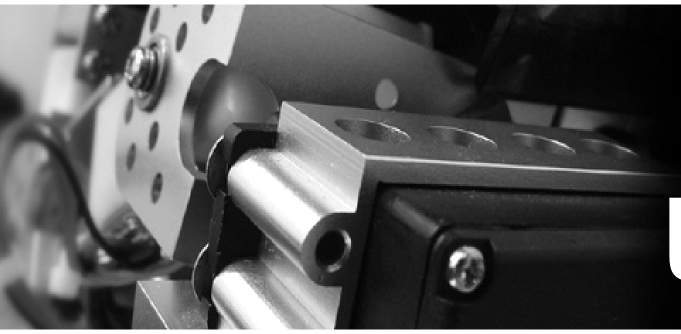
Ver 1.00

Closer to Real ROBOTIS

BIOLOID

User's Guide



Contents

1. Before Starting

1-1. A Word of Caution

A Note on Safety

The user is responsible for any accidents that occur while building the robot. Before starting, please remember the following.

- Read and study the manual before starting.
- The recommended age for this product is 12 years and older. Those under 15 years must work under supervision.
- Only use the recommended tools and do not use any dangerous tools, such as knives or drills.
- Do not work on this product if you are feeling sick or feel fatigue, and especially under influence of alcohol.
- Keep the robot away from your face.
- Keep the robot or its parts away from children.
- Be careful not to get your finger be caught between the joints.
- The product is not waterproof so be cautious when handling near water.
- Only operate the robot indoors.
- Do not operate or store it in under direct sunlight.
- Do not operate or store it near open flames or in humid environments.

Robot Malfunction

If any of the following occurs, immediately turn off the power and contact a supervisor or the company.

- When you see smoke coming out of the product.
- When the LED does not blink after power is connected.
- When water or foreign substances enter the robot.
- When you detect an unnatural smell from the product.
- When the robot is damaged.

Recharge Problem

After powering on, when you connect to SMPS and press Ⓤ button, the Power LED will blink and will began recharging. If there is a problem with recharging, make sure that fuse is not out. [Refer to the QuickStart for exchanging a fuse]

Notes Please note the following.

- The beginner should not use self-made cables.
- Only use the right size screw drivers.
- Do not use excess force when tightening the bolts or assembling the parts.
- Turn off the power immediately to avoid damage to the robot if a joint gets twisted caused by the inappropriate motion settings during development.
- If this is the first time you are building a robot, please build a robot in the QuickStart following the instructions. A custom-built robot should only be attempted if you have at least six months experience with the robots.
- To prevent the robots from falling, do not place the robot on a high location such as on top of a table or desk. Always operate the robot on the ground. If the robot is damaged due to a fall, it will be ineligible for free repair.
- The joints of the robot and the gears inside the Dynamixel are susceptible to wear. After a period of time, the backlash of the robot will increase, especially if excessive load is applied.
- When operating the robot with the SMPS, make sure the robot doesn’ t fall and refrain from excessive movements. This can cause the SMPS cable to break.

Recommended Tools

- Phillips head screwdriver: M2 size
- Flat head slotted screwdriver: Use if the bolt’ s groove wears out. Do not use any other tools. The use of dangerous tools can cause accidents.

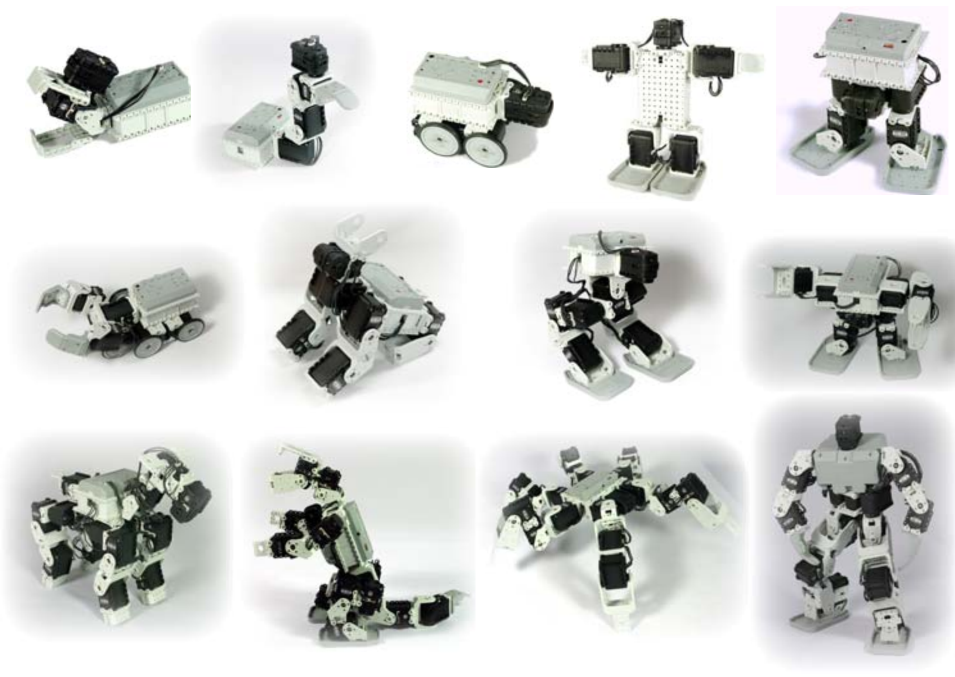
Building a Robot

- Don’ t attempt to make a robot with more than ten joints if you are a beginner. You may need many practices before trying to build a complex robot.
- This User’ s Guide shows how to build the Robot Arm (3-degree of freedom), and the Walking Droid(4- degree of freedom).

1-2. What is Bioloid?

Bioloid The Bioloid is a robot kit where the user can build anything they desire, just like the Lego sets. But unlike the Lego sets, the robot is built with blocks that are actuated, so the joints can move. The name “Bioloid” comes from the words “Bio” + “all” + “oid” meaning that any living thing can be built in the form of a robot.

The following are some examples of what can be built with the Bioloid kit. In addition to below, many other forms of robots can be built.



[Examples of Bioloid Robots]

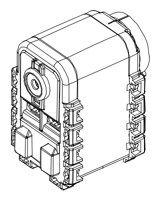
Function With the use of a distance sensor, sound sensor, and feedback from the joints, the robot can be programmed to operate autonomously. For example, you can build a robot dog that gets up when it hears a clap and sits down when it hears two claps, or a robot that bows when a person comes close. You can also make a robot that avoids obstacles or a robot that plays with a ball. A robot that can move by the pressing of buttons or by using the remote control (option) can also be built. Using the provided software, even people without a background in robotics can easily program these kinds of robot movements.

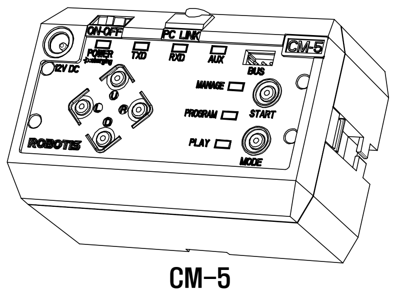
1-3. Things to Understand Before Starting

Before putting the robot together there are a few basics that you have to understand. First of all, let’ s study about the Bioloid’ s hardware and software. The terms used here will be mentioned often throughout the manual so it is important that you understand them.

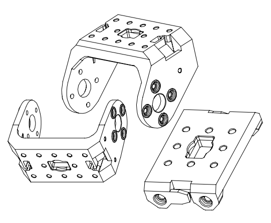
Hardware The hardware of the Bioloid consists of three types.

- Dynamixel: This is the basic unit of the Bioloid which acts as a joint or a sensor. The AX-12 Dynamixel is an actuator that is used as a joint. The AX-S1 Dynamixel is a sensor unit that can sense both distance and sound.
- CM-5: This is the main controller of Bioloid robot. Batteries that are placed here supply power to the connected Dynamixel.
- Frame: The frame connects the robot units. The Dynamixel can be connected together with the use of the frame. Also, the frame connects the Dynamixel and the CM-5.


Dynamixel



CM-5


Frame

Software There are three pieces of software supplied for use with the Bioloid.

- Behavior Control Programmer: This is used for creating a program that controls the robot behavior. The program is used to implement the motion the robot follows according to the information received through input devices such as sensors.
- Motion editor: It is not easy constructing the complex motion of multi-joints robots with the Behavior Control Programmer only. Motion Editor is software that helps the creation of robot motions and that calls it whenever it is required.
- Robot Terminal: This is a type of a serial communication terminal program where advanced users often use to see information displayed on the screen sent by the robot and also to send the characters typed on the keyboard to the robot.

Assembling Process

The step of developing a Bioloid robot

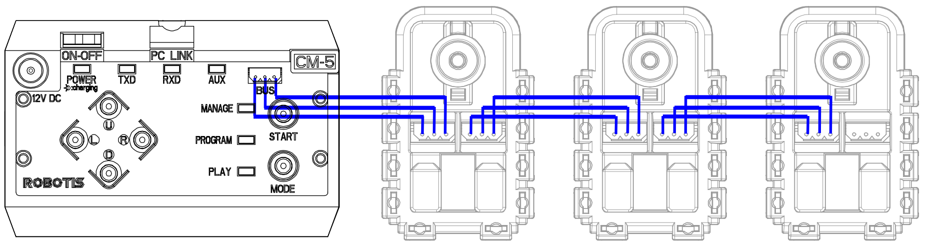
Step 1	Read the User’ s Guide
Step 2	Decide the configuration and function of the robot
Step 3	Connecting the Dynamixel units around the CM-5
Step 4	Cable connection (beware of connector direction and length of the cable)
Step 5	Simple motion verification (use Motion Editor)
Step 6	Behavior control programming
Step 7	Editing the motion (use Motion Editor)

Step 1 Read and completely understand the manual before trying to build a robot. The manual consists of nine chapters and the first five chapters are intended for beginners. The User's Guide presents examples of developing the Robot Arm and the Walking Droid (two-legged robot). Once you have completely understood the User's Guide, you can explore various robot configurations and experiment with the program in depth.

Step 2 This is where you decide what kind of robot you will be building. With the Bioloid kit, you can make all kinds of robots. If you are a beginner, however, we recommend that you first make the robot that is shown in this User's Guide.

Step 3 Step 3 is the building stage. First, connect the Dynamixel units to the CM-5 unit as the center unit. Connect other Dynamixel units to this to create joints and expand to complete the robot. Secure each part with nuts and bolts.

Step 4 After the configuration of the robot is completed, the next step is to connect the cables. Having the CM-5 unit as the center, the wires are connected to, and through the Dynamixel units. Each cable is made up of three wires. Two of the wires are for power and one is for communication. Make sure the cables are long enough so that they allow the joints to bend all the way in either direction. Connect the cables in a daisy chain fashion as shown in the figure below.



Control Box "CM-5"

The steps up to here complete the building of the robot hardware. The configuration of the robot is decided in Steps 1, 2, 3, 4 and the function of the robot will be decided through programming in Steps 5, 6, 7.

Step 5 After you are done building the robot, use the motion editor program to make sure the robot is properly put together. Make sure all the Dynamixels are communicating with the CM-5 unit properly. To check if the joints are working properly, test them by moving each of the joints slightly.

Step 6 In step 6, you will create the behavior control program of the robot. Behavior control is simply telling the robot to take some kind of action when it enters a certain state. For example, when the robot is walking forward through a narrow path, you can make it walk through the center of a path. Or, if there is something blocking the way, you can make it turn around and go the other way. The behavior control program takes the form of rules which defines the appropriate motion the robot should output for specific input information it receives from the sensors.

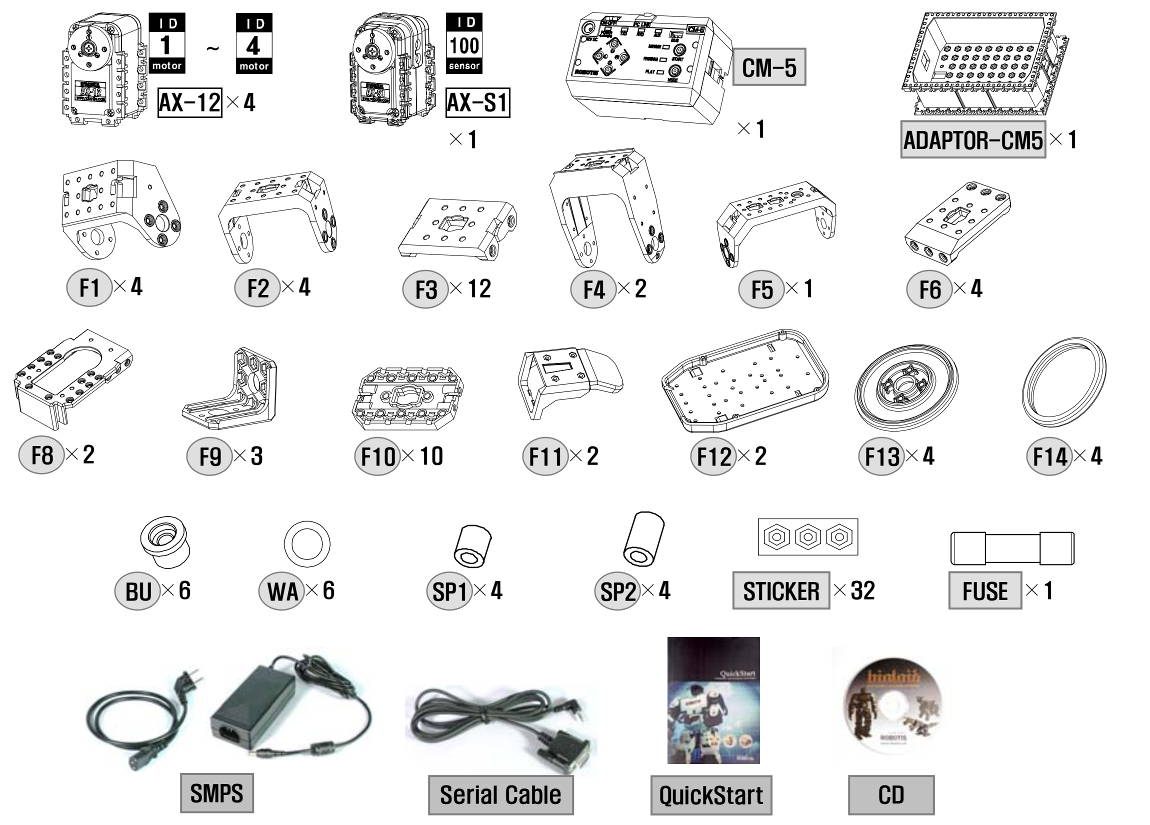
Step 7 In step 7, you will create the motion of the robot. For robots that use wheels, step 7 is unnecessary because all you have to do to move the wheels is to set the position or velocity settings of the Dynamixel. But it would be difficult to make a puppy robot sit or a humanoid robot walk by changing each of its joint angles individually. In order to move a complicated multi-jointed robot, you have to "call" a pre-made movement." This "pre-made movements" (motions) are what you will create here in step 7. The behavior control program of step 6 will be able to "call" these "pre-made movement" (motions) of step 7. Step 7 shouldn't be taken after you are done with step 6, but rather together with step 6.

PC Requirements

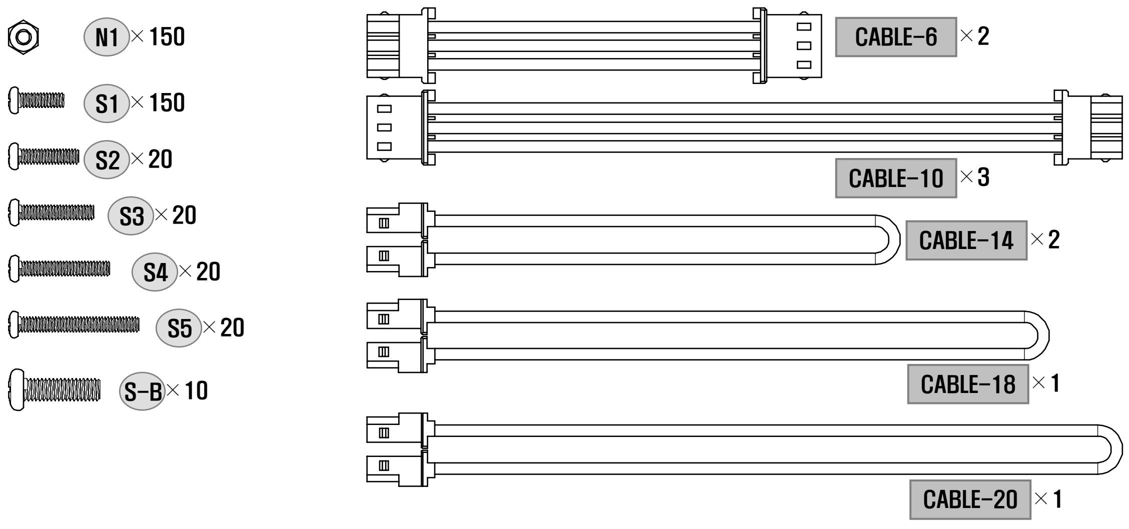
▪ PC : IBM compatible	(Required)
▪ OS : Windows 2000 or Windows XP	(Required)
▪ CPU: Intel Pentium III 1GHz or AMD Athlon XP 1GHz or higher	(Recommended)
▪ RAM: 256MB or higher	(Recommended)
▪ Graphic Card : 3D acceleration function (Direct 3D supported)	(Required)
▪ HDD free space : at least 300MBytes	(Recommended)
▪ Direct X 8.0 or higher	(Required)

1-4. Package Contents

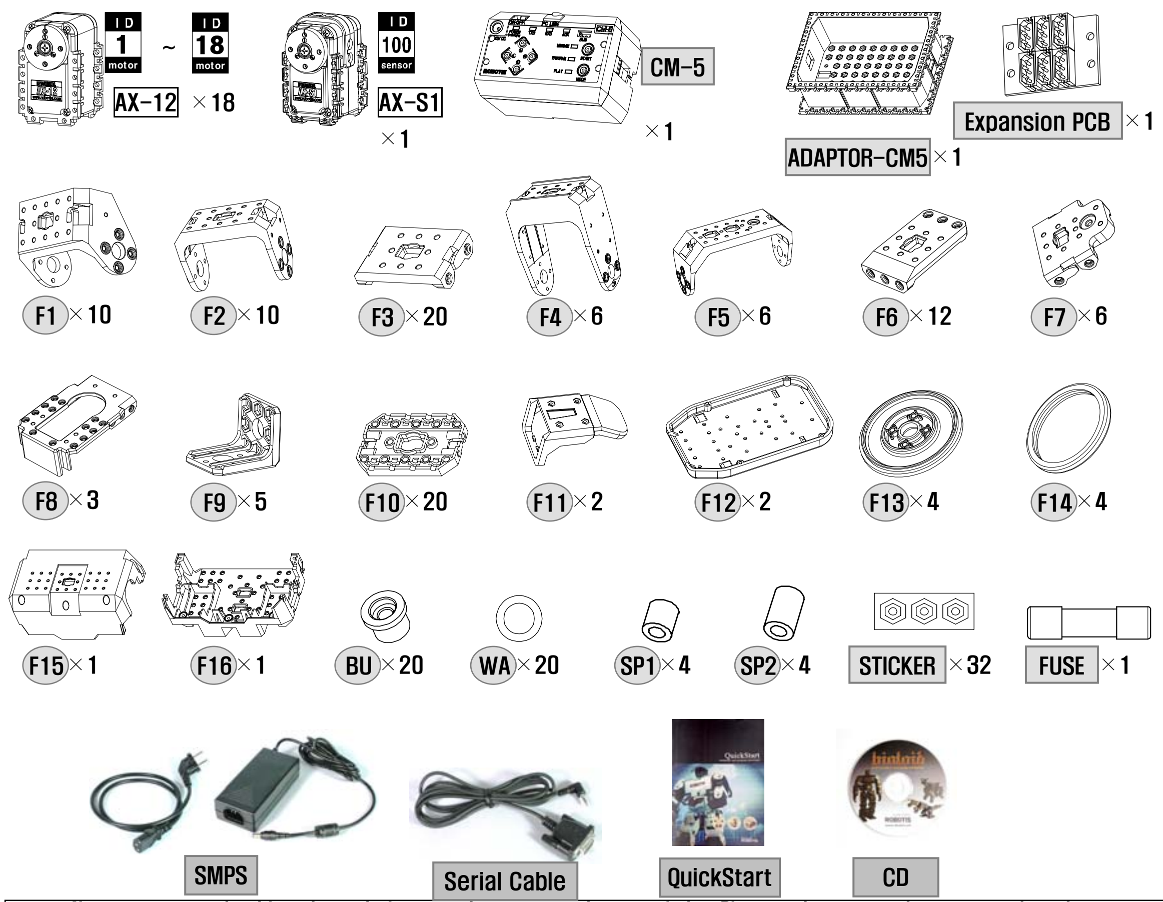
(Beginner kit' s parts)



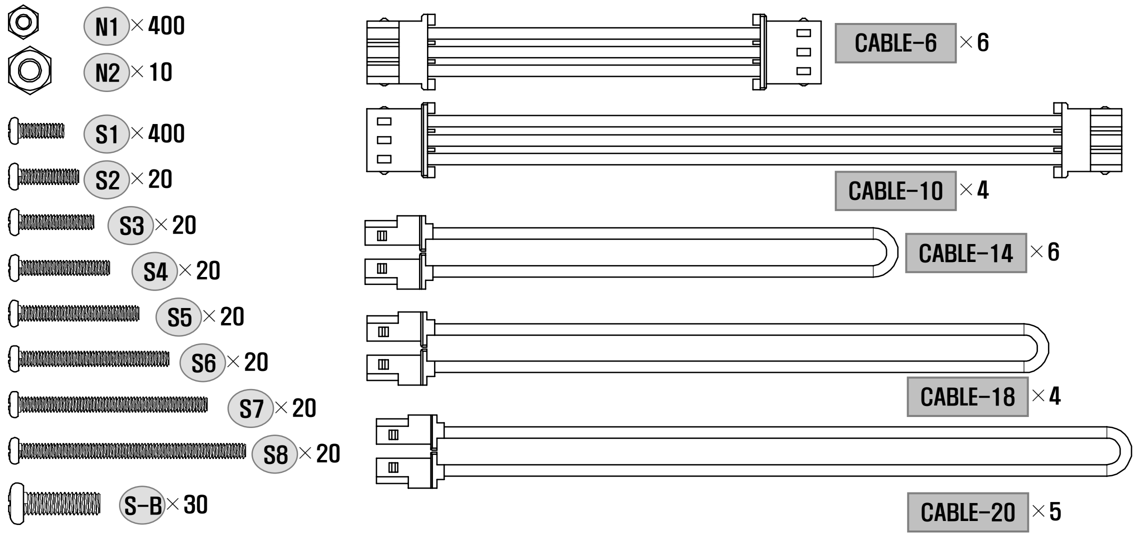
※ Nuts, screws, and cables shown below are the same as the actual size. Place and measure the parts against the below illustration to choose the correct assembly part.



[Comprehensive kit' s parts]



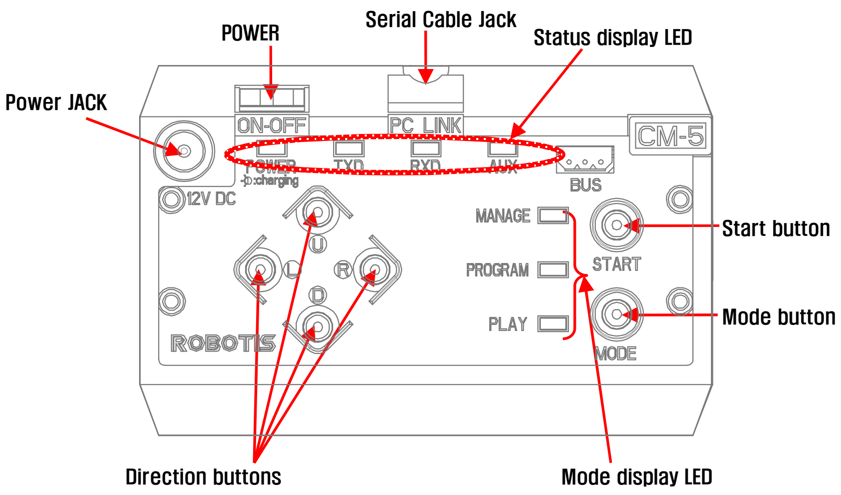
※ Nuts, screws, and cables shown below are the same as the actual size. Place and measure the parts against the below illustration to choose the correct assembly part.



2. Learning the Basic Operations

2-1. The CM-5 and Its Operation Mode.

CM-5 The CM-5 is the main controller for the Bioloid. As mentioned previously, the robot is built by connecting the Dynamixels to the CM-5 as the central unit. In order to understand how the Bioloid works, you first have to understand how the CM-5 operates.



[Top view of the CM-5

Applying Power Let’ s now apply power to the CM-5 unit. Plug the SMPS into the power jack on the upper left corner. Then turn the power switch on. One of the mode display LEDs should be blinking. As you press the mode change button, the mode LED will change sequentially. Currently, it is in standby mode.

Operating Modes The operating modes of the CM-5 unit is as follows

- Manage mode: This is used when you want to know the status of the CM-5 unit or the Dynamixels, or when you want to test the motion. This mode should only be used by advanced users who are very confident with operating the robot.
- Program mode: The mode used for editing the motion.
- Play mode: The mode used for running the behavior control program created.
- Standby mode: The mode before running the other three modes.
- Charging mode: In standby mode, if the SMPS is connected, battery charging will begin when you press the Ⓤ button.

Execution If you press the start button during standby mode the CM-5 unit will go into the mode that you have selected. To go back into standby mode you can press the mode change button or turn the power switch off and then back on again.

TIP The mode change button is the reset button for the CPU inside the CM-5 unit. Therefore, when the power is on, the CM-5 will go back to standby mode whenever the mode change button is pressed.

Serial Cable In order to communicate with a PC, the CM-5 unit has to be connected to it using a serial cable.

- When using a laptop: Most laptops do not have a serial port, thus you will have to purchase and use a USB to serial converter device. USB2Serial can be purchase at local computer stores.

Status Display LED

There are the four LEDs that indicate the status of the CM-5 unit. The definitions of each are as follows.

- Power: If the power is on, this LED will be on. The LED will blink when the batteries inside the CM-5 unit are charging. Recharging starts when the SMPS is connected to the power jack and the ⏻ button is pressed in standby mode.
- TXD: This LED is on when the CM-5 unit is transmitting data.
- RXD: This LED is on when the CM-5 unit is receiving data.
- AUX: This LED is assigned for user programming. It can be turned on or off with the behavior control program.

Direction Buttons

These buttons are also assigned for user programming like the AUX LED.

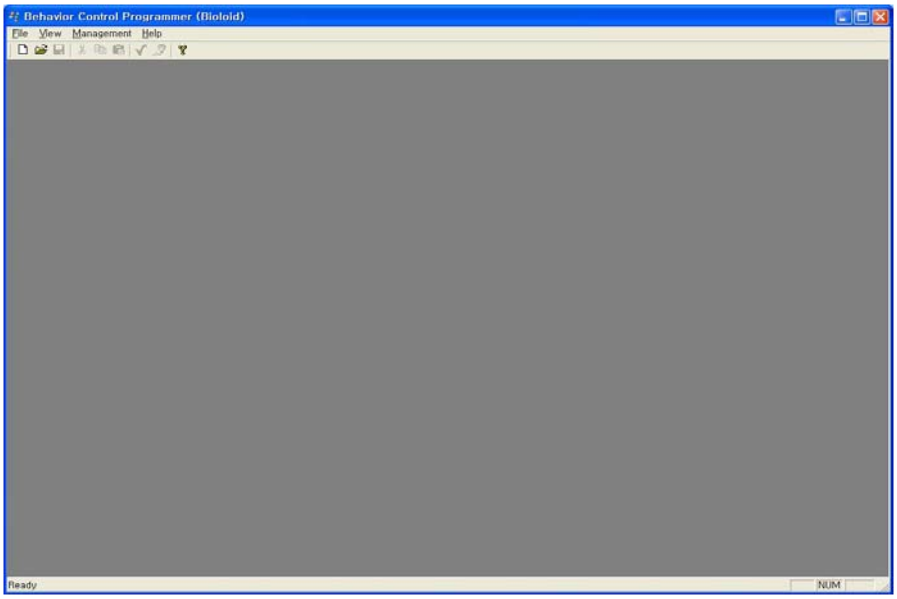
2-2. Behavior Control Program

A robot is a machine that can behave in various ways. However, it can do so only when there is a program that tells how the robot should act for a certain situation. This program is called the "behavior control program." A behavior control program is a series of rules that define the action a robot should take for the given state. Insert the provided CD into the PC and install the software used for creating the behavior control program.

Installation **You can install following three programs.**

- Behavior Control Programmer
- Motion Editor
- Robot Terminal

The following screen appears when you run the behavior control programmer.


Behavior Control Programmer (Bioloid)
File View Management Help
Ready
NUM

Input and Output The behavior control program takes the form of a series of rules that mutually connect the input and output. For example, let' s say that we want to build a robot dog that stands up when you clap once and sits down when you clap twice. The input item (clap once or twice) and the output item (stand up or sit down) have to be predefined. Also, behavior rules need to be defined that tells what behavior to output for the given input item (clap once or twice).

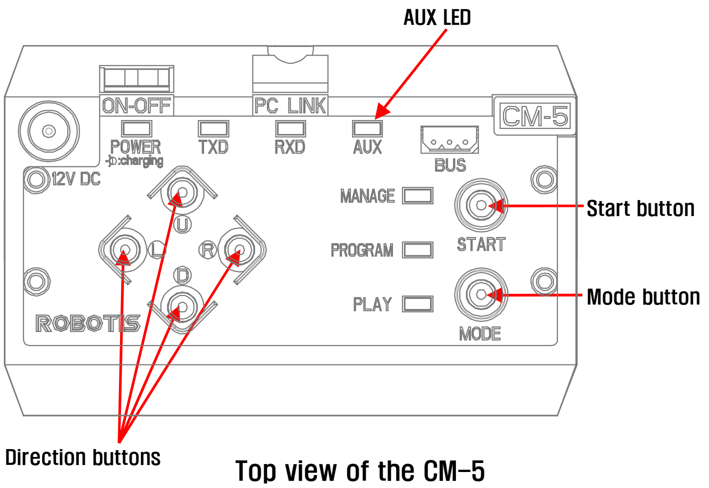
The followings are the general expression sentences for these behaviors.

- If input item 1 occurs, then execute output A.
- If input item 2 occurs, then execute output B.

Thus, understanding what types of input and output items are available is important to writing a behavior control program. Learning how to use the Bioloid is to learn about the input and output items. Let' s practice creating a simple behavior control program using the CM-5 unit' s input and output items now.

2-3. Examples of Simple Behavior Control

There are several buttons and LEDs on the CM-5 unit that is user definable. Five out of the total six buttons (except the mode change button) and the AUX LED can be used for such purposes in Behavior Control Programmer.

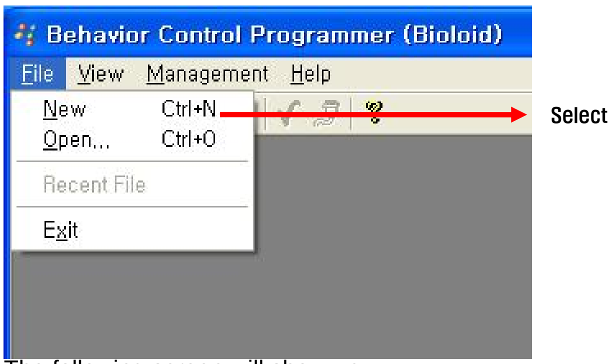


Top view of the CM-5

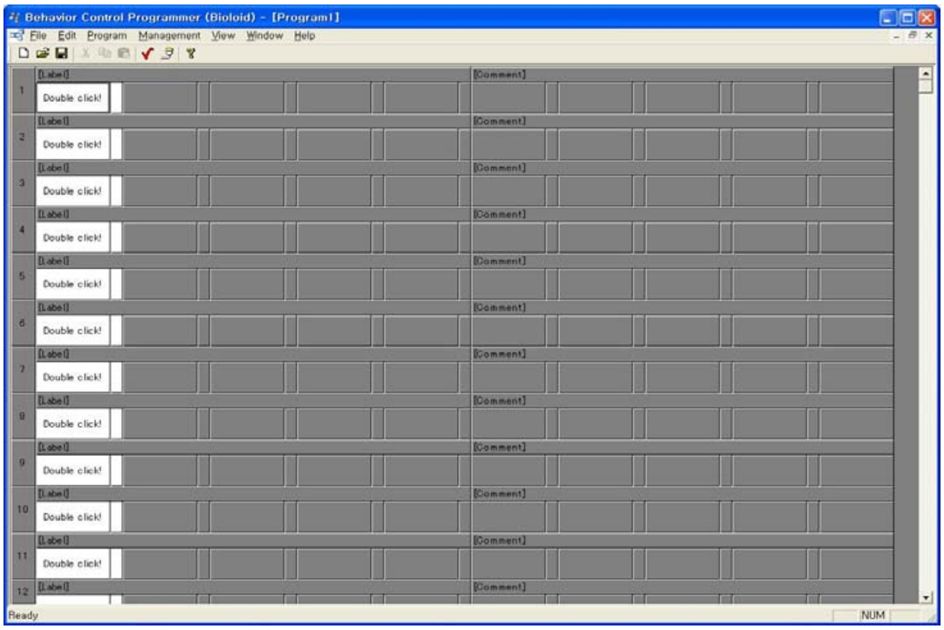
Let's create a behavior control program as the following.

- Make a program that will turn on the AUX LED when the Ⓤ button is pressed and turn off the AUX LED when the Ⓓ button is pressed.

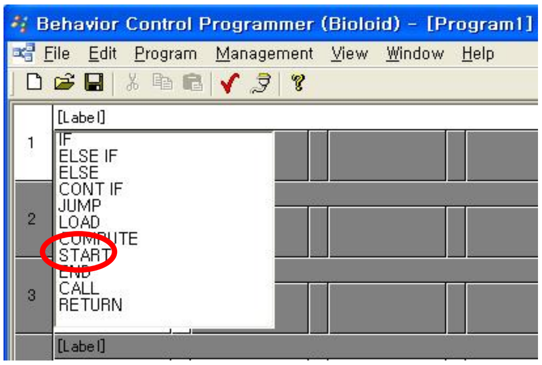
Create a new file from the Behavior Control Programmer, as shown in the figure below.


Behavior Control Programmer (Bioloid)
File View Management Help
New Ctrl+N
Open... Ctrl+O
Recent File
Exit
Select

The following screen will show up.


Behavior Control Programmer (Bioloid) - [Program1]
File Edit Program Management View Window Help
[Label]
Double click!
[Comment]
Ready
NUM

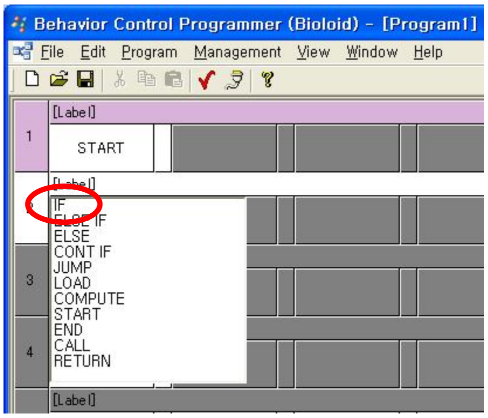
"Start" Left double-clicks the first cell in the Behavior Control Programmer. The following commands should appear. Select "START."


Behavior Control Programmer (Bioloid) - [Program1]
File Edit Program Management View Window Help
[Label]
IF
ELSE IF
ELSE
CONT IF
JUMP
LOAD
COMPUTE
START
END
CALL
RETURN

The "START" command does not instruct a specific action, but rather it tells that the program is starting. You can see that the first cell is used for commands. There are around ten commands that are available for the behavior control program and we have already learned one of them.

"If" The command that we want to input is "if the Ⓤ button is pressed."

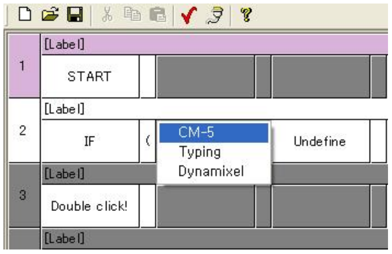
Double-clicks the first cell of the following line and select the command "If."


