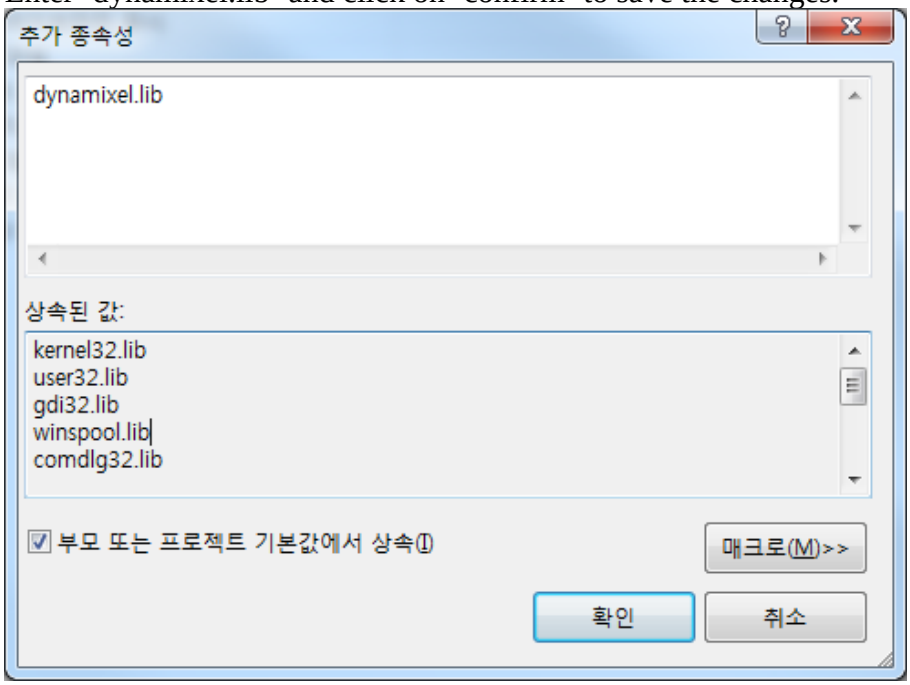
- Go to project properties (see illustration below).




- On ‘Properties’, click ‘Linker -> Input.
- Click on the ‘Additional Dependancies’, then select ‘Edit.’



- Enter ‘dynamixel.lib’ and click on ‘confirm’ to save the changes.




- Dynamixel SDK programming setup is complete.

ii. Inialize and terminate the connnection with USB-to-Dynamixel dongle

- First, open a .cpp file and copy the code below. The **COM_PORT_NUM** and **BAUD_RATE_NUM** values need to match the proper USB-to-Dynamixel dongle in use.

```
#include "dynamixel.h"

#define    COM_PORT_NUM      17          //Comport Number of USB2DXL
#define    BAUD_RATE_NUM      1          //Baudrate Number of Dynamixel PRO
```

- The relationship between the baud rate number and the baud rate is indicated below.

| Baudrate Number | Bps(= bit per seconds) |
|---|---|
| 0 | 2400 bps |
| 1 | 57600 bps |
| 2 | 115200 bps |
| 3 | 1 Mbps |
| 4 | 2 Mbps |
| 5 | 3 Mbps |
| 6 | 4 Mbps |
| 7 | 4.5 Mbps |
| 8 | 10.5 Mbps |

- Declare the ‘SerialPort’ type variable and then initialize all the values to zero.
- Declare the ‘SerialPort’ pointer type variable, then initialize it with addresss of the predefined variable. Please refer to the source code shown below.

```
SerialPort sp = {0,0,0,0,0};
SerialPort *Port = &sp;
```

- Next, implement functions, **dxl_initialize** and **dxl_terminate**, to connect and disconnect to USB-to-Dynamixel dongle.
- The program’s entire source is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM      17          //Comport Number of USB2DXL
#define    BAUD_RATE_NUM      1          //Baudrate Number of Dynamixel PRO. 1 is 57600 bps
int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;

        //Open the port of USB2DXL
```

```
        if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
                printf("Succeed to open USB2Dynamixel!\n");
        else
        {
                printf( "Failed to open USB2Dynamixel!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        printf( "Press any key to terminate...\n" );
        getch();

        //Close the port of USB2DXL
        dxl_terminate(Port);

        return 0;
}
```

- **COMM_RXSUCCESS** is returned when USB-to-Dynamixel dongle ~~is~~ successfully communicates with Dynamixel PRO.
- When using the Dynamixel SDK to communicate with USB-to-Dynamixel you can check if communication is successful.

### 1.2.2 Basic functions of Dynamixel PRO.

i. Turning the torque On/Off of Dynamixel PRO.

- As explained in 1.1, Dynamixel PRO's Control Table contains Torque Enable function that controls the torque to be on/off. The address for Torque Enable function is 562. Torque Enable ~~is~~ has 1 byte assigned.
- Therefore, implement **`dxl_write_byte`** function (See 2.4.3)
- The program's entire source is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define   COM_PORT_NUM      17          //Comport Number of USB2DXL
#define   BAUD_RATE_NUM      1          //Baudrate Number of Dynamixel PRO. 1 is 57600 bps

#define P_TORQUE_ENABLE     562         //Address of Torque Enable in Control Table
#define ID                          1         //ID of Dynamixel PRO you use

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;

        //Open the port of USB2DXL
        if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
                printf("Succeed to open USB2Dynamixel!\n");
        else
        {
                printf( "Failed to open USB2Dynamixel!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        int ErrorStatus;
        int Result;

        //Torque ON
        printf( "Press any key to turn on the torque...\n" );
        _getch();
        Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);

        //Torque OFF
        printf( "Press any key to turn off the torque...\n" );
        _getch();
        Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
```

```
        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);

        return 0;
}
```

- The functions, such as dxl_initialize, dxl_write_byte, etc…, are used to communicate with Dynamixel PRO. These functions return communication results.
- **COMM_RXSUCCESS** is returned for successful communications.

ii. Operating Dynamixel PRO using C programming language.

- As explained in 1.1, Dynamixel PRO's Control Table's Goal Position address is 562. Also, Goal Position is assigned 4 bytes of memory.
- Therefore, a function called **`dxl_write_dword`** function controls Goal position. (See 2.4.3)
- The program's entire source is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM       17            //Comport Number of USB2DXL
#define    BAUD_RATE_NUM       1            //Baudrate Number of Dynamixel PRO. 1 is 57600 bps

#define P_TORQUE_ENABLE     562          //Address of Torque Enable in Control Table
#define P_GOAL_POSITION     596          //Address of Goal Position in Control Table
#define ID                          1          //ID of Dynamixel PRO you use

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;

        //Open the port of USB2DXL
        if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
                printf("Succeed to open USB2Dynamixel!\n");
        else
        {
                printf( "Failed to open USB2Dynamixel!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        int Result, ErrorStatus;

        //Torque ON
        printf( "Torque On...\n" );
        Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
```

```
	int Goal_Pos1 = 100000, Goal_Pos2 = -100000;

	//Change the vlaue of goal position
	printf("Press any key to roatate the Dynamixel PRO to position 1\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, Goal_Pos1, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( " Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//Change the vlaue of goal position
	printf("Press any key to roatate the Dynamixel PRO to position 2\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, Goal_Pos2, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( " Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//Torque OFF
	printf( "Press any key to turn off the torque...\n" );
	_getch();
	Result = dxl_write_dword(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( " Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//Close the port of USB2DXL
	printf( "Press any key to terminate...\n" );
	_getch();
	dxl_terminate(Port);