Behavior Control Programmer (Bioloid) - [Program1]
File Edit Program Management View Window Help
[Label]
1
START
IF
ELSE IF
ELSE
CONT IF
JUMP
LOAD
COMPUTE
START
END
CALL
RETURN
3
4

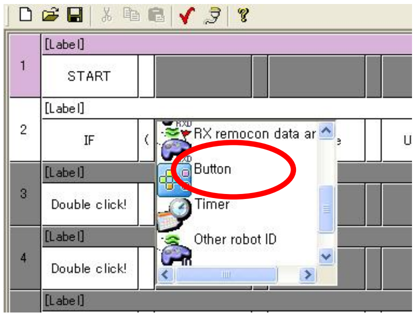
Parameters There are some commands that need to be used together with a number or symbol. For example, the command "START" does not require any number or symbol to be used with, but the a command like "compute 1 + 2" requires the numbers "1" , "2" and the symbol "+." The command "If" compares two items so it requires several parameters. Let' s implement the input command "IF Ⓤ is pressed" using several parameters.

- "IF (button status = Ⓤ button) then"

As mentioned earlier, in order to understand and program a behavior control program, you have to learn about the available input and output items. Here, we have to find the "Ⓤ button" input item that has been used as a parameter.
To select the button status parameter, double-clicks on "undefined" and select CM-5. Then select the "CM-5 button" item.

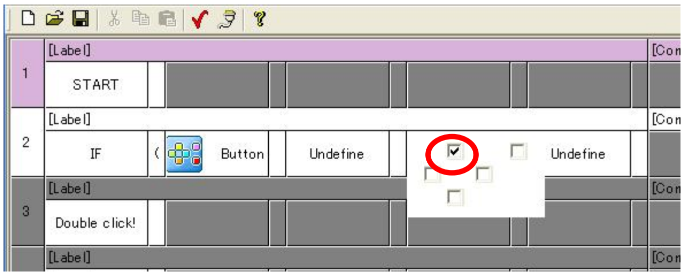


[Select CM-5 first.]

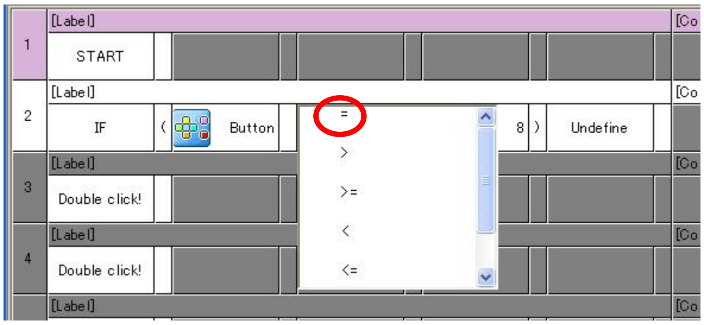


[Then select item]

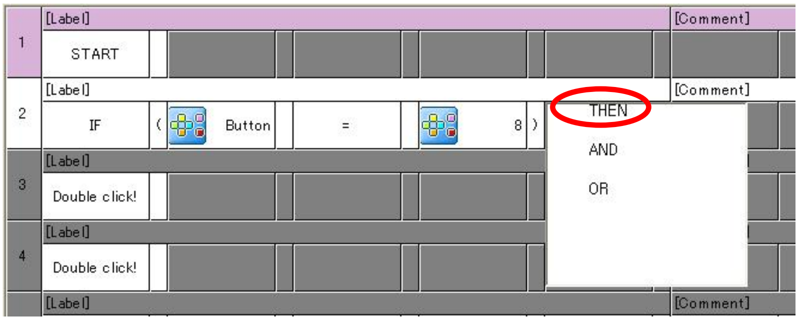
Now select the Ⓤ button. Double-click the parameter on the opposite side and check the location of the Ⓤ button.



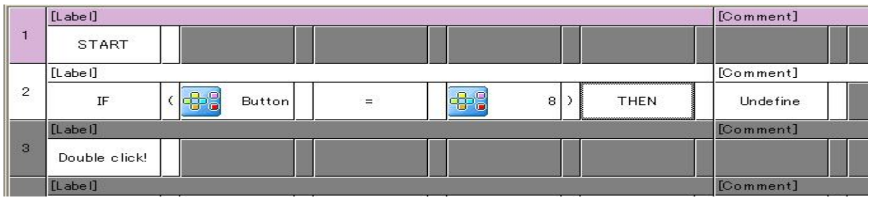
The next step is to select the operator that compares the two parameters. Select the equal sign.



Finally, select the undefined item and double-click “THEN.”



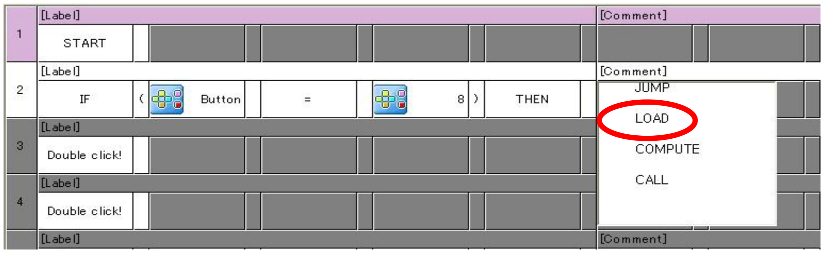
If you have done everything correctly, your screen should look like the figure below.



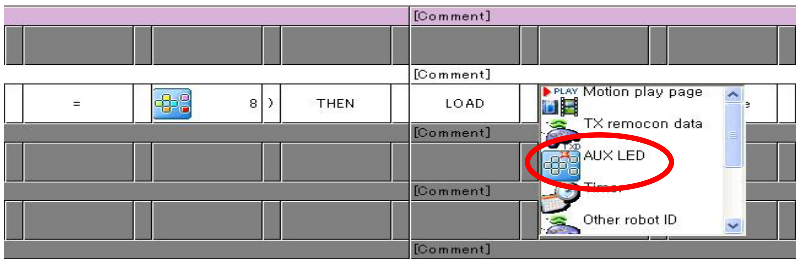
0|

Now we have finished making the complete command sentence, "If (button status = Ⓤ button), then." It seems difficult at first but after practicing several times it should become natural.

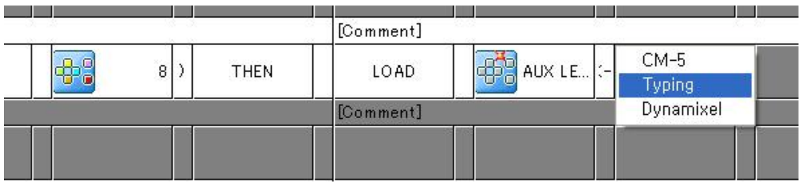
"Load" The behavior control program expresses the command, "Turn on the LED" as "Load LED ← 1." Loading 1 mean "ON" and loading 0 means "OFF" for the LED. Double-click the undefined cell and type in the command sentence. Select the "LOAD" command.



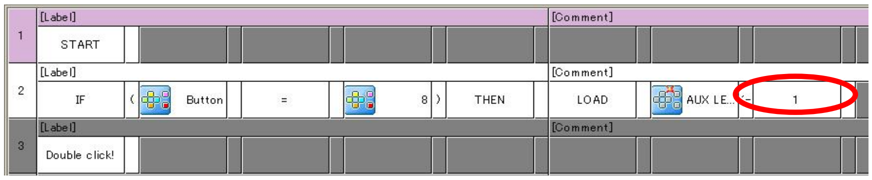
Select the "AUX LED" item from the CM-5 item as the left parameter of "LOAD."



For the right parameter a number is needed, so first select "INPUT" and then type in the number 1.

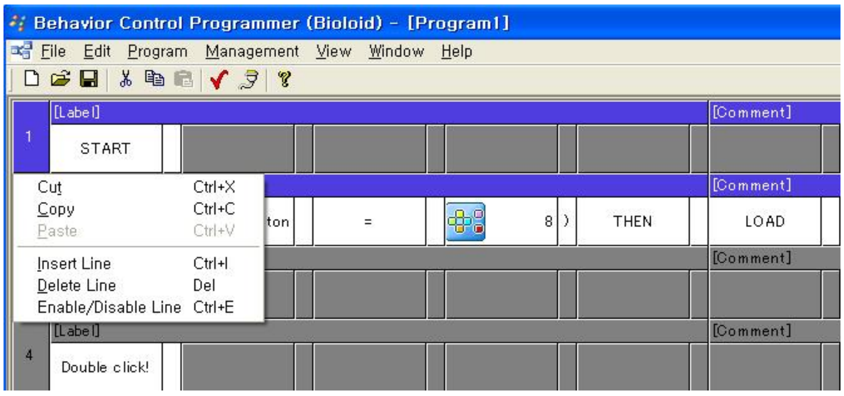


그
러
You have finished creating the following command.



Editing 행동제어 프로그램을 작성하다가 보면, 문장 전체, 심지어는 몇 개의 문장을 통째로 지우거나 복사

Select the number of the sentence that you want to edit. If you press the shift button and click on a sentence and then another sentence down the list, then both sentences and all the sentences between the two will be selected altogether. After you have selected the sentences that need to be edited you can right click to display to the edit menu as shown below.

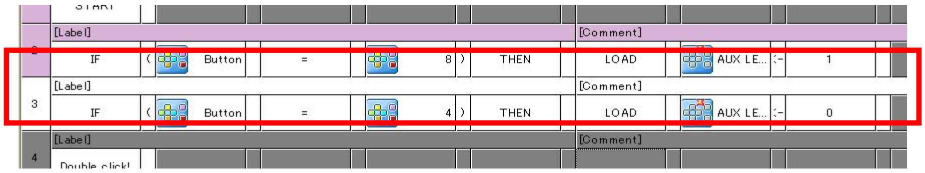


Let' s create the second behavior control sentence using the same method as we used for the first one.

- "If you push the Ⓓ button the LED will turn off."

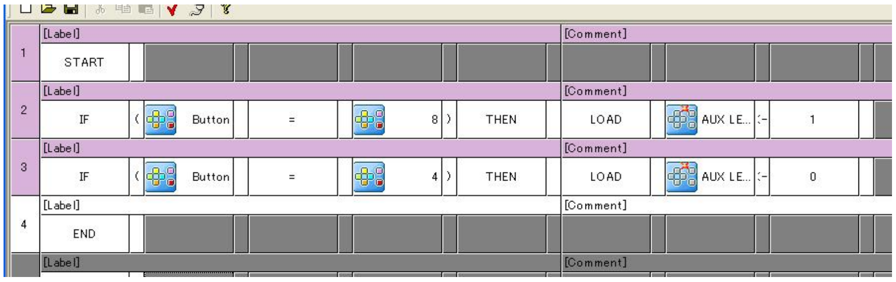
This can be expressed in a behavior control program as the following.

- If (CM-5 button = Ⓓ button) then LOAD CM-5 LED ← 0.
  If you do everything correctly it should look like this.



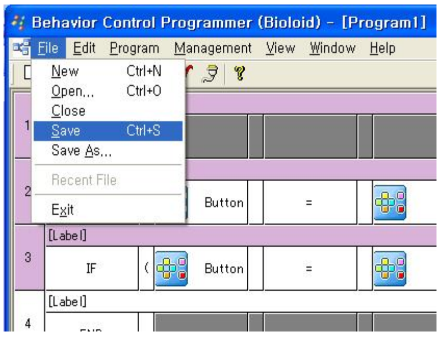
2	IF	(	Button	=	8	)	THEN	LOAD	AUX LE...	<-	1
3	IF	(	Button	=	4	)	THEN	LOAD	AUX LE...	<-	0
4	Double click!										

Finally you have to state the end of the program. Everything that you have done so far should be like the following.

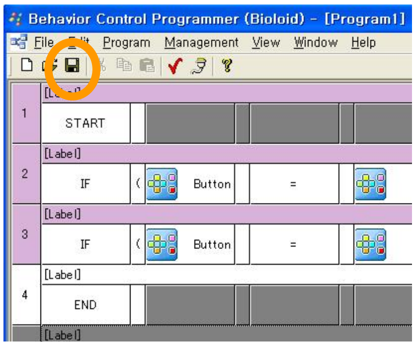


1	START										
2	IF	(	Button	=	8	)	THEN	LOAD	AUX LE...	<-	1
3	IF	(	Button	=	4	)	THEN	LOAD	AUX LE...	<-	0
4	END										

Save Let' s save everything that we have done so far. To save a program, you can either use the menu or click on the tool bar icon. Saving is done in the same way as in any other computer program. The figure below shows how this is done.

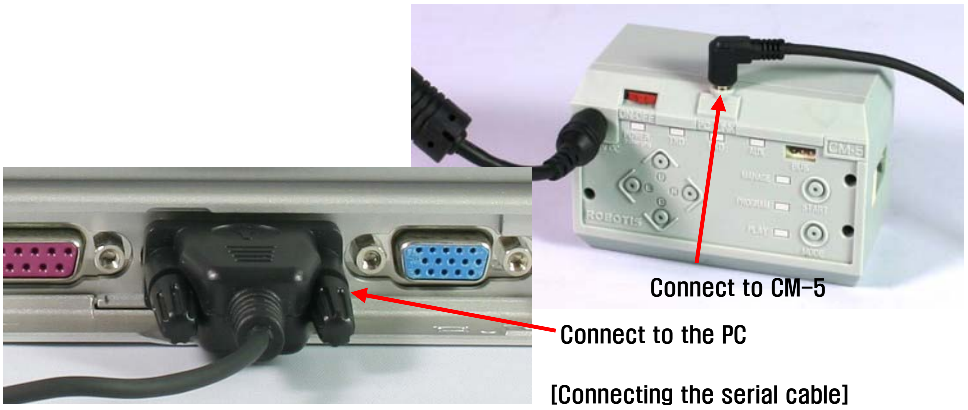

Behavior Control Programmer (Bioloid) - [Program1]
File Edit Program Management View Window Help
New Ctrl+N
Open... Ctrl+O
Close
Save Ctrl+S
Save As...
Recent File
Exit

[Saving with the use of the menu]


Behavior Control Programmer (Bioloid) - [Program1]
File Edit Program Management View Window Help

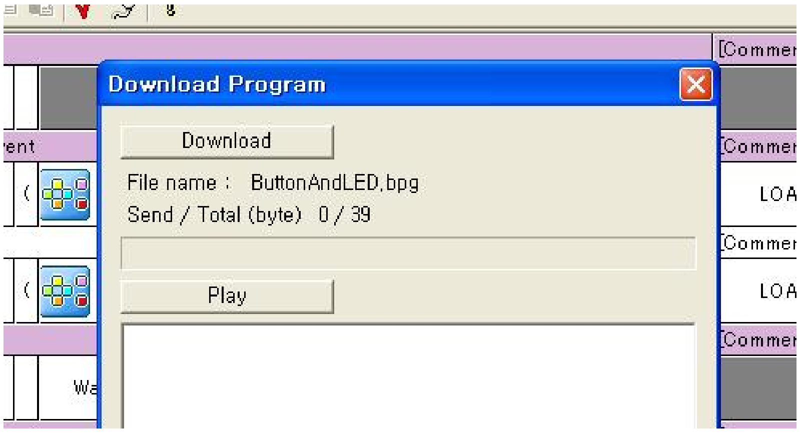
[Saving by clicking on the toolbar]

Download To download the behavior control program onto the robot, the robot and the computer needs to be connected to each other with a serial cable as shown in the figure below. To download the behavior control program, the Bioloid robot has to be turned on.



[Connecting the serial cable]

Click on the download toolbar icon. The following dialog box will show up.



Download Program

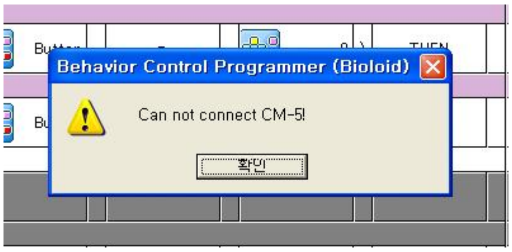
Download

File name : ButtonAndLED.bpg

Send / Total (byte) 0 / 39

Play

If the CM-5 unit and the behavior control programmer are not connected properly, then the following error message will show up.



Behavior Control Programmer (Bioloid)

Can not connect CM-5!

확인

In such cases check the following in order.

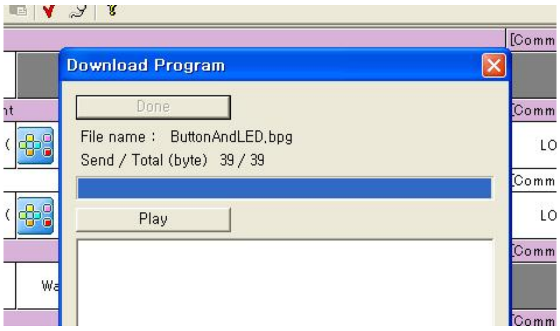
- Is the serial cable connected properly?
- Is the robot' s power on?
- Is any other program running?

- Is the COM port on your PC set up properly?

  This is how to set up the COM port on your PC.

- With the serial cable disconnected, click on the COM port set up button, which is on the bottom left corner of the program download screen.
- Select the correct COM port that the cable is connected, such as COM1 and COM2.

Finish If the download is finished successfully, the following screen will appear and the robot will be in standby mode. If you press the play button, the behavior control program that was just downloaded will execute.



File name : ButtonAndLED.bpg
Send / Total (byte) 39 / 39

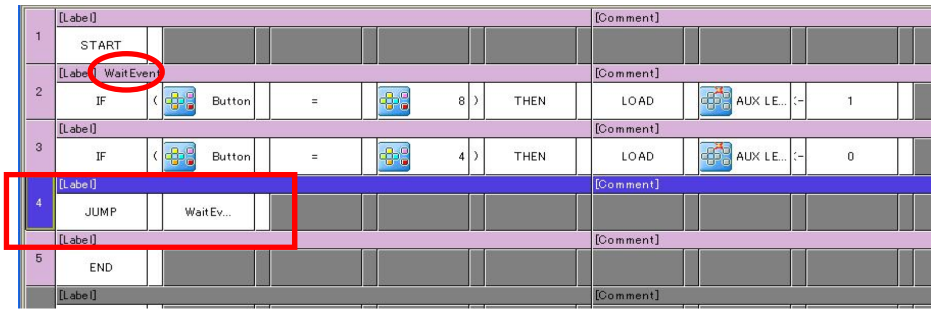
How did the program run?

It may seem like the program didn't run at all and the robot is still in standby mode. But actually, the robot did indeed enter the play mode for a very short period of time and then exit back to the standby mode.

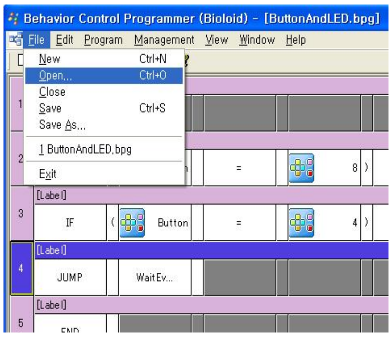
Program Editing

If you think about it carefully, you should realize that this is exactly what was supposed to happen. After the "START" command, two "IF" statements were executed and then the program was terminated.

Jump For the program to work properly, the two "IF" statements need to be repeated continuously. For the second statement (the first "IF" statement), type in an appropriate name in its label. Then, insert a line between the third statement (the second "IF" statement) and the fourth statement ("END") and create a "JUMP" command. In the parameter after the "JUMP" command, type in the name of the item of the line you want the program to jump to.



1	[Label]						[Comment]		
	START								
2	[Label] WaitEvent						[Comment]		
	IF	(Button	=	8)	THEN		LOAD	AUX LE... <-	1
3	[Label]						[Comment]		
	IF	(Button	=	4)	THEN		LOAD	AUX LE... <-	0
4	[Label]						[Comment]		
	JUMP	WaitEv...							
5	[Label]						[Comment]		
	END								



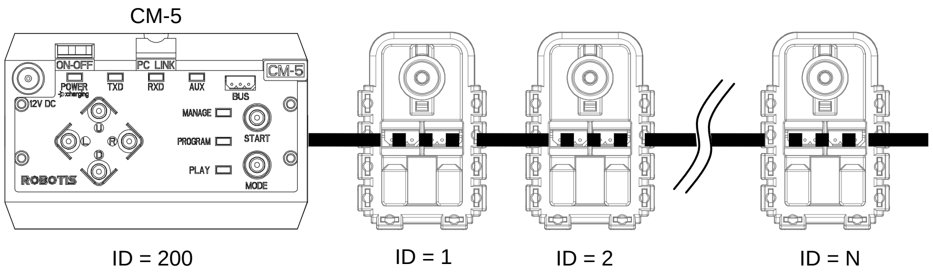
Now download the program again and see how it executes. If it does not work properly, go over the program and check if there are any errors. If it still doesn't work then open and download [Examples\Example[Button and LED.bpg] from the provided CD. This file is the source file for the example shown above.

TIP We have pressed the play button and execute the program by entering the play mode while the PC was connected to the Behavior Control Programmer. When a PC is not connected, you can execute a program by manually entering the play mode by pressing the mode change button and then pressing the start button.

2-4. Understanding ID, Address, and Data

So far we have learned how to create a simple behavior control program. Now let' s systematically learn about the available input and output items.

ID As mentioned previously, the Bioloid is made up of Dynamixel units and a CM-5 unit. The input and output items all exist inside these devices. All of these robot modules are connected to one bus and each of them have their own unique ID.



Address Each Dynamixel units has many input and output items. To access them, each item is numbered in consecutive order. These numbers are called addresses.

Data Data is the value of each input and output item. If you look at the behavior control program created previously (Button and LED). bpg, you can see that the LED is located in the LED item (Address 24) of the CM-5 (unit with ID = 200). We have made an "If" statement to see if the data it contains is 8 (the data value for the Ⓤ button). Let' s review the behavior control program again and look at the "If" statement with this fact in mind.

Setting the input and output items means specifying the ID and address.

The Bioloid has three units: CM-5, AX-12, and AX-S1. Let take a look at the available input and output items.

ID Assignment The following are the IDs that are assigned to each unit.

- AX-12: ID = 1 ~ 19 (allowed range is 0~30)
- AX-S1: ID = 100 (allowed range is 100~109)
- CM-5: ID = 200

For the AX-12 and AX-S1, you can set or reset the ID as needed. The AX-12s that come with the Bioloid already have their IDs set in sequential order, but the AX-12s that come directly from the factory have their IDs set to 1. If you have a problem with one of these units and have to order a new one, you have to reset the IDs accordingly. To reset the ID, refer to the "Management Mode," in User's Guide or 3-3. Changing Dynamixel ' s ID in QuickStart and "Help Files\ID changing .wmv " video clip.

CM-5 input output items and their addresses

Address	Icon	Name	Function	input/output
24	PLAY	Start robot motion	The motion will start when the page number is loaded	input/output
25	STATUS	Playing robot motion	During motion play:1, otherwise:0	input/output
26	TXD	Wireless data to be sent	Wireless data to be sent	input/output
28	RXD	Received wireless data	Received wireless data	input
30	RXD	New wireless data	Arrival of new wireless data [When received: 1 will change to 0 after the received data is read	input
31		AUX LED	Loading 1 will turn it on and loading 0 will turn it off	input/output
32		CM-5 button	The value changes depending on the five buttons on the CM-5 pressed	input
33		Timer	When a value is loaded, the value decreases by 1 every 0.1 seconds	input/output
34	ID	Wireless ID of another robot	The wireless ID of the robot that you want to communicate with	input/output
36	ID	Wireless ID of my robot	The wireless ID of my robot [cannot be changed	input
37		Print screen	If you load a value here, it will be displayed on the screen	output
38		Change lines after printing screen	If you load a value here, it will be displayed on the screen and the line on the screen will change	output

You do not have to understand all the items at the beginning. You do not have to memorize all the address numbers to create a program since most items have icons and names. You will learn the items of the CM-5 unit one by one as you read this manual. You don' t have to understand all of the items for the AX-S1 and AX-12 units either when you first start learning about Bioloid. Just refer to the manual when necessary.

AX-S1 input output items and their addresses

Address	Icon	Name	Function	Input, Output
26	LEFT	Value of left side distance sensor	Value for sensing distance of left side distance sensor	Input
27	CENTER	Value of center side distance sensor	Value for sensing distance of center distance sensor	Input
28	RIGHT	Value of right side distance sensor	Value for sensing distance of right side distance sensor	Input
29	LEFT	Left side light brightness	Value for left side light brightness	Input
30	CENTER	Center light brightness	Value for center light brightness	Input
31	RIGHT	Right side light brightness	Value for right side light brightness	Input
32		Obstacle sensor	Set if the the value is larger than the standard value [bit-RCL]	Input
33		Brightness sensor	Set if the the value is larger than the standard value [bit-RCL]	Input
35		Sound volume	Value for the sensed sound volume	Input
36	MAX	Maximum sound value	Largest volume sensed so far	Input, Output
37		Number of times sound is sensed	Number of times a sound is sensed, such as the number of claps	Input, Output
38		The time when sound occurred	Time when sound occurred	Input, Output
40		Buzzer scale	0~52 (goes up half a not starting at "La")	Input, Output

41		Duration of buzzer sound	Time interval is 0.1 seconds and a maximum of 50 intervals is possible. If it is 255, then a prerecorded sound will be played (buzzer scal 0~27)	Input, Output
42		Supplied voltage	Supplied voltage X 10 (12V will be read as 120)	Input
43	C F	Internal temperature	The internal temperature of the Dynamixel (Celsius)	Input
46	RXD	Arrival of new information from the remote controller	If new data arrives the setting will be 1 and if the data is read the setting will change to 0	Input, Output
48	RXD	Received remote controller data	The value of received remote controller data	Input
50	TXD	Remote controller data to be sent	The value of remote controller data to be sent	Input, Output
52	≥	Standard value of distance for the distance sensor	Becomes the standard for the data setting of address 32	Input
53	≥	Standard value of brightness for the light sensor	Becomes the standard for the data setting of address 33	Input

AX-12 input output items and their addresses

Address	Icon	Name	Function	Input, Output
24		Turn on motor	Motor torque will engage when set to 1	Input, Output
25		LED	Turns on when 1 is loaded and turns off when 0 is loaded	Input, Output
26	MARGIN	CW Margin	Clockwise compliance range	Input, Output
27	MARGIN	CCW Margin	Counter clockwise compliance range	Input, Output
28	SLOPE	CW Slope	CW compliance slope	Input, Output
29	SLOPE	CCW Slope	CCW compliance slope	Input, Output

30	POS	Destination position	Joint position from 0° to 300° (300° when 1023)	Input, Output
32	SPEED	Speed	Speed when moving (values from 0~1023)	Input, Output
34	TORQUE	Torque control	Set maximum torque (values are 0~1023)	Input, Output
36	POS	Current position	Value of current position (0~1023)	Input
38	SPEED	Current speed	Value of current speed (0~1023)	Input
40		Current load	Value of external load (0~1023)	Input
42		Supplied voltage	Supplied voltage X 10 (12V will be read as 120)	Input
43	C F	Internal temperature	The internal temperature of the Dynamixel (Celsius)	Input
46	MOVING	Existence of movement	1 when executing a move command, otherwise 2	Input

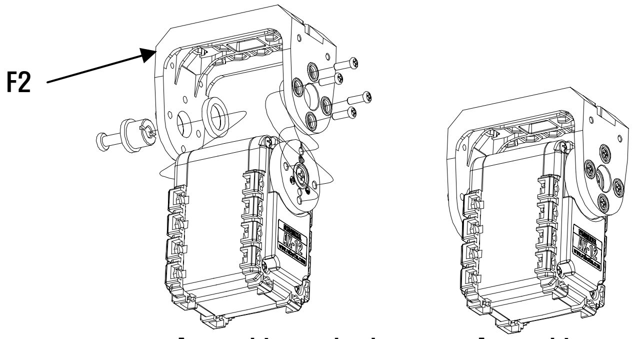
3. Assembling Simple Robot

3-1. Connecting the Frames

In this chapter, we are going to learn the basics of robot assembling. Let' s make the Robot Arm that has three degrees of freedom. Degree of freedom means the number of joints. If the robot has many degrees of freedom it can move in many different ways, however, the robot will also become heavy and slow and will become hard to operate. The recommended size of a Bioloid robot is as follows.

- Robot weight: No heavier than 2 kg
- Robot height: No higher than 350 mm

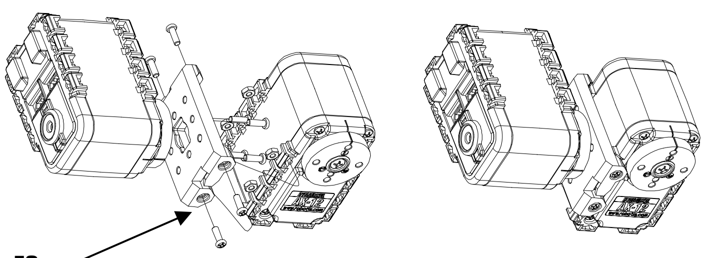
F2 The figure below shows how to attach the most basic frame F2 (hinge) to the AX-12.



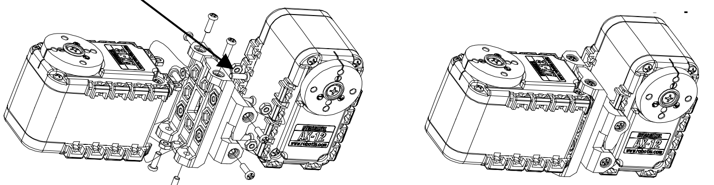
Assembly method Assembly complete

There are four ways to attach the hinge at an angle increment of 90 degrees.

F3



How to connect a horn and a body Assembly completed



How to connect a body and a body Assembly completed

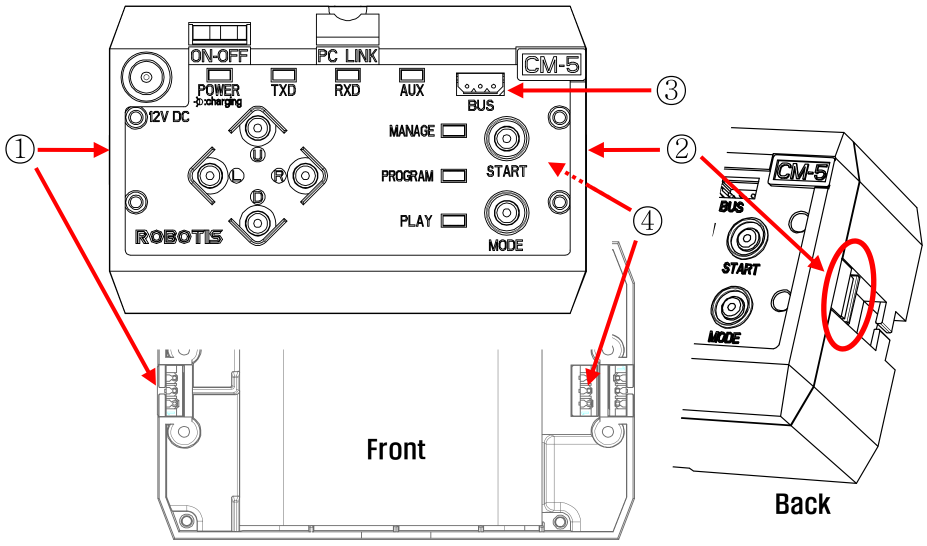
The F3 frame can be connected to three side of the Dynamixel at a 90 degree angle. There are a total of four ways to connect the frame and the Dynamixel. There are other additional ways to construct the frame. Refer to the construction diagram of the robot for more specific information.

3-2. Wiring and Power

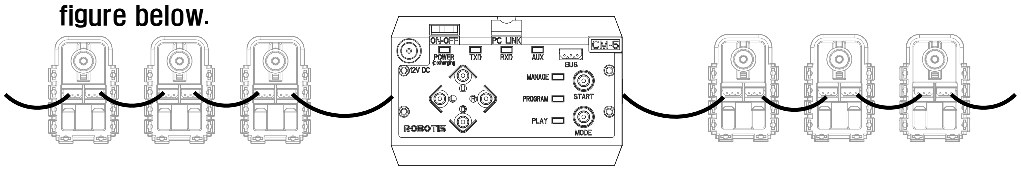
Wiring

Wiring is an important part of building the robot. Majority of the problems that occur with the robot are due to faulty wiring. Severed cable due to insufficient length, cables getting stuck in between the robot' s joints are some of the problems that may occur (latter problem mostly occurs due to users wanting to assemble robot in uncluttered fashion, wiring the cable in narrow joint' s space.) Also, self-made cables are often the cause of problems. Sufficient cables are provided so try not to use your own self-made cable. Connect the Dynamixels to the CM-5 unit. As in the figure below, there are four places on the CM-5 unit where the cables can be connected.

Four wiring locations for CM-5 are illustrated as follows.

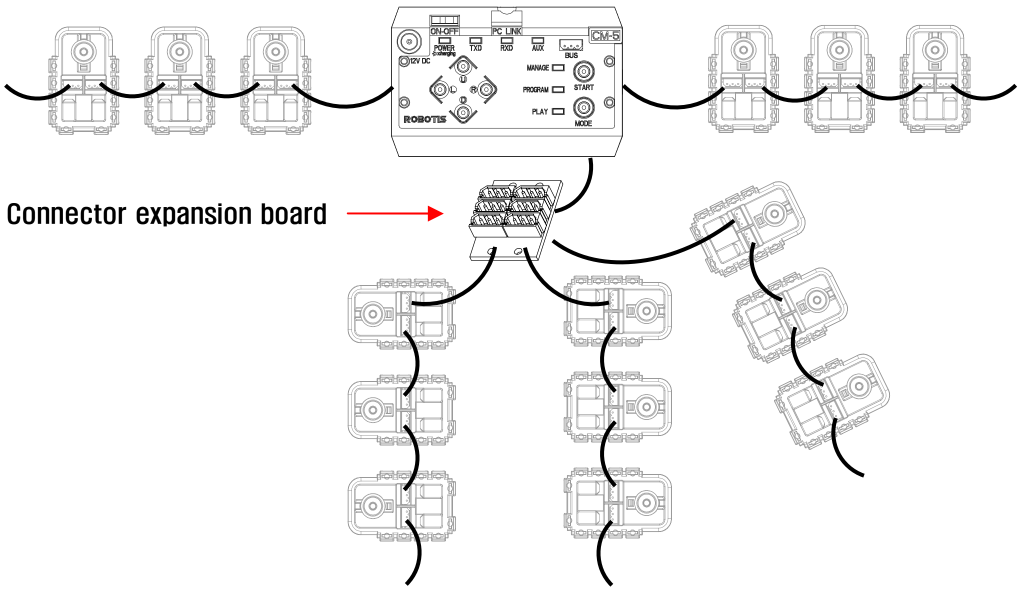


As in the figure above, the number 1 and 2 bus connectors are located on the side of the CM-5 unit and the number 4 bus connector is located on the bottom. The cables should be connected in a daisy chain configuration, as shown in the figure below.



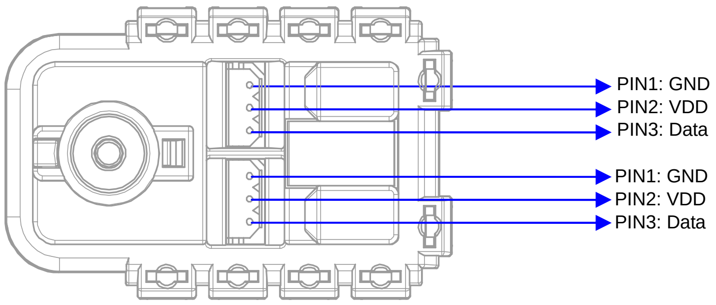
Bus Expansion A robot can have legs, a head, and sometimes a tail. Thus sometimes there will be a need for more 3 line bus connectors on the CM-5. In such cases a bus expansion board can be used (connector expansion board is not included in the beginner' s kit.)

The figure below shows how the connector expansion board is used in wiring.

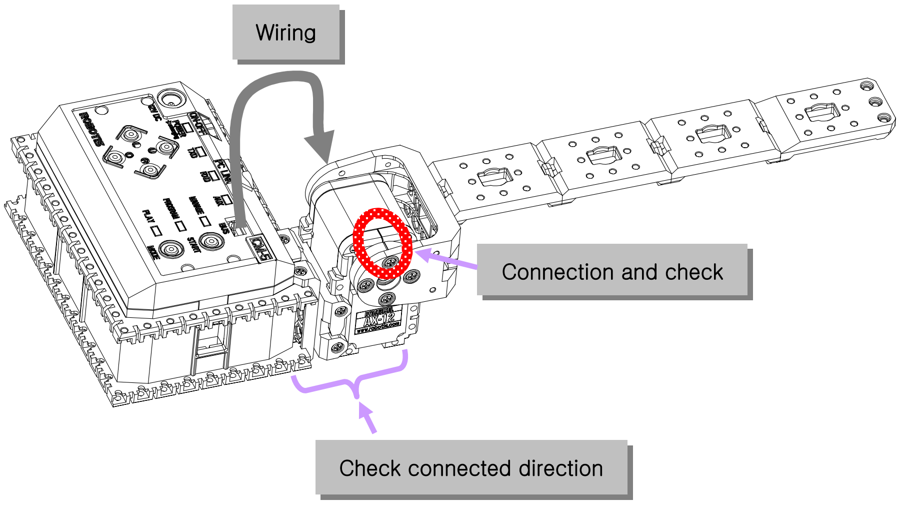


Arrangement of Pins

The figure below shows how the pins on a Dynamixel unit are arranged. Two of the connectors inside the Dynamixel are connected pin to pin. This is how connecting in a daisy chain configuration is possible. Pin1 and Pin 2 are where the wires for the power source are connected. Make sure that the wires are connected securely.