	return 0;
}
```

iii. Modifying Dynamixel PRO's ID using C programming language.

- As explained in 1.1, the address for the ID is 7 on the Control Table. ID can be modified with the function called **`dxl_write_byte`**.
- However, ID cannot be changed while the Torque Enable is on, so the torque must be turned off in order to change its value.
- The program's entire source is shown below.

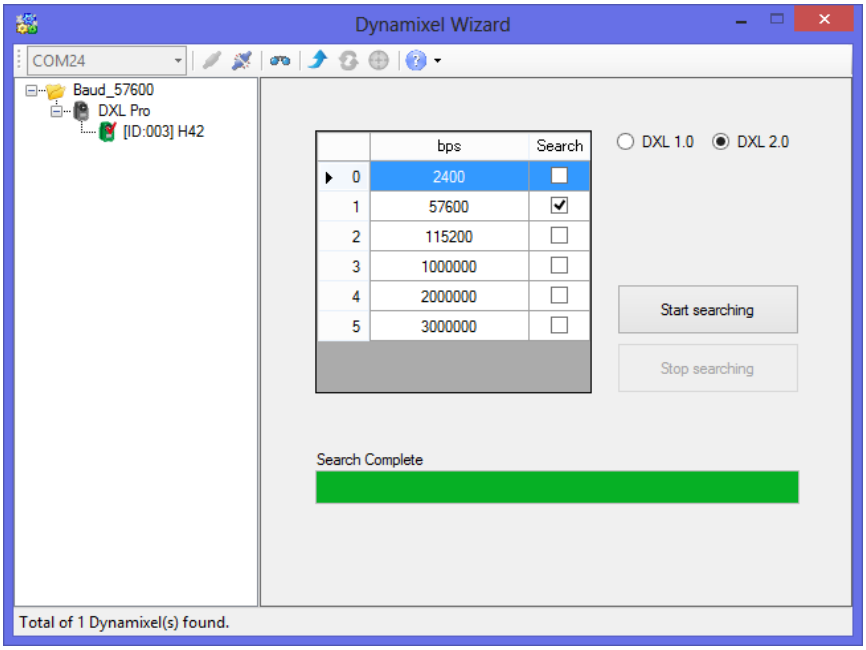
main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM          17          //Comport Number of USB2DXL
#define    BAUD_RATE_NUM          1          //Baudrate Number of Dynamixel PRO

#define P_ID                     7          //Address of ID in Contorl Table
#define P_TORQUE_ENABLE        562          //Address of Torque Enable in Control Table

#define ID                       1          //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
    if(ErrorCode & ERRBIT_VOLTAGE)
        printf("Input voltage error!\n");

    if(ErrorCode & ERRBIT_ANGLE)
        printf("Angle limit error!\n");

    if(ErrorCode & ERRBIT_OVERHEAT)
        printf("Overheat error!\n");

    if(ErrorCode & ERRBIT_RANGE)
        printf("Out of range error!\n");

    if(ErrorCode & ERRBIT_CHECKSUM)
        printf("Checksum error!\n");

    if(ErrorCode & ERRBIT_OVERLOAD)
        printf("Overload error!\n");

    if(ErrorCode & ERRBIT_INSTRUCTION)
        printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
```

```
SerialPort *Port = &sp;

//Open the port of USB2DXL
if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
	printf("Succeed to open USB2Dynamixel!\n");
else
{
	printf( "Failed to open USB2Dynamixel!\n" );
	printf( "Press any key to terminate...\n" );
	_getch();
	return 0;
}

int Result, ErrorStatus;

//Torque Off
printf( "Torque Off...\n" );
Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
	printf( "Failed to write!\n" );
	printf( "Press any key to terminate...\n" );
	_getch();
	return 0;
}


int desired_ID = 3;

//Change the ID of Dynamixel PRO you use
printf("Press any key to change the ID of Dynamixel PRO you use\n");
_getch();
Result = dxl_write_byte(Port, ID, P_ID, desired_ID, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
	printf( "Failed to write!\n" );
	printf( "Press any key to terminate...\n" );
	_getch();
	return 0;
}
else
{
	if(ErrorStatus != 0 )
		PrintErrorCode(ErrorStatus);
	else
		printf("Succeed to chagne the ID of Dynamixel PRO you use!\n");
}

//Close the port of USB2DXL
printf( "Press any key to terminate...\n" );
```

```
        _getch();
        dxl_terminate(Port);

        return 0;
}
```

- The return pointer of the read/write functions returns the error/status code sent from Dynamixel PRO.
- Please refer to 2.4.2 for detailed information regarding error values.
- Check the ID that is modified using Dynamixel Wizard.




iv. Modifying the baud rate of Dynamixel PRO.

- Please use Dynamixel Wizard to change Dynamixel PRO ID to 1.
- As explained in 1.1, the baud rate of Dynamixel PRO on the Control Table is in address #8. Also, the baud rate is assigned with 1 byte so it can be modified by implementing **`dxl_write_byte`** function.
- The baud rate cannot be changed while Dynamixel PRO is in torque on status. Torque must be turned off in order to change its value.
- The program's entire source is shown below.

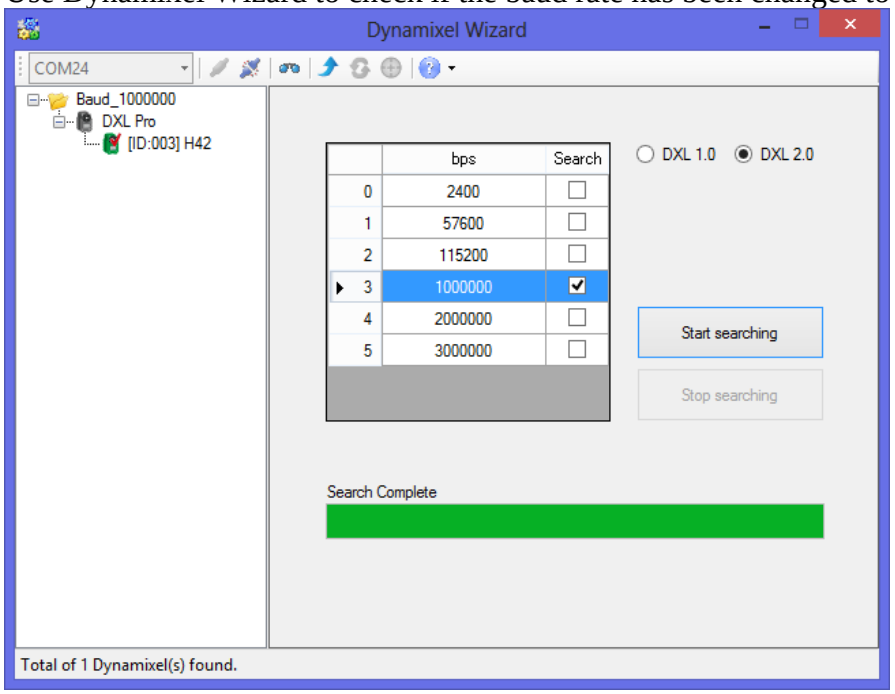
main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define   COM_PORT_NUM        17        //Comport Number of USB2DXL
#define   BAUD_RATE_NUM        1        //Baudrate Number of Dynamixel PRO

#define   P_BAUD_RATE          8        //Address of ID in Contorl Table
#define   P_TORQUE_ENABLE     562        //Address of Torque Enable in Control Table
```

```
#define ID                      1       //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
    if(ErrorCode & ERRBIT_VOLTAGE)
        printf("Input voltage error!\n");

    if(ErrorCode & ERRBIT_ANGLE)
        printf("Angle limit error!\n");

    if(ErrorCode & ERRBIT_OVERHEAT)
        printf("Overheat error!\n");

    if(ErrorCode & ERRBIT_RANGE)
        printf("Out of range error!\n");

    if(ErrorCode & ERRBIT_CHECKSUM)
        printf("Checksum error!\n");

    if(ErrorCode & ERRBIT_OVERLOAD)
        printf("Overload error!\n");

    if(ErrorCode & ERRBIT_INSTRUCTION)
        printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;

        //Open the port of USB2DXL
        if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
                printf("Succeed to open USB2Dynamixel!\n");
        else
        {
                printf( "Failed to open USB2Dynamixel!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        int Result, ErrorStatus;

        //Torque Off
        printf( "Torque Off...\n" );
        Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
```

```
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}


	int desired_Baudrate = 3; //1 Mbps

	//Change the ID of Dynamixel PRO you use
	printf("Press any key to change the baudrate of Dynamixel PRO you use\n");
	_getch();
	Result = dxl_write_byte(Port, ID, P_BAUD_RATE, 3, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
		else
			printf("Succeed to chage the baudrate of Dynamixel PRO you use!\n");
	}

	//Close the port of USB2DXL
	printf( "Press any key to terminate...\n" );
	_getch();
	dxl_terminate(Port);

	return 0;
}
```

- Use Dynamixel Wizard to check if the baud rate has been changed to 1 Mbps.




- If the baud rate has not been changed check if Dynamixel PRO is in torque off status.
- The following content assumes Dynamixel PRO's baud rate set to 1 Mbps.

v. LED control of Dynamixel PRO

- The baud rate is now 1 Mbps therefore the code must be implemented as follows:

```
#define   COM_PORT_NUM        17        //Comport Number of USB2DXL
#define   BAUD_RATE_NUM        3        //Baudrate Number of Dynamixel PRO
```

- As explained in 1.1, Dynamixel PRO Control Table contrains 3 addresses for the LEDs, each assigned with 1 byte of memory: LED_RED (#563), LED_GREEN (#564), and LED_BLUE (#565). Implement **`dxl_write_byte`** function to control its values.
- Since LED is assigned with 1 byte of memory select values between 0~255 to control the brightness of the LED.
- The program's entire source is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define   COM_PORT_NUM        17        //Comport Number of USB2DXL
#define   BAUD_RATE_NUM        3        //Baudrate Number of Dynamixel PRO

#define   P_LED_RED          563        //Address of LED_RED in Contorl Table
#define   P_LED_GREEN        564        //Address of LED_RED in Contorl Table
#define   P_LED_BLUE         565        //Address of LED_RED in Contorl Table

#define ID                    1         //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
    if(ErrorCode & ERRBIT_VOLTAGE)
        printf("Input voltage error!\n");

    if(ErrorCode & ERRBIT_ANGLE)
        printf("Angle limit error!\n");

    if(ErrorCode & ERRBIT_OVERHEAT)
        printf("Overheat error!\n");

    if(ErrorCode & ERRBIT_RANGE)
        printf("Out of range error!\n");

    if(ErrorCode & ERRBIT_CHECKSUM)
        printf("Checksum error!\n");

    if(ErrorCode & ERRBIT_OVERLOAD)
        printf("Overload error!\n");
```

```
        if(ErrorCode & ERRBIT_INSTRUCTION)
                printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;

        //Open the port of USB2DXL
        if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
                printf("Succeed to open USB2Dynamixel!\n");
        else
        {
                printf( "Failed to open USB2Dynamixel!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        int Result, ErrorStatus;

        //Turn off the LED
        Result = dxl_write_byte(Port, ID, P_LED_RED,      0, &ErrorStatus);
        Result = dxl_write_byte(Port, ID, P_LED_GREEN, 0, &ErrorStatus);
        Result = dxl_write_byte(Port, ID, P_LED_BLUE,    0, &ErrorStatus);

        //Turn on and change the color of LED in Dynamixel PRO
        printf("Press any key to change the color of LED in Dynamixel PRO\n");
        _getch();
        Result = dxl_write_byte(Port, ID, P_LED_RED, 255, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
        else
        {
                if(ErrorStatus != 0 )
                        PrintErrorCode(ErrorStatus);
                else
                        printf("Succeed to change the color of LED in Dynamixel PRO!\n");
        }

        printf("Press any key to change the color of LED in Dynamixel PRO\n");
        _getch();
        Result = dxl_write_byte(Port, ID, P_LED_RED,          0, &ErrorStatus);
        Result = dxl_write_byte(Port, ID, P_LED_GREEN, 255, &ErrorStatus);
```

```
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
        else
        {
                if(ErrorStatus != 0 )
                        PrintErrorCode(ErrorStatus);
                else
                        printf("Succeed to change the color of LED in Dynamixel PRO!\n");
        }

        printf("Press any key to change the color of LED in Dynamixel PRO\n");
        _getch();
        Result = dxl_write_byte(Port, ID, P_LED_GREEN,    0, &ErrorStatus);
        Result = dxl_write_byte(Port, ID, P_LED_BLUE, 255, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
        else
        {
                if(ErrorStatus != 0 )
                        PrintErrorCode(ErrorStatus);
                else
                        printf("Succeed to change the color of LED in Dynamixel PRO!\n");
        }

        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```

- Try controlling the intensity of the LED light.

vi. Modifying the P gain value of Dynamixel PRO

- Let's control the P gain value of Dynamixel PRO's position.
- Dynamixel PRO implements PID control to maniputate its motor; therefore, the movement changes cooresponding to the gain values.
- On Dynamixel PRO's Control Table, the Position_P_Gain address is set as 594 and it utilizes 2 bytes (1 word) of memory.
- Thus, **`dxl_write_word`** function, which modifies 2 bytes of memory, changes the P gain value.
- The program's entire source is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define   COM_PORT_NUM            17        //Comport Number of USB2DXL
#define   BAUD_RATE_NUM           3         //Baudrate Number of Dynamixel PRO
#define   P_TORQUE_ENABLE         562       //Address of Torque Enable in Control Table
#define   P_POSITION_P_GAIN       594       //Address of Position P Gain in Contorl Table
#define   P_GOAL_POSITION         596       //Address of Goal Position in Contorl Table

#define ID                        1         //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
    if(ErrorCode & ERRBIT_VOLTAGE)
        printf("Input voltage error!\n");

    if(ErrorCode & ERRBIT_ANGLE)
        printf("Angle limit error!\n");

    if(ErrorCode & ERRBIT_OVERHEAT)
        printf("Overheat error!\n");

    if(ErrorCode & ERRBIT_RANGE)
        printf("Out of range error!\n");

    if(ErrorCode & ERRBIT_CHECKSUM)
        printf("Checksum error!\n");

    if(ErrorCode & ERRBIT_OVERLOAD)
        printf("Overload error!\n");

    if(ErrorCode & ERRBIT_INSTRUCTION)
        printf("Instruction code error!\n");
}
```

```
int main(void)
{
	SerialPort sp = {0,0,0,0,0};
	SerialPort *Port = &sp;

	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result, ErrorStatus;

	//Torque on
	printf( "Torque on...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//change the position p gain
	printf("Press any key to change the position p gain\n");
	_getch();
	Result = dxl_write_word(Port, ID, P_POSITION_P_GAIN, 8, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
	}

	//change the goal position value
	printf("Press any key to change the goal position\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, 100000, &ErrorStatus);
```

```
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
	}

	//change the position p gain
	printf("Press any key to change the position p gain\n");
	_getch();
	Result = dxl_write_word(Port, ID, P_POSITION_P_GAIN, 256, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
	}

	//change the goal position value
	printf("Press any key to change the goal position\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, -100000, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
	}

	//Close the port of USB2DXL
	printf( "Press any key to terminate...\n" );
	_getch();
```

```
        dxl_terminate(Port);
        return 0;
}
```

- Observe the diffenence in Dynamixel PRO's movement speed relevant to the change in P Gain values.

vii. Operating Dynamixel PRO in various speeds

- To control the movement speed in Joint Mode change the Goal Velocity.
- There are two methods in changing the Goal Position and Goal Velocity of Dynamixel PRO.
- First, implement **dxl_write_dword** function twice to change the values. Second, implement **dxl_write** function to simultaneously change the Goal Position and Goal Velocity (see 2.44).
- Since we have gone over **dxl_write_byte**, **dxl_write_word**, and **dxl_write_dword**, please try the second method to change the Goal Position and Goal Velocity of Dynamixel PRO.
- Goal Position and Goal Velocity both use 2 bytes each (or 1 word each).
- Thus, in order to simultaneously change the Goal Position and Goal Velocity, 8 bytes of fata needs to be transferred. You need to modify the data accordingly.
- The data can be written like the example below.

```
int position, velocity;
position = 100000;
velocity =    10000;

//Make a tx data
unsigned char data[8];
data[0] = DXL_LOBYTE(DXL_LOWORD(position));
data[1] = DXL_HIBYTE(DXL_LOWORD(position));
data[2] = DXL_LOBYTE(DXL_HIWORD(position));
data[3] = DXL_HIBYTE(DXL_HIWORD(position));
data[4] = DXL_LOBYTE(DXL_LOWORD(velocity));
data[5] = DXL_HIBYTE(DXL_LOWORD(velocity));
data[6] = DXL_LOBYTE(DXL_HIWORD(velocity));
data[7] = DXL_HIBYTE(DXL_HIWORD(velocity));
```

- The goal position and goal velocity values are entered into the `unsigned char` type array implementing **DXL_LOWORD**, **DXL_HIWORD**, **DXL_LOBYTE**, and **DXL_HIBYTE** functions.
- Next, use **dxl_write** function to send the configured data to Dynamixel PRO.

```
dxl_write(Port, ID, P_GOAL_POSITION, 8, data, &ErrorStatus);
```

- To simultaneously change the Goal Position and Goal Velocity implement **dxl_write** function to modify 8 bytes of memory.
- The program's entire source is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM              17          //Comport Number of USB2DXL
```

```
#define    BAUD_RATE_NUM              3          //Baudrate Number of Dynamixel PRO
#define    P_TORQUE_ENABLE          562          //Address of Torque Enable in Control Table
#define    P_GOAL_POSITION          596          //Address of Goal Position in Contorl Table

#define ID                          1          //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
    if(ErrorCode & ERRBIT_VOLTAGE)
        printf("Input voltage error!\n");

    if(ErrorCode & ERRBIT_ANGLE)
        printf("Angle limit error!\n");

    if(ErrorCode & ERRBIT_OVERHEAT)
        printf("Overheat error!\n");

    if(ErrorCode & ERRBIT_RANGE)
        printf("Out of range error!\n");

    if(ErrorCode & ERRBIT_CHECKSUM)
        printf("Checksum error!\n");

    if(ErrorCode & ERRBIT_OVERLOAD)
        printf("Overload error!\n");

    if(ErrorCode & ERRBIT_INSTRUCTION)
        printf("Instruction code error!\n");
}

int main(void)
{
    SerialPort sp = {0,0,0,0,0};
    SerialPort *Port = &sp;

    //Open the port of USB2DXL
    if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
        printf("Succeed to open USB2Dynamixel!\n");
    else
    {
        printf( "Failed to open USB2Dynamixel!\n" );
        printf( "Press any key to terminate...\n" );
        _getch();
        return 0;
    }

    int Result, ErrorStatus;

    //Torque on
```

```
printf( "Torque on...\n" );
Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
	printf( "Failed to write!\n" );
	printf( "Press any key to terminate...\n" );
	_getch();
	return 0;
}

int position, velocity;
position = 100000;
velocity =    10000;

//Make a tx data
unsigned char data[8];
data[0] = DXL_LOBYTE(DXL_LOWORD(position));
data[1] = DXL_HIBYTE(DXL_LOWORD(position));
data[2] = DXL_LOBYTE(DXL_HIWORD(position));
data[3] = DXL_HIBYTE(DXL_HIWORD(position));
data[4] = DXL_LOBYTE(DXL_LOWORD(velocity));
data[5] = DXL_HIBYTE(DXL_LOWORD(velocity));
data[6] = DXL_LOBYTE(DXL_HIWORD(velocity));
data[7] = DXL_HIBYTE(DXL_HIWORD(velocity));

//change the position value and moving speed
printf( "Press any key to change the position value and moving speed...\n" );
_getch();
Result = dxl_write(Port, ID, P_GOAL_POSITION, 8, data, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
	printf( "Failed to write!\n" );
	printf( "Press any key to terminate...\n" );
	_getch();
	return 0;
}
else
{
	if(ErrorStatus != 0 )
		PrintErrorCode(ErrorStatus);
}


position = -100000;
velocity =       2000;
//Make a tx data
data[0] = DXL_LOBYTE(DXL_LOWORD(position));
data[1] = DXL_HIBYTE(DXL_LOWORD(position));
data[2] = DXL_LOBYTE(DXL_HIWORD(position));
data[3] = DXL_HIBYTE(DXL_HIWORD(position));
```

```
        data[4] = DXL_LOBYTE(DXL_LOWORD(velocity));
        data[5] = DXL_HIBYTE(DXL_LOWORD(velocity));
        data[6] = DXL_LOBYTE(DXL_HIWORD(velocity));
        data[7] = DXL_HIBYTE(DXL_HIWORD(velocity));

        //change the position value and moving speed
        printf( "Press any key to change the position value and moving speed...\n" );
        _getch();
        Result = dxl_write(Port, ID, P_GOAL_POSITION, 8, data, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
        else
        {
                if(ErrorStatus != 0 )
                        PrintErrorCode(ErrorStatus);
        }

        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```

- The velocity should be fast when rotating towards the first position and slower when rotating towards the second position.

viii. Internal temperature feedback of Dynamixel PRO

- So far, we have covered how to use the Write command to send commands to Dynamixel PRO.
- Next, we will go over how to use the Read command to receive data from Dynamixel PRO.
- The address of Present Temperature on the Control Table is 625 and is assigned 1 byte of memory.
- Thus, dxl_read_byte function can be used to obtain the Current Temperature of Dynamixel PRO.
- The program's entire source is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define   COM_PORT_NUM              17        //Comport Number of USB2DXL
#define   BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO
#define   P_PRESENT_TEMPERATURE    625        //Address of Present Temperature in Contorl
Table

#define ID                                1        //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
      if(ErrorCode & ERRBIT_VOLTAGE)
            printf("Input voltage error!\n");

      if(ErrorCode & ERRBIT_ANGLE)
            printf("Angle limit error!\n");

      if(ErrorCode & ERRBIT_OVERHEAT)
            printf("Overheat error!\n");

      if(ErrorCode & ERRBIT_RANGE)
            printf("Out of range error!\n");

      if(ErrorCode & ERRBIT_CHECKSUM)
            printf("Checksum error!\n");

      if(ErrorCode & ERRBIT_OVERLOAD)
            printf("Overload error!\n");

      if(ErrorCode & ERRBIT_INSTRUCTION)
            printf("Instruction code error!\n");
}
```

```
int main(void)
{
	SerialPort sp = {0,0,0,0,0};
	SerialPort *Port = &sp;

	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result, ErrorStatus;


	printf("Press any key to terminate...\n" );
	printf("\n");
	int temp;
	while(true)
	{
		if(_kbhit())
			break;

		//Read the present temperature
		Result = dxl_read_byte(Port, ID,
				P_PRESENT_TEMPERATURE, &temp, &ErrorStatus);
		if( Result != COMM_RXSUCCESS )
		{
			printf( "Failed to write!\n" );
			printf( "Press any key to terminate...\n" );
			_getch();
			return 0;
		}
		else
		{
			PrintErrorCode(ErrorStatus);
		}

		printf("\r");
		printf("current temperature : %d", temp);
	}

	//Close the port of USB2DXL
	dxl_terminate(Port);
	return 0;
}
```

- The present temperature is constantly read.

-

ix. Read the Present Position of Dynamixel PRO

- Present Position indicates the present position of Dynamixel PRO Its address is 611 and is assigned 4 bytes of memory.
- Therefore, implement **`dxl_read_dword`** function present position of Dynamixel PRO.
- The program's entire source is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define   COM_PORT_NUM              17        //Comport Number of USB2DXL
#define   BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO
#define   P_PRESENT_POSITION       611       //Address of Present Position in Contorl Table

#define ID                          1        //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
    if(ErrorCode & ERRBIT_VOLTAGE)
        printf("Input voltage error!\n");

    if(ErrorCode & ERRBIT_ANGLE)
        printf("Angle limit error!\n");

    if(ErrorCode & ERRBIT_OVERHEAT)
        printf("Overheat error!\n");

    if(ErrorCode & ERRBIT_RANGE)
        printf("Out of range error!\n");

    if(ErrorCode & ERRBIT_CHECKSUM)
        printf("Checksum error!\n");

    if(ErrorCode & ERRBIT_OVERLOAD)
        printf("Overload error!\n");

    if(ErrorCode & ERRBIT_INSTRUCTION)
        printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;
```

```
	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result, ErrorStatus;


	printf("Press any key to terminate...\n" );
	printf("\n");
	int temp;
	while(true)
	{
		if(_kbhit())
			break;

		//Read the present position
		Result = dxl_read_dword(Port, ID, P_PRESENT_POSITION, (unsigned*)&temp,
&ErrorStatus);
		if( Result != COMM_RXSUCCESS )
		{
			printf( "Failed to write!\n" );
			printf( "Press any key to terminate...\n" );
			_getch();
			return 0;
		}
		else
		{
			PrintErrorCode(ErrorStatus);
		}

		printf("\r");
		printf("present position : %d", temp);
	}

	//Close the port of USB2DXL
	dxl_terminate(Port);
	return 0;
}
```

- The Present Position is shown when the program is initiated.

-

x. Changing velocity of Dynamixel PRO in Wheel Mode

- On Dynamixel PRO's Control Table the address for Operating Mode is 11. It requires 1 byte of memory. By changing the value of the Operating Mode Dynamixel PRO can be set to Joint Mode, Wheel Mode, or Torque Mode,
- Implement **`dxl_write_byte`** to change Operating Mode. It is a part of the EEPROM area so the Torque Enable must be off to modifiy its value.
- Also, as shown in 1.1, Torque Enable needs to be on to activate Dynamixel PRO in Wheel Mode. The speed of Dynamixel PRO can be controlled by modifying the Goal_Velocity values.
- The program's entire source code is shown below.

main.cpp

```
#include <Windows.h>
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM              17        //Comport Number of USB2DXL
#define    BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO

#define    P_OPERATING_MODE          11        //Address of Operation Mode in Control Table
#define    P_TORQUE_ENABLE          562        //Address of Torque Enable in Control Table
#define    P_GOAL_VELOCITY          600        //Address of Goal Velocity in Control Table

#define ID                                1        //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");

        if(ErrorCode & ERRBIT_CHECKSUM)
                printf("Checksum error!\n");

        if(ErrorCode & ERRBIT_OVERLOAD)
                printf("Overload error!\n");

        if(ErrorCode & ERRBIT_INSTRUCTION)
```

```
		printf("Instruction code error!\n");
}

int main(void)
{
	SerialPort sp = {0,0,0,0,0};
	SerialPort *Port = &sp;

	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result, ErrorStatus;

	//Torque Off
	printf( "Torque Off...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	printf("Press any key to change the operating mode...\n");
	_getch();
	Result = dxl_write_byte(Port, ID, P_OPERATING_MODE, 1, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
		else
			printf("Succeed to chage the operationg mode!\n");
	}
```

```
	//Torque On
	printf( "Torque On...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//Change the Goal Velocity
	printf("Press any key to change goal velocity...\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_GOAL_VELOCITY, 5000, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
		else
			printf("Succeed to chage the operationg mode!\n");
	}

	//Close the port of USB2DXL
	printf( "Press any key to terminate...\n" );
	_getch();

	//Torque Off
	printf( "Torque Off...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	dxl_terminate(Port);
	return 0;
}
```

xi. Checking the current position and current velocity of Dynamixel PRO

- There are two methods to check the present position of Dynamixel PRO.
- First, is to implement **`dxl_read_dword`** function twice to change the values. Second, is to implement **`dxl_read`** function to simultaneously read the Present Position and Present Velocity.
- You have gone over **`dxl_read_byte`**, **`dxl_read_word`**, and **`dxl_tead_dword`**, This time implement the second method to change the Goal Position and Goal Velocity of Dynamixel PRO.
- Both Goal Position and Goal Velocity require 4 bytes.
- Thus, to simultaneously read the Present Position and Present Velocity you need to modify the data length accordingly.
- Data can be read by entering the data type as shown on the example below.

```
unsigned char data[8];
dxl_read(Port, ID, P_PRESENT_POSITION, 8, data, &ErrorStatus);

int present_position, present_velocity;
present_position=DXL_MAKEDWORD(DXL_MAKEWORD(data[0],data[1]),DXL_MAKEWORD(da
ta[2],data[3]));
present_velocity=DXL_MAKEDWORD(DXL_MAKEWORD(data[4],data[5]),DXL_MAKEWORD(da
ta[6],data[7]) );
```