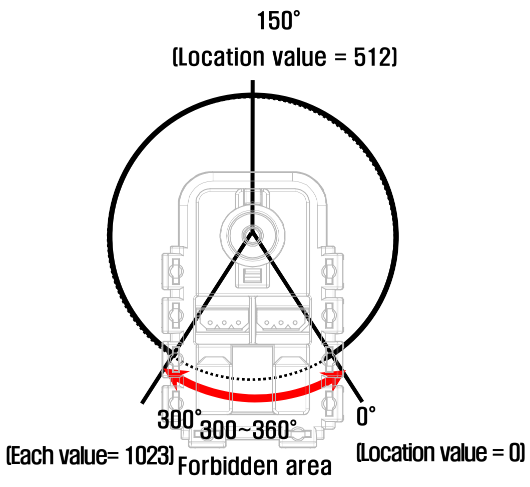
Crossing gate Let’ s build a simple device as shown below. It is a crossing bar where if you press Ⓤ button, the bar goes up and down when you press Ⓓ button.



We have to keep in mind of three things when we assemble the robots. first of all, when the actuator is activated, make sure that the cable line is not too short or does not get stuck between the joints. Second, keep mind of angle when you are assembling horn for it can be assembled at an interval of 90° ; and when the horn’ s assembled angle is incorrect, there exist possibility of over-current flow, affecting joint adversely (for instance, it may burn out the fuse in CM-5). You can check whether the horn is assembled correctly by taking a closer look at the groove of both horn and AX-12. Additionally, you have to consider the direction when you assemble Dynamixel. That is, as Dynamixel is symmetrical, users may inadvertently assemble robot in a reverse direction. Thus, pay attention to the direction of horn when you are assembling. For details, refer to the manual (2-2-1).

3-3. Frequently Used Behavior Control Routines

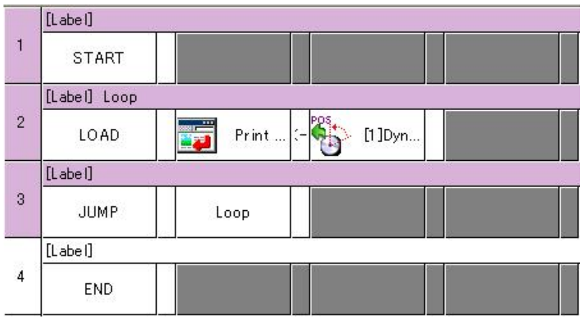
Location Unit In order to operate Dynamixel (AX-12), you have to understand the underlying principle of angle unit. The AX-12 can control 300° by 1024 unit steps. When the groove of both horn and AX-12



correspond, the location value will be 512.

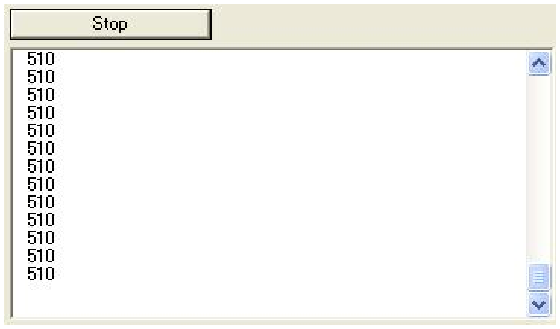
Print Screen

Let's check the above value. There will be times when you want to find out the value for input and output while creating behavior control program. In this case, load the value in "print screen" item of CM-5. (Refer to "Examples\Example(Joint position(Crossing Gate)).bpg" inside the CD)



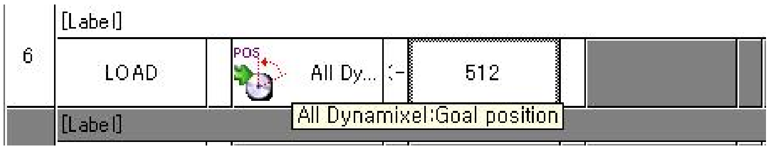
1	[Label]			
	START			
2	[Label] Loop			
	LOAD	Print ...	<-	[1]Dyn...
3	[Label]			
	JUMP	Loop		
4	[Label]			
	END			

Just like above example, load the current location item onto print screen. As we need to examine the values in detail, execute the print repeatedly. After downloading and running above program, bring your hands closer. You will get the following results.



```
510
510
510
510
510
510
510
510
510
510
510
510
510
```

Desired Position Up to now, we experimented with finding the location value of Dynamixel. Now, let's take a look at ways to drive the Dynamixel. If you load the value to "desired position," the Dynamixel will move in the direction of its set value. If you set the value to 512 as shown below, it will move to center position.



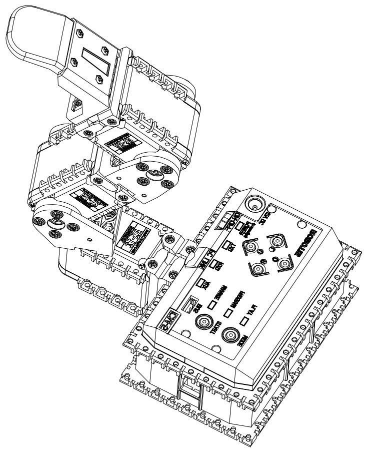
6	[Label]			
	LOAD	All Dy...	<-	512
	[Label]			

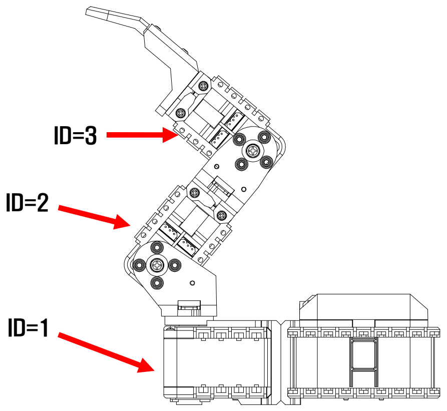
All Dynamixel:Goal position

Above example implemented "all Dynamixel." In this case, instead of assigning values one by one, you can select “all Dynamixels” and assign values to all the Dynamixels at once. In crossing gate, we use one Dynamixel. However for robots that use many joints, you can take advantage of “all Dynamixel.” Refer to the QuickStart for crossing gate example program.

3-4. Assembling Robot Arm

Example Now that we have learned the basics of building, let’ s go ahead and build a robot arm. First, refer to the “2-2-8. Robot Arm” of QuickStart and build a robot arm.





[Side view]

- After you are finished building, make sure that the cables that come out from the CM-5 bus and into the Dynamixels are all connected properly.

After you are finished building the arm, apply power to the CM-5 unit. Plug in the power jack into the SMPS and turn the power switch on. The blinking of the Dynamixel’ s LED means that the power has been applied properly.

If the LED does not blink, then check the following.

- Is the CM-5 unit in standby mode? (make sure the mode LED is blinking) If not, it means that the power was not supplied properly.
- Are the Dynamixels connected properly? Make sure the wiring on the AX-12 units is done properly. The direction of the three line cables does not matter.

Charging Once the SMPS is connected the robot it can use the outside power source and also recharge the internal batteries at the same time. Pressing the Ⓤ button in standby mode will start the recharging. Among the status LED’ s of the CM-5 unit, there is one labeled Power. This will be ON when running on battery power and will blink during recharging. When the recharging is almost finished, the blinking will become faster. When the recharging is done, the blinking will become slow again. Refer to the “3-2 Charging CM-5” or “Help Files\Charging CM-5. wmv ” video clip.

Caution Frequent recharging and draining of the battery will reduce the battery life quickly. This is because of the memory effect and the recharging cycle limit of the battery. The recommended number of times the battery should be recharged is 300. When charging, do not disconnect the SMPS until the recharging is completely finished. Disconnecting the SMPS and plugging it back again often will reduce battery life.
Also, do not charge the batteries without it being inside the CM-5 unit. A temperature sensor inside the CM-5 measures the temperature of the battery and determines whether the charging cycle is finished. If you charge the battery outside the CM-5 unit, there is a danger of over-charging.

Low Battery Sign

The power LED will blink when the robot is low in battery power.

Let’ s create a behavior control program that makes the robot arm do the following.

- Set the Dynamixel so that it can output 1/4 of its max torque.
- Set the position of all the Dynamixels at their center position.

Torque Limit Address number 34 of the AX-12 decides the torque limit. The max data value is 1023. In other words, if you load 1023 into address 34 of a AX-12 unit, then it will move while outputting the maximum torque possible. If you want one fourths of the max torque, then load 256, which is one fourths of 1023.

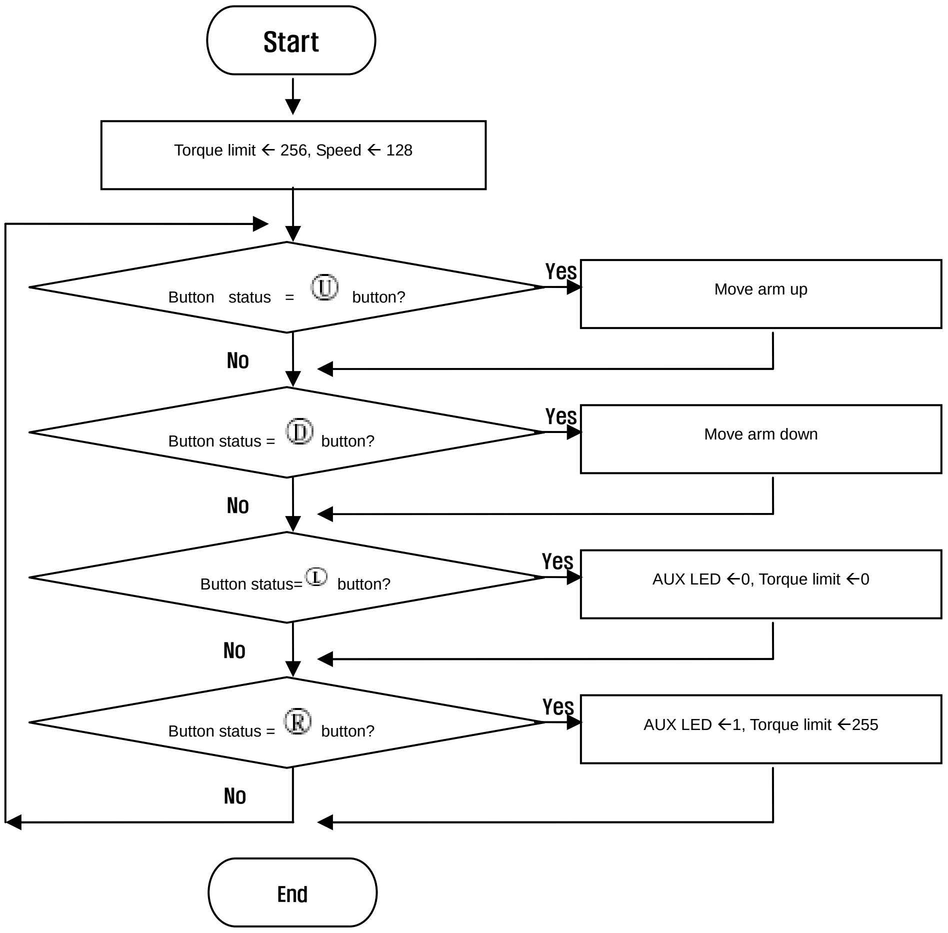
TIP Only integers can be used when inputting numbers for the CM-5 unit. One fourths of 1023 is 255.75 but you are not allowed to input this number thus the integer number 256 should be used instead.

Why would one want to reduce the maximum torque limit? This is because you could damage the robot if a joint moves to a position over its physically allowed angle limit of a joint. To prevent this, test the robot with reduced torque first. If you are sure that there is no problem with the movement, then you can increase the torque back to normal.

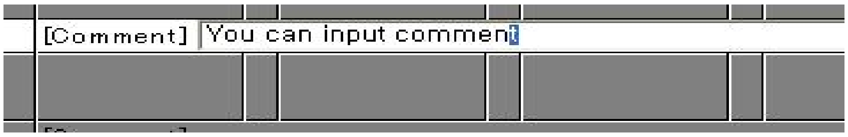
Now let’ s build a more complicated behavior control program that does the following.

- Set the motor torques to one fourths of its max value.
- Set the motor speeds to one eighths of its max value.
- If the U button is pressed, move the arm up.
- If the D button is pressed, move the arm down.
- If the L button is pressed, turn the AUX LED off and release the power in the arm.
- If the R button is pressed, turn the AUX LED back on and engage power back to the arm.

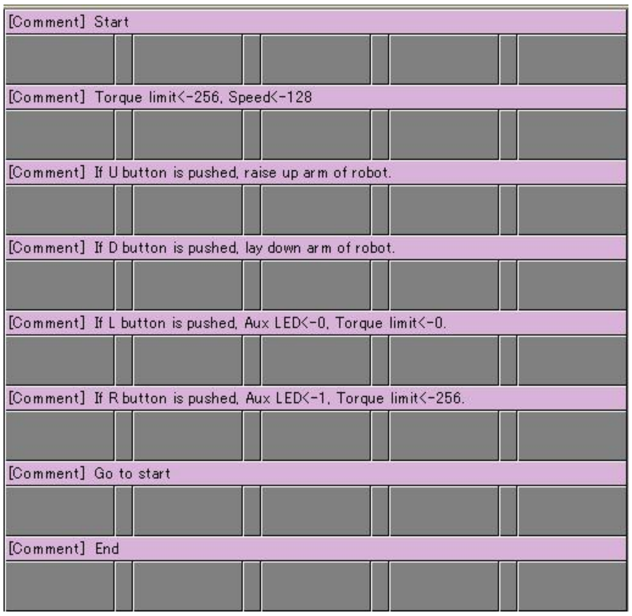
This behavior control program can be represented in the form of a flow chart as shown below.


Start
Torque limit ← 256, Speed ← 128
Button status = Ⓤ button?
Yes
Move arm up
No
Button status = Ⓓ button?
Yes
Move arm down
No
Button status= Ⓛ button?
Yes
AUX LED ←0, Torque limit ←0
No
Button status = Ⓡ button?
Yes
AUX LED ←1, Torque limit ←255
No
End

Comments The flow chart shown above is much more complicated than the behavior control program that we created before. Thus, to prevent mistakes and to make it more convenient, you can use comments. A comment does not affect execution of the program but it is rather a memo for the user.
As in the figure below, you can double-click on the cell labeled "Comment" to type comments in.


[Comment] You can input comment

Comments written here are only for the user. It will not affect the execution of the program.
The following figure shows the comments for the flow chart from above.

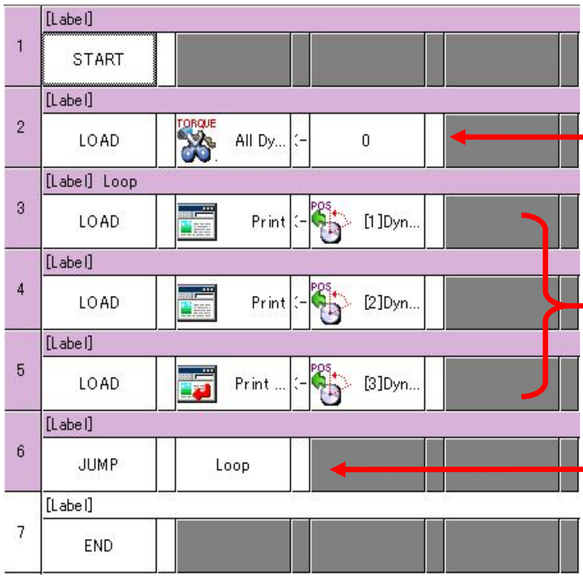

[Comment] Start
[Comment] Torque limit<-256, Speed<-128
[Comment] If U button is pushed, raise up arm of robot.
[Comment] If D button is pushed, lay down arm of robot.
[Comment] If L button is pushed, Aux LED<-0, Torque limit<-0.
[Comment] If R button is pushed, Aux LED<-1, Torque limit<-256.
[Comment] Go to start
[Comment] End

You can slove the problem step by step if you write in the comments line by line first as shown above. Let' s now create the behavior control program based on the comments that we have just typed in.
There is one part of the program that is difficult to accomplish using what we have learned so far. It is the command that makes the arm move up or down. It would be easier if we knew the position value of the joints as the arm moves up or down.

Now, let’ s make a program that prints the value of joint of robot arm on screen. Follow the following steps.

- Set the torque limit to 0.
- Display the present position of a joint on screen continually.

The following figure shows how this is done. Refer to the “Examples\Example(Joint position(Robot Arm).bpg)



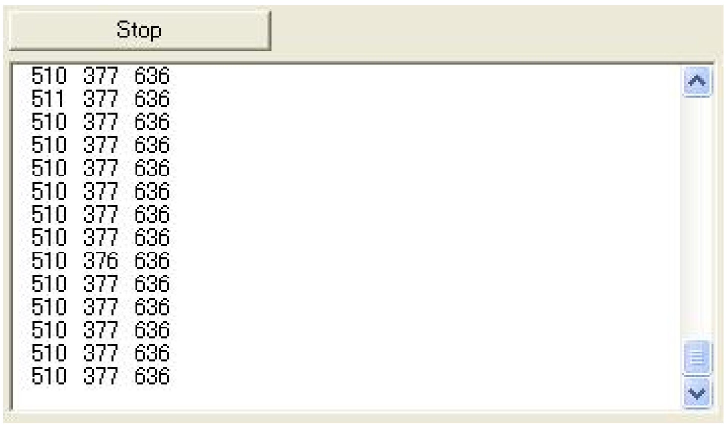
1	[Label] START			
2	[Label] LOAD	TORQUE All Dy...	<-	0
3	[Label] Loop LOAD	Print	<-	POS [1]Dyn...
4	[Label] LOAD	Print	<-	POS [2]Dyn...
5	[Label] LOAD	Print ...	<-	POS [3]Dyn...
6	[Label] JUMP	Loop		
7	[Label] END			

All value of joint’ s torque set at 0.

Load the value of joints to print screen item.

Continue print out the changed value on screen

Download the program and run it. Moving the robot arm will output the value for the joint position. With this program, you can select the appropriate values for moving the arm up and down.



```
510  377  636
511  377  636
510  377  636
510  377  636
510  377  636
510  377  636
510  377  636
510  377  636
510  376  636
510  377  636
510  377  636
510  377  636
510  377  636
510  377  636
```

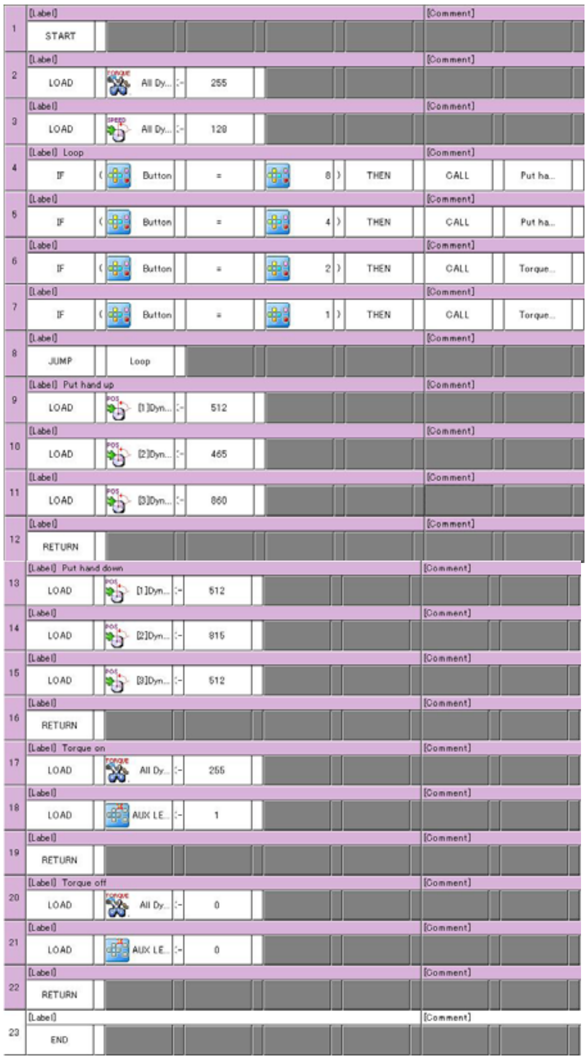
Appropriate joint position values for moving up the arm.

- Value of joint position ID = 1: 512
- Value of joint position ID = 2: 465
- Value of joint position ID = 3: 860

Appropriate joint position values for moving down the arm

- Value of joint position ID = 1: 512
- Value of joint position ID = 2: 815
- Value of joint position ID = 3: 512

Input these values and complete the program. The following figure shows the source of the finished program. Refer to the "Examples\Example(Robot Arm).bpg)



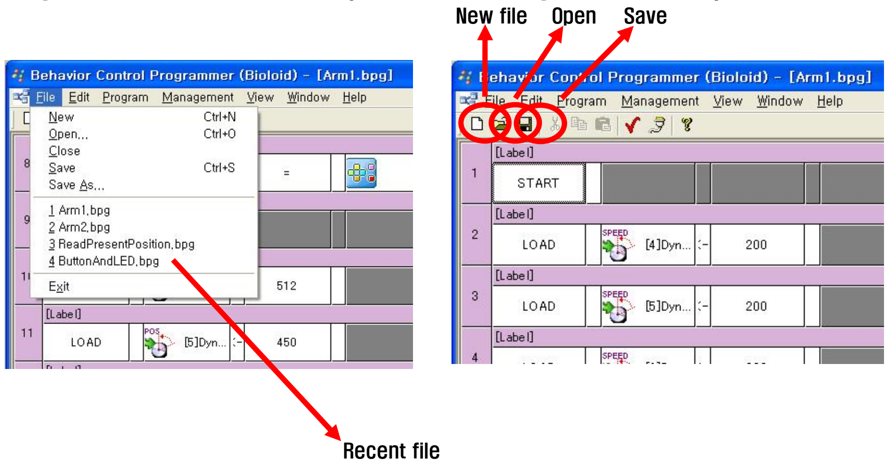
```
[Label]                                                    [Comment]
START
[Label]                                                    [Comment]
LOAD   All Dy... :-  255
[Label]                                                    [Comment]
LOAD   All Dy... :-  128
[Label] Loop                                               [Comment]
IF  ( Button  =  8 )  THEN                                 CALL  Put ha...
[Label]                                                    [Comment]
IF  ( Button  =  4 )  THEN                                 CALL  Put ha...
[Label]                                                    [Comment]
IF  ( Button  =  2 )  THEN                                 CALL  Torque...
[Label]                                                    [Comment]
IF  ( Button  =  1 )  THEN                                 CALL  Torque...
[Label]                                                    [Comment]
JUMP   Loop
[Label] Put hand up                                        [Comment]
LOAD   [1]Dyn... :-  512
[Label]                                                    [Comment]
LOAD   [2]Dyn... :-  465
[Label]                                                    [Comment]
LOAD   [3]Dyn... :-  860
[Label]                                                    [Comment]
RETURN
[Label] Put hand down                                      [Comment]
LOAD   [1]Dyn... :-  512
[Label]                                                    [Comment]
LOAD   [2]Dyn... :-  815
[Label]                                                    [Comment]
LOAD   [3]Dyn... :-  512
[Label]                                                    [Comment]
RETURN
[Label] Torque on                                          [Comment]
LOAD   All Dy... :-  255
[Label]                                                    [Comment]
LOAD   AUX LE... :-  1
[Label]                                                    [Comment]
RETURN
[Label] Torque off                                         [Comment]
LOAD   All Dy... :-  0
[Label]                                                    [Comment]
LOAD   AUX LE... :-  0
[Label]                                                    [Comment]
RETURN
[Label]                                                    [Comment]
END
```

4. Behavior Control Programmer

So far we have made simple behavior control programs and learned how to use the Behavior Control Programmer. In this chapter we will be learning the functions of the behavior control program one by one.

4-1. Opening a File

In the menu, select “file” and the following items can be seen. The items here are very similar to most other PC programs so they won’ t be explained in detail.



New This function creates a new behavior control program file. Several program files can be opened at the same time and copying and pasting between files is possible.

Open This function opens a previously saved program.

Save This function saves the program that you are working on. When saving for the first time, you will be asked to name the file. You can also save a file by clicking on the “save” icon in the toolbar.

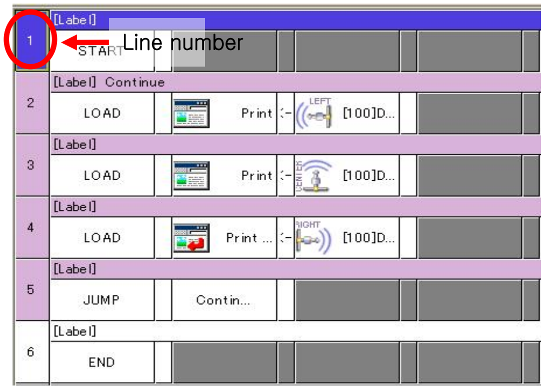
Recent File The program keeps a record of the paths to recently opened files. Instead of searching for a file and then opening it, you can use this function to take a shortcut.

4-2. Editing Function of the Behavior Control Programmer

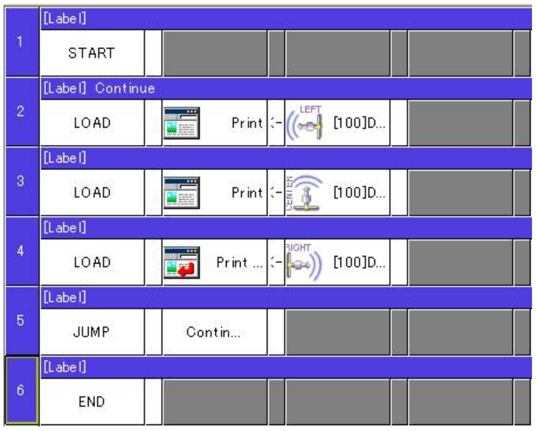
Selecting a Sentence

Before copying, moving, or deleting a sentence you have to select a sentence. You can do this by clicking on the number at the very beginning of the sentence. To select several sentences at the same time press the SHIFT key and click on the numbers. The following figures show how this is done.

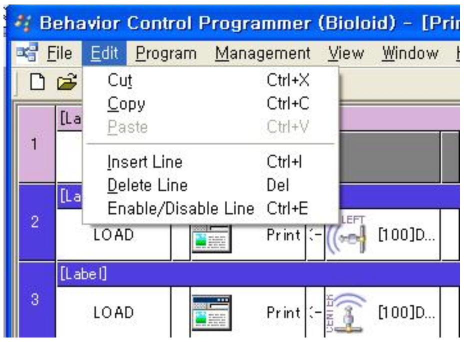
Step 1: Click on the line number at the beginning of the sentence that you want to select.

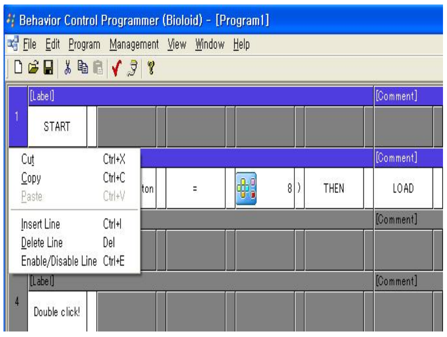

Line number
[Label]
1 START
[Label] Continue
2 LOAD Print <- LEFT [100]D...
[Label]
3 LOAD Print <- CENTER [100]D...
[Label]
4 LOAD Print ... <- RIGHT [100]D...
[Label]
5 JUMP Contin...
[Label]
6 END

Step 2: While pressing the SHIFT key, click on the last command sentence that needs to be selected. Then all the sentences that come between the two will all be selected.

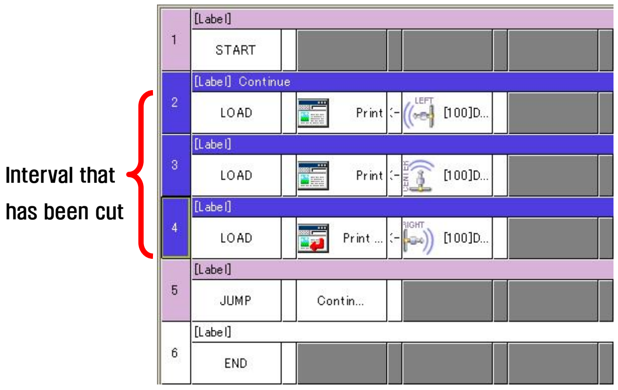

[Label]
1 START
[Label] Continue
2 LOAD Print <- LEFT [100]D...
[Label]
3 LOAD Print <- CENTER [100]D...
[Label]
4 LOAD Print ... <- RIGHT [100]D...
[Label]
5 JUMP Contin...
[Label]
6 END

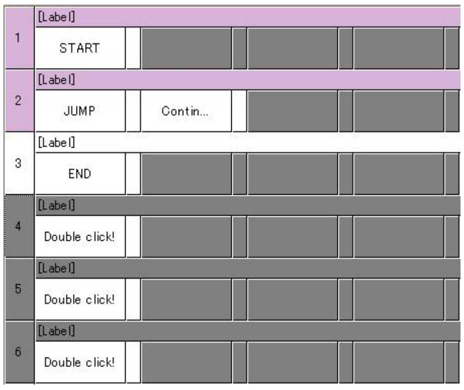
Editing menu After selecting a command sentence, go to the menu and click on EDIT to see the editing menu items. This can also be done by right clicking on the selected sentences.




Behavior Control Programmer (Bioloid) - [Program1]
File Edit Program Management View Window Help
[Label]
[Comment]
START
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Insert Line Ctrl+I
Delete Line Del
Enable/Disable Line Ctrl+E
THEN
LOAD
Double click!

Cut Cutting a selected sentence will save it and then delete it. This sentence can be used again elsewhere by using the paste function.

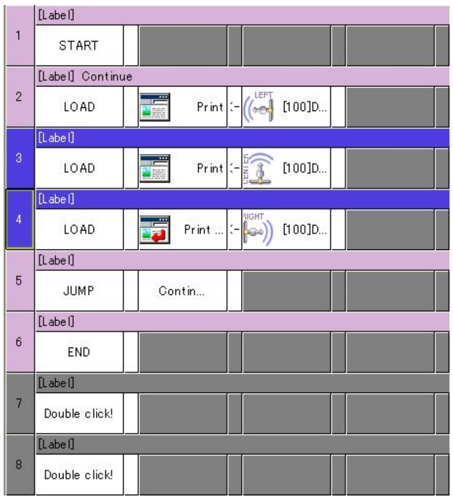


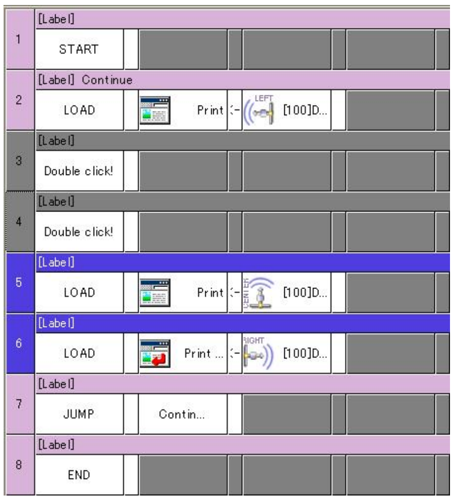


Copy When you copy a sentence, it might look like nothing is happening, but actually the selected sentences are being saved. To use paste function, it can be saved elsewhere.

Paste After cutting or copying a sentence you can paste it somewhere else. You can also copy or cut and paste from one program to another. Thus, you can copy command sentences from previously created programs and reuse them by pasting them to the program you are currently working on.

Insert When you want to put in another sentence between two sentences you can use the insert function. Select the number of sentences that you want to insert and then press "insert."



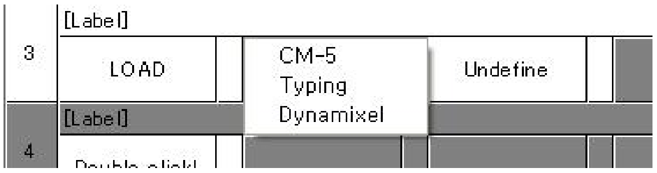


[Insert Instruction in two lines selected state makes two blank line]

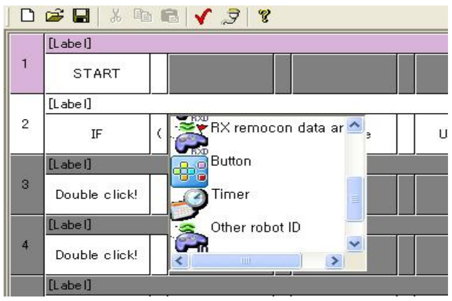
Erase Use this function when you want to delete a sentence.

Parameter input

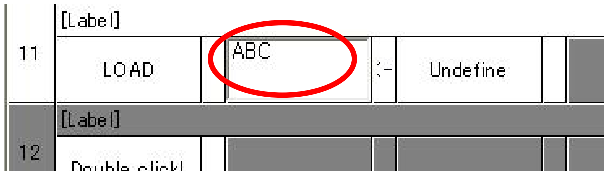
If you double-click a block for the parameters, the following menu will show up.



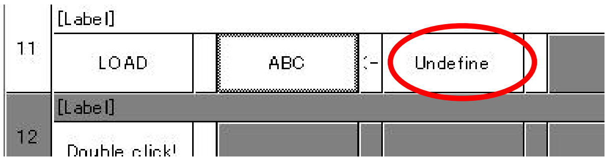
As in the figure above, you can select between "CM-5," "Input," or "Dynamixel." If you select "CM-5" the following menu will show up. Here, select the desired CM-5 item.



If you select "input" instead of "CM-5" you will be able to input something via the keyboard instead, as shown in the figure below. Selecting "input" in the menu means that the program is expecting you to input a constant or a variable. When inputting a constant, just type in numbers and when inputting a variable, type in an appropriate variable name. Just remember that for the variable names the first word has to start with a letter (numbers 0~9 should not be used as the first character). After typing in and editing the contents, press Enter and the information will be entered into the parameter block.

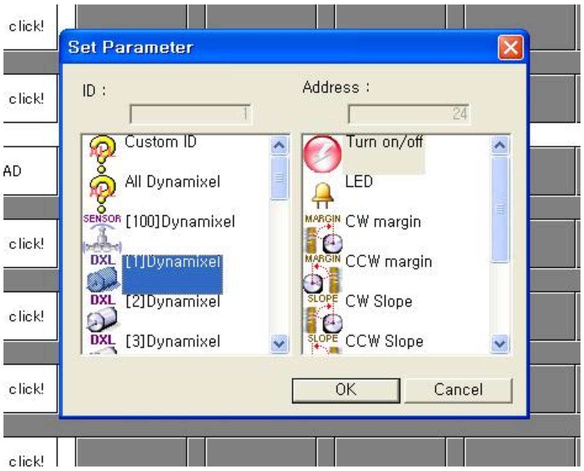


[Example of inputting a constant]

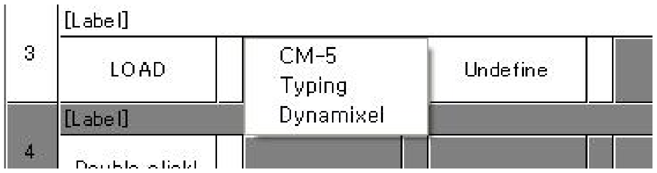


[[Example of inputting a constant

When you select "Dynamixel" as an input item, a pop up box will appear and here you can select the ID and address items.



Correcting To correct an element of a parameter command after you are done editing, first left click the sentence that you want to correct and then right click to see the first menu again to correct it.

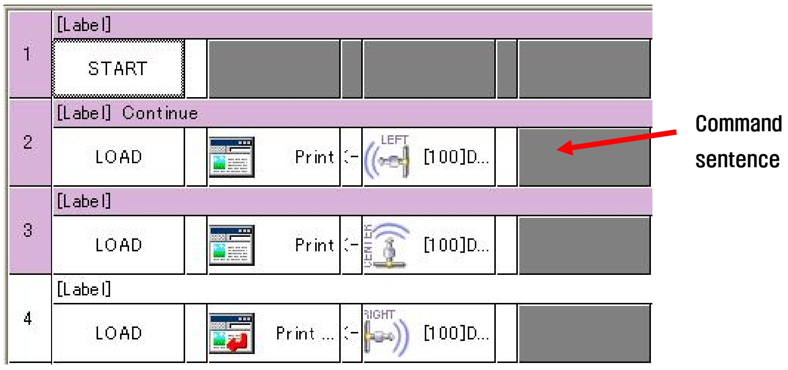


4-3. Syntax of the Behavior Control Program

We have learned several commands for the behavior control program with several examples. Let' s now review it systematically.

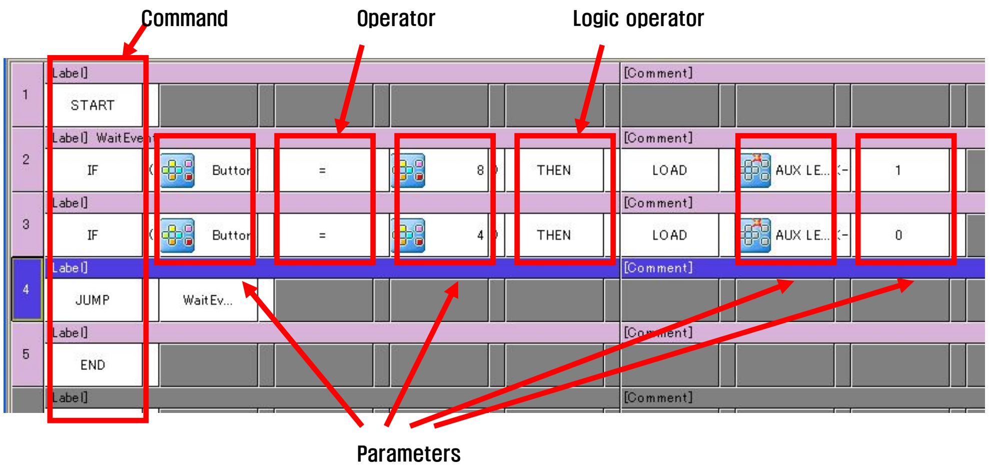
Command Sentence

The basic unit of a behavior control program is the command sentence. Each line tells what the robot should do.



Inputting a Sentence

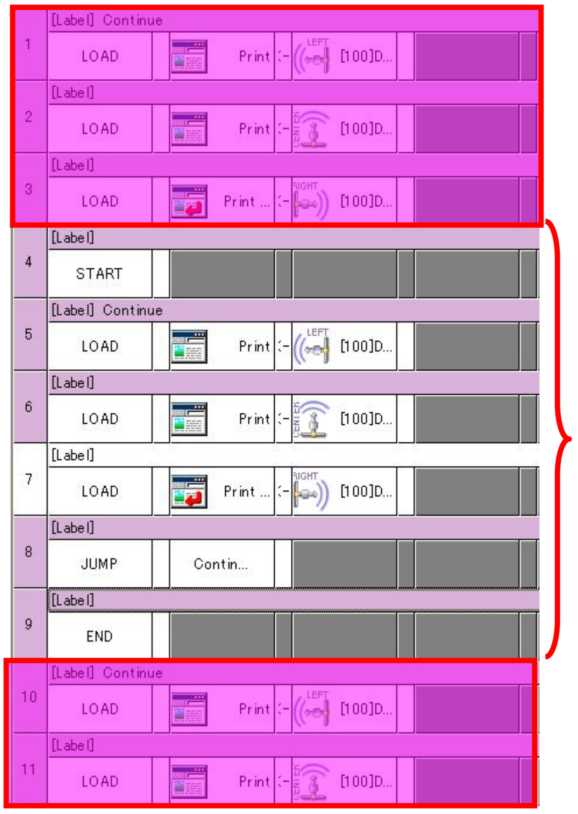
You can create a command sentence by clicking on an empty cell and selecting what you want from the item list. A sentence is made of a command with an operator, logic operator, or parameters.



Command There are three types of commands: execution, condition, and branch.

- Condition: IF, ELSE IF, ELSE, CONT IF
- Execution: START, END, LOAD, COMPUTE
- Branch: JUMP, CALL, RETURN

Start, End These are the very basic commands. The behavior control program will consider everything between the start and the end command as the program that will actually be used. The command sentences that are above the start command or below the end command will be ignored. Also, if there are two start commands, or if start and end does not exist, an error will occur during the rule check routine.



1	[Label] Continue	LOAD	Print	:-	LEFT	[100]D...
2	[Label]	LOAD	Print	:-	CENTER	[100]D...
3	[Label]	LOAD	Print ...	:-	RIGHT	[100]D...
4	[Label]	START				
5	[Label] Continue	LOAD	Print	:-	LEFT	[100]D...
6	[Label]	LOAD	Print	:-	CENTER	[100]D...
7	[Label]	LOAD	Print ...	:-	RIGHT	[100]D...
8	[Label]	JUMP	Contin...			
9	[Label]	END				
10	[Label] Continue	LOAD	Print	:-	LEFT	[100]D...
11	[Label]	LOAD	Print	:-	CENTER	[100]D...

Segment that will be actually be executed.

IF IF command is a condition decision command. The command statement has the following format.

IF | Parameter1 | Operator | Parameter2 | Logic Operator

Parameter1 and Parameter2 are the objects to be compared, and the operator decides what type of comparing will to be performed. For example, for a sentence "If A is larger than B," "A" is paramter1, "B" is parameter2, and "larger" is the operator.

Let’ s say that there is a traffic signal made of red, orange, and green lights. We want to make a robot that stops at red, gets ready at orange, and moves at green. A program using the IF commands would look like the following.

- IF signal = red THEN, robot stops.
- IF signal = orange THEN, robot gets ready.
- IF signal = green THEN, robot moves.

ELSE IF, ELSE

In the program above, the robot has to check all three color conditions. However, if the signal is red, the robot does not need to check whether the lights are orange or green. If the signal is orange, the robot does not have to check whether the lights are red or green. Using the ELSE IF and the ELSE command, we can simplify the program by eliminating this need for checking all three conditions.

- IF signal = red THEN, robot stops.
- ELSE IF signal = orange THEN, robot gets ready.
- ELSE robot goes.

CONT IF

If you want to change the behavior control program above to implement "If the signal is red, or if a car is coming, then stop the robot" you can do so using the CONT IF command. What would happen if we create a program as the following?

- IF, signal = red AND
- ELSE IF, car is coming THEN, stop robot.

The program above tells the robot to stop only when the signal is red and at the same time if a car is coming. For the command AND, the lines after THEN will be executed only when both statements are true. In order for the program to work properly, it has to be changed as the following.

- IF, signal = red OR
- ELSE IF, car is coming THEN, stop robot.

LOAD The command LOAD will load one item onto another. The sentence structure is as the following.

LOAD | Parameter 1 | Parameter 2

This commands to load parameter 2 into parameter 1. LOAD is an execution command which is used like "start the robot motion on page 10" or "Set the Dynamixel destination position to 500." For these examples, parameter 1 would be "start motion" or "destination position" and parameter 2 would be "page 10" or "500." Also, LOAD is a write command which can be used like "write the current position of the Dynamixel on the screen," or "write 20 to variable A." In these examples, parameter 1 would be "write on screen" or "variable A" and parameter 2 would be "current position" or "20."

Compute This command is used when executing an operator. The command sentence has the structure as the following.

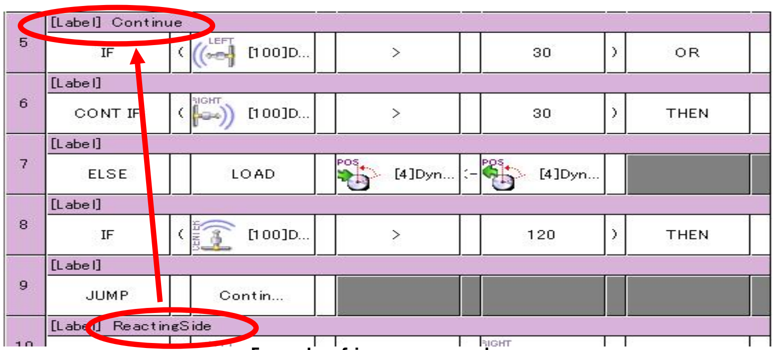
Compute | Parameter1 | Parameter2 | Operator | Parameter3

The format of the compute command would look like "A = B + C." In this case, "A" would be parameter 1 and is the answer that will be stored. The objects to be added are "B" and "C" which correspond to parameter 2 and parameter 3. There are many kinds of operators including "+" , "–" , "*" , "/" , and they can be selected from the menu.

JUMP This command is used when the execution order of the sentences need to be changed. The structure of a JUMP command statement looks like the following.

Command | Parameter

For the parameter, you have to put in a label of the sentence that you want the program to jump to. The name of the label has to be unique or there will be an error during the rule check routine.



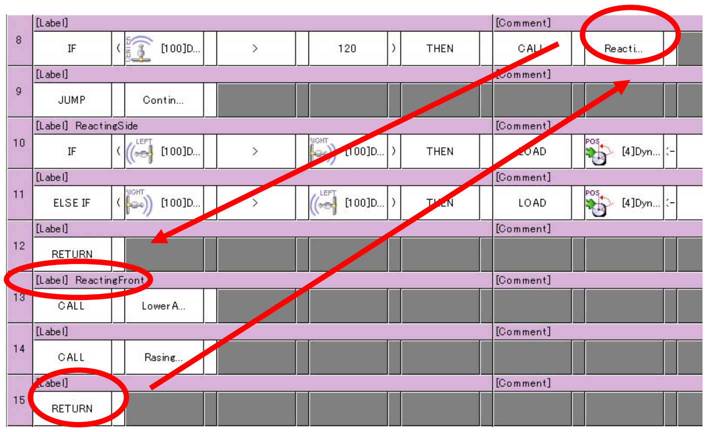
```
5  [Label] Continue
   IF       ( [100]D...   >          30          )  OR
6  [Label]
   CONT IF  ( [100]D...   >          30          )  THEN
7  [Label]
   ELSE       LOAD        [4]Dyn...  <-  [4]Dyn...
8  [Label]
   IF       ( [100]D...   >          120         )  THEN
9  [Label]
   JUMP       Contin...
10 [Label] ReactingSide
```

Example of jump command

Call, Return

Sometimes you will have to execute an identical code of a program several times. You can make such a code a subroutine and call it whenever you need to execute it to simplify the program and for convenience. This technique is used very often in programming and it becomes especially useful as the program becomes lengthier.

The CALL command is very similar to the JUMP command, however the difference is that, for the CALL command, after executing the subroutine the execution will go back to where the CALL was executed, using the RETURN command. When using the CALL command, the subroutine should always end with the RETURN command.

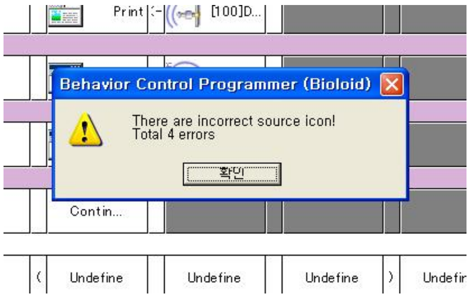


Row	[Label]							[Comment]	
8	IF	([100]D...	>	120	)	THEN		CALL	Reacti...
9	JUMP	Contin...							
10	[Label] ReactingSide								
	IF	([100]D...	>	[100]D...	)	THEN		LOAD	[4]Dyn... <-
11	ELSE IF	([100]D...	>	[100]D...	)	THEN		LOAD	[4]Dyn... <-
12	RETURN								
13	[Label] ReactingFront								
	CALL	LowerA...							
14	CALL	Rasing...							
15	RETURN								

Example of call command

Rule Check

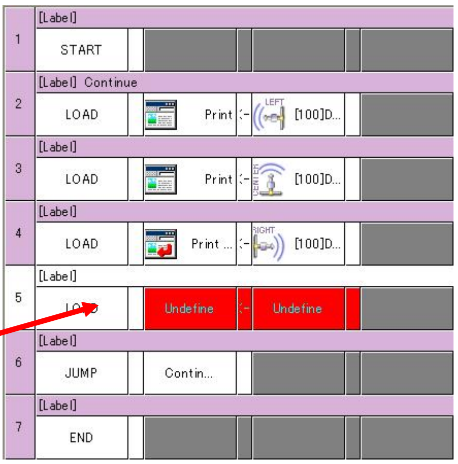
After finishing creating a behavior control program, you always have to check if there are any errors in the syntax. A program with an error cannot be downloaded to the robot. You can select the rule check item from the program menu. Places with an error will be highlighted in red. Fix the errors and then run rule check again.



Behavior Control Programmer (Bioloid)

There are incorrect source icon!
Total 4 errors

확인



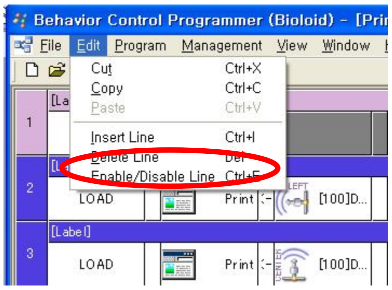
Row	[Label]				
1	START				
2	[Label] Continue				
	LOAD	Print	<-	[100]D...	
3	LOAD	Print	<-	[100]D...	
4	LOAD	Print ...	<-	[100]D...	
5	LOAD	Undefine	<-	Undefine	
6	JUMP	Contin...			
7	END				

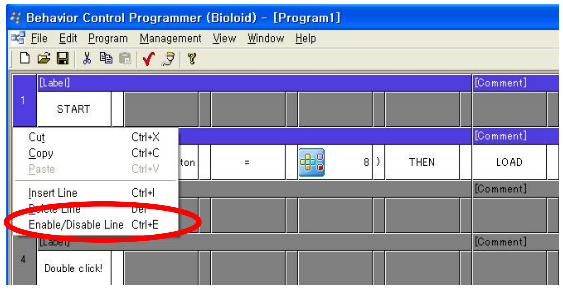
Error Line

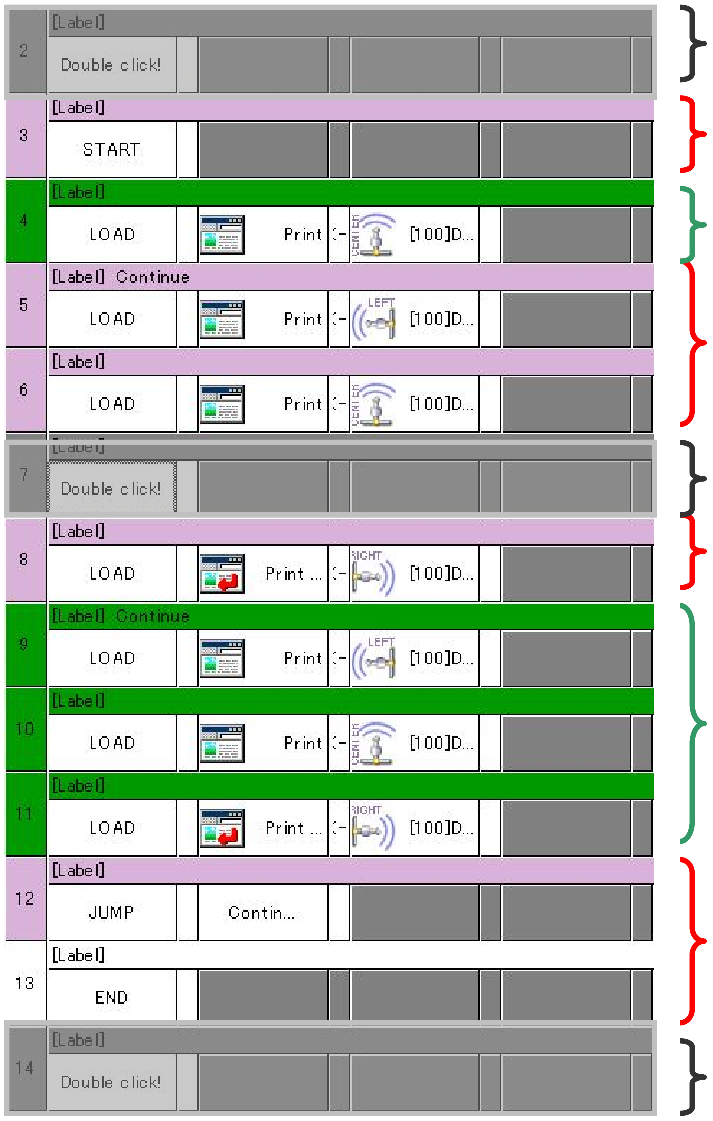
TIP When you download a program to the robot, rule check will run automatically.

Enable/Disable Code

When creating a program, sometimes you will have to prevent some parts of the code from being changed. This can be done with enabling or disabling the code segments. This function can be very useful in programming and can speed up the development process. The only parts of the program that are changeable are the parts between “START” and “END.” One way to prevent a part from changing is to move it outside of “START” and “STOP,” but this would be a troublesome process. Instead, you can use the “locking” function to protect the part from being changed.


Behavior Control Programmer (Bioloid) - [Pri
File Edit Program Management View Window
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Insert Line Ctrl+I
Delete Line Del
Enable/Disable Line Ctrl+E
LOAD
Print
[100]D...


Behavior Control Programmer (Bioloid) - [Program1]
File Edit Program Management View Window Help
START
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Insert Line Ctrl+I
Delete Line Del
Enable/Disable Line Ctrl+E
THEN
LOAD
Double click!



Portions that will be ignored

Portion that will be ignored

Enable / Disable Code

5. Using Sensors

A robot cannot truly be a robot if it simply moves by remote control. A robot has to be able to move and do things all by itself. The most important thing in making a robot autonomous is to give it the ability to sense and gather information. For example, if you want to build a robot that can avoid obstacles, the robot would first need to have the ability to sense the obstacle.

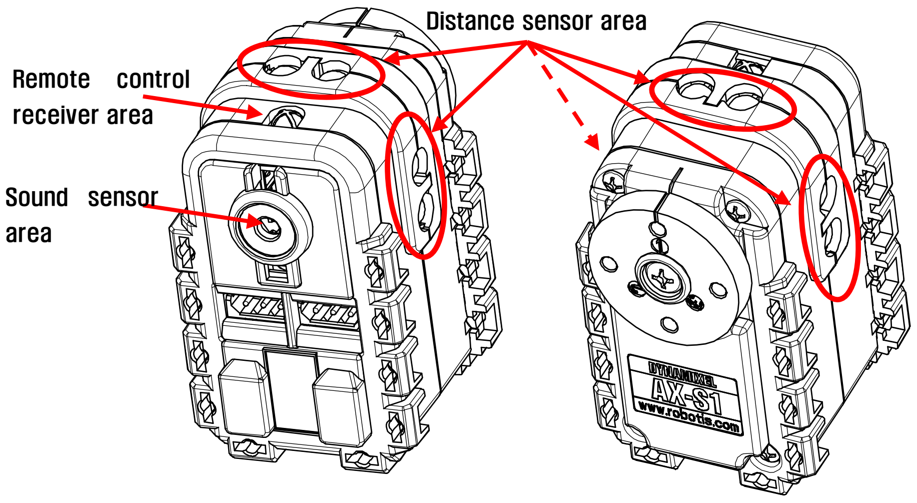
A device that can sense information is called a sensor. A sensor not only has the ability to sense objects, but also people or other robots. The process of a robot sensing outside information and reacting to it via outputting a movement is called robot interaction.

TIP The interaction between a robot and human is called HRI (Human-Robot Interaction). Voice and face recognition is also part of HRI. In order to have robots live with humans, the advanced HRI development is essential

5-1. The AX-S1 Sensor Module

Earlier, we have created a program that controls the robot behavior based on the robot' s input and output. The output items that we have tested were LED, motor position, and printing on screen. The input items we have used were mostly using buttons. In this chapter we will learn more input items using the sensor module AX-S1.

The figure below shows the AX-S1. The AX-S1 has a distance sensor (3 directions), sound sensor, remote control receiver, and a buzzer.

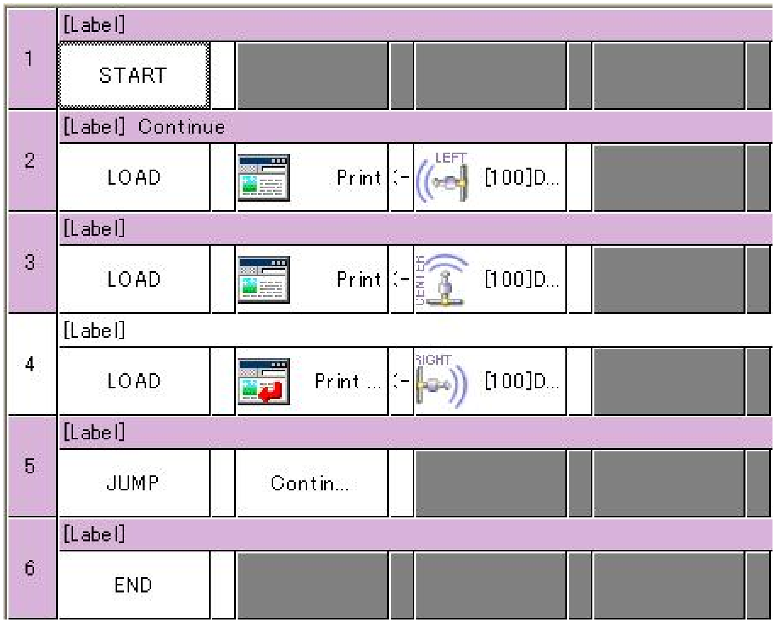


[External view of the AX-S1 module: Top view, bottom view]

The AX-S1 has the identical mechanical and electrical universal expansion structure as the AX-12 which can be connected to other Dynamixel units.

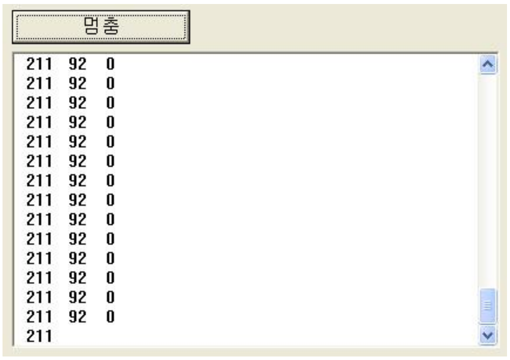
5-2. Distance Sensing Function

Let’ s create a behavior control program that will print the distance sensor values on the screen. There are three distance sensors. Let’ s print the value of each one by one on the screen. Select LOAD for the command, CM-5 screen as the left parameter, and sensor 1 of the Dynamixel as the right parameter.
The finished program should look like the following. Refer to the “Examples\Example(Read IR sensor value.bpg” inside the CD.



1	[Label]					
	START					
2	[Label] Continue					
	LOAD		Print	:-	LEFT	[100]D...
3	[Label]					
	LOAD		Print	:-	CENTER	[100]D...
4	[Label]					
	LOAD		Print ...	:-	RIGHT	[100]D...
5	[Label]					
	JUMP		Contin...			
6	[Label]					
	END					

A beeping sound will be made when the AX-S1 is connected to the CM-5 unit and the power is turned on. When you run the program the result will look like the following.



```
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211  92   0
211
```

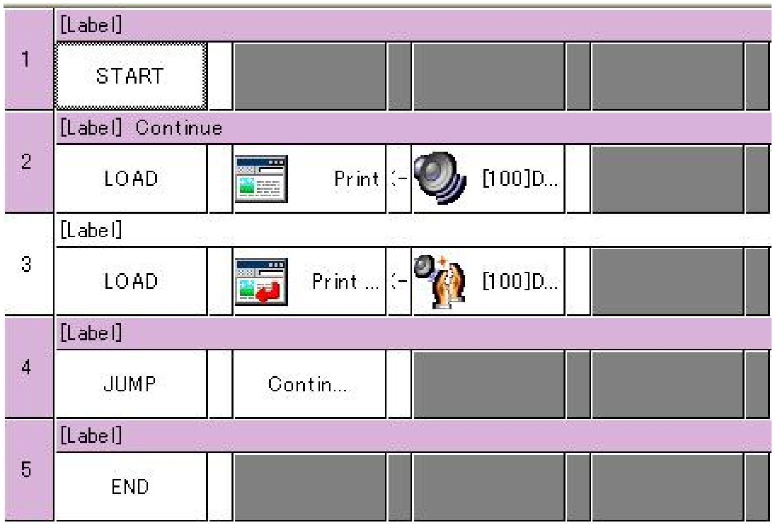
Bring your hand closer to the distance sensor area. When the distance between the sensor and your hand decreases, the value printed on the screen will increase, up to maximum value of 255.
As the distance increases, the value will decrease down to zero. If the value does

not go down to zero, this means that the lighting in the surrounding area is too bright or the sensor is sensing the wall or ceiling.
One thing that you have to be careful about is the fact that the sensitivity is different between white objects and black objects and thus this value can not be assumed to be the absolute distance to the object.

There are also many other items related to distance sensing. Refer to the AX-S1 manual for more information.

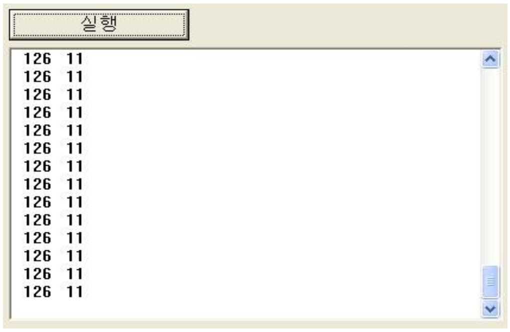
5-3. Sound Sensing Function

The AX-S1 can also sense sound from the surroundings and can count how many claps it hears. Let' s create a program that will demonstrate this.
As in the example above, this program will print the noise level of the surroundings and number of claps onto the screen.
The program will look like the following. Refer to the "Examples\Example(Read sound and count clap).bpg" inside the CD.



```
1  [Label]
   START
2  [Label] Continue
   LOAD  Print  <-  [100]D...
3  [Label]
   LOAD  Print ...  <-  [100]D...
4  [Label]
   JUMP  Contin...
5  [Label]
   END
```

Run the program and try clapping or making a sound. The output on the screen will look like the picture on the left.
There are also many other items related to sound sensing. Refer to the AX-S1 manual for more information.

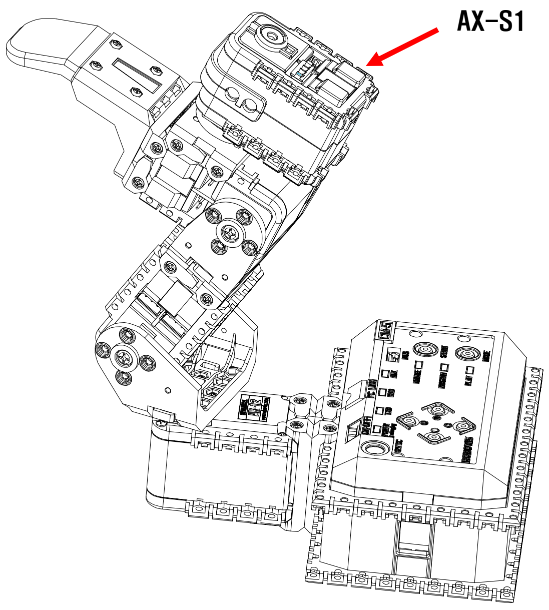


```
126  11
126  11
126  11
126  11
126  11
126  11
126  11
126  11
126  11
126  11
126  11
126  11
126  11
126  11
```

5-4. Assembling Attacking Duck that Uses Sensor

We are now going to build a very interesting robot. Let’ s make a duck robot that attacks an object that approaches it.

Attach an AX-S1 unit to the robot arm built in chapter 3 as shown in the figure below. Refer to the “2-2-11. Attacking Duck” of QuickStart whenever it is necessary.

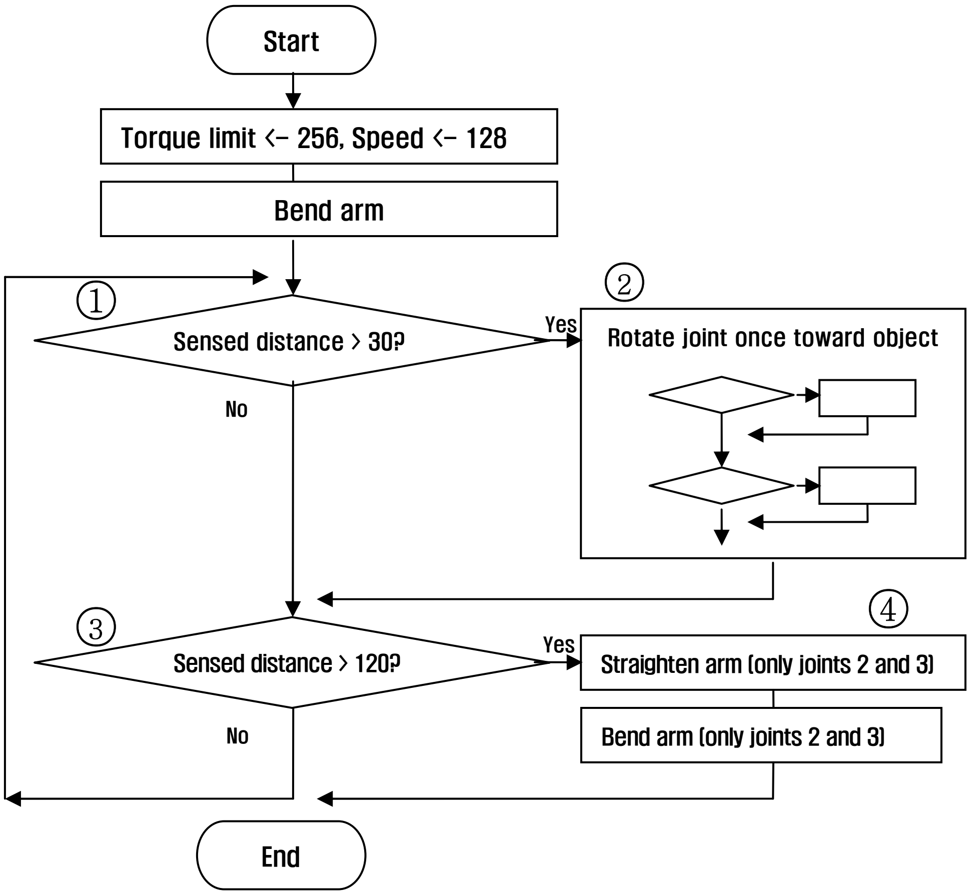


Caution While working with the robot never put your face close to the robot.

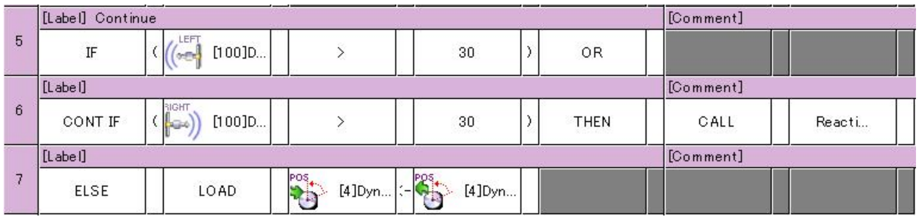
The behavior control of above can be divided into the following.

- Set the max joint speed to 128 and torque limit to 256.
- Implement arm bending motion.
- Repeat the two functions below.
- If the value of the distance sensor is greater than 30, move joint number 4 towards the sensed direction.
- If the value of the distance sensor is greater than 120, then straighten the joints for number 5 and number 6 and then bend them back.

Since it will be difficult to make the two repeating items into a single command, it will be better to divide it into small parts. Refer to the following flow chart which labels the complicated parts as ①, ②, ③, and ④.

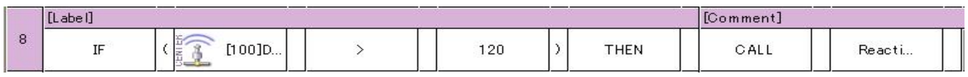

Start
Torque limit <- 256, Speed <- 128
Bend arm
①
Sensed distance > 30?
Yes
No
②
Rotate joint once toward object
③
Sensed distance > 120?
Yes
No
④
Straighten arm (only joints 2 and 3)
Bend arm (only joints 2 and 3)
End

The figure below shows the behavior control program for part 1. If either of the left or right distance sensors has a value greater than 30, then part 2 will be executed. To do this, several condition statements need to be used together. This can be done by connecting the sentences with “OR.” If not, make the robot remain in its position.



5	[Label] Continue									[Comment]		
	IF	(	LEFT [100]D...		>		30	)	OR			
6	[Label]									[Comment]		
	CONT IF	(	RIGHT [100]D...		>		30	)	THEN	CALL		Reacti...
7	[Label]									[Comment]		
	ELSE		LOAD		POS [4]Dyn...	<-	POS [4]Dyn...					

Next, let’ s go into more detail for part 3. Here, it is sensing objects straight in front so only the center sensor value needs to be checked.

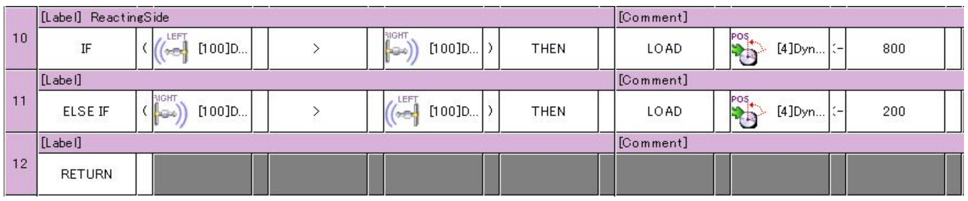


8	[Label]						[Comment]			
	IF	(	CENTER [100]D...	>	120	)	THEN	CALL	Reacti...	

We have now completed with the overall structure of the program. But we still have to finish part 2 and part 4.
Take a look at part 2. There are several ways to make joint 4 rotate towards the direction of the object. Here, we will be using the following algorithm.

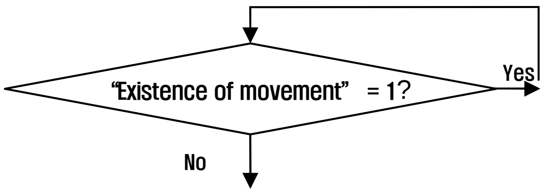
- “If “Left distance sensor value > Right distance sensor value” then move to position 800. (rotate left until 800)
- If “Right distance sensor value > Left distance sensor value” then move to position 200. (rotate right until 200)

The rotation limit was set to 800 and 200 to prevent the motor from rotating too much and damaging the wires. If we create the code for this, it will look like the following.

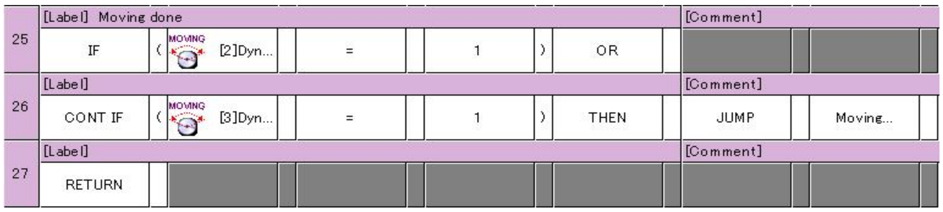


10	[Label] ReactingSide							[Comment]			
	IF	(	LEFT [100]D...	>	RIGHT [100]D...	)	THEN	LOAD	POS [4]Dyn...	<-	800
11	[Label]							[Comment]			
	ELSE IF	(	RIGHT [100]D...	>	LEFT [100]D...	)	THEN	LOAD	POS [4]Dyn...	<-	200
12	[Label]							[Comment]			
	RETURN										

Finally, we have to finish part 4. The difficult part here is that the arm bending has to be started after the arm finishes straightening out. The program speed is much faster than the speed of the actual robot moving, so even before the arm is finished straightening out the program will set the Dynamixel position value to where the arm will bend. To prevent this from happening, there is an item in the Dynamixel called “Existence of movement.” When the Dynamixel is moving, the “Existence of movement” is set to 1, and 0 when not moving. Therefore, we need a routine that will make the program standby until the value is 0.



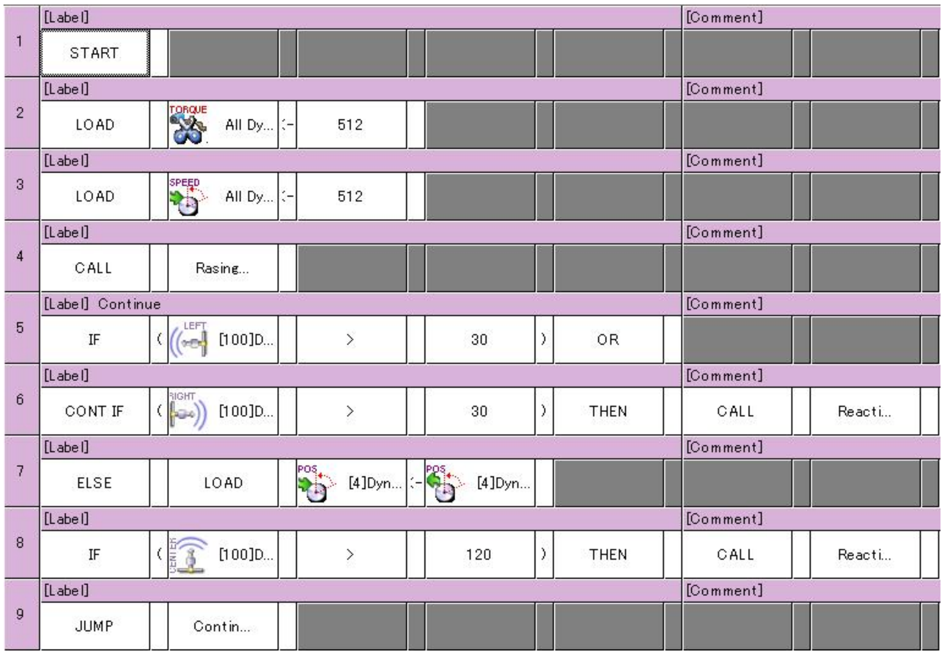
After setting the destination position, implement the code for the flowchart shown on the right. The behavior control program code will look like the following.