- The `unsigned char` type array data obtained is converted into the present position and velocity with **`DXL_MAKEWORD`**, **`DXL_MAKEDWORD`** (refer to the source code above).
- The program's entire source code is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define   COM_PORT_NUM              17        //Comport Number of USB2DXL
#define   BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO

#define   P_OPERATING_MODE          11        //Address of Operation Mode in Control Table
#define   P_TORQUE_ENABLE          562        //Address of Torque Enable in Control Table
#define   P_GOAL_POSITION          596       //Address of Goal Position in Control Table
#define   P_GOAL_VELOCITY          600        //Address of Goal Velocity in Control Table
#define   P_PRESENT_POSITION       611       //Address of Present Position in Control Table
#define   P_PRESENT_VELOCITY       615        //Address of Present Velocity in Control Table

#define ID                          1        //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
```

```
	if(ErrorCode & ERRBIT_VOLTAGE)
		printf("Input voltage error!\n");

	if(ErrorCode & ERRBIT_ANGLE)
		printf("Angle limit error!\n");

	if(ErrorCode & ERRBIT_OVERHEAT)
		printf("Overheat error!\n");

	if(ErrorCode & ERRBIT_RANGE)
		printf("Out of range error!\n");

	if(ErrorCode & ERRBIT_CHECKSUM)
		printf("Checksum error!\n");

	if(ErrorCode & ERRBIT_OVERLOAD)
		printf("Overload error!\n");

	if(ErrorCode & ERRBIT_INSTRUCTION)
		printf("Instruction code error!\n");
}

int main(void)
{
	SerialPort sp = {0,0,0,0,0};
	SerialPort *Port = &sp;

	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result, ErrorStatus;

	//Torque Off
	printf( "Torque Off...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
```

```
printf("Press any key to change the operating mode...\n");
_getch();
Result = dxl_write_byte(Port, ID, P_OPERATING_MODE, 1, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
	printf( "Failed to write!\n" );
	printf( "Press any key to terminate...\n" );
	_getch();
	return 0;
}
else
{
	if(ErrorStatus != 0 )
		PrintErrorCode(ErrorStatus);
	else
		printf("Succeed to chage the operationg mode!\n");
}

//Torque On
printf( "Torque On...\n" );
Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
	printf( "Failed to write!\n" );
	printf( "Press any key to terminate...\n" );
	_getch();
	return 0;
}

//Change the Goal Velocity
printf("Press any key to change goal velocity...\n");
_getch();
Result = dxl_write_dword(Port, ID, P_GOAL_VELOCITY, 5000, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
	printf( "Failed to write!\n" );
	printf( "Press any key to terminate...\n" );
	_getch();
	return 0;
}
else
{
	if(ErrorStatus != 0 )
		PrintErrorCode(ErrorStatus);
	else
		printf("Succeed to chage the operationg mode!\n");
}
```

```
	printf("Press any key to terminate...\n" );
	printf("\n");
	while(true)
	{
		if(_kbhit())
			break;

		unsigned char data[8];
		dxl_read(Port, ID, P_PRESENT_POSITION, 8, data, &ErrorStatus);

		int present_position, present_velocity;
		present_position = DXL_MAKEDWORD( DXL_MAKEWORD(data[0], data[1]),
DXL_MAKEWORD(data[2], data[3]) );
		present_velocity = DXL_MAKEDWORD( DXL_MAKEWORD(data[4], data[5]),
DXL_MAKEWORD(data[6], data[7]) );

		printf("\r");
		printf("present position : %d, presen velocity : %d", present_position,
present_velocity);
	}
	printf("\n");

	//Close the port of USB2DXL
	printf( "Press any key to terminate...\n" );
	_getch();

	//Torque Off
	printf( "Torque Off...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	dxl_terminate(Port);
	return 0;
}
```

### 1.2.3 C programming language funtions.

i. Modifying the zero value of Dynamixel PRO

- You can modify the zero-position of Dynamixel PRO.
- Change the Control table's Homing Offset to modify Dynamixel PRO's zero-position.
- Homing Offset address on the Control Tableis 13 and requires 4 bytes of memory. Therefore, implement **`dxl_write_dword`** to modify the Homing Offset value.
- Homing Offset is a part of the EEPROM area, which requires the Torque Enable to be turned off be for changes.
- When modifying the Homing Offset value, designate the zero position by inverting the desired zero position with a negative sign (-).
- For instance, to set the Position Valu 50000 as the zero position, -50000 needs to be entered as the Homing Offset value.
- The program's entire source code is shown below.
- Check if the Operation Mode is set as 'Joint Mode' before running the program below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM              17        //Comport Number of USB2DXL
#define    BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO

#define    P_HOMING_OFFSET           13        //Address of Homing Offset in Control Table
#define    P_TORQUE_ENABLE          562        //Address of Torque Enable in Control Table
#define    P_GOAL_POSITION          596        //Address of Goal Position in Control Table


#define ID                                1        //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");
```

```
	if(ErrorCode & ERRBIT_CHECKSUM)
		printf("Checksum error!\n");

	if(ErrorCode & ERRBIT_OVERLOAD)
		printf("Overload error!\n");

	if(ErrorCode & ERRBIT_INSTRUCTION)
		printf("Instruction code error!\n");
}

int main(void)
{
	SerialPort sp = {0,0,0,0,0};
	SerialPort *Port = &sp;

	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result, ErrorStatus;

	//Torque Off
	printf( "Torque Off...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//Change the zero point
	printf("Press any key to change the zero point...\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_HOMING_OFFSET, -50000, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
```

```
        else
        {
                if(ErrorStatus != 0 )
                        PrintErrorCode(ErrorStatus);
                else
                        printf("Succeed to chage the homing offset!\n");
        }

        //Torque On
        printf( "Torque On...\n" );
        Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }


        //Change the Goal position
        printf("Press any key to change the Goal Position...\n");
        _getch();
        Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, 0, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
        else
                PrintErrorCode(ErrorStatus);


        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```

- If the program does not run properly check if the Operating Mode is set to Joint Mode.

ii. Limiting the operating range of Dynamixel PRO

- Please refer to 1.1 for information regarding +, - Position Limit.
- The program's entire source code is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define   COM_PORT_NUM              17        //Comport Number of USB2DXL
#define   BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO

#define   P_PLUS_POSITION_LIMIT     36       //Address of Plus Position Limit in Control Table
#define   P_MINUS_POSITION_LIMIT    40       //Address of Minus Position Limit in Control
Table
#define   P_TORQUE_ENABLE           562      //Address of Torque Enable in Control Table
#define   P_GOAL_POSITION           596      //Address of Goal Position in Control Table


#define ID                          1        //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");

        if(ErrorCode & ERRBIT_CHECKSUM)
                printf("Checksum error!\n");

        if(ErrorCode & ERRBIT_OVERLOAD)
                printf("Overload error!\n");

        if(ErrorCode & ERRBIT_INSTRUCTION)
                printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
```

```
	SerialPort *Port = &sp;

	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result, ErrorStatus;

	//Torque Off
	printf( "Torque Off...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//Change the Plus Position Limit
	printf("Press any key to change the plus position limit...\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_PLUS_POSITION_LIMIT, 100000, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
		else
			printf("Succeed to chage the plus position limit!\n");
	}

	//Change the Minus Position Limit
	printf("Press any key to change the minus position limit...\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_MINUS_POSITION_LIMIT, -100000, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
```

```
{
        printf( "Failed to write!\n" );
        printf( "Press any key to terminate...\n" );
        _getch();
        return 0;
}
else
{
        if(ErrorStatus != 0 )
                PrintErrorCode(ErrorStatus);
        else
                printf("Succeed to chage the minus position limit!\n");
}


//Torque On
printf( "Torque On...\n" );
Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
        printf( "Failed to write!\n" );
        printf( "Press any key to terminate...\n" );
        _getch();
        return 0;
}

//Change the Goal position
printf("Press any key to change the Goal Position to 120000...\n");
_getch();
Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, 120000, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
        printf( "Failed to write!\n" );
        printf( "Press any key to terminate...\n" );
        _getch();
        return 0;
}
else
        PrintErrorCode(ErrorStatus);

//Change the Goal position
printf("Press any key to change the Goal Position to 100000\n");
_getch();
Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, 100000, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
        printf( "Failed to write!\n" );
        printf( "Press any key to terminate...\n" );
        _getch();
        return 0;
```

```
        }
        else
                PrintErrorCode(ErrorStatus);

        //Change the Goal position
        printf("Press any key to change the Goal Position to -120000\n");
        _getch();
        Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, -120000, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
        else
                PrintErrorCode(ErrorStatus);


        //Change the Goal position
        printf("Press any key to change the Goal Position to -100000\n");
        _getch();
        Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, -100000, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
        else
                PrintErrorCode(ErrorStatus);


        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```

- If the program does not run properly check if the Operating Mode is set to Joint Mode.

iii. Extending the operating range of Dynamixel PRO.

- Please refer to 1.1 for information regarding +/- Position Limits.
- The program's entire source code is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM                17          //Comport Number of USB2DXL
#define    BAUD_RATE_NUM                3          //Baudrate Number of Dynamixel PRO

#define    P_PLUS_POSITION_LIMIT     36          //Address of Plus Position Limit in Control Table
#define    P_MINUS_POSITION_LIMIT    40          //Address of Minus Position Limit in Control
Table
#define    P_TORQUE_ENABLE           562         //Address of Torque Enable in Control Table
#define    P_GOAL_POSITION           596         //Address of Goal Position in Control Table

#define ID                                1          //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");

        if(ErrorCode & ERRBIT_CHECKSUM)
                printf("Checksum error!\n");

        if(ErrorCode & ERRBIT_OVERLOAD)
                printf("Overload error!\n");

        if(ErrorCode & ERRBIT_INSTRUCTION)
                printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;
```

```
//Open the port of USB2DXL
if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
        printf("Succeed to open USB2Dynamixel!\n");
else
{
        printf( "Failed to open USB2Dynamixel!\n" );
        printf( "Press any key to terminate...\n" );
        _getch();
        return 0;
}

int Result, ErrorStatus;

//Torque Off
printf( "Torque Off...\n" );
Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
        printf( "Failed to write!\n" );
        printf( "Press any key to terminate...\n" );
        _getch();
        return 0;
}

//Change the Plus Position Limit
printf("Press any key to change the plus position limit...\n");
_getch();
Result = dxl_write_dword(Port, ID, P_PLUS_POSITION_LIMIT, 5000000, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
        printf( "Failed to write!\n" );
        printf( "Press any key to terminate...\n" );
        _getch();
        return 0;
}
else
{
        if(ErrorStatus != 0 )
                PrintErrorCode(ErrorStatus);
        else
                printf("Succeed to chage the plus position limit!\n");
}

//Change the Minus Position Limit
printf("Press any key to change the minus position limit...\n");
_getch();
Result = dxl_write_dword(Port, ID, P_MINUS_POSITION_LIMIT, -500000, &ErrorStatus);
if( Result != COMM_RXSUCCESS )
{
```

```
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
		else
			printf("Succeed to chage the minus position limit!\n");
	}

	//Torque On
	printf( "Torque On...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//Change the Goal position
	printf("Press any key to change the Goal Position to 500000...\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, 500000, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
		PrintErrorCode(ErrorStatus);

	//Change the Goal position
	printf("Press any key to change the Goal Position to -500000\n");
	_getch();
	Result = dxl_write_dword(Port, ID, P_GOAL_POSITION, -500000, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
```

```
                PrintErrorCode(ErrorStatus);

        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```

- If the program does not run properly check if the Operating Mode is set to Joint Mode.

### 1.2.4 Indirect Addressing function of Dynamixel PRO

i. Change the position, velocity, and acceleration using Indirect Address fucntion.

- On Dynamixel PRO's Control Table-Goal Position, Goal Velocity, and Goal Acceleration addresses-are not arranged in consecutive order. Thus, **`dxl_write`** and **`dxl_write_dword`** need to be implemented 3 times to change all 3 addresses.
- To resolve this issue, Dynamixel PRO has Indirect Address function.
- Indirect Address assignes another address to the default one.
- For example, Goal Position (#596) address can be assigned another address (#634).
- Start by writing the value of the desired address in the EEPROM area of the Indirect Address.
- The written value is the default address. Please refer to the source code below