25	[Label] Moving done								[Comment]		
	IF	(	[2]Dyn...		=		1	)	OR		
26	[Label]								[Comment]		
	CONT IF	(	[3]Dyn...		=		1	)	THEN	JUMP	Moving...
27	[Label]								[Comment]		
	RETURN										

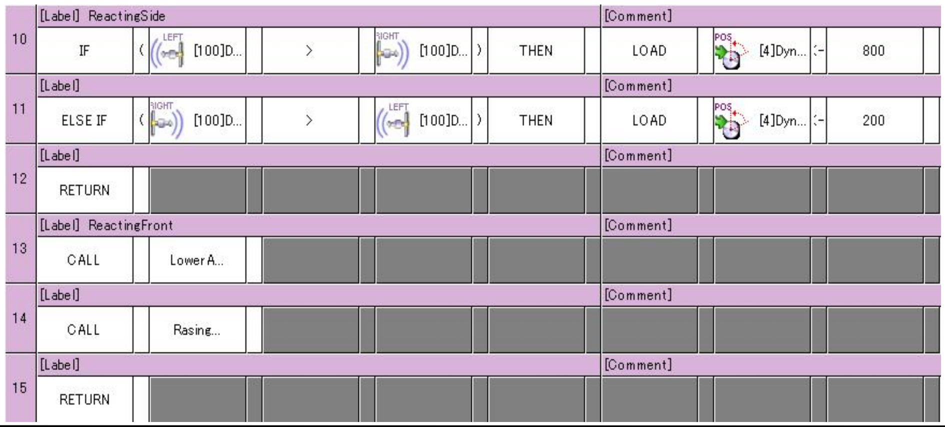
The figure below shows the entire program by putting together all the codes developed so far. Refer to the "Examples\Example(Attacking Duck).bpg" inside the CD.

Main routine



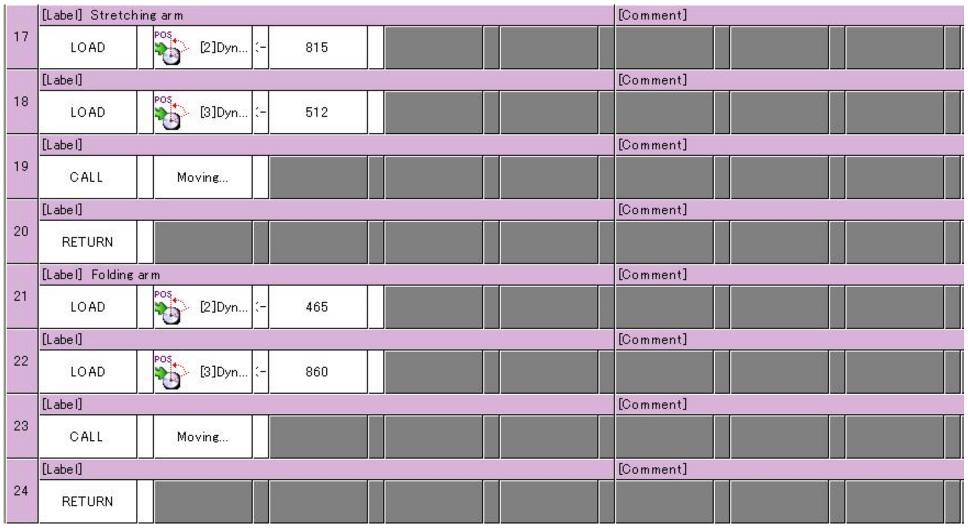
1	[Label]								[Comment]		
	START										
2	[Label]								[Comment]		
	LOAD		All Dy...	<-	512						
3	[Label]								[Comment]		
	LOAD		All Dy...	<-	512						
4	[Label]								[Comment]		
	CALL		Rasing...								
5	[Label] Continue								[Comment]		
	IF	(	[100]D...		>		30	)	OR		
6	[Label]								[Comment]		
	CONT IF	(	[100]D...		>		30	)	THEN	CALL	Reacti...
7	[Label]								[Comment]		
	ELSE		LOAD		[4]Dyn...	<-	[4]Dyn...				
8	[Label]								[Comment]		
	IF	(	[100]D...		>		120	)	THEN	CALL	Reacti...
9	[Label]								[Comment]		
	JUMP		Contin...								

Routine for front-side/left-side reaction to the sensors



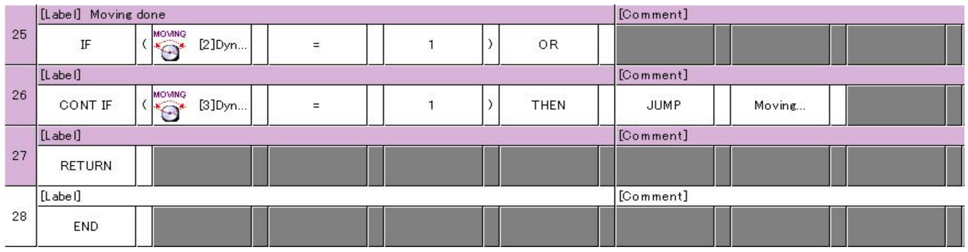
10	[Label] ReactingSide						[Comment]				
	IF	(	[100]D...	>	[100]D...	)	THEN	LOAD	[4]Dyn...	<-	800
11	[Label]						[Comment]				
	ELSE IF	(	[100]D...	>	[100]D...	)	THEN	LOAD	[4]Dyn...	<-	200
12	[Label]						[Comment]				
	RETURN										
13	[Label] ReactingFront						[Comment]				
	CALL		Lower A...								
14	[Label]						[Comment]				
	CALL		Rasing...								
15	[Label]						[Comment]				
	RETURN										

Arm bending/straightening routine to front side reaction



17	[Label] Stretching arm					[Comment]
	LOAD	POS [2]Dyn...	<-	815		
18	[Label]					[Comment]
	LOAD	POS [3]Dyn...	<-	512		
19	[Label]					[Comment]
	CALL	Moving...				
20	[Label]					[Comment]
	RETURN					
21	[Label] Folding arm					[Comment]
	LOAD	POS [2]Dyn...	<-	465		
22	[Label]					[Comment]
	LOAD	POS [3]Dyn...	<-	860		
23	[Label]					[Comment]
	CALL	Moving...				
24	[Label]					[Comment]
	RETURN					

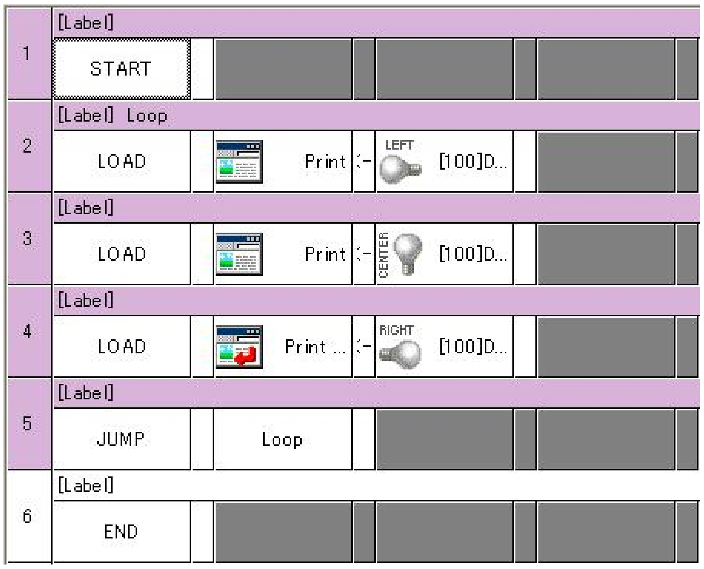
Routine that waits until one motion is finished



25	[Label] Moving done						[Comment]		
	IF	(MOVING [2]Dyn...	=	1	)	OR			
26	[Label]						[Comment]		
	CONT IF	(MOVING [3]Dyn...	=	1	)	THEN	JUMP	Moving...	
27	[Label]						[Comment]		
	RETURN								
28	[Label]						[Comment]		
	END								

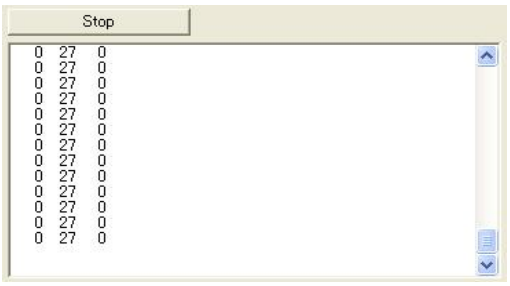
5-5. Surrounding Light Sensing Function

Let's create a behavior control program that will print the surround light sensor values on the screen. There are three light sensors. Let's print the value of each, one by one on the screen. Select LOAD for the command, CM-5 screen as the left parameter, and sensor 1 of the Dynamixel as the right parameter.
The program will look like the following. Refer to the "Examples\Example(Read light sensor value).bpg" inside the CD.



1	[Label]			
	START			
2	[Label] Loop			
	LOAD	Print	:-	LEFT [100]D...
3	[Label]			
	LOAD	Print	:-	CENTER [100]D...
4	[Label]			
	LOAD	Print ...	:-	RIGHT [100]D...
5	[Label]			
	JUMP	Loop		
6	[Label]			
	END			

Depending on the brightness of light, it will take values ranging from 0 to 255. Because the sunlight is very strong, the test outdoor may not work. In indoor, sensor can be very sensitive to devices that emit strong light, including flashlight, incandescent lamp and others.
There are also many other items related to light sensing. Refer to the AX-S1 manual for more information.



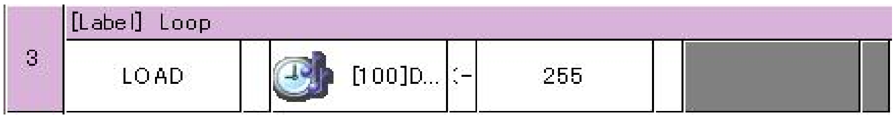
```
0  27  0
0  27  0
0  27  0
0  27  0
0  27  0
0  27  0
0  27  0
0  27  0
0  27  0
0  27  0
0  27  0
0  27  0
0  27  0
```

5-6. Melody Playing Function

AX-S1 can make a sound as there is a buzzer inside. Let’ s make a robot that can make sound or that exhibit expressions using the melody playing function.

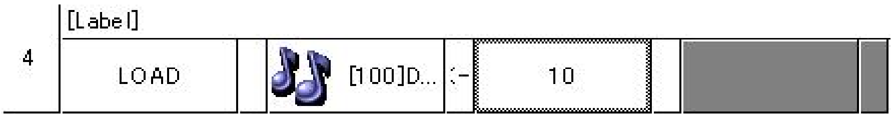
Special Melody Play

Special melody is already built inside and there are 27 melody sounds that can be played by AX-S1.To play the special melody sound, the “buzzer sound” value must be set to 255 first.



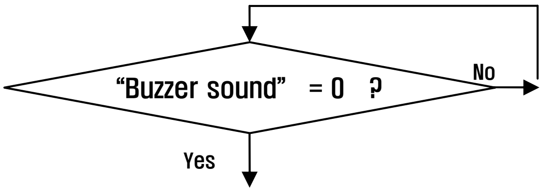
3	[Label] Loop				
	LOAD		[100]D...	:-	255

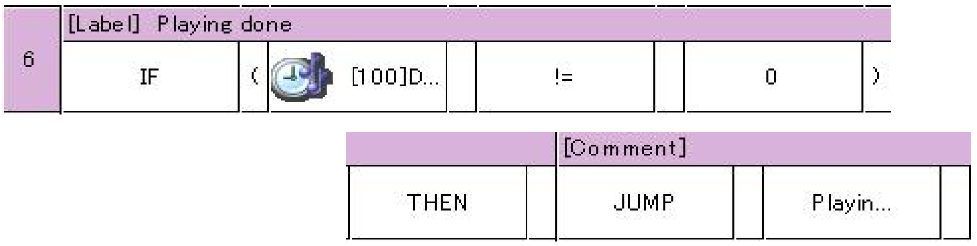
Next, input the value between 0 and 26 to produce the sound.



4	[Label]				
	LOAD		[100]D...	:-	10

If you play other sound before the previous sound completes, the previous sound will just finish. To solve this problem, you have to wait until the melody sound completes. For this, check whether “buzzer sound” is set to 0.

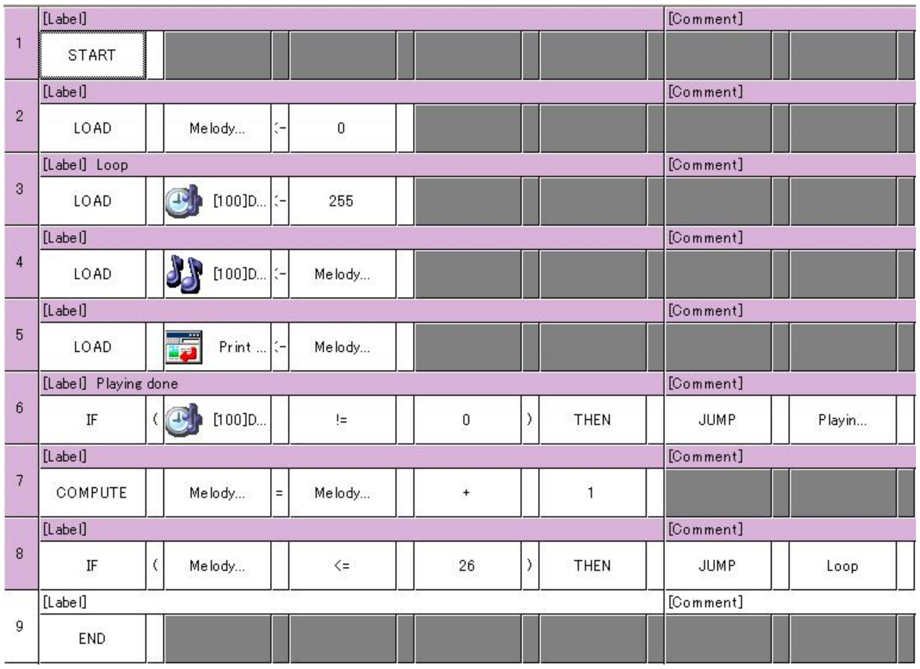




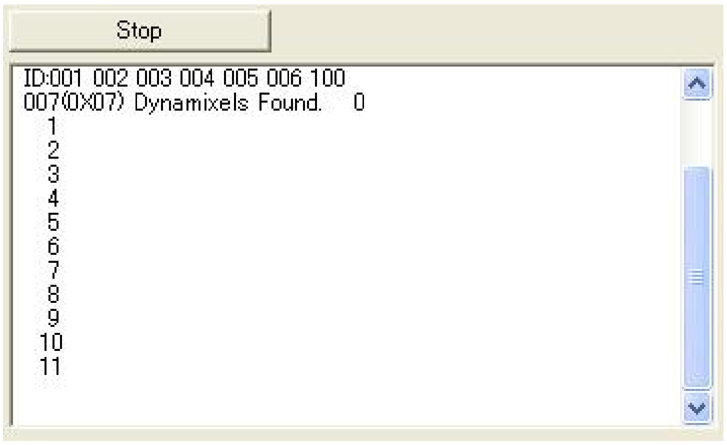
6	[Label] Playing done						
	IF	(	[100]D...	!=		0	)

	[Comment]			
THEN		JUMP		Playin...

Now, let’ s make a program that plays the special melody sounds from 0 to 26 and that print the numbers corresponding to the sounds.
The program will look like the following. Refer to the “Examples\Example(Play special melody sound).bpg].bpg” inside the CD



1	[Label]								[Comment]	
	START									
2	[Label]								[Comment]	
	LOAD		Melody...	<-	0					
3	[Label] Loop								[Comment]	
	LOAD		[100]D...	<-	255					
4	[Label]								[Comment]	
	LOAD		[100]D...	<-	Melody...					
5	[Label]								[Comment]	
	LOAD		Print ...	<-	Melody...					
6	[Label] Playing done								[Comment]	
	IF	(	[100]D...		!=	0	)	THEN	JUMP	Playin...
7	[Label]								[Comment]	
	COMPUTE		Melody...	=	Melody...	+		1		
8	[Label]								[Comment]	
	IF	(	Melody...		<=	26	)	THEN	JUMP	Loop
9	[Label]								[Comment]	
	END									



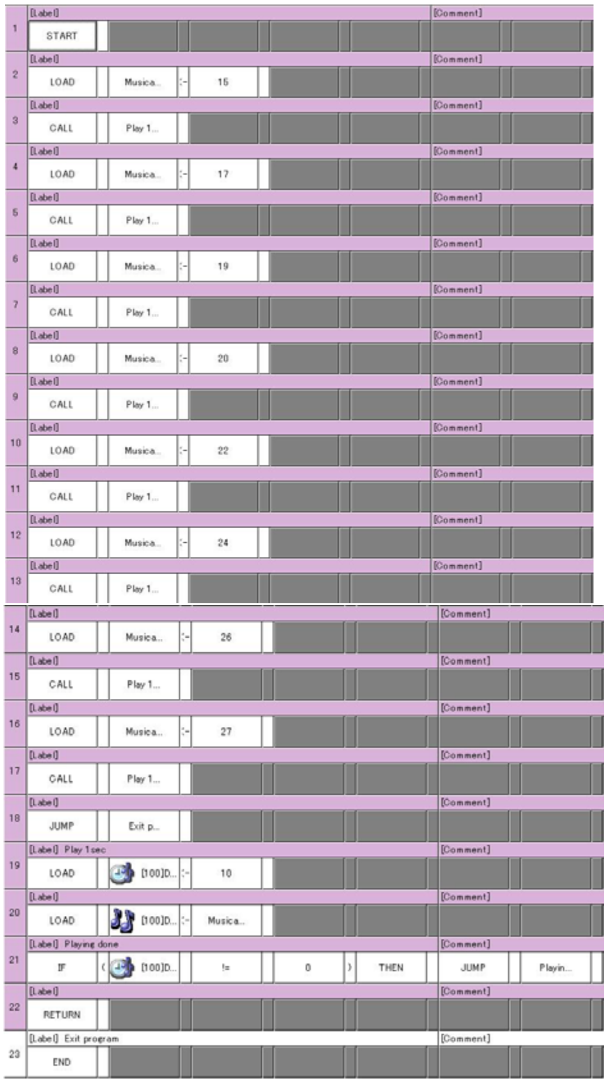
```
ID:001 002 003 004 005 006 100
007(0X07) Dynamixels Found.    0
   1
   2
   3
   4
   5
   6
   7
   8
   9
  10
  11
```

Musical Notes

While working with the robot never put your face close to the robot. Musical notes consist of “C D E F G A B C” and AX-S1 has a range of three octaves, consisting of 52 notes. For more details related to musical notes, refer to the AX-S1 manual.

When you input the numbers ranging from 0 to 253, it will make a sound lasting “0.1 X “Buzzer sound.” That is, if you input the value of 10, the sound will last for one second (0.1 X 10 = 1). However, if you input the value of 0, instead of 0 (0.1 X 0 = 0), it will last for 0.3 second. If you want to make a sound that continues forever, input the value of 254; and to stop, input 0.

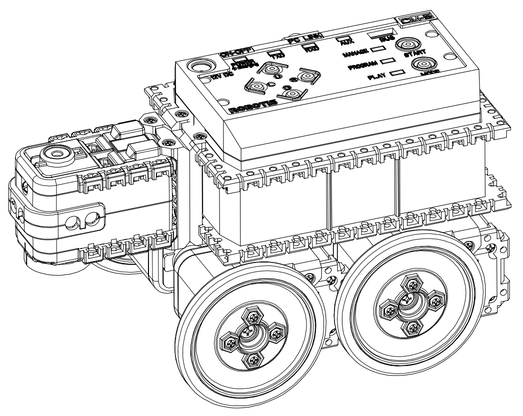
An example below will play the musical notes “C D E F G A B C” each for one second and will end it. Refer to “Examples\Example(Play musical note).bpg” inside the CD.



```
1   [Label]                    [Comment]
    START
2   [Label]                    [Comment]
    LOAD    Musica...  :=  15
3   [Label]                    [Comment]
    CALL    Play 1...
4   [Label]                    [Comment]
    LOAD    Musica...  :=  17
5   [Label]                    [Comment]
    CALL    Play 1...
6   [Label]                    [Comment]
    LOAD    Musica...  :=  19
7   [Label]                    [Comment]
    CALL    Play 1...
8   [Label]                    [Comment]
    LOAD    Musica...  :=  20
9   [Label]                    [Comment]
    CALL    Play 1...
10  [Label]                    [Comment]
    LOAD    Musica...  :=  22
11  [Label]                    [Comment]
    CALL    Play 1...
12  [Label]                    [Comment]
    LOAD    Musica...  :=  24
13  [Label]                    [Comment]
    CALL    Play 1...
14  [Label]                    [Comment]
    LOAD    Musica...  :=  26
15  [Label]                    [Comment]
    CALL    Play 1...
16  [Label]                    [Comment]
    LOAD    Musica...  :=  27
17  [Label]                    [Comment]
    CALL    Play 1...
18  [Label]                    [Comment]
    JUMP    Exit p...
19  [Label] Play 1sec          [Comment]
    LOAD    [100]D...  :=  10
20  [Label]                    [Comment]
    LOAD    [100]D...  :=  Musica...
21  [Label] Playing done       [Comment]
    IF  (  [100]D...  !=  0  )  THEN  JUMP  Playin...
22  [Label]                    [Comment]
    RETURN
23  [Label] Exit program       [Comment]
    END
```

5-7. Assembling Intelligent Car that Uses Sensor

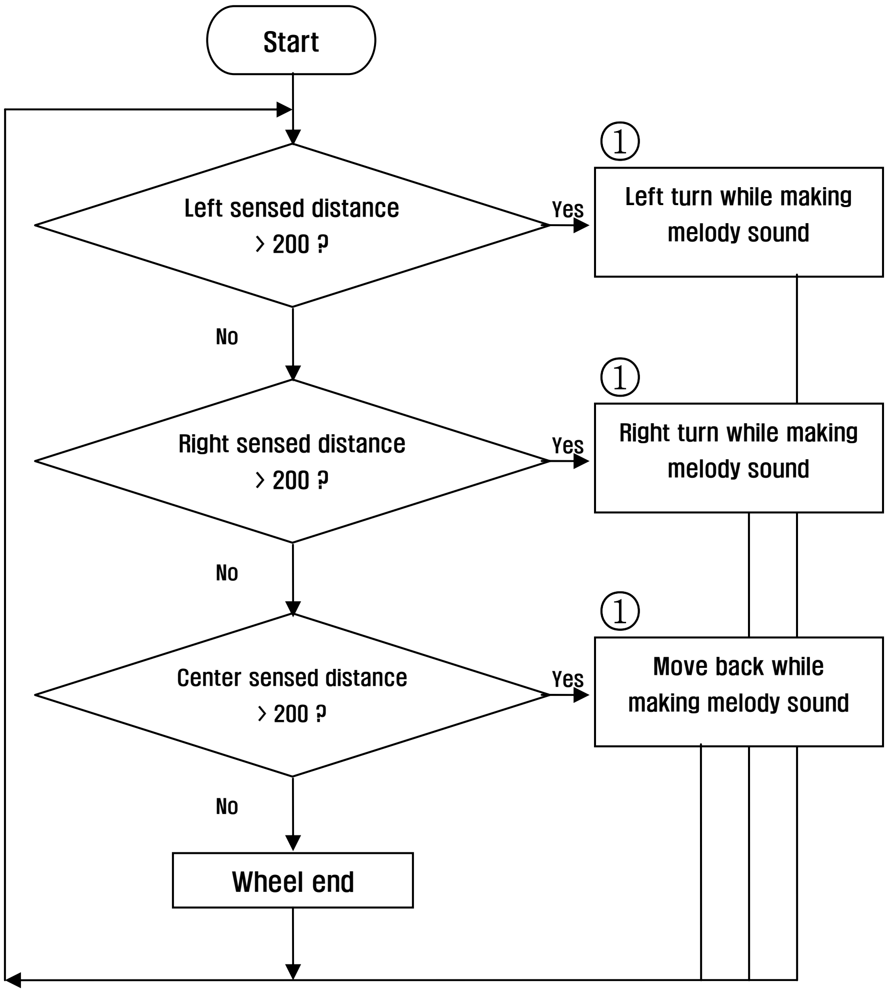
We are now going to build a very interesting robot. Let’ s make an intelligent car that makes a melody sound and that moves in opposite direction when an object moves towards it.
Refer to the “2-2-9. Obstacle Detection Car” of QuickStart whenever it is necessary.

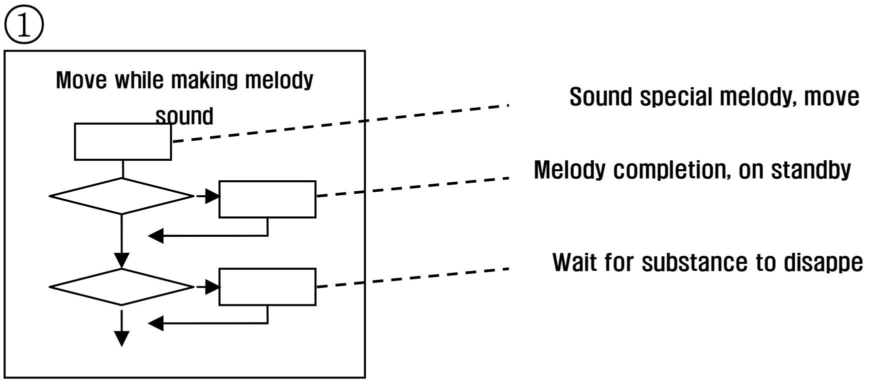


Detail behavior controls are as follows.

- If the left sensed distance value is greater than 200, it will make left turn while making melody sound.
- If the right sensed distance value is greater than 200, it will make right turn while making melody sound.
- If the center sensed distance value is greater than 200, it will go backward while making melody sound.
- If there are no changes, the car will stop.
- The melody will be special melody play and upon call will generate sound only once.

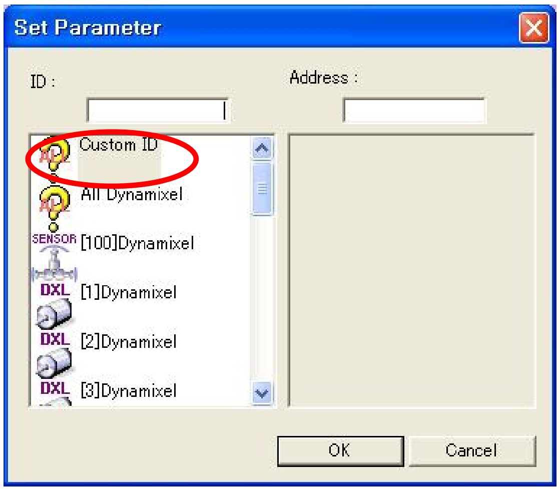
As the behavior control of above robot involves complex behavior, it is recommended that you use function.


Start
Left sensed distance > 200 ?
Yes
①
Left turn while making melody sound
No
Right sensed distance > 200 ?
Yes
①
Right turn while making melody sound
No
Center sensed distance > 200 ?
Yes
①
Move back while making melody sound
No
Wheel end

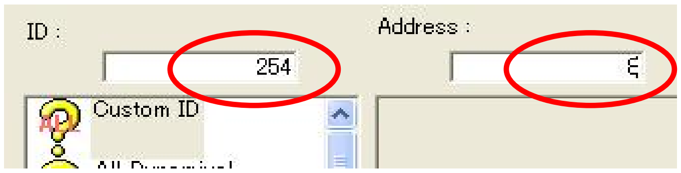

①
Move while making melody sound
Sound special melody, move
Melody completion, on standby
Wait for substance to disappe

AX-12 Continuous Turn Mode

To use the wheel motor of AX-12, it is required to set the "continuous turn mode" rather than the "joint mode." For more details on the "continuous turn mode," refer to the AX-12 manual.
Among the items of AX-12, if you set the "CCW angle limit" 0, it will be set as "continuous turn mode," and it will be set to "joint mode" if you input values other than the 0. If you look closely however, you will notice that there is no "CCW angle limit" item. For items not available, use "Custom ID" item to directly control the behavior. In this case, to use "continuous turn mode," you have to select "Custom ID."

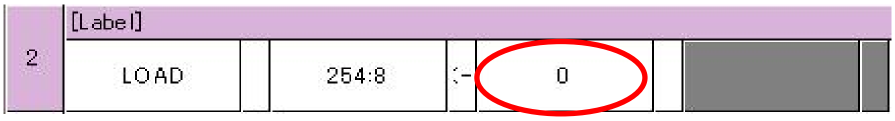


Here, you input the value 254, the value representing all Dynamixel in "ID" and for "address," input the value of 8, the designated value for "CCW Angle Limit." If you want to select specified ID of Dynamixel, input the applicable value in "ID."



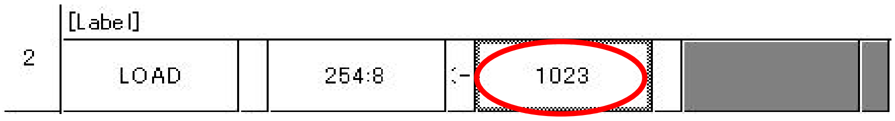
ID : 254
Address : 8

Input the value of 0 for wheel mode for the AX-12 parameter. Refer to "Examples\Example(Change endless turn mode).bpg" inside the CD.



```
2  [Label]
   LOAD    254:8  <-  0
```

If you want to change back the AX-12 to "joint mode," input the value of 1023. Refer to "Examples\Example(Change joint mode).bpg" inside the CD.



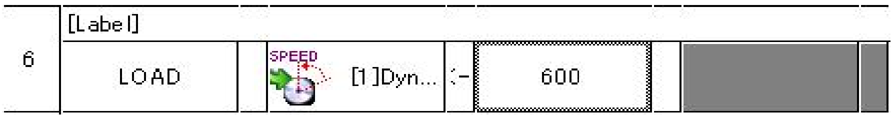
	[Label]						
2	LOAD		254:8	<-	1023		

Be aware as once you set the AX-12 mode, it will be in a set mode until you change it directly. Thus, if you try to control the AX-13 by the "continuous turn mode" when it is set as the "joint mode," it will not work properly, and vice versa.

AX-12 Continuous Turn Mode Control

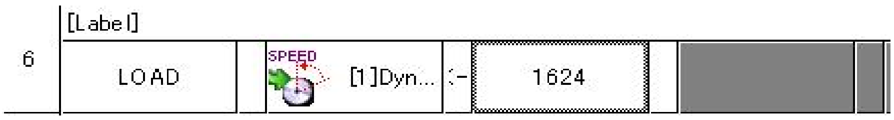
When the AX-12 is in "continuous turn mode," it will be controlled in the "motion speed," not "desired position." If you input the value from 0 to 1023 in "motion speed," AX-12 will rotate clockwise. Of course, the value of 0 will make the AX-12 stationary. However, if you put the value above 1024, it will rotate counter-clockwise corresponding to the inputted value. For example, if you input 600, it will rotate clockwise corresponding to the speed of 600, whereas, if you input 1624, it will rotate counter-clockwise, once again at the speed of 600(600+1024).

[정 회전]



	[Label]						
6	LOAD		SPEED [1]Dyn...	<-	600		

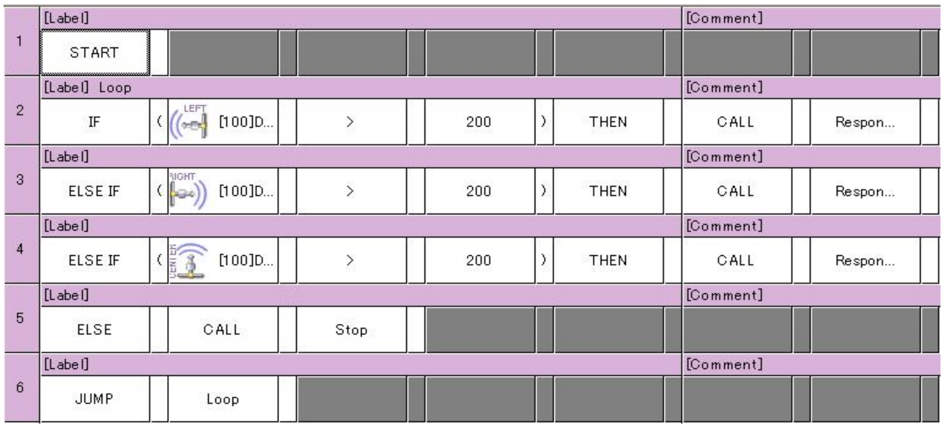
[역 회전]



	[Label]						
6	LOAD		SPEED [1]Dyn...	<-	1624		

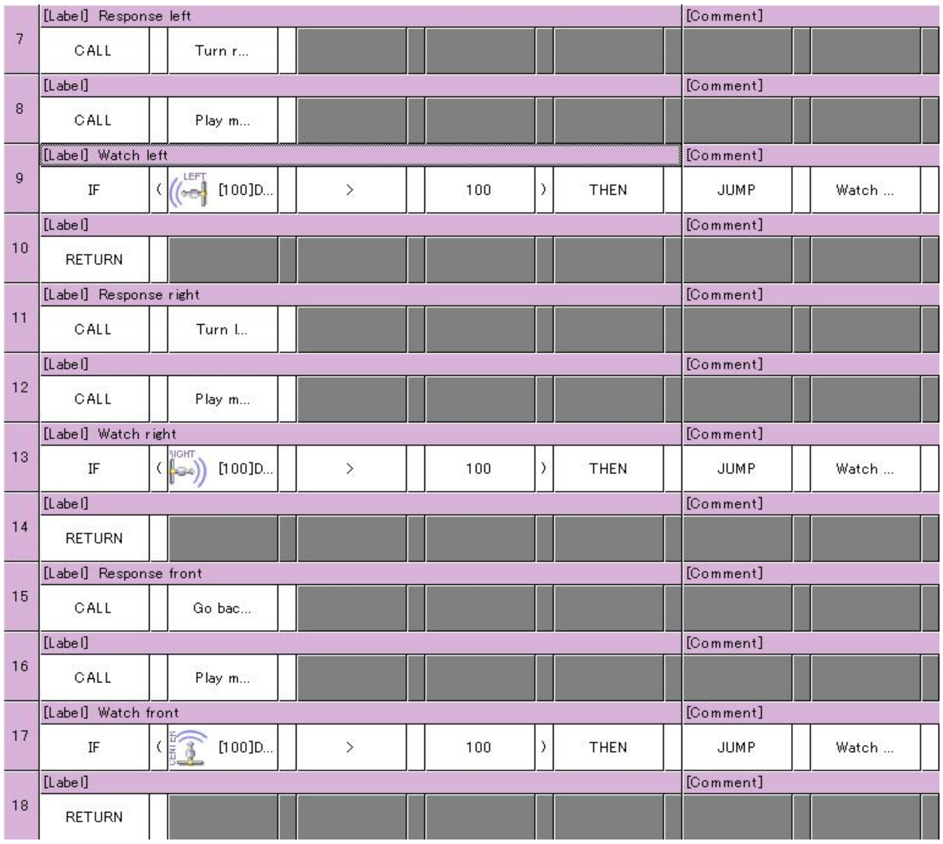
For the intelligent car example here, AX-12 will be set to the “continuous turn mode.” To do so, refer to “Examples\Example(Change endless turn mode).bpg” inside the CD. Next will be the intelligent car behavior control program. For that, refer to “Examples\Example(Intelligent Car).bpg” inside the CD.

Main routine



	[Label]							[Comment]
1	START							
2	[Label] Loop							[Comment]
	IF	(LEFT [100]D...	>	200)	THEN	CALL	Respon...	
3	ELSE IF	(RIGHT [100]D...	>	200)	THEN	CALL	Respon...	[Comment]
4	ELSE IF	(CENTER [100]D...	>	200)	THEN	CALL	Respon...	[Comment]
5	ELSE	CALL	Stop					[Comment]
6	JUMP	Loop						[Comment]

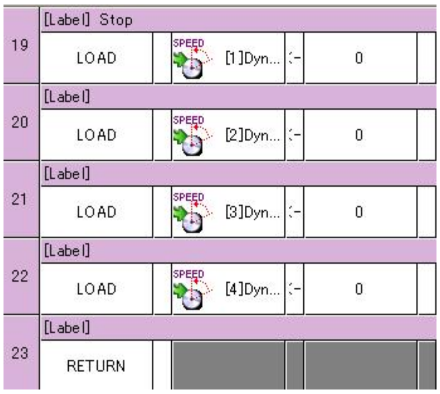
Routine for left, right and center reaction to the sensors



	[Label]							[Comment]
7	[Label] Response left							[Comment]
	CALL	Turn r...						
8	CALL	Play m...						[Comment]
9	[Label] Watch left							[Comment]
	IF	(LEFT [100]D...	>	100)	THEN	JUMP	Watch ...	
10	RETURN							[Comment]
11	[Label] Response right							[Comment]
	CALL	Turn l...						
12	CALL	Play m...						[Comment]
13	[Label] Watch right							[Comment]
	IF	(RIGHT [100]D...	>	100)	THEN	JUMP	Watch ...	
14	RETURN							[Comment]
15	[Label] Response front							[Comment]
	CALL	Go bac...						
16	CALL	Play m...						[Comment]
17	[Label] Watch front							[Comment]
	IF	(CENTER [100]D...	>	100)	THEN	JUMP	Watch ...	
18	RETURN							[Comment]

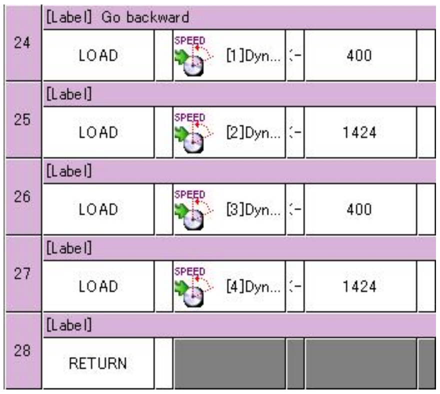
Move routine

[Stop]



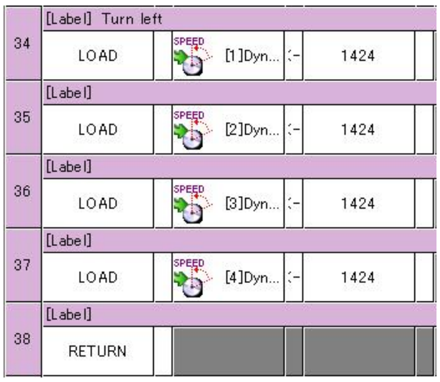
19	[Label] Stop				
	LOAD	SPEED	[1]Dyn...	<-	0
20	[Label]				
	LOAD	SPEED	[2]Dyn...	<-	0
21	[Label]				
	LOAD	SPEED	[3]Dyn...	<-	0
22	[Label]				
	LOAD	SPEED	[4]Dyn...	<-	0
23	[Label]				
	RETURN				

[Backward]



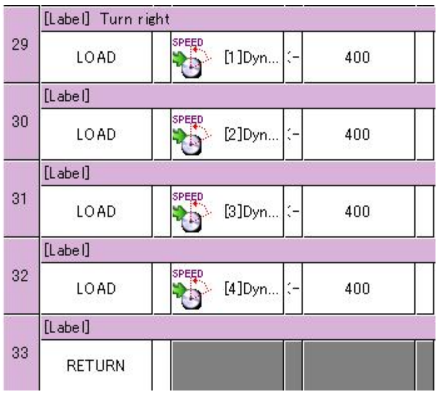
24	[Label] Go backward				
	LOAD	SPEED	[1]Dyn...	<-	400
25	[Label]				
	LOAD	SPEED	[2]Dyn...	<-	1424
26	[Label]				
	LOAD	SPEED	[3]Dyn...	<-	400
27	[Label]				
	LOAD	SPEED	[4]Dyn...	<-	1424
28	[Label]				
	RETURN				

[Left turn]



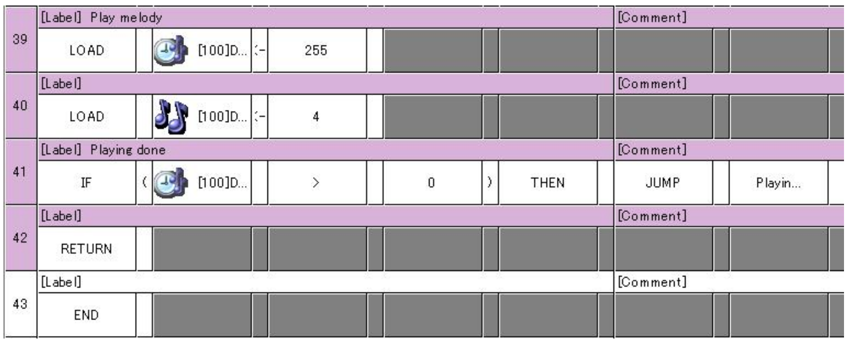
34	[Label] Turn left				
	LOAD	SPEED	[1]Dyn...	<-	1424
35	[Label]				
	LOAD	SPEED	[2]Dyn...	<-	1424
36	[Label]				
	LOAD	SPEED	[3]Dyn...	<-	1424
37	[Label]				
	LOAD	SPEED	[4]Dyn...	<-	1424
38	[Label]				
	RETURN				

[Right turn]



29	[Label] Turn right				
	LOAD	SPEED	[1]Dyn...	<-	400
30	[Label]				
	LOAD	SPEED	[2]Dyn...	<-	400
31	[Label]				
	LOAD	SPEED	[3]Dyn...	<-	400
32	[Label]				
	LOAD	SPEED	[4]Dyn...	<-	400
33	[Label]				
	RETURN				

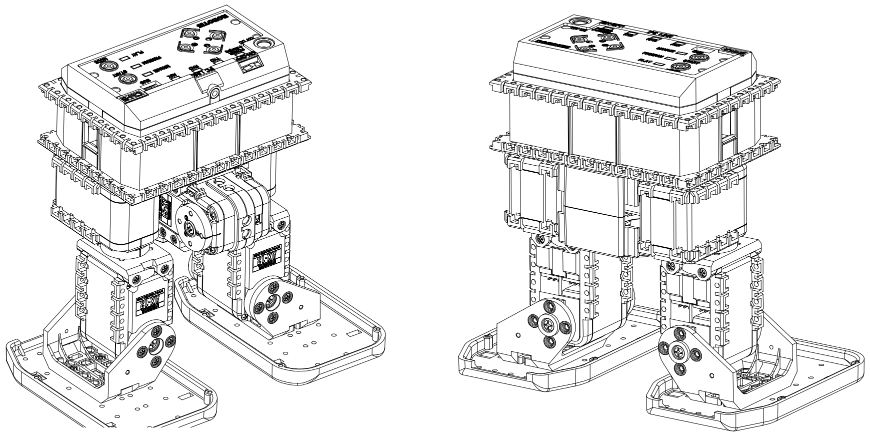
Melody sound routine



39	[Label] Play melody								[Comment]	
	LOAD		[100]D...	<-	255					
40	[Label]								[Comment]	
	LOAD		[100]D...	<-	4					
41	[Label] Playing done								[Comment]	
	IF	(	[100]D...		>	0	)	THEN	JUMP	Playin...
42	[Label]								[Comment]	
	RETURN									
43	[Label]								[Comment]	
	END									

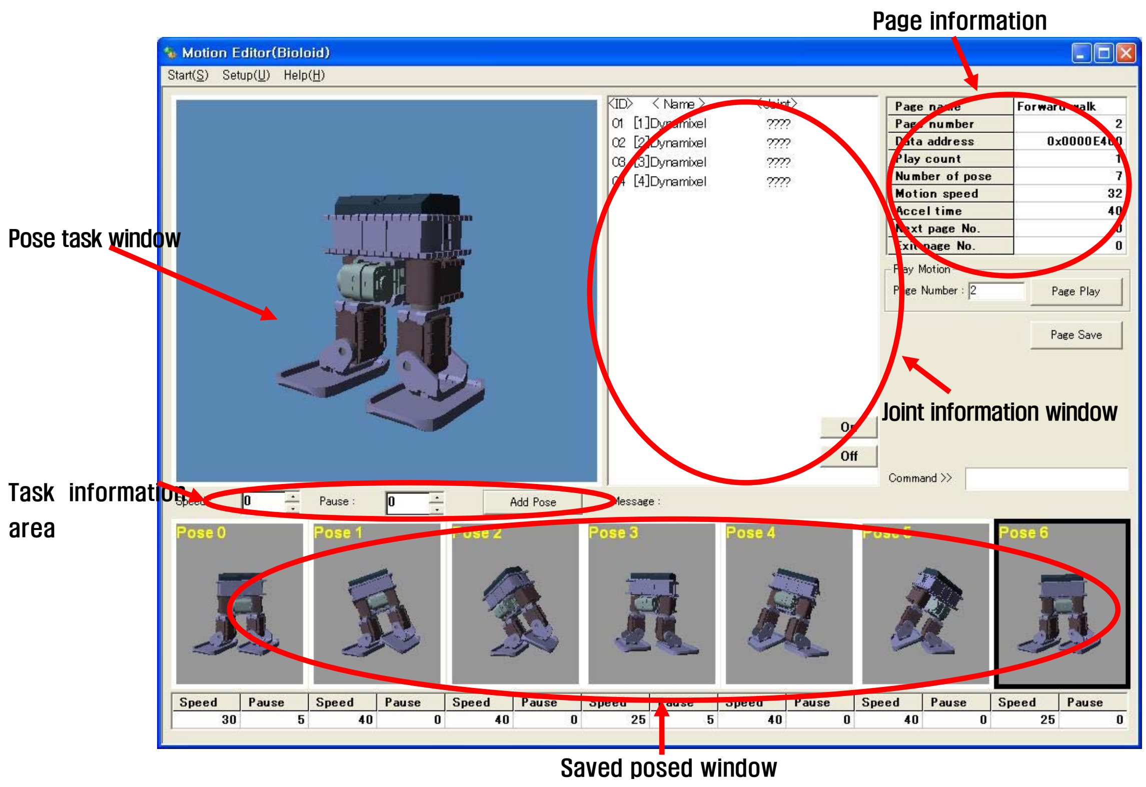
6. Motion Editor

There are two ways to edit the motion. One is to use the motion editor and the other way is to execute the program mode using the robot terminal. The motion editor has a graphical user interface so it is easy for beginners to use. The robot terminal uses text mode, so it lets users see all information all at once, thus more useful to advanced users.
For this chapter, we will take a look at the example of simple two-legged walking Droid. Refer to the “2-2-14 Walking Droid,” of the QuickStart and make the hardware part in advance.



6-1. Using the Motion Editor

Motion Editor The motion editor has a graphical user interface that allows the user to edit a multi-jointed robot made up of Dynamixels. A user can create and edit motions by moving the joints by hand and saving each pose using the motion editor. The user can also connect or repeat edited motions.
The following screen will show up when you run the motion editor.



<ID>	< Name >	<Joint>
01	[1]Dynamixel	????
02	[2]Dynamixel	????
03	[3]Dynamixel	????
04	[4]Dynamixel	????

Page name	Forward walk
Page number	2
Data address	0x0000E400
Play count	1
Number of pose	7
Motion speed	32
Accel time	40
Next page No.	0
Exit page No.	0

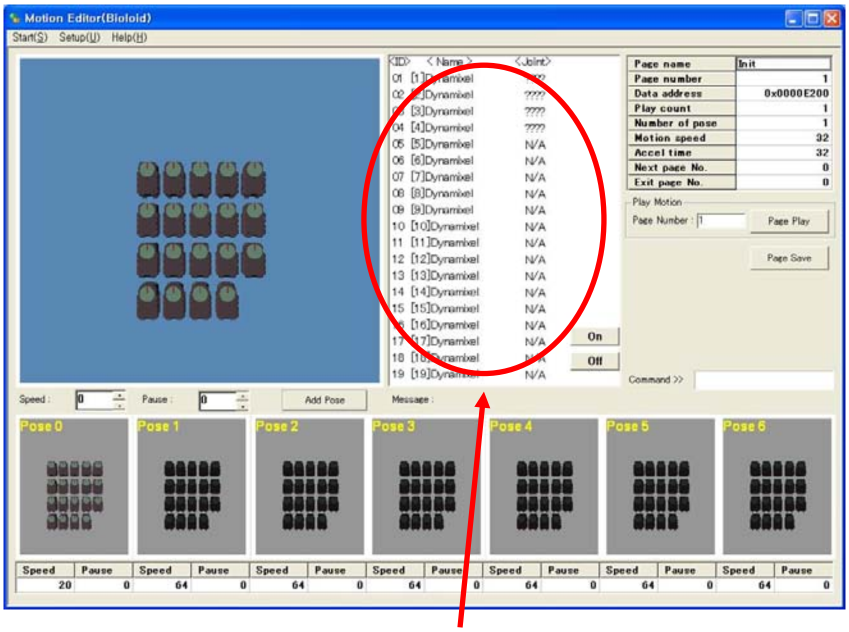
Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause
30	5	40	0	40	0	25	5	40	0	40	0	25	0

Connection In order to use the motion editor, the robot has to be connected to a PC in standby mode. If the connection fails, go to the start menu and set the "Com port" correctly and check if the "Com port" is in use or not.

Robot Profile

If you cannot see above screen when the motion editor is running and there is no problem with the connection, it means that the robot profile is not set properly. The robot profile is a file that contains the information of the robot composition elements that defines how many joints it has, the name of them, and the ID number assigned to them. This information will be different for robot to robot, thus a different robot profile is needed for a different robot. The robot profile

information file will be used by the behavior control programmer and in the motion editor.
The file extension for the robot profile is *.rbt
Below screen is, after the program is installed, when the motion editor is executed and is applied to the default robot profile (default.rbt).



<ID>	< Name >	<Joint>
01	[1]Dynamixel	????
02	[2]Dynamixel	????
03	[3]Dynamixel	????
04	[4]Dynamixel	????
05	[5]Dynamixel	N/A
06	[6]Dynamixel	N/A
07	[7]Dynamixel	N/A
08	[8]Dynamixel	N/A
09	[9]Dynamixel	N/A
10	[10]Dynamixel	N/A
11	[11]Dynamixel	N/A
12	[12]Dynamixel	N/A
13	[13]Dynamixel	N/A
14	[14]Dynamixel	N/A
15	[15]Dynamixel	N/A
16	[16]Dynamixel	N/A
17	[17]Dynamixel	N/A
18	[18]Dynamixel	N/A
19	[19]Dynamixel	N/A

Page name	Init
Page number	1
Data address	0x0000E200
Play count	1
Number of pose	1
Motion speed	32
Accel time	32
Next page No.	0
Exit page No.	0

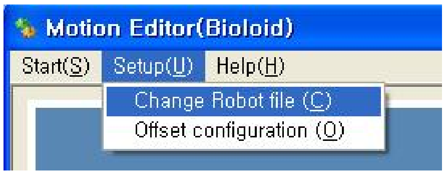
Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause
20	0	64	0	64	0	64	0	64	0	64	0	64	0

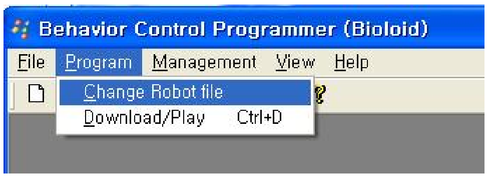
[Part where robot profile is used in the motion editor]

Robot Profile Change

If a specific robot information file has not been selected after installing the program, then the robot profile file called default.rbt will be selected by default. Default.rbt contains the information for 1 AX-S1 and 18 AX-12s, and the names of the joints are set as Dynamixel [ID]. Let' s change this so we can use it on a "walking Droid.rbt." with 4 joints.

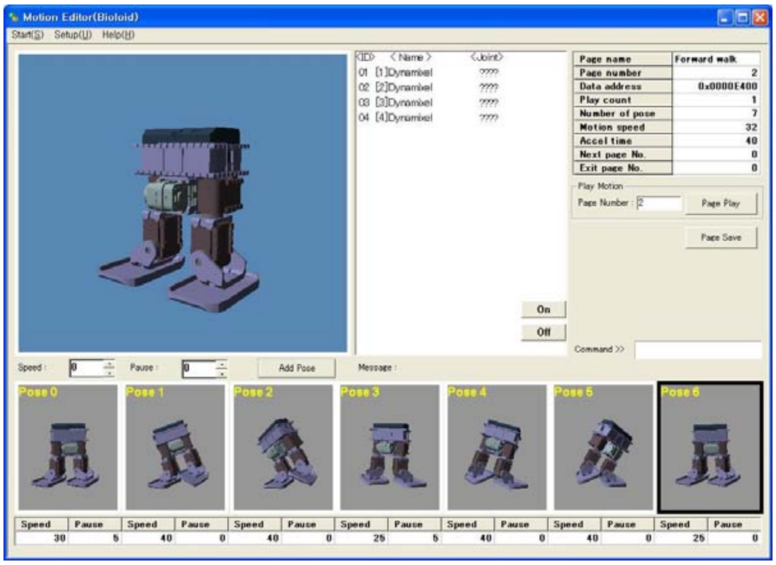
In the setup menu select "change robot file" and click on "walking Droid.rbt."
One thing to keep in mind is that the changed robot information will be applied starting with the next program run. In the behavior control programmer, "Change Robot File" can be found under the menu "program."




Behavior Control Programmer (Bioloid)
File Program Management View Help
Change Robot file
Download/Play Ctrl+D

Re-setup of robot profile

After selecting the "Walking Droid.rbt." the robot profile information will look like the following.

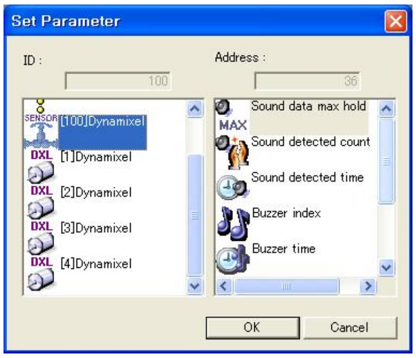


<ID>	< Name >	<Joint>
01	[1]Dynamixel	????
02	[2]Dynamixel	????
03	[3]Dynamixel	????
04	[4]Dynamixel	????

Page name	Forward walk
Page number	2
Data address	0x0000E400
Play count	1
Number of pose	7
Motion speed	32
Accel time	40
Next page No.	0
Exit page No.	0

Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause
30	5	40	0	40	0	25	5	40	0	40	0	25	0

[Walking Droid. rbt. executed in the motion



[Walking Droid. rbt. executed in the Behavior Control Programmer]

Pose

The pose is an instance of a motion. For example, in the figure right, you will need many poses to make the robot move one side step.

Pose 0
Pose 1
Pose 2
Pose 3
Pose 4

Creating a Pose

On the motion editor screen, the big picture on the left is the pose work area, and it shows the current state of the robot. Here, you create a motion by setting the desired pose and dragging it to the saved pose area. Connecting the poses in the saved pose area smoothly creates a motion.

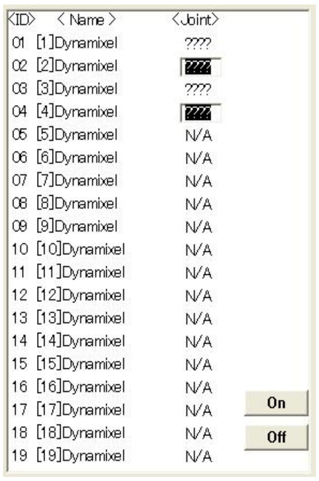
The numbers and poses that are shown on a motion editor screen are the information for a single motion page. A motion page is made up of 7 poses and 64 bytes of page information. The Bioloid has up to 127 motion pages.

- The Bioloid' s motion memory = 127 motion pages
- 1 motion page = 7 poses + page information (64 bytes)
- 1 pose = Maximum 30 joints information (position, velocity, and stop time)

The behavior control program plays the motion using its page number.

The numbers in the joint information window are the position values of the robot joints. The robot inside the pose task window is the picture of the actual current robot. Therefore the joint position values shown inside the joint information window is the actual current joint position of the robot.

Inside the joint information window there is a button turning ON or OFF the torque of the joints.



<ID>	< Name >	<Joint>
01	[1]Dynamixel	????
02	[2]Dynamixel	????
03	[3]Dynamixel	????
04	[4]Dynamixel	????
05	[5]Dynamixel	N/A
06	[6]Dynamixel	N/A
07	[7]Dynamixel	N/A
08	[8]Dynamixel	N/A
09	[9]Dynamixel	N/A
10	[10]Dynamixel	N/A
11	[11]Dynamixel	N/A
12	[12]Dynamixel	N/A
13	[13]Dynamixel	N/A
14	[14]Dynamixel	N/A
15	[15]Dynamixel	N/A
16	[16]Dynamixel	N/A
17	[17]Dynamixel	N/A
18	[18]Dynamixel	N/A
19	[19]Dynamixel	N/A

Off This will turn off the torque of the selected joint. After pressing the OFF button you can move the joints by hand.

On After moving the joint to the desired position, press the ON button to see the joint angle value on the joint information window. The torque will be back on, locking its joint position.

You can select several joints at the same time for this operation. Hold the Ctrl key and select the joints that you want to turn ON or OFF as once.

Shown below is a summary of the process for creating a motion.

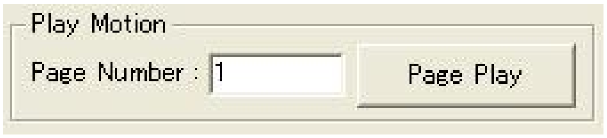
Joint OFF → Create the desired pose → Joint On → Add pose

Pose Speed Even if the joint values have been set properly, you still might not be able to get the desired motion unless you set the speed between poses correctly. On the bottom of the screen, there is a place where you can set the pose speed.

Spot Time Sometimes you will want to stop while moving from one pose to another. If you give the robot a stop time, it will stop moving for 7.8 msec per this value and then play the next pose.

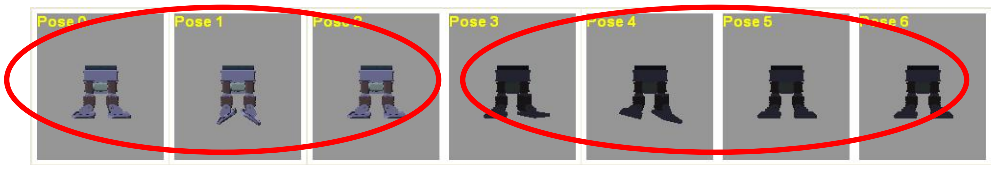
Add Pose If you press the “Add Pose” button in the task information area, the joint values in the pose task window will be added to the saved pose window. Let’ s create some more poses using the ON, OFF commands.

Play Now let’ s connect the inputted poses to create a motion. If you press the play button the motion will be played using the poses saved in the saved pose window. This function is useful when you want to test the motion that is currently being created. If you press the play button and the motion is executed, the button will changed to stop. If you press the stop button, the motion that is currently running will stop. Also, if you input the page number, motion that has been saved in that page will be activated.



Editing the Pose

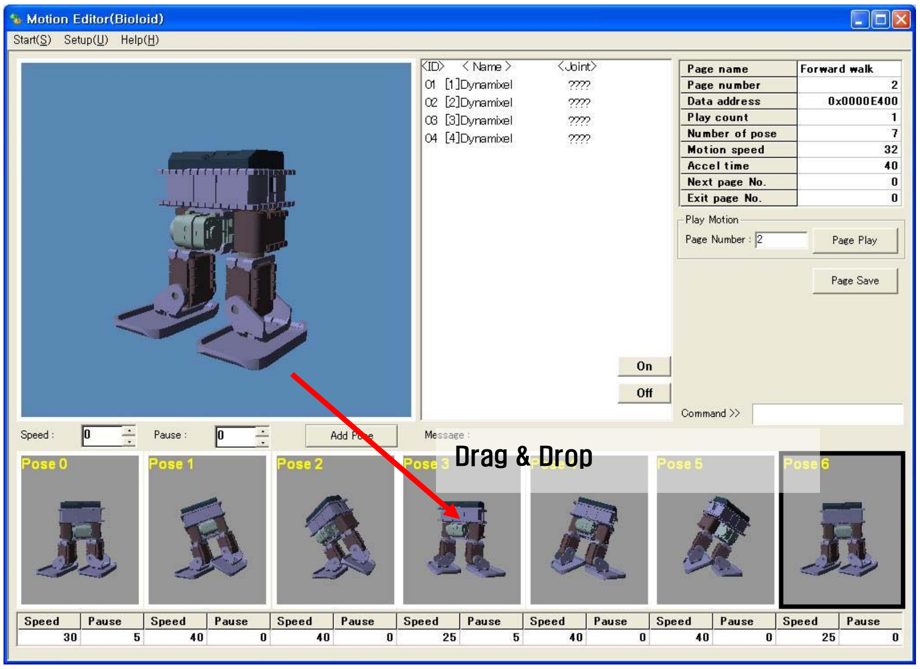
If you want to execute the pose that has been saved in the save pose window, double-click the applicable pose. Take note however that to prevent robot from breaking down due to execution of invalid pose, the program will request for the confirmation. By checking the color of robot as shown below, you can tell which poses are valid. For invalid poses, it is indicated by the color black. For more details, refer to a next page when it covers the valid pose number.



Valid poses Invalid poses

Pose Save

Addition of pose always goes in the end of pose save window. If you want to put the pose in particular place, however, you can drag and drop as shown below. Take note however that the previous pose will be overwritten.



<ID>	< Name >	<Joint>
01	[1]Dynamixel	????
02	[2]Dynamixel	????
03	[3]Dynamixel	????
04	[4]Dynamixel	????

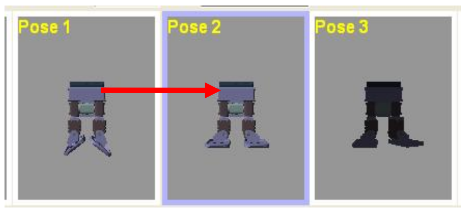
Page name	Forward walk
Page number	2
Data address	0x0000E400
Play count	1
Number of pose	7
Motion speed	32
Accel time	40
Next page No.	0
Exit page No.	0

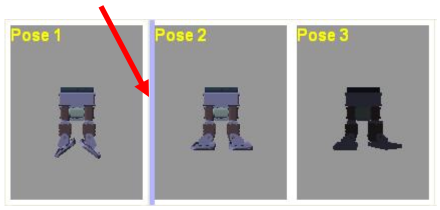
Page Number : 2

Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause	Speed	Pause
30	5	40	0	40	0	25	5	40	0	40	0	25	0

Pose Move and Insertion

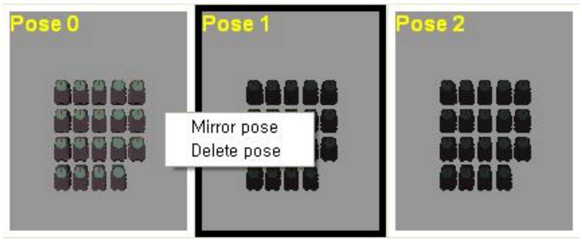
As you can see from the pictures shown below, you can move the pose by dragging to applicable place. Also, if you want to insert the pose, just move it between appropriate poses.
This function is also possible when you want to move pose from the pose window.





Pose move Pose insert

Delete Right click the pose to be removed and select delete.



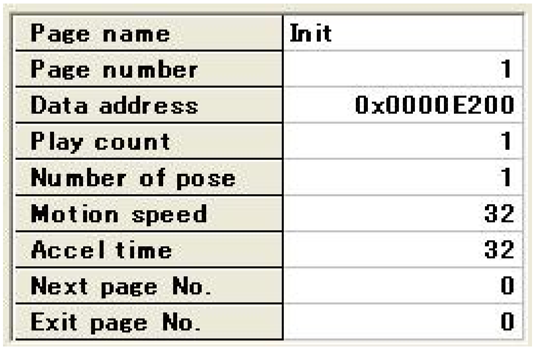
The following is a summary of the above.

- The key to motion editing is posing editing and saving.
- A pose is created using the ON, OFF button and actually moving the robot joint by hand. Also, you have to set the pose speed and stop time.
- You can move to change order or insert a pose by dragging a pose with your mouse.
- The motion can be verified by playing it by pressing the Play button.

Save Page During editing, the motion data is stored inside the CM-5' s RAM. When you are finished creating the motion, use the save command to save the motion page to the flash memory.

Page information

You can find the page information on the upper right corner of the screen. To edit the page information double-clicks the area you want to edit and type in the value.



Page name	Init
Page number	1
Data address	0x0000E200
Play count	1
Number of pose	1
Motion speed	32
Accel time	32
Next page No.	0
Exit page No.	0

Page Name You can give a name to each page. The page name is empty as default. If you give a name for each page it will be convenient for later use.

Page Number This is the unique number (1~127) of each page. If you change the number, it is possible to move to applicable page.

Start Address This is the location of the code memory of the page that you are currently working on. You do not have to be concerned about this information.

Play Count This item tells how many times the motion will be played. The default value is 1.

Number of Poses

When you play the motion, the poses from 0 to the valid number of poses will be played. Inside the saved pose window, the poses that are not valid are in black color. Sometimes you will need to change the number of valid poses during editing, for example, if you want to play the first few poses instead of the whole thing.

Motion Speed Use this when you want to adjust the play speed of the whole page. The default value is 32. Setting the value to 64 would be the same as doubling the speed of each of the poses individually.

Acceleration Time

Every time a motion is played, the joints go through a process of acceleration constant velocity deceleration. The acceleration time is the time of acceleration and deceleration (ramp up time plus ramp down time). Reducing the acceleration time will make the motors acceleration faster and stress on the joints. Increasing the acceleration time might create an interval which makes it impossible to complete a motion.

Next Page No.

You can set the next motion page to be executed after the current page is finished being played. This can be done by indicating the page to be played next in the “next page No.” If you don’ t need to play another motion page then set the value to 0 (default value is 0). If you set “Next Page No.” as the number of the current page, then playback will continue infinitely in a loop. In this case, you can press the ESC key to stop the playback. Playback will be stopped after the current page motion is completed.

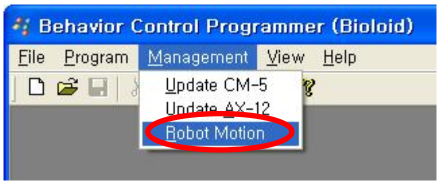
Final Page No.

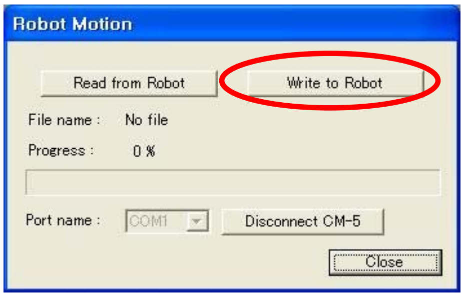
During motion play, if a signal (such as from a remote controller) is received, the playback will end after the executing the final page defined by this. If this feature is not needed, then set its value to 0. To use the “motion stop” command in behavior control program, load 0 in CM-5 “robot motion” item.

Motion Data Download

To download created robot motion to PC, use the “manage” menu of behavior control programmer.

Motion data filename extension is mtn and to download, select "manage" from the behavior control program menu and choose 'motion data download." Here, open "Examples\Example(Walking Droid)..mtn in the CD and download it. To download the motion, follow the following steps. Also, refer to the "2-1-2. Robot Program Download" of the manual.

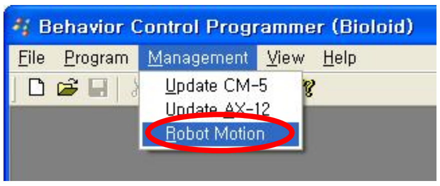


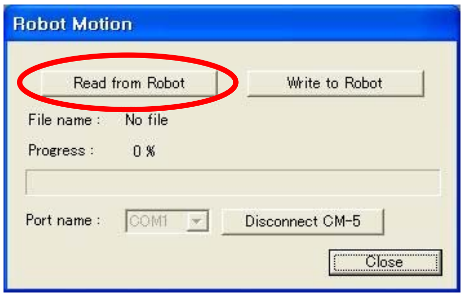


After download is completed, click close to finish the program.

Motion Data Upload

If you created the motion with the motion editor and saved the motion, there is a chance of data loss as modified motion data is inside the robot. To save in more secure fashion, it is necessary to the save motion data in PC. The motion data upload is used in such case.





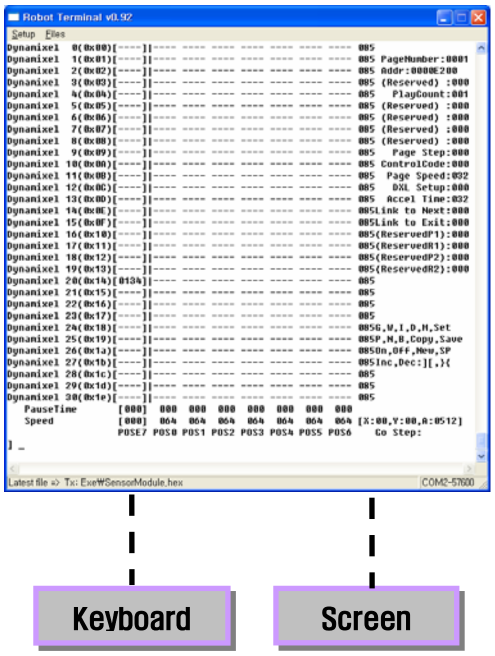
As the motion data upload is more complicated than the download, keep attention of the following.

- Click "Write to Robot" and input the file name.
- When "Start" button of CM-5 is clicked, the motion data upload will begin.
- When upload is completed, close the motion data dialogue box and click the "Mode" button of CM-5.

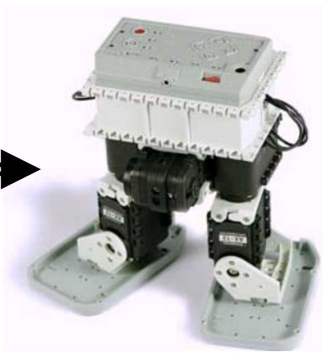
6-2. Motion Editing Using the Robot Terminal

Once you get familiar with the robot you will want to edit motion in the text interface instead of the graphic user interface - being able to see all the information at once can be helpful. In this chapter we will learn how to edit motion using the robot terminal program.

Robot Terminal The Robot Terminal is a program that connects the CM-5 and the PC. The CM-5 does not have a screen or a keyboard, but the Robot Terminal program will allow you to input and output information using your the PC. The information outputted from the CM-5 will go to the PC through the serial cable and then printed on screen through the Robot Terminal. Also, information inputted in the Robot Terminal via the keyboard will be sent to the CM-5 through the serial cable.