```
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_0, P_GOAL_POSITION,      &ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_1, P_GOAL_POSITION+1, &ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_2, P_GOAL_POSITION+2, &ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_3, P_GOAL_POSITION+3, &ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_4, P_GOAL_VELOCITY,      &ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_5, P_GOAL_VELOCITY+1, &ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_6, P_GOAL_VELOCITY+2, &ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_7, P_GOAL_VELOCITY+3, &ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_8,    P_GOAL_ACCELERATION,
&ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_9,    P_GOAL_ACCELERATION+1,
&ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_10, P_GOAL_ACCELERATION+2,
&ErrorStatus);
dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_11, P_GOAL_ACCELERATION+3,
&ErrorStatus);
```

- The program's entire source code is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM              17        //Comport Number of USB2DXL
#define    BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO

#define    P_INDIRECT_ADDRESS_0      49        //Address of 1st Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_1      51        //Address of 2nd Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_2      53        //Address of 3rd Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_3      55        //Address of 4th Indirect Address in Control
Table
```

```
#define    P_INDIRECT_ADDRESS_4      57          //Address of 5th Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_5      59          //Address of 6th Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_6      61          //Address of 7th Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_7      63          //Address of 8th Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_8      65          //Address of 9th Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_9      67          //Address of 10th Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_10     69          //Address of 11th Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_11     71          //Address of 12th Indirect Address in Control
Table

#define    P_TORQUE_ENABLE           562         //Address of Torque Enable in Control Table
#define    P_GOAL_POSITION           596         //Address of Goal Position in Control Table
#define    P_GOAL_VELOCITY           600         //Address of Goal Velocity in Control Table
#define    P_GOAL_ACCELERATION       606         //Address of Goal Acceleration in Control
Table
#define    P_INDIRECT_DATA_0         634         //Address of Goal Indirect Data in Control Table

#define ID                           1           //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");

        if(ErrorCode & ERRBIT_CHECKSUM)
                printf("Checksum error!\n");

        if(ErrorCode & ERRBIT_OVERLOAD)
                printf("Overload error!\n");

        if(ErrorCode & ERRBIT_INSTRUCTION)
                printf("Instruction code error!\n");
}
```

```
int main(void)
{
	SerialPort sp = {0,0,0,0,0};
	SerialPort *Port = &sp;

	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) == COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result, ErrorStatus;

	//Torque Off
	printf( "Torque Off...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//Set the Indirect Address for Goal Position
	printf("Press any key to set the indirect address for goal position...\n");
	_getch();
	Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_0, P_GOAL_POSITION,
&ErrorStatus);
	Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_1, P_GOAL_POSITION+1,
&ErrorStatus);
	Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_2, P_GOAL_POSITION+2,
&ErrorStatus);
	Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_3, P_GOAL_POSITION+3,
&ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
```

```
                if(ErrorStatus != 0 )
                        PrintErrorCode(ErrorStatus);
                else
                        printf("Succeed to set the indirect address for goal position!\n");
        }

        //Set the Indirect Address for Goal Velocity
        printf("Press any key to set the indirect address for goal velocity...\n");
        _getch();
        Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_4, P_GOAL_VELOCITY,
&ErrorStatus);
        Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_5, P_GOAL_VELOCITY+1,
&ErrorStatus);
        Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_6, P_GOAL_VELOCITY+2,
&ErrorStatus);
        Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_7, P_GOAL_VELOCITY+3,
&ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
        else
        {
                if(ErrorStatus != 0 )
                        PrintErrorCode(ErrorStatus);
                else
                        printf("Succeed to set the indirect address for goal velocity!\n");
        }

        //Set the Indirect Address for Goal Acceleration
        printf("Press any key to set the indirect address for goal acceleration...\n");
        _getch();
        Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_8,
P_GOAL_ACCELERATION,     &ErrorStatus);
        Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_9,
P_GOAL_ACCELERATION+1, &ErrorStatus);
        Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_10,
P_GOAL_ACCELERATION+2, &ErrorStatus);
        Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_11,
P_GOAL_ACCELERATION+3, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
```

```
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
		else
			printf("Succeed to set the indirect address for goal acceleration!\n");
	}


	//Torque On
	printf( "Torque On...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int position, velocity, acceleration;
	position = 100000;
	velocity =    10000;
	acceleration = 16;

	//Make a tx data
	unsigned char data[12];
	data[0]    = DXL_LOBYTE(DXL_LOWORD(position));
	data[1]    = DXL_HIBYTE(DXL_LOWORD(position));
	data[2]    = DXL_LOBYTE(DXL_HIWORD(position));
	data[3]    = DXL_HIBYTE(DXL_HIWORD(position));
	data[4]    = DXL_LOBYTE(DXL_LOWORD(velocity));
	data[5]    = DXL_HIBYTE(DXL_LOWORD(velocity));
	data[6]    = DXL_LOBYTE(DXL_HIWORD(velocity));
	data[7]    = DXL_HIBYTE(DXL_HIWORD(velocity));
	data[8]    = DXL_LOBYTE(DXL_LOWORD(acceleration));
	data[9]    = DXL_HIBYTE(DXL_LOWORD(acceleration));
	data[10] = DXL_LOBYTE(DXL_HIWORD(acceleration));
	data[11] = DXL_HIBYTE(DXL_HIWORD(acceleration));

	//change the position value, moving speed, acceleration using indirect addr
	printf( "Press any key to change the position, speed, acceleration...\n" );
	_getch();
	Result = dxl_write(Port, ID, P_INDIRECT_DATA_0, 12, data, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
```

```
        }
        else
        {
                if(ErrorStatus != 0 )
                        PrintErrorCode(ErrorStatus);
        }


        position = -100000;
        velocity =      1000;
        acceleration =   2;

        //Make a tx data
        data[0]   = DXL_LOBYTE(DXL_LOWORD(position));
        data[1]   = DXL_HIBYTE(DXL_LOWORD(position));
        data[2]   = DXL_LOBYTE(DXL_HIWORD(position));
        data[3]   = DXL_HIBYTE(DXL_HIWORD(position));
        data[4]   = DXL_LOBYTE(DXL_LOWORD(velocity));
        data[5]   = DXL_HIBYTE(DXL_LOWORD(velocity));
        data[6]   = DXL_LOBYTE(DXL_HIWORD(velocity));
        data[7]   = DXL_HIBYTE(DXL_HIWORD(velocity));
        data[8]   = DXL_LOBYTE(DXL_LOWORD(acceleration));
        data[9]   = DXL_HIBYTE(DXL_LOWORD(acceleration));
        data[10] = DXL_LOBYTE(DXL_HIWORD(acceleration));
        data[11] = DXL_HIBYTE(DXL_HIWORD(acceleration));

        //change the position value, moving speed, acceleration using indirect addr
        printf( "Press any key to change the position, speed, acceleration...\n" );
        _getch();
        Result = dxl_write(Port, ID, P_INDIRECT_DATA_0, 12, data, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }
        else
        {
                if(ErrorStatus != 0 )
                        PrintErrorCode(ErrorStatus);
        }

        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```

ii. Reading the temperature and the current position using Indirect Address

- Implement Indirect Address to obtain read outputs.
- Read the Present Position and Present Temperature by using Indirect Address.
- The program's entire source code is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM                17          //Comport Number of USB2DXL
#define    BAUD_RATE_NUM                3          //Baudrate Number of Dynamixel PRO

#define    P_OPERATING_MODE            11          //Address of Operating Mode in Control Table
#define    P_INDIRECT_ADDRESS_0        49          //Address of 1st Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_1        51          //Address of 2nd Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_2        53          //Address of 3rd Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_3        55          //Address of 4th Indirect Address in Control
Table
#define    P_INDIRECT_ADDRESS_4        57          //Address of 5th Indirect Address in Control
Table

#define    P_TORQUE_ENABLE             562         //Address of Torque Enable in Control Table
#define    P_PRESENT_POSITION          611         //Address of Present Position in Control Table
#define    P_PRESENT_TEMPERATURE       625         //Address of Present Temperature in Control
Table

#define    P_INDIRECT_DATA_0                634          //Address of Goal Indirect Data in
Control Table

#define ID                                  1          //ID of Dynamixel PRO you use

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");
```

```
        if(ErrorCode & ERRBIT_CHECKSUM)
                printf("Checksum error!\n");

        if(ErrorCode & ERRBIT_OVERLOAD)
                printf("Overload error!\n");

        if(ErrorCode & ERRBIT_INSTRUCTION)
                printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;

        //Open the port of USB2DXL
        if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) ==
COMM_RXSUCCESS )
                printf("Succeed to open USB2Dynamixel!\n");
        else
        {
                printf( "Failed to open USB2Dynamixel!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        int Result, ErrorStatus;

        //Torque Off
        printf( "Torque Off...\n" );
        Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 0, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        //change the operating mode to wheel mode
        printf( "Make the Mode of Dynamixel PRO to Wheel Mode...\n" );
        Result = dxl_write_byte(Port, ID, P_OPERATING_MODE, 1, &ErrorStatus);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
```

```
	}


	//Set the Indirect Address for Present Temperature
	printf("Press any key to set the indirect address for temperature...\n");
	_getch();
	Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_0,
P_PRESENT_TEMPERATURE, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
		else
			printf("Succeed to set the indirect address for temperature!\n");
	}

	//Set the Indirect Address for Goal Velocity
	printf("Press any key to set the indirect address for present position...\n");
	_getch();
	Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_1, P_PRESENT_POSITION,
&ErrorStatus);
	Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_2,
P_PRESENT_POSITION+1, &ErrorStatus);
	Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_3,
P_PRESENT_POSITION+2, &ErrorStatus);
	Result = dxl_write_word(Port, ID, P_INDIRECT_ADDRESS_4,
P_PRESENT_POSITION+3, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}
	else
	{
		if(ErrorStatus != 0 )
			PrintErrorCode(ErrorStatus);
		else
			printf("Succeed to set the indirect address for present position!\n");
	}

	//Torque On
```

```
	printf( "Torque On...\n" );
	Result = dxl_write_byte(Port, ID, P_TORQUE_ENABLE, 1, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	//Rotating Start
	printf("Rotating Start\n");
	Result = dxl_write_dword(Port, ID, 600, 5000, &ErrorStatus);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	printf("Press any key to terminate...\n" );
	printf("\n");
	while(true)
	{
		if(_kbhit())
			break;

		unsigned char data[5];
		dxl_read(Port, ID, P_INDIRECT_DATA_0, 5, data, &ErrorStatus);

		int temp, present_position;
		temp = data[0];
		present_position = DXL_MAKEDWORD( DXL_MAKEWORD(data[1], data[2]),
DXL_MAKEWORD(data[3], data[4]) );

		printf("\r");
		printf("present temperature : %d, presen position : %d", temp, present_position);
	}
	printf("\n");

	//Close the port of USB2DXL
	dxl_terminate(Port);
	return 0;
}
```

### 1.2.5 Using Multiple Dynamixel PROs

i. Controlling the LEDs of 3 Dynamixel PROs

- Implement **`dxl_synch_write`** to simultaneously send a command to multiple Dynamixel PROs.
- The program's entire source code is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define   COM_PORT_NUM              17        //Comport Number of USB2DXL
#define   BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO

#define   P_LED_RED                563        //Address of LED RED in Control Table

#define ID_1                       1        //ID of Dynamixel PRO you use
#define ID_2                       2
#define ID_3                       3

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");

        if(ErrorCode & ERRBIT_CHECKSUM)
                printf("Checksum error!\n");

        if(ErrorCode & ERRBIT_OVERLOAD)
                printf("Overload error!\n");

        if(ErrorCode & ERRBIT_INSTRUCTION)
                printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
```

```
	SerialPort *Port = &sp;

	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) ==
COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result;

	unsigned char param[6];
	param[0] = ID_1;
	param[1] = 255;
	param[2] = ID_2;
	param[3] = 255;
	param[4] = ID_3;
	param[5] = 255;

	//LED On
	printf( "Press any to turn on the LED...\n" );
	_getch();
	Result = dxl_sync_write(Port, P_LED_RED, 1, param, 6);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	param[0] = ID_1;
	param[1] = 0;
	param[2] = ID_2;
	param[3] = 0;
	param[4] = ID_3;
	param[5] = 0;

	//LED Off
	printf( "Press any to turn off the LED...\n" );
	_getch();
	Result = dxl_sync_write(Port, P_LED_RED, 1, param, 6);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
```

```
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```

ii. Controlling the Goal Position of 3 Dynamixel PRO

- The program's entire source code is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM              17        //Comport Number of USB2DXL
#define    BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO

#define    P_TORQUE_ENABLE          562        //Address of Torque Enable in Control Table
#define    P_GOAL_POSITION          596        //Address of Goal Position in Control Table

#define ID_1                        1         //ID of Dynamixel PRO you use
#define ID_2                        2
#define ID_3                        3

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");

        if(ErrorCode & ERRBIT_CHECKSUM)
                printf("Checksum error!\n");

        if(ErrorCode & ERRBIT_OVERLOAD)
                printf("Overload error!\n");

        if(ErrorCode & ERRBIT_INSTRUCTION)
                printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;

        //Open the port of USB2DXL
```

```
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) ==
COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result;
	unsigned char param[15];
	//Torque ON
	printf("Torque On\n");
	param[0] = ID_1;
	param[1] = 1;
	param[2] = ID_2;
	param[3] = 1;
	param[4] = ID_3;
	param[5] = 1;
	Result = dxl_sync_write(Port, P_TORQUE_ENABLE, 1, param, 6);
	if( Result != COMM_RXSUCCESS )
	{
		printf( "Failed to write!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}


	param[0] = ID_1;
	param[1] = DXL_LOBYTE(DXL_LOWORD(100000));
	param[2] = DXL_HIBYTE(DXL_LOWORD(100000));
	param[3] = DXL_LOBYTE(DXL_HIWORD(100000));
	param[4] = DXL_HIBYTE(DXL_HIWORD(100000));
	param[5] = ID_2;
	param[6] = DXL_LOBYTE(DXL_LOWORD(50000));
	param[7] = DXL_HIBYTE(DXL_LOWORD(50000));
	param[8] = DXL_LOBYTE(DXL_HIWORD(50000));
	param[9] = DXL_HIBYTE(DXL_HIWORD(50000));
	param[10]= ID_3;
	param[11]= DXL_LOBYTE(DXL_LOWORD(-80000));
	param[12]= DXL_HIBYTE(DXL_LOWORD(-80000));
	param[13]= DXL_LOBYTE(DXL_HIWORD(-80000));
	param[14]= DXL_HIBYTE(DXL_HIWORD(-80000));

	printf( "Press any to change goal position...\n" );
	_getch();
	Result = dxl_sync_write(Port, P_GOAL_POSITION, 4, param, 15);
```