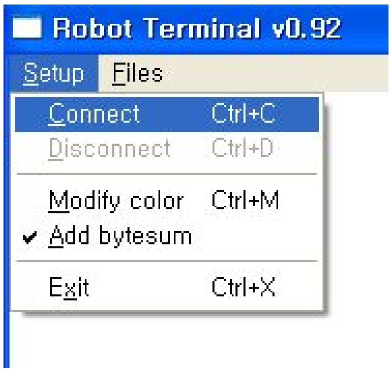


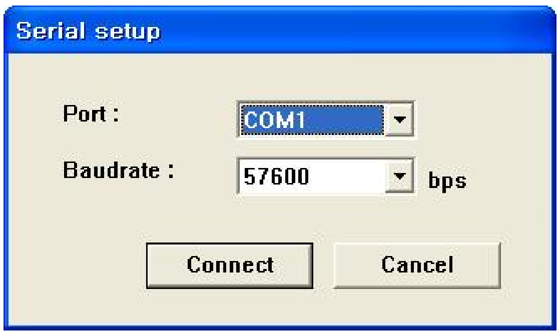
```
Dynamixel  0(0x00)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel  1(0x01)[----]|---- ---- ---- ---- ---- ---- ---- 085 PageNumber:0001
Dynamixel  2(0x02)[----]|---- ---- ---- ---- ---- ---- ---- 085 Addr:0000E200
Dynamixel  3(0x03)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  4(0x04)[----]|---- ---- ---- ---- ---- ---- ---- 085   PlayCount:001
Dynamixel  5(0x05)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  6(0x06)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  7(0x07)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  8(0x08)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  9(0x09)[----]|---- ---- ---- ---- ---- ---- ---- 085   Page Step:000
Dynamixel 10(0x0A)[----]|---- ---- ---- ---- ---- ---- ---- 085 ControlCode:000
Dynamixel 11(0x0B)[----]|---- ---- ---- ---- ---- ---- ---- 085  Page Speed:032
Dynamixel 12(0x0C)[----]|---- ---- ---- ---- ---- ---- ---- 085   DXL Setup:000
Dynamixel 13(0x0D)[----]|---- ---- ---- ---- ---- ---- ---- 085  Accel Time:032
Dynamixel 14(0x0E)[----]|---- ---- ---- ---- ---- ---- ---- 085Link to Next:000
Dynamixel 15(0x0F)[----]|---- ---- ---- ---- ---- ---- ---- 085Link to Exit:000
Dynamixel 16(0x10)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedP1):000
Dynamixel 17(0x11)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedR1):000
Dynamixel 18(0x12)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedP2):000
Dynamixel 19(0x13)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedR2):000
Dynamixel 20(0x14)[0134]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 21(0x15)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 22(0x16)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 23(0x17)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 24(0x18)[----]|---- ---- ---- ---- ---- ---- ---- 085G,W,I,D,M,Set
Dynamixel 25(0x19)[----]|---- ---- ---- ---- ---- ---- ---- 085P,N,B,Copy,Save
Dynamixel 26(0x1a)[----]|---- ---- ---- ---- ---- ---- ---- 085On,Off,New,SP
Dynamixel 27(0x1b)[----]|---- ---- ---- ---- ---- ---- ---- 085Inc,Dec:][,)(
Dynamixel 28(0x1c)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 29(0x1d)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 30(0x1e)[----]|---- ---- ---- ---- ---- ---- ---- 085
   PauseTime       [000]  000  000  000  000  000  000  000
   Speed           [000]  064  064  064  064  064  064  064 [X:00,Y:00,A:0512]
                   POSE7 POS0 POS1 POS2 POS3 POS4 POS5 POS6    Go Step:
] _
Latest file => Tx: Exe₩SensorModule.hex                        COM2-57600
```



Setting the Comport

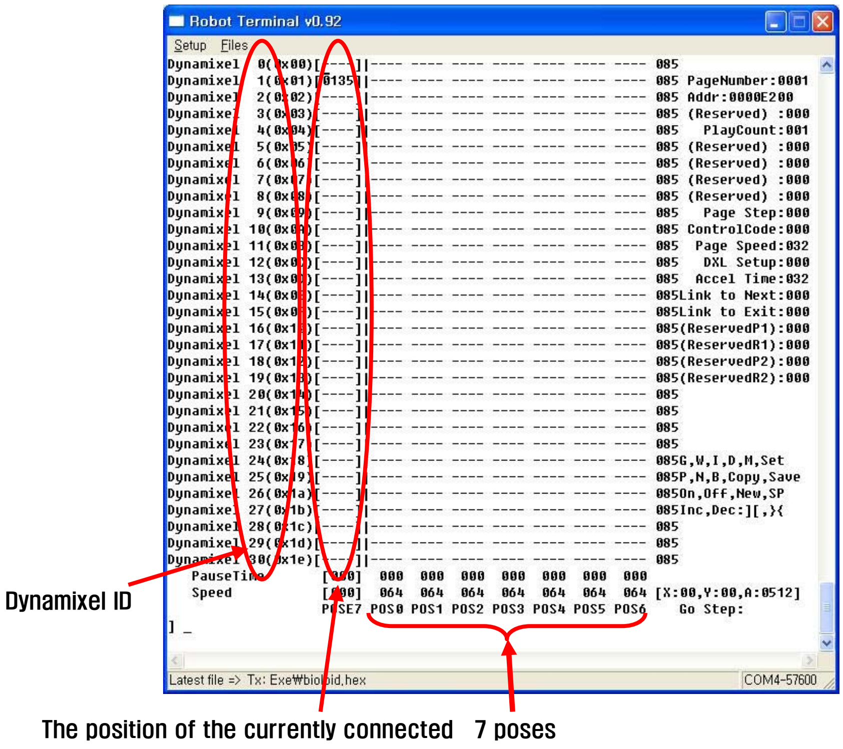
After running the Robot Terminal program, if the connection fails, go to the setup menu and select Connect to set up the Com port, as shown in the figure below. Set it to the appropriate Com port and set the communication speed to 57600 bps. You only have to set the Com port once since this information will be saved inside the program.





Port : COM1
Baudrate : 57600 bps

Apply power to the CM-5 and run the program mode. To do this, go into program mode by pressing the MODE button and then press the START button. The following screen should appear.



```
Dynamixel  0(0x00)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel  1(0x01)[0135]|---- ---- ---- ---- ---- ---- ---- 085 PageNumber:0001
Dynamixel  2(0x02)[----]|---- ---- ---- ---- ---- ---- ---- 085 Addr:0000E200
Dynamixel  3(0x03)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  4(0x04)[----]|---- ---- ---- ---- ---- ---- ---- 085   PlayCount:001
Dynamixel  5(0x05)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  6(0x06)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  7(0x07)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  8(0x08)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  9(0x09)[----]|---- ---- ---- ---- ---- ---- ---- 085   Page Step:000
Dynamixel 10(0x0a)[----]|---- ---- ---- ---- ---- ---- ---- 085 ControlCode:000
Dynamixel 11(0x0b)[----]|---- ---- ---- ---- ---- ---- ---- 085  Page Speed:032
Dynamixel 12(0x0c)[----]|---- ---- ---- ---- ---- ---- ---- 085   DXL Setup:000
Dynamixel 13(0x0d)[----]|---- ---- ---- ---- ---- ---- ---- 085  Accel Time:032
Dynamixel 14(0x0e)[----]|---- ---- ---- ---- ---- ---- ---- 085Link to Next:000
Dynamixel 15(0x0f)[----]|---- ---- ---- ---- ---- ---- ---- 085Link to Exit:000
Dynamixel 16(0x10)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedP1):000
Dynamixel 17(0x11)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedR1):000
Dynamixel 18(0x12)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedP2):000
Dynamixel 19(0x13)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedR2):000
Dynamixel 20(0x14)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 21(0x15)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 22(0x16)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 23(0x17)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 24(0x18)[----]|---- ---- ---- ---- ---- ---- ---- 085G,W,I,D,M,Set
Dynamixel 25(0x19)[----]|---- ---- ---- ---- ---- ---- ---- 085P,N,B,Copy,Save
Dynamixel 26(0x1a)[----]|---- ---- ---- ---- ---- ---- ---- 085On,Off,New,SP
Dynamixel 27(0x1b)[----]|---- ---- ---- ---- ---- ---- ---- 085Inc,Dec:][,}{
Dynamixel 28(0x1c)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 29(0x1d)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 30(0x1e)[----]|---- ---- ---- ---- ---- ---- ---- 085
   PauseTime          [000]  000  000  000  000  000  000  000
   Speed              [000]  064  064  064  064  064  064  064 [X:00,Y:00,A:0512]
                      POSE7 POS0 POS1 POS2 POS3 POS4 POS5 POS6    Go Step:
] _
Latest file => Tx: Exe₩bioloid.hex                                    COM4-57600
```

The first column on the left is the ID of the Dynamixel. The robot profile

information is not shown here. The next column (which we call POSE 7) shows the values of the current angle positions of the Dynamixel units. By the numbers in POSE7 you can see that only one Dynamixel of an ID 1 is connected.
The figure above shows the screen for editing a single motion page. A motion page is made of 7 poses and 64 bytes of page information. The size of one page is 512 bytes.

Pose The pose is an instance of a motion. For example, in the figure below, you will need many poses to make the robot move one side step. A motion connects these poses smoothly.

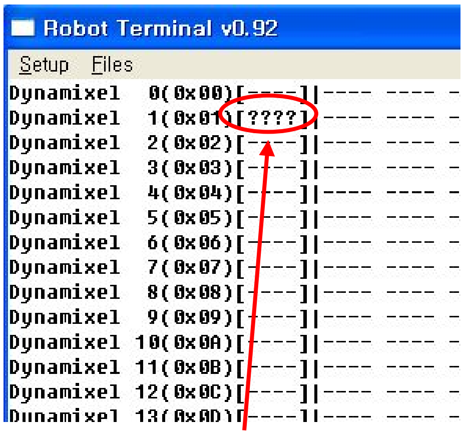
Pose 0
Pose 1
Pose 2
Pose 3
Pose 4

Command The following commands are available.

- Commands related to creating poses: ON, OFF, WRITE, SET, STEP, PLAY, GO, INSERT, MOVE, NAME, SAVE
- Commands related to editing pages: PAGE, BEFORE, NEXT, COPY, NEW

A multi-jointed robot motion can be edited using these commands. Let' s take a look at each one.

Off This will turn off the torque of the joint. If you input OFF, the joint angle value for POSE7 will disappear (see figure on the right). You can now move this joint by hand.



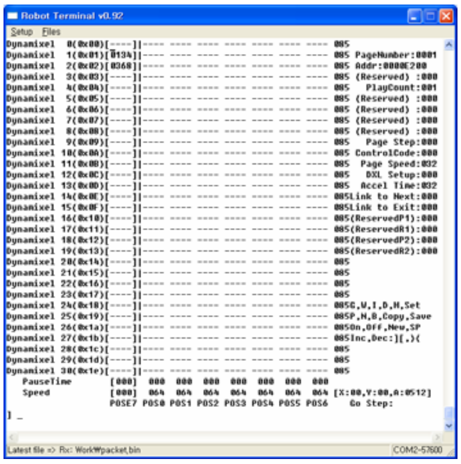
```
Dynamixel  0(0x00)[----]|---- ---- -
Dynamixel  1(0x01)[????]|---- ---- -
Dynamixel  2(0x02)[----]|---- ---- -
Dynamixel  3(0x03)[----]|---- ---- -
Dynamixel  4(0x04)[----]|---- ---- -
Dynamixel  5(0x05)[----]|---- ---- -
Dynamixel  6(0x06)[----]|---- ---- -
Dynamixel  7(0x07)[----]|---- ---- -
Dynamixel  8(0x08)[----]|---- ---- -
Dynamixel  9(0x09)[----]|---- ---- -
Dynamixel 10(0x0A)[----]|---- ---- -
Dynamixel 11(0x0B)[----]|---- ---- -
Dynamixel 12(0x0C)[----]|---- ---- -
Dynamixel 13(0x0D)[----]|---- ---- -
```

Torque Off state with OFF instruction

On After moving the joint to the desired position, type the ON command to see the joint angle value at POSE7. The torque will be back on, locking its joint position.

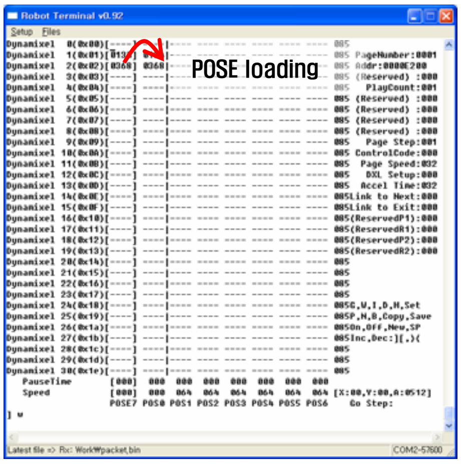
TIP You can list several Dynamixel IDs after an OFF, ON command to turn them on or off all at once.

Write After setting the joint angles to the desired positions, type in the WRITE command. The joint angle values of the POSE7 will be added to the pose. Let's make several more poses this way.



```
Robot Terminal v0.92
Setup  Files
Dynamixel  0(0x00)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel  1(0x01)[0134]|---- ---- ---- ---- ---- ---- ---- 085 PageNumber:0001
Dynamixel  2(0x02)[0368]|---- ---- ---- ---- ---- ---- ---- 085 Addr:0000E200
Dynamixel  3(0x03)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  4(0x04)[----]|---- ---- ---- ---- ---- ---- ---- 085   PlayCount:001
Dynamixel  5(0x05)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  6(0x06)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  7(0x07)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  8(0x08)[----]|---- ---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  9(0x09)[----]|---- ---- ---- ---- ---- ---- ---- 085   Page Step:000
Dynamixel 10(0x0A)[----]|---- ---- ---- ---- ---- ---- ---- 085 ControlCode:000
Dynamixel 11(0x0B)[----]|---- ---- ---- ---- ---- ---- ---- 085  Page Speed:032
Dynamixel 12(0x0C)[----]|---- ---- ---- ---- ---- ---- ---- 085   DXL Setup:000
Dynamixel 13(0x0D)[----]|---- ---- ---- ---- ---- ---- ---- 085  Accel Time:032
Dynamixel 14(0x0E)[----]|---- ---- ---- ---- ---- ---- ---- 085Link to Next:000
Dynamixel 15(0x0F)[----]|---- ---- ---- ---- ---- ---- ---- 085Link to Exit:000
Dynamixel 16(0x10)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedP1):000
Dynamixel 17(0x11)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedR1):000
Dynamixel 18(0x12)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedP2):000
Dynamixel 19(0x13)[----]|---- ---- ---- ---- ---- ---- ---- 085(ReservedR2):000
Dynamixel 20(0x14)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 21(0x15)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 22(0x16)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 23(0x17)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 24(0x18)[----]|---- ---- ---- ---- ---- ---- ---- 085G,W,I,D,H,Set
Dynamixel 25(0x19)[----]|---- ---- ---- ---- ---- ---- ---- 085P,N,B,Copy,Save
Dynamixel 26(0x1a)[----]|---- ---- ---- ---- ---- ---- ---- 085On,OFF,New,SP
Dynamixel 27(0x1b)[----]|---- ---- ---- ---- ---- ---- ---- 085Inc,Dec:][,)(
Dynamixel 28(0x1c)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 29(0x1d)[----]|---- ---- ---- ---- ---- ---- ---- 085
Dynamixel 30(0x1e)[----]|---- ---- ---- ---- ---- ---- ---- 085
   PauseTime       [000]  000  000  000  000  000  000  000
   Speed           [000]  064  064  064  064  064  064  064 [X:00,Y:00,A:0512]
                   POSE7 POS0 POS1 POS2 POS3 POS4 POS5 POS6    Go Step:
] _
Latest file => Rx: Work\packet.bin                                   COM2-57600
```

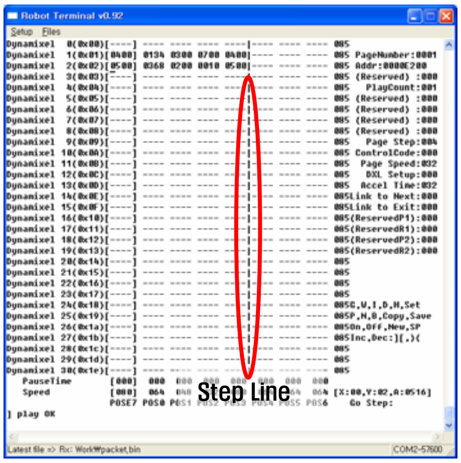
After executing the ON command



```
Robot Terminal v0.92
Setup  Files
Dynamixel  0(0x00)[----]  ----|---- ---- ---- ---- ---- ---- 085
Dynamixel  1(0x01)[0134]  0134|---- ---- ---- ---- ---- ---- 085 PageNumber:0001
Dynamixel  2(0x02)[0368]  0368|---- ---- ---- ---- ---- ---- 085 Addr:0000E200
Dynamixel  3(0x03)[----]  ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  4(0x04)[----]  ----|---- ---- ---- ---- ---- ---- 085   PlayCount:001
Dynamixel  5(0x05)[----]  ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  6(0x06)[----]  ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  7(0x07)[----]  ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  8(0x08)[----]  ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  9(0x09)[----]  ----|---- ---- ---- ---- ---- ---- 085   Page Step:001
Dynamixel 10(0x0A)[----]  ----|---- ---- ---- ---- ---- ---- 085 ControlCode:000
Dynamixel 11(0x0B)[----]  ----|---- ---- ---- ---- ---- ---- 085  Page Speed:032
Dynamixel 12(0x0C)[----]  ----|---- ---- ---- ---- ---- ---- 085   DXL Setup:000
Dynamixel 13(0x0D)[----]  ----|---- ---- ---- ---- ---- ---- 085  Accel Time:032
Dynamixel 14(0x0E)[----]  ----|---- ---- ---- ---- ---- ---- 085Link to Next:000
Dynamixel 15(0x0F)[----]  ----|---- ---- ---- ---- ---- ---- 085Link to Exit:000
Dynamixel 16(0x10)[----]  ----|---- ---- ---- ---- ---- ---- 085(ReservedP1):000
Dynamixel 17(0x11)[----]  ----|---- ---- ---- ---- ---- ---- 085(ReservedR1):000
Dynamixel 18(0x12)[----]  ----|---- ---- ---- ---- ---- ---- 085(ReservedP2):000
Dynamixel 19(0x13)[----]  ----|---- ---- ---- ---- ---- ---- 085(ReservedR2):000
Dynamixel 20(0x14)[----]  ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 21(0x15)[----]  ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 22(0x16)[----]  ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 23(0x17)[----]  ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 24(0x18)[----]  ----|---- ---- ---- ---- ---- ---- 085G,W,I,D,H,Set
Dynamixel 25(0x19)[----]  ----|---- ---- ---- ---- ---- ---- 085P,N,B,Copy,Save
Dynamixel 26(0x1a)[----]  ----|---- ---- ---- ---- ---- ---- 085On,OFF,New,SP
Dynamixel 27(0x1b)[----]  ----|---- ---- ---- ---- ---- ---- 085Inc,Dec:][,)(
Dynamixel 28(0x1c)[----]  ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 29(0x1d)[----]  ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 30(0x1e)[----]  ----|---- ---- ---- ---- ---- ---- 085
   PauseTime       [000]  000  000  000  000  000  000  000
   Speed           [000]  000  064  064  064  064  064  064 [X:00,Y:00,A:0512]
                   POSE7 POS0 POS1 POS2 POS3 POS4 POS5 POS6    Go Step:
] w
Latest file => Rx: Work\packet.bin                                   COM2-57600
```

After executing the WRITE command

Play We will now check the motion that connects the inputted poses. If you type in "play," the motion will be played. The poses from POS0 to the last inputted pose will be played. You can see where the last inputted pose is by the Step line.



```
Robot Terminal v0.92
Setup  Files
Dynamixel  0(0x00)[----] ---- ---- ---- ----|---- ---- ---- 085
Dynamixel  1(0x01)[0400] 0134 0300 0700 0A00|---- ---- ---- 085 PageNumber:0001
Dynamixel  2(0x02)[0500] 0368 0200 0010 0500|---- ---- ---- 085 Addr:0000E200
Dynamixel  3(0x03)[----] ---- ---- ---- ----|---- ---- ---- 085 (Reserved) :000
Dynamixel  4(0x04)[----] ---- ---- ---- ----|---- ---- ---- 085   PlayCount:001
Dynamixel  5(0x05)[----] ---- ---- ---- ----|---- ---- ---- 085 (Reserved) :000
Dynamixel  6(0x06)[----] ---- ---- ---- ----|---- ---- ---- 085 (Reserved) :000
Dynamixel  7(0x07)[----] ---- ---- ---- ----|---- ---- ---- 085 (Reserved) :000
Dynamixel  8(0x08)[----] ---- ---- ---- ----|---- ---- ---- 085 (Reserved) :000
Dynamixel  9(0x09)[----] ---- ---- ---- ----|---- ---- ---- 085   Page Step:004
Dynamixel 10(0x0A)[----] ---- ---- ---- ----|---- ---- ---- 085 ControlCode:000
Dynamixel 11(0x0B)[----] ---- ---- ---- ----|---- ---- ---- 085  Page Speed:032
Dynamixel 12(0x0C)[----] ---- ---- ---- ----|---- ---- ---- 085   DXL Setup:000
Dynamixel 13(0x0D)[----] ---- ---- ---- ----|---- ---- ---- 085  Accel Time:032
Dynamixel 14(0x0E)[----] ---- ---- ---- ----|---- ---- ---- 085Link to Next:000
Dynamixel 15(0x0F)[----] ---- ---- ---- ----|---- ---- ---- 085Link to Exit:000
Dynamixel 16(0x10)[----] ---- ---- ---- ----|---- ---- ---- 085(ReservedP1):000
Dynamixel 17(0x11)[----] ---- ---- ---- ----|---- ---- ---- 085(ReservedR1):000
Dynamixel 18(0x12)[----] ---- ---- ---- ----|---- ---- ---- 085(ReservedP2):000
Dynamixel 19(0x13)[----] ---- ---- ---- ----|---- ---- ---- 085(ReservedR2):000
Dynamixel 20(0x14)[----] ---- ---- ---- ----|---- ---- ---- 085
Dynamixel 21(0x15)[----] ---- ---- ---- ----|---- ---- ---- 085
Dynamixel 22(0x16)[----] ---- ---- ---- ----|---- ---- ---- 085
Dynamixel 23(0x17)[----] ---- ---- ---- ----|---- ---- ---- 085
Dynamixel 24(0x18)[----] ---- ---- ---- ----|---- ---- ---- 085G,W,I,D,H,Set
Dynamixel 25(0x19)[----] ---- ---- ---- ----|---- ---- ---- 085P,N,B,Copy,Save
Dynamixel 26(0x1a)[----] ---- ---- ---- ----|---- ---- ---- 085On,OFF,New,SP
Dynamixel 27(0x1b)[----] ---- ---- ---- ----|---- ---- ---- 085Inc,Dec:][,)(
Dynamixel 28(0x1c)[----] ---- ---- ---- ----|---- ---- ---- 085
Dynamixel 29(0x1d)[----] ---- ---- ---- ----|---- ---- ---- 085
Dynamixel 30(0x1e)[----] ---- ---- ---- ----|---- ---- ---- 085
   PauseTime       [000]  000  000  000  000  000  000  000
   Speed           [000]  064  048                 064  064 [X:00,Y:02,A:0516]
                   POSE7 POS0 POS1 POS2 POS3 POS4 POS5 POS6    Go Step:
] play OK
Latest file => Rx: Work\packet.bin                                   COM2-57600
```

[Playing from Pose 0 to Pose 3]

TIP You can play another motion page by typing in "Play [page number]." For example, "Play 3" will play the motion of page number 3.

Go After playing a motion, sometimes you will want to edit it. Here, you can use the "go" command. This command will take you to a certain pose. For example, "go 1" will make the robot move to the configuration of pose1 and the joint angle values of POSE1 will be copied into POSE7. The Dynamixels will move at a constant speed.

Write [pose number]

After using the GO command and edit the pose data using the OFF, ON commands, you will want to save the new joint values of POSE7. This can be done by typing in

"Write [pose number]." And the joint values for POSE7 will be saved in the specified "pose number.

Insert While editing, you will sometimes want to place the current pose (POSE7) between two other poses. To do this, use the "insert" command. The format is "insert [pose number]."

Delete A pose can be deleted using the "delete" command. Typing in "delete" without a parameter will move the step line up one column. If you want to delete a certain pose then type in "Delete [number]."

Step Sometimes during motion editing you will want to change the location of the Step line. For example, say that you made 4 poses but you want to run only the first two. Typing in "step 2" will move the step line to the beginning of pose 2 and only POSE0 and POSE1 will be played.

Name With this function you can give a name to a page. This will be useful later.

Save During editing, the motion data is stored inside the CM-5' s RAM. When you are finished creating the motion, use the save command to save the motion page to the flash memory.

Page Type in "page [page number]" to jump to another page.

Before Moves to previous page.

Next Moves to the next page.

Note. Make sure to save the motion data before moving to another page since it will be deleted if not saved.

Copy To copy the data of a certain page onto the current page, type in "Copy [page number]." The copied data is not saved in the flash memory yet.

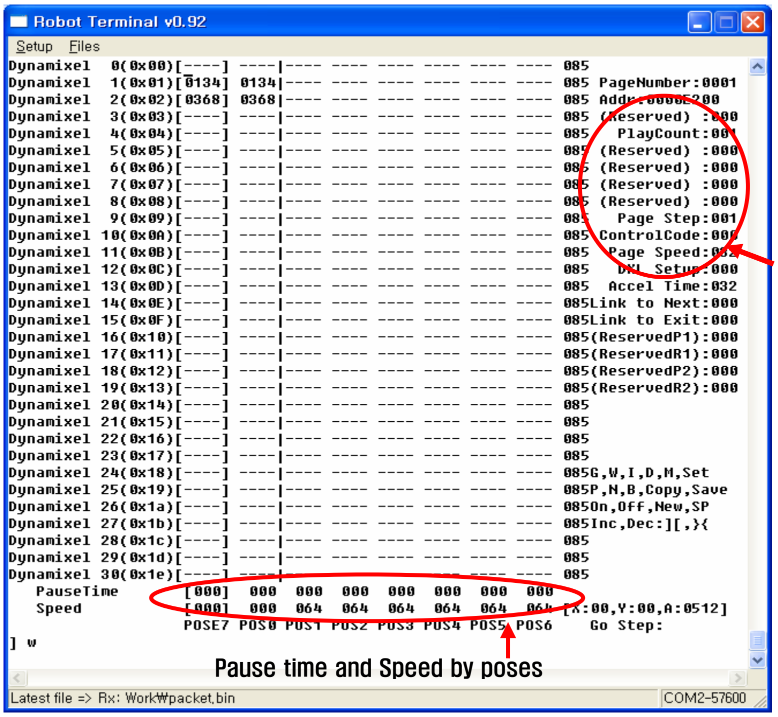
New This command will erase all the information inputted on the current page.

Next we will be learning about editing the page information. The page information is located on the upper right corner of the screen. The following are names of the items and are not commands. They can be edited using the "Set" command.

Speed Even though the joint values have been inputted properly, you might still not be able to get the desired motion unless you set the speed between poses correctly. You can do this by setting the value of the speed at the bottom of the screen. The following shows how this is done.

Move the cursor to the item that you want to set.
Press the "]" or "[" key to increase or decrease the value of the item.
Press the "{ " or "}" key to change the magnitude.
By typing in "Set [value]" the value can be set at once.

The method above can be used to set not only the speed but also all the other items below.


Robot Terminal v0.92
Setup Files
Page information
Pause time and Speed by poses
Latest file => Rx: Work₩packet.bin
COM2-57600

```
Dynamixel  0(0x00)[----] ----|---- ---- ---- ---- ---- ---- 085
Dynamixel  1(0x01)[0134] 0134|---- ---- ---- ---- ---- ---- 085 PageNumber:0001
Dynamixel  2(0x02)[0368] 0368|---- ---- ---- ---- ---- ---- 085 Addr:0000E200
Dynamixel  3(0x03)[----] ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  4(0x04)[----] ----|---- ---- ---- ---- ---- ---- 085   PlayCount:001
Dynamixel  5(0x05)[----] ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  6(0x06)[----] ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  7(0x07)[----] ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  8(0x08)[----] ----|---- ---- ---- ---- ---- ---- 085 (Reserved) :000
Dynamixel  9(0x09)[----] ----|---- ---- ---- ---- ---- ---- 085   Page Step:001
Dynamixel 10(0x0A)[----] ----|---- ---- ---- ---- ---- ---- 085 ControlCode:000
Dynamixel 11(0x0B)[----] ----|---- ---- ---- ---- ---- ---- 085  Page Speed:032
Dynamixel 12(0x0C)[----] ----|---- ---- ---- ---- ---- ---- 085   DXL Setup:000
Dynamixel 13(0x0D)[----] ----|---- ---- ---- ---- ---- ---- 085  Accel Time:032
Dynamixel 14(0x0E)[----] ----|---- ---- ---- ---- ---- ---- 085Link to Next:000
Dynamixel 15(0x0F)[----] ----|---- ---- ---- ---- ---- ---- 085Link to Exit:000
Dynamixel 16(0x10)[----] ----|---- ---- ---- ---- ---- ---- 085(ReservedP1):000
Dynamixel 17(0x11)[----] ----|---- ---- ---- ---- ---- ---- 085(ReservedR1):000
Dynamixel 18(0x12)[----] ----|---- ---- ---- ---- ---- ---- 085(ReservedP2):000
Dynamixel 19(0x13)[----] ----|---- ---- ---- ---- ---- ---- 085(ReservedR2):000
Dynamixel 20(0x14)[----] ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 21(0x15)[----] ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 22(0x16)[----] ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 23(0x17)[----] ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 24(0x18)[----] ----|---- ---- ---- ---- ---- ---- 085G,W,I,D,M,Set
Dynamixel 25(0x19)[----] ----|---- ---- ---- ---- ---- ---- 085P,N,B,Copy,Save
Dynamixel 26(0x1a)[----] ----|---- ---- ---- ---- ---- ---- 085On,Off,New,SP
Dynamixel 27(0x1b)[----] ----|---- ---- ---- ---- ---- ---- 085Inc,Dec:][,}{
Dynamixel 28(0x1c)[----] ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 29(0x1d)[----] ----|---- ---- ---- ---- ---- ---- 085
Dynamixel 30(0x1e)[----] ----|---- ---- ---- ---- ---- ---- 085
   PauseTime          [000]  000  000  000  000  000  000  000
   Speed              [000]  000  064  064  064  064  064  064 [X:00,Y:00,A:0512]
                      POSE7 POS0 POS1 POS2 POS3 POS4 POS5 POS6    Go Step:
] w
```

PauseTime Sometimes you will want to pause for a short time after a pose. For example, you might want the robot to pause for a while after it has finished doing a bowing motion. If you give it a "pause time" value, it will pause for 7.8 msec per the set value and then move on to the next pose. The figure above shows an example of the robot pausing for 0.5 s (40*7.8 msec) after POSE1 is played.

Accel. Time Every time a motion is played, the joints go through a process of acceleration constant velocity deceleration. The acceleration time is the time of acceleration and deceleration (ramp up time plus ramp down time). Reducing the acceleration time will make the motors acceleration faster and stress on the joints. Increasing the acceleration time might create an interval which makes it impossible to complete a motion.

Page Speed Use this when you want to adjust the play speed of the whole page. The default value is 32. Setting the value to 64 would be the same as doubling the speed of each of the poses individually.

Link to Next You can set the next motion page to be executed after the current page is finished being played. This can be done by inputting the page to be played next in the in the “Link to Next” item. If you don’ t need to play another motion page then set the value to 0 (default value is 0). If you set the value for “Link to Next” as the number of the current page, then playback will continue infinitely in a loop. In this case, you can input the key stop the playback. Playback will be stopped after the Link to Exit motion is completed.

Link to Exit During motion play, if a signal is received, the playback will end after the executing the page defined by this. If this feature is not needed, then set its value to 0.

Other Things to Keep in Mind

Page 0, 1 Page 0 and page 1 have special functions so it is recommended that you do not use them.

“Are you sure?”
The “Are you sure?” message will appear in the following cases.

- When you move to another page without saving.
- When you use the “go” command with a pose that is outside of the step line.
- When you use the “new” command.

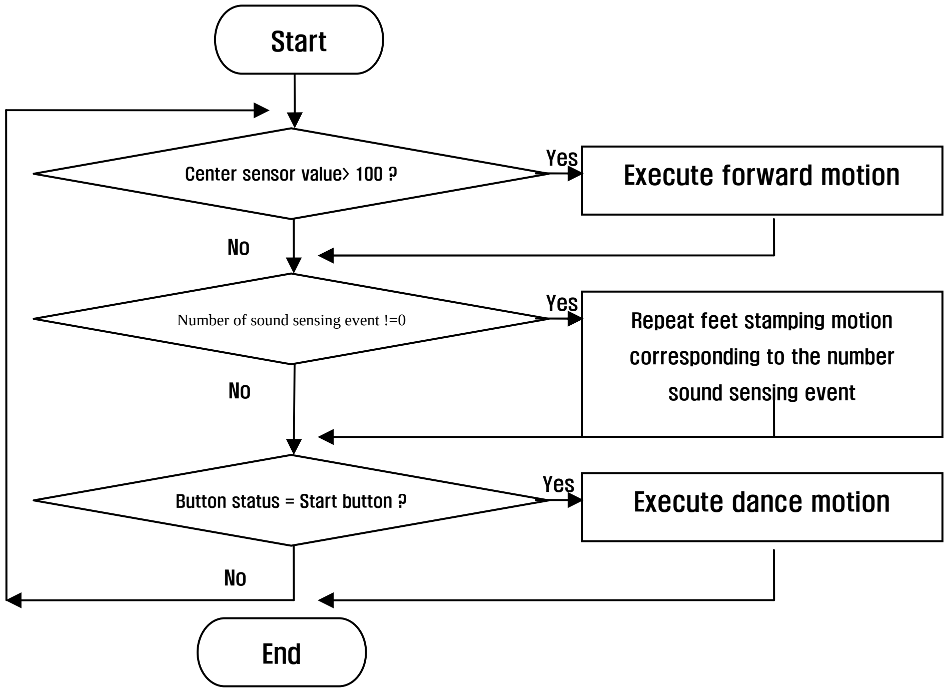
Abbreviation You can use only the first letter of the commands that are used often.

6-3. Walking Droid ' s Program

Now let' s build a walking droid as follows.

- When an object is sensed by center sensor, the robot will move forward.
- When handclapped, walking droid will stamp its feet corresponding to a number of handclaps.
- When "Start" button is clicked, the robot will dance.

Let' s look at the flow chart below.

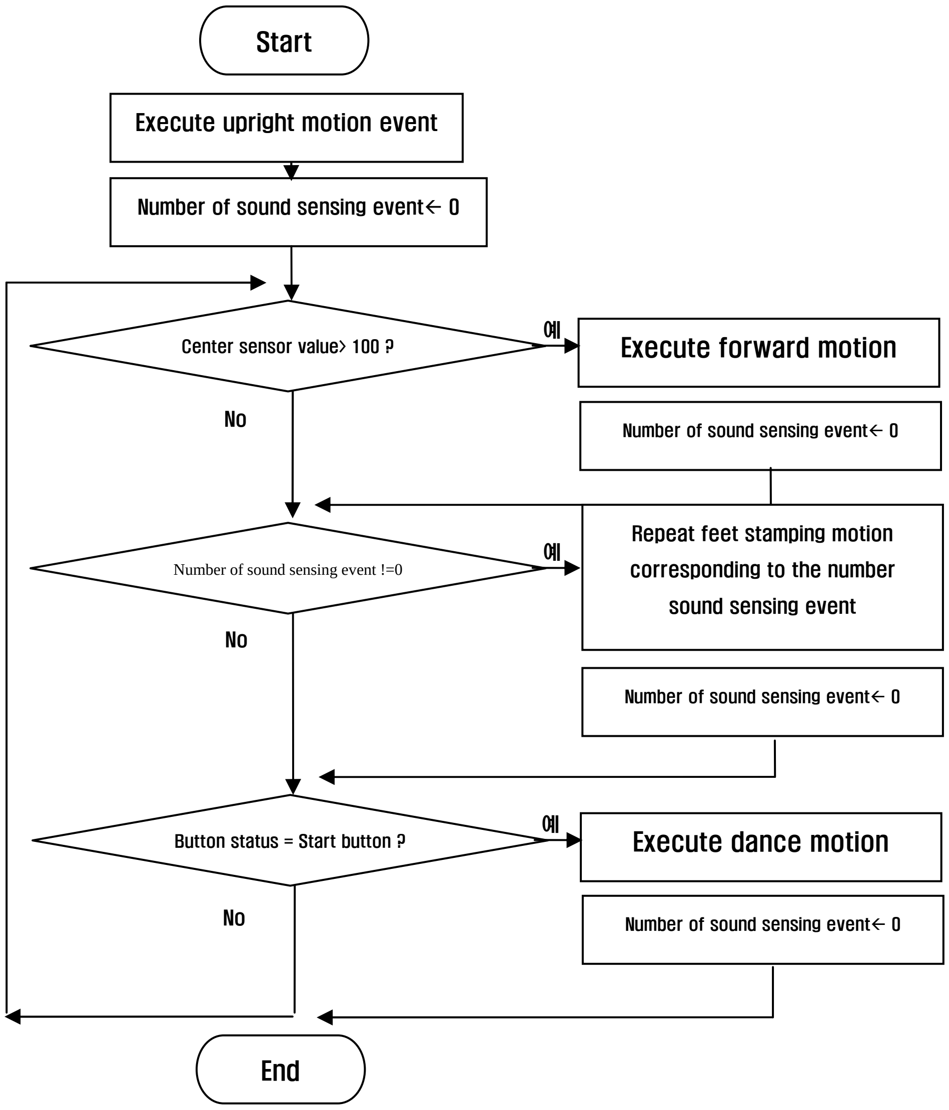

Start
Center sensor value> 100 ?
Yes
Execute forward motion
No
Number of sound sensing event !=0
Yes
Repeat feet stamping motion corresponding to the number sound sensing event
No
Button status = Start button ?
Yes
Execute dance motion
No
End

At first behavior control program does not seem to have any problems with it. But there will be several problems when you run it as is.

The first problem is the position configuration of the humanoid robot when the program is started. The robot could be in a pose which will make it difficult to execute a certain motion. Thus it needs a motion to make the humanoid robot stand straight upright.