```
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        param[0] = ID_1;
        param[1] = DXL_LOBYTE(DXL_LOWORD(0));
        param[2] = DXL_HIBYTE(DXL_LOWORD(0));
        param[3] = DXL_LOBYTE(DXL_HIWORD(0));
        param[4] = DXL_HIBYTE(DXL_HIWORD(0));
        param[5] = ID_2;
        param[6] = DXL_LOBYTE(DXL_LOWORD(0));
        param[7] = DXL_HIBYTE(DXL_LOWORD(0));
        param[8] = DXL_LOBYTE(DXL_HIWORD(0));
        param[9] = DXL_HIBYTE(DXL_HIWORD(0));
        param[10]= ID_3;
        param[11]= DXL_LOBYTE(DXL_LOWORD(0));
        param[12]= DXL_HIBYTE(DXL_LOWORD(0));
        param[13]= DXL_LOBYTE(DXL_HIWORD(0));
        param[14]= DXL_HIBYTE(DXL_HIWORD(0));

        printf( "Press any to change goal position...\n" );
        _getch();
        Result = dxl_sync_write(Port, P_GOAL_POSITION, 4, param, 15);
        if( Result != COMM_RXSUCCESS )
        {
                printf( "Failed to write!\n" );
                printf( "Press any key to terminate...\n" );
                _getch();
                return 0;
        }

        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```

iii. Reading the the Current Position of 3 Dynamixel PROs

- Implement **`dxl_bulk_read`** to simultaneously read the values of multiple Dynamixel PROs.
- The program's entire source code is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM              17        //Comport Number of USB2DXL
#define    BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO

#define    P_PRESENT_POSITION       611        //Address of Goal Position in Control Table

#define ID_1                        1         //ID of Dynamixel PRO you use
#define ID_2                        2
#define ID_3                        3

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");

        if(ErrorCode & ERRBIT_CHECKSUM)
                printf("Checksum error!\n");

        if(ErrorCode & ERRBIT_OVERLOAD)
                printf("Overload error!\n");

        if(ErrorCode & ERRBIT_INSTRUCTION)
                printf("Instruction code error!\n");
}

int main(void)
{
        SerialPort sp = {0,0,0,0,0};
        SerialPort *Port = &sp;
```

```
	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) ==
COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result;
	int position_length=4;
	unsigned char param[15];
	param[0] = ID_1;
	param[1] = DXL_LOBYTE(P_PRESENT_POSITION);
	param[2] = DXL_HIBYTE(P_PRESENT_POSITION);
	param[3] = DXL_LOBYTE(position_length);
	param[4] = DXL_HIBYTE(position_length);
	param[5] = ID_2;
	param[6] = DXL_LOBYTE(P_PRESENT_POSITION);
	param[7] = DXL_HIBYTE(P_PRESENT_POSITION);
	param[8] = DXL_LOBYTE(position_length);
	param[9] = DXL_HIBYTE(position_length);
	param[10] = ID_3;
	param[11] = DXL_LOBYTE(P_PRESENT_POSITION);
	param[12] = DXL_HIBYTE(P_PRESENT_POSITION);
	param[13] = DXL_LOBYTE(position_length);
	param[14] = DXL_HIBYTE(position_length);


	BulkData bd[256];
	BulkData *pbd[256];
	for(unsigned int i = 0 ; i <256 ; i++)
	{
		pbd[i] = &bd[i];
	}

	dxl_bulk_read(Port, param, 15, pbd);

	int present_position1;
	int present_position2;
	int present_position3;

	dxl_get_bulk_dword(pbd, ID_1, P_PRESENT_POSITION, (unsigned*)&present_position1);
	dxl_get_bulk_dword(pbd, ID_2, P_PRESENT_POSITION, (unsigned*)&present_position2);
	dxl_get_bulk_dword(pbd, ID_3, P_PRESENT_POSITION, (unsigned*)&present_position3);
```

```
        printf("Present Position\n");
        printf("ID_1 : %d\n", present_position1);
        printf("ID_2 : %d\n", present_position2);
        printf("ID_3 : %d\n", present_position3);


        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```

iv. Read temperature of the first, position of the second, and present current of the third Dynamixel PRO

- Implement **`dxl_bulk_read`** to simultaneously read the values of multiple Dynamixel PROs.
- The program's entire source code is shown below.

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include "dynamixel.h"

#define    COM_PORT_NUM              17        //Comport Number of USB2DXL
#define    BAUD_RATE_NUM              3        //Baudrate Number of Dynamixel PRO

#define    P_PRESENT_POSITION        611       //Address of Goal Position in Control Table
#define    P_PRESENT_CURRENT         621       //Address of Present Current in Control Table
#define    P_PRESENT_TEMPERATURE     625       //Address of Present Temperature in Control
Table

#define ID_1                         1         //ID of Dynamixel PRO you use
#define ID_2                         2
#define ID_3                         3

// Print error bit of status packet
void PrintErrorCode(int ErrorCode)
{
        if(ErrorCode & ERRBIT_VOLTAGE)
                printf("Input voltage error!\n");

        if(ErrorCode & ERRBIT_ANGLE)
                printf("Angle limit error!\n");

        if(ErrorCode & ERRBIT_OVERHEAT)
                printf("Overheat error!\n");

        if(ErrorCode & ERRBIT_RANGE)
                printf("Out of range error!\n");

        if(ErrorCode & ERRBIT_CHECKSUM)
                printf("Checksum error!\n");

        if(ErrorCode & ERRBIT_OVERLOAD)
                printf("Overload error!\n");

        if(ErrorCode & ERRBIT_INSTRUCTION)
                printf("Instruction code error!\n");
}

int main(void)
```

```
{
	SerialPort sp = {0,0,0,0,0};
	SerialPort *Port = &sp;

	//Open the port of USB2DXL
	if(dxl_initialize(Port, COM_PORT_NUM, BAUD_RATE_NUM) ==
COMM_RXSUCCESS )
		printf("Succeed to open USB2Dynamixel!\n");
	else
	{
		printf( "Failed to open USB2Dynamixel!\n" );
		printf( "Press any key to terminate...\n" );
		_getch();
		return 0;
	}

	int Result;
	int current_length = 2, position_length = 4, temperature_length = 1;
	unsigned char param[15];
	param[0] = ID_1;
	param[1] = DXL_LOBYTE(P_PRESENT_CURRENT);
	param[2] = DXL_HIBYTE(P_PRESENT_CURRENT);
	param[3] = DXL_LOBYTE(current_length);
	param[4] = DXL_HIBYTE(current_length);
	param[5] = ID_2;
	param[6] = DXL_LOBYTE(P_PRESENT_POSITION);
	param[7] = DXL_HIBYTE(P_PRESENT_POSITION);
	param[8] = DXL_LOBYTE(position_length);
	param[9] = DXL_HIBYTE(position_length);
	param[10] = ID_3;
	param[11] = DXL_LOBYTE(P_PRESENT_TEMPERATURE);
	param[12] = DXL_HIBYTE(P_PRESENT_TEMPERATURE);
	param[13] = DXL_LOBYTE(temperature_length);
	param[14] = DXL_HIBYTE(temperature_length);


	BulkData bd[256];
	BulkData *pbd[256];
	for(unsigned int i = 0 ; i <256 ; i++)
		pbd[i] = &bd[i];

	dxl_bulk_read(Port, param, 15, pbd);

	int present_current;
	int present_position;
	int present_temperature;

	dxl_get_bulk_word( pbd, ID_1, P_PRESENT_CURRENT,
&present_current);
	dxl_get_bulk_dword(pbd, ID_2, P_PRESENT_POSITION,
```

```
(unsigned*)&present_position);
        dxl_get_bulk_byte( pbd, ID_3, P_PRESENT_TEMPERATURE,
&present_temperature);
        present_current = (short int) present_current

        printf("Present Position\n");
        printf("ID_1 : %d\n", present_current);
        printf("ID_2 : %d\n", present_position);
        printf("ID_3 : %d\n", present_temperature);


        //Close the port of USB2DXL
        printf( "Press any key to terminate...\n" );
        _getch();
        dxl_terminate(Port);
        return 0;
}
```