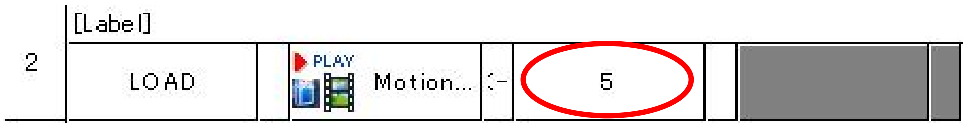
The second problem is the number of times the sound is sensed. The starting value for the number of sound sensing event might not be set to zero. Also, the robot could sense its own clapping sound as a clap made by the user. If the number of sound sensing is not set to 0, it will clap endlessly unless there is a part of the code that will set this value to 0.
The third problem is that when making the robot execute a motion, it will be almost impossible to set each joint value individually to create the many motions required.

A flow chart to solve the first and second problems looks like the following.


Start
Execute upright motion event
Number of sound sensing event← 0
Center sensor value> 100 ?
예
Execute forward motion
Number of sound sensing event← 0
No
Number of sound sensing event !=0
예
Repeat feet stamping motion corresponding to the number sound sensing event
No
Number of sound sensing event← 0
Button status = Start button ?
예
Execute dance motion
No
Number of sound sensing event← 0
End

Motion Execution

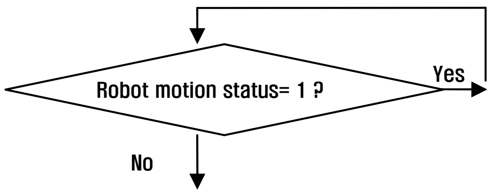
To use the motion data that was created with the motion editor, it is necessary to use the motion execution function. The motion execution involves inputting a motion page number of user' s choice in the "Robot Motion" item of CM-5. When the page numbers that will be used in the motion data are managed separately, it will be very useful when you want to create behavior control program.

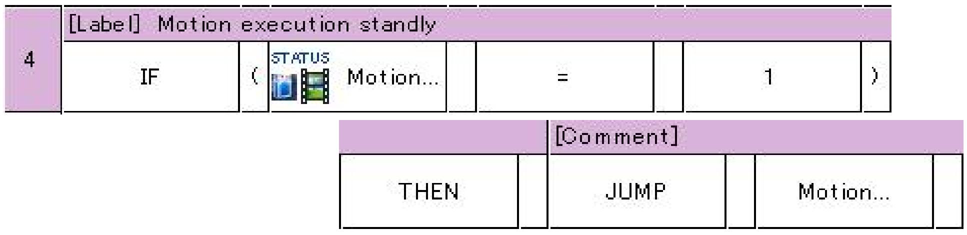


	[Label]				
2	LOAD	PLAY Motion...	<-	5	

Motion Execution Standby

There will be a time when you may want to stop behavior control program while the particular motion is running. For that, you first need to know whether the motion execution is completed. By checking the "Robot Motion Status," you can find out. If the "Robot Motion Status" is set to 0, it means that the motion execution is completed, and if not, it will be 1.

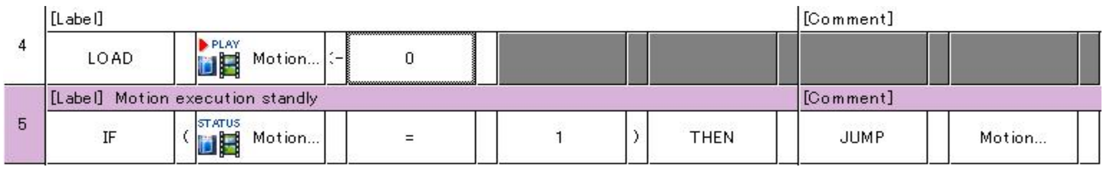




	[Label] Motion execution standly					
4	IF	(	STATUS Motion...	=	1	)
			THEN	[Comment] JUMP	Motion...	

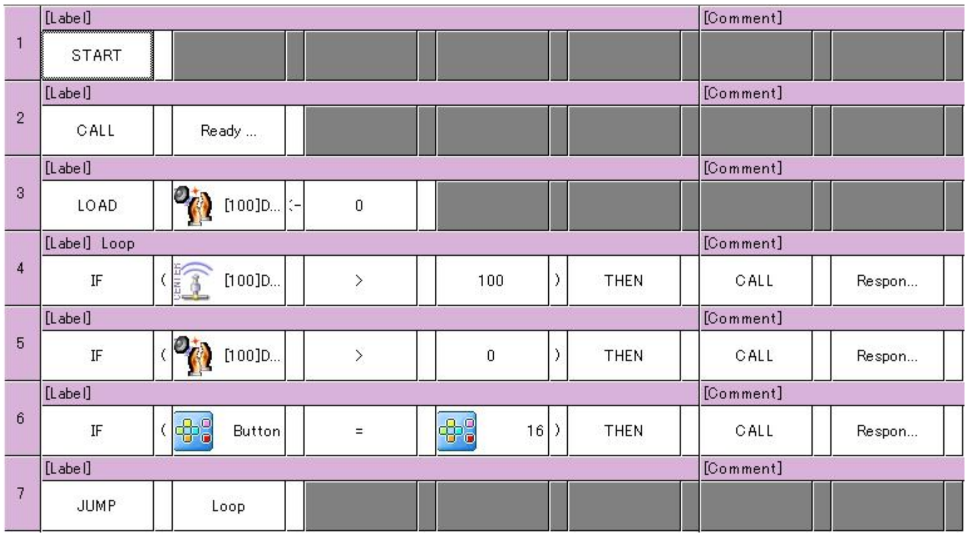
Motion Execution Stop

There is a function that will allow forceful stop without waiting for the motion to complete. If you put input the value of 0 in "robot motion" item, the robot will respond very fast to external events. However, keep in mind that even if you use motion execution stop function, you will have to wait for all current motion pages and designated final pages to be completed. Therefore, in order to completely end the motion, you have to use motion execution standby.



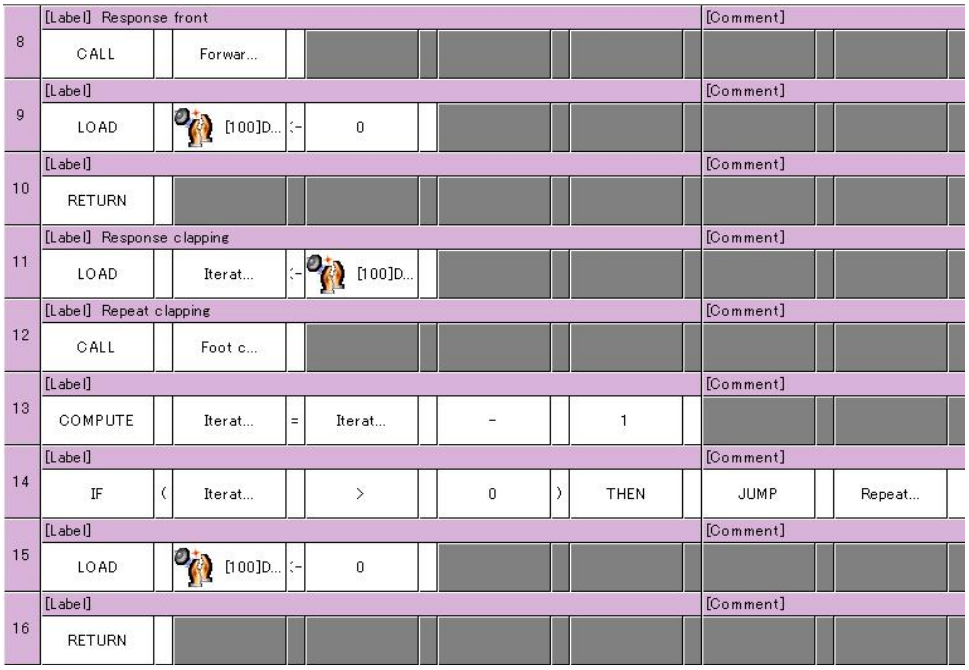
	[Label]						[Comment]	
4	LOAD	PLAY Motion...	<-	0				
	[Label] Motion execution standly						[Comment]	
5	IF	(STATUS Motion...	=	1	)	THEN	JUMP	Motion...

To create above program, follow the following steps.
Refer to the "Examples\Example(Walking Droid).bpg" inside the CD.
Main routine



	[Label]									[Comment]
1		START								
2		CALL		Ready ...						
3		LOAD		[100]D...	:-	0				
4	Loop	IF	(	[100]D...	>	100	)	THEN	CALL	Respon...
5		IF	(	[100]D...	>	0	)	THEN	CALL	Respon...
6		IF	(	Button	=	16	)	THEN	CALL	Respon...
7		JUMP		Loop						

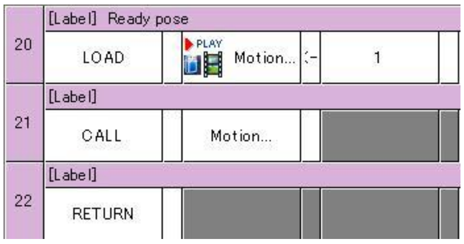
Button reaction routine for the center reaction, handclap reaction to the sensors



	[Label]									[Comment]
8	Response front	CALL		Forwar...						
9		LOAD		[100]D...	:-	0				
10		RETURN								
11	Response clapping	LOAD		Iterat...	:-	[100]D...				
12	Repeat clapping	CALL		Foot c...						
13		COMPUTE		Iterat...	=	Iterat...	-	1		
14		IF	(	Iterat...	>	0	)	THEN	JUMP	Repeat...
15		LOAD		[100]D...	:-	0				
16		RETURN								

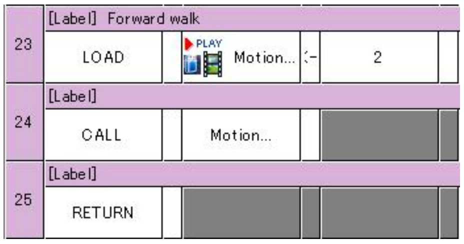
Motion execution routine

[Upright]



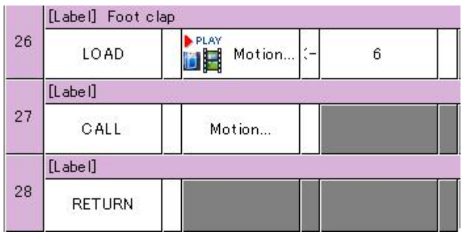
20	[Label] Ready pose			
	LOAD	PLAY Motion...	<-	1
21	[Label]			
	CALL	Motion...		
22	[Label]			
	RETURN			

[Forward]



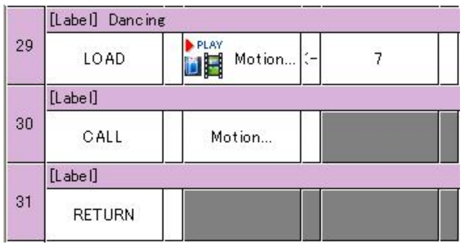
23	[Label] Forward walk			
	LOAD	PLAY Motion...	<-	2
24	[Label]			
	CALL	Motion...		
25	[Label]			
	RETURN			

[Foot stamping]



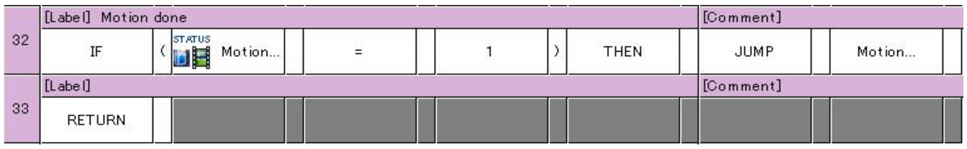
26	[Label] Foot clap			
	LOAD	PLAY Motion...	<-	6
27	[Label]			
	CALL	Motion...		
28	[Label]			
	RETURN			

[Dancing]



29	[Label] Dancing			
	LOAD	PLAY Motion...	<-	7
30	[Label]			
	CALL	Motion...		
31	[Label]			
	RETURN			

Motion completion standby routine

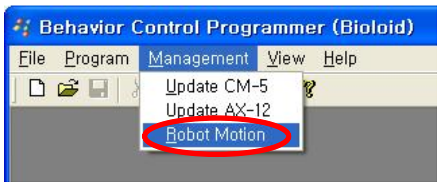


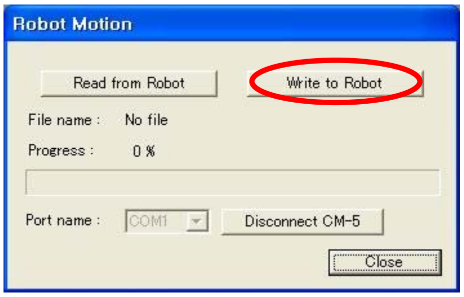
32	[Label] Motion done							[Comment]		
	IF	(	STATUS Motion...	=	1	)	THEN	JUMP	Motion...	
33	[Label]							[Comment]		
	RETURN									

Now it is time to generate the required motions.
If you look at the source above, you can see that the following motion is needed.

- Motion page 1: Stand straight up motion
- Motion page 2: Forward motion
- Motion page 6: Foot stamping motion
- Motion page 7: Dancing motion

The Bioloid has motion pages from 1 to 127. When you create the desired motion onto the page and call this page from the behavior control program, the motion will be played. We are going to learn how to create a motion in the next chapter, but first download the provided motion onto the robot and let' s check if the program works. To download, go to the menu bar of the behavior control program and select maintenance and then select motion data download. Here, open and download the provided "Examples\Example(Walking Droid).mtn" data. The figure below shows how to download a motion.





After download is complete, execute the play mode by clicking a button.
Go ahead and test by handclapping and putting a hand closer to the robot' s waist. Click once again the start button.

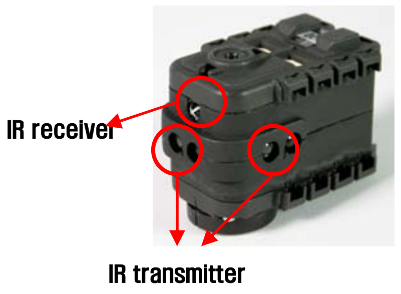
7. Building a Wireless Remote Control

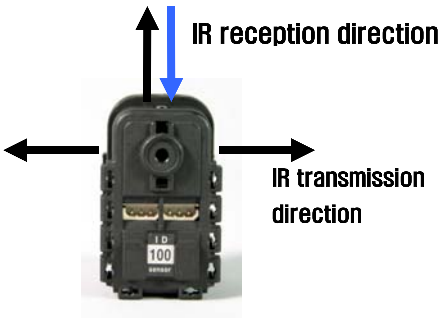
Bioloid supports two types of wireless communication. With these methods, user can control the robots remotely or allow the Bioloids to send and receive the data between each others.
The first method involves sending data using IR (infrared rays) transmitter-receiver function of AX-S1 and by attaching the Zigbee module ZIG-100, dedicated Bioloid wireless device, to CM-5. With this attachment, Bioloid can communicate via RF method.

7-1. Infrared Communication Program Using the AX-S1

IR Communication

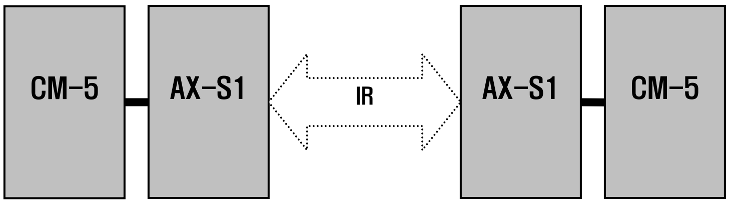
In AX-S1, there is a transmitter-receiver built-in that allows IR communication. Although IR communication is often used for short distance, as it is strongly influenced by the direction and the location of its devices, users have to keep in mind of above limitation when transmitting. As the images below show, IR can send data in three directions, only one direction is allowed in receiving data.





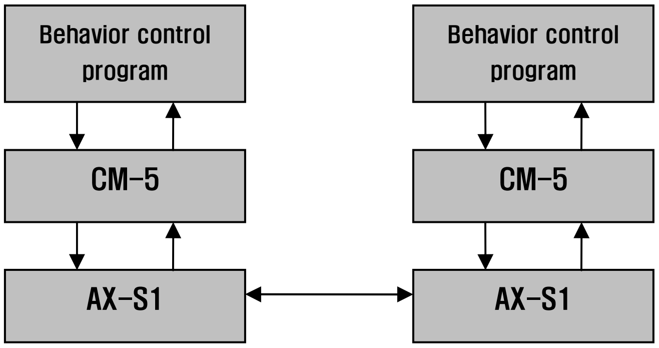
IR Communication H/W

It is the sensor module AX-S1, rather than the CM-5, central control device of Bioloid, that actually handles the communication. To communicate via infrared, you need at least two AX-S1s and two CM-5s.



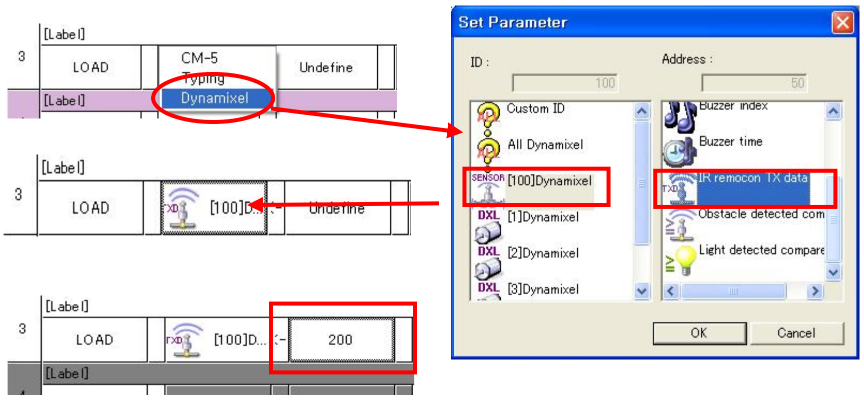
IR Communication S/W

The result of communication between AX-S1 can be checked through behavior control program of CM-5. That is, to control the Bioloid via IR transmission, you need behavior control program that communicates between each other using AX-S1.



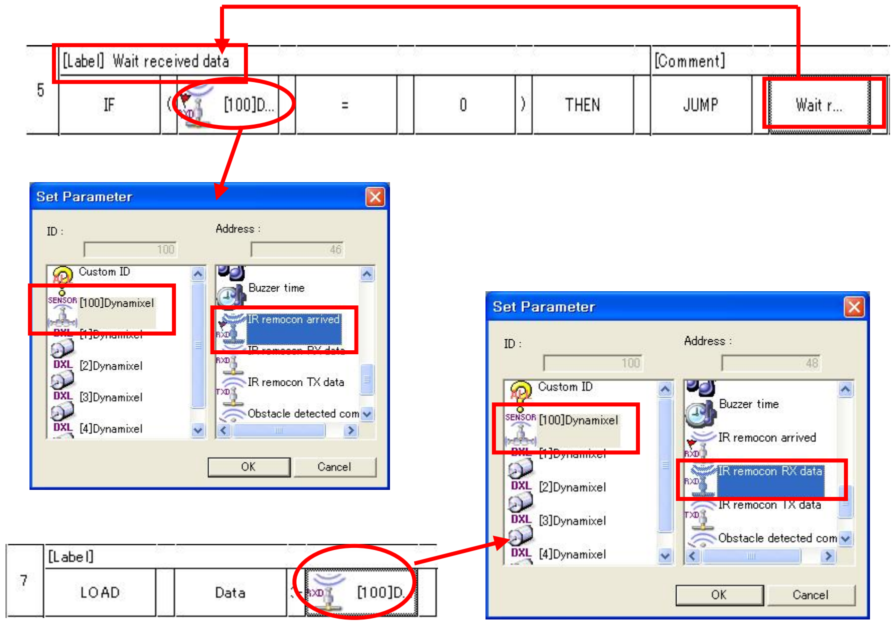
Data Transmission

As previously mentioned, AX-S1 is the one that actually handles the communication. Following that, you have to load the data to AX-S1 in order to send the data. The values that can be sent are between 0 and 65535. To send the data, in behavior control program, load the "to be sent remote control data" address of AX-S1 of ID 100. Upon doing so, AX-S1 will immediately send the data.



Data Reception

Data reception is much more complicated than the data transmission. Simple reason being that although data send is determined by the user, no one can predict when there will be data transmission from outside source. Accordingly, reception standby routine is required. With the proper use of flag available in AX-S1, user can create reception standby routine. Here, if the value is 0, it means that data did not arrive, whereas, if the value is other than 0, it indicates that data has arrived. In behavior control program, Check the arriving new date flag .if it is not going to be change to o should read value of remote control data address what you received.


[Label] Wait received data [Comment]

5 IF ([100]D... = 0) THEN JUMP Wait r...

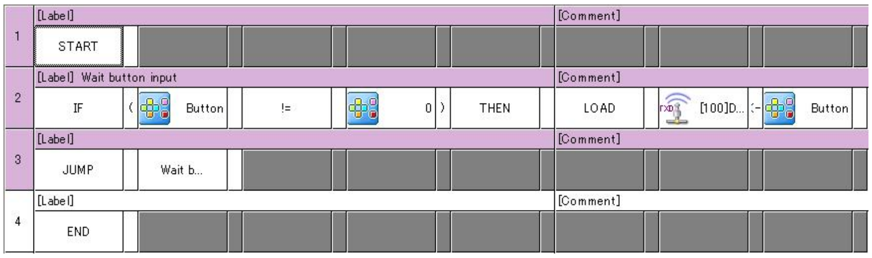
Set Parameter — ID : 100, Address : 46 — [100]Dynamixel, IR remocon arrived

7 [Label] LOAD Data [100]D.

Set Parameter — ID : 100, Address : 48 — [100]Dynamixel, IR remocon RX data

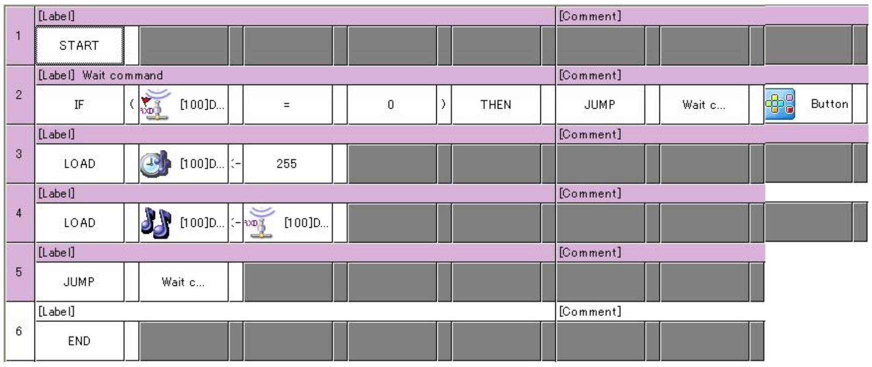
Example Let's take a look at the simple example of AX-S1 transmitter-receiver function. As below picture shows, it is simply composed of AX-S1 and CM-5. In transmitter, if the button is pressed, it will send the data via IR, whereas in receiver, when it receives the data, it will play the melody. Refer to the "Examples\Example(IR receiver).bpg" and "Examples\Example(IR transmitter).bpg" inside the CD.

[Transmitter behavior control program]



	[Label]								[Comment]			
1	START											
	[Label] Wait button input								[Comment]			
2	IF	(	Button	!=	0	)	THEN		LOAD	TXD [100]D...	<-	Button
	[Label]								[Comment]			
3	JUMP		Wait b...									
	[Label]								[Comment]			
4	END											

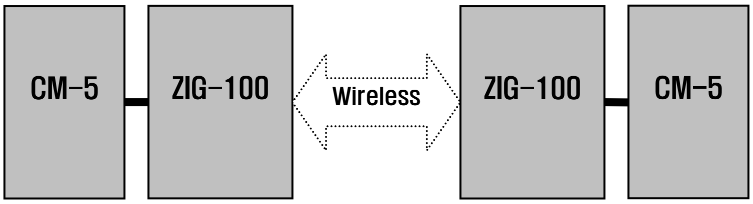
[Receiver behavior control program]

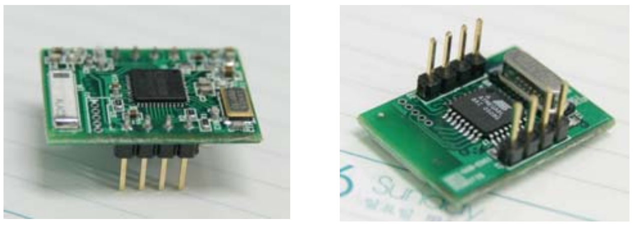


	[Label]								[Comment]		
1	START										
	[Label] Wait command								[Comment]		
2	IF	(	RXD [100]D...	=	0	)	THEN		JUMP	Wait c...	Button
	[Label]								[Comment]		
3	LOAD		[100]D...	<-	255						
	[Label]								[Comment]		
4	LOAD		[100]D...	<-	RXD [100]D...						
	[Label]								[Comment]		
5	JUMP		Wait c...								
	[Label]								[Comment]		
6	END										

7-2. Assembling RF Remote Control Using ZIG-100

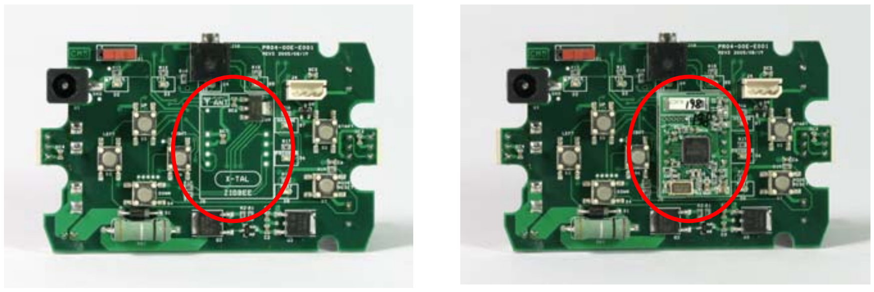
Zigbee Module Zigbee, just like Bluetooth, is a frequently used PAN(Personal Area Network) communication technology. Zigbee module ZIG-100 is a communication module of Bioloid and as such, with its communication technology, it enables Bioloid to transmit the data and control the robots in various ways.





CM-5 and ZIG-100

By default CM-5, the central control device of Bioloid, does not have ZIG-100. Thus, for the IR wireless communication between Bioloid, you need to have ZIG-100 attach to the CM-5. In order to attach it, you need to dissemble CM-5 and solder the ZIG-100 on Zigbee circuit board, as shown on below pictures. You need at least two sets of CM-5s and ZIG-100 modules to send and receive data.



Zigbee ID

ZIG-100 modules each have their own unique IDs. Following that, in order to communicate between each others, they need to know the IDs of respective devices. In general, with known IDs, devices can communicate one to one and additionally, you can send broadcasting messages to all ZIG-100s. For further details, you can check out the ZIG-100 manual.

You can change the communication mode setup through the behavior control program.

CM-5 Setup

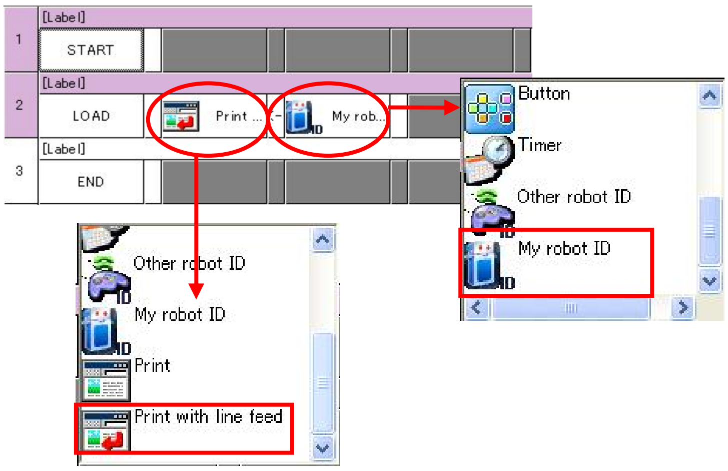
To setup the IDs of other ZIG-100 from ZIG-100, you have to use behavior control program. Both CM-5s must have ZIG-100 built-in and must know the unique IDs of each others.

Additionally, both must set and save the IDs of each others through behavior control program before communicating.

For example, let' suppose that robot A has ID of 120 and robot B has ID of 121. For robot A to send data to robot B, the robot A has to set and save robot B's ID of 121. Likewise, the robot B has to set and save the ID of robot A, which is 120.

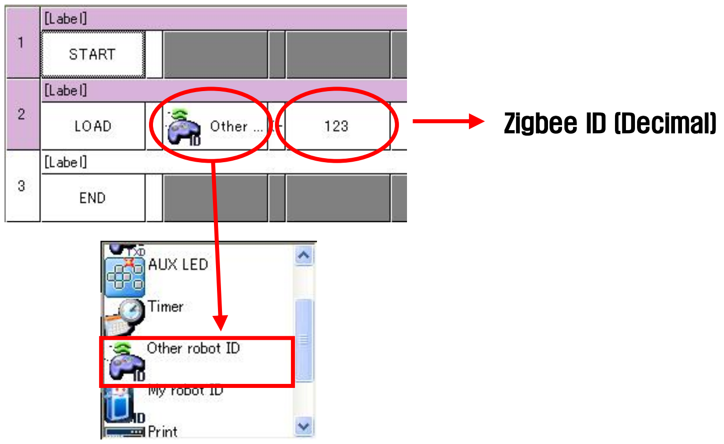
To find own ID and set the ID of others, follow the instruction below. Also refer to "Examples\Example(Read my RF ID).bpg" and "Examples\Example[Set other RF ID].bpg" inside the CD.

[Finding own Zigbee's ID through the behavior control program]



Own ID cannot be changed as it is read from the ROM of ZIG-100.

[Setting up other Zigbee's ID through the behavior control program]



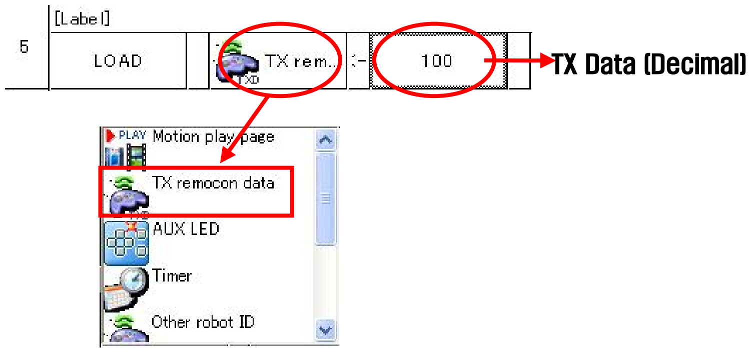
Once other Zigbee's ID set, it is maintained even after power is turned off. As other ID is used for simple communication, the ID can be changed accordingly by users.

Transmitting through the behavior control program

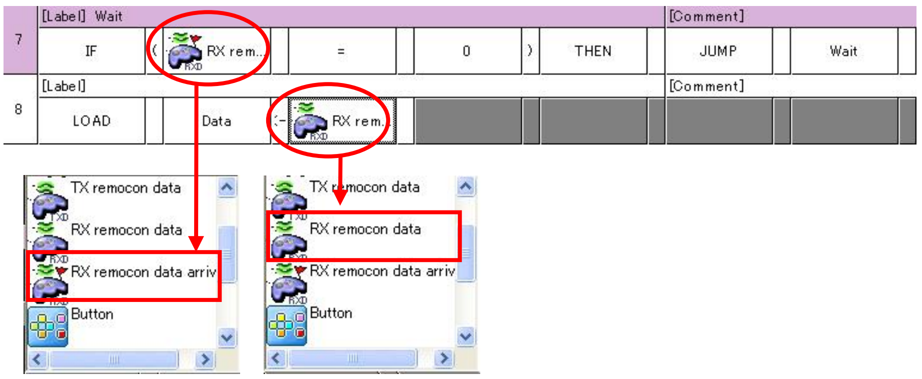
When you are sending data through the behavior control program, you write the value (0~65535) in "to be sent data" address. When receiving data, similar to AX-S1, you first check the "new wireless data arrival" address. If it changes to value other than 0, it means that new data has arrived and you can read "received wireless data."

Take note however that wireless communication by itself means simple transmission of data between 0 and 65535. Thus, if you want Bioloid to receive data and to behave accordingly, you must create behavior control program and set the protocol.

[Sending data]



[Receiving data]



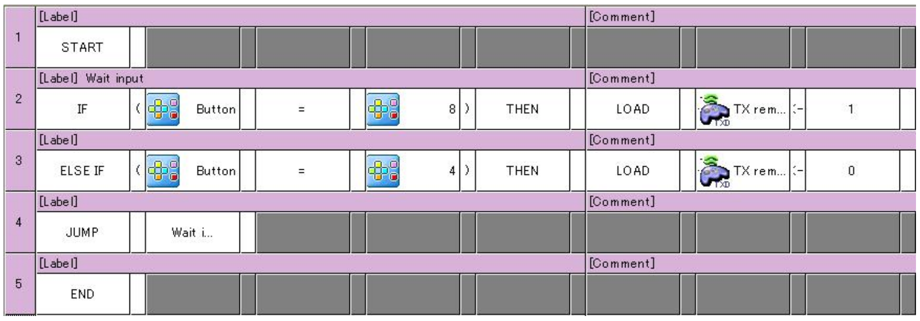
7	[Label] Wait							[Comment]		
	IF	(	RX rem...	=	0	)	THEN	JUMP		Wait
8	[Label]							[Comment]		
	LOAD		Data	<-	RX rem.					

< New wireless data flag >
-> 0: No arrived data
-> 1: Data arrived

Example

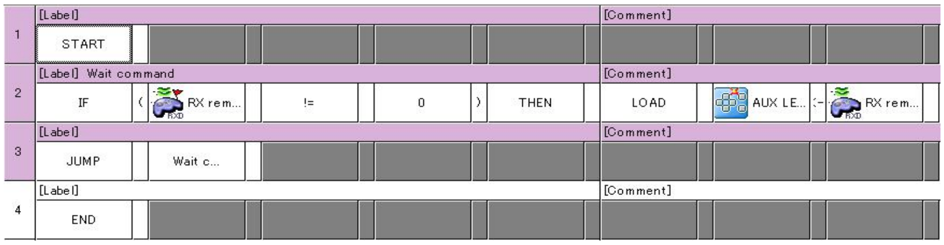
Let’ s expand what we have learned before when we created program that turn on and off AUX LED by pressing a button. Here, let’ s create a program that can control AUX LED by wireless. Refer to “Examples\Example(IR receiver).bpg” and “Examples\Example(IR transmitter).bpg” inside the CD.

[Sending data]



1	[Label]							[Comment]			
	START										
2	[Label] Wait input							[Comment]			
	IF	(	Button	=	8	)	THEN	LOAD	TX rem...	<-	1
3	[Label]							[Comment]			
	ELSE IF	(	Button	=	4	)	THEN	LOAD	TX rem...	<-	0
4	[Label]							[Comment]			
	JUMP		Wait i...								
5	[Label]							[Comment]			
	END										

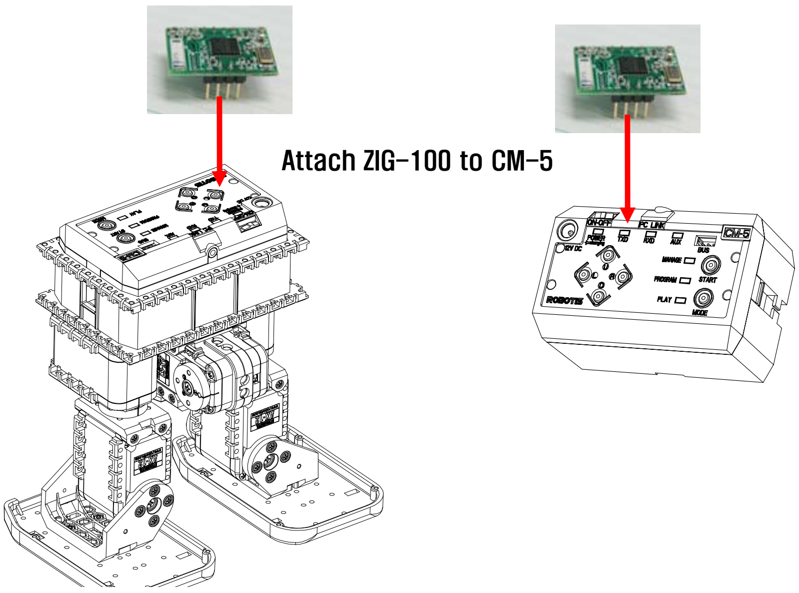
[Receiving data]



1	[Label]							[Comment]			
	START										
2	[Label] Wait command							[Comment]			
	IF	(	RX rem...	!=	0	)	THEN	LOAD	AUX LE...	<-	RX rem...
3	[Label]							[Comment]			
	JUMP		Wait c...								
4	[Label]							[Comment]			
	END										

7-3. Walking Droid Program Controlled by RF Wireless Remote Control

Let' s apply what we have learned so far in wireless communication to making a remote control to control the robot remotely.
Refer to "2-2-14 Walking Droid" and build hardware. Additionally, prepare by purchasing the pair of ZIG-100(not included) and additional CM-5.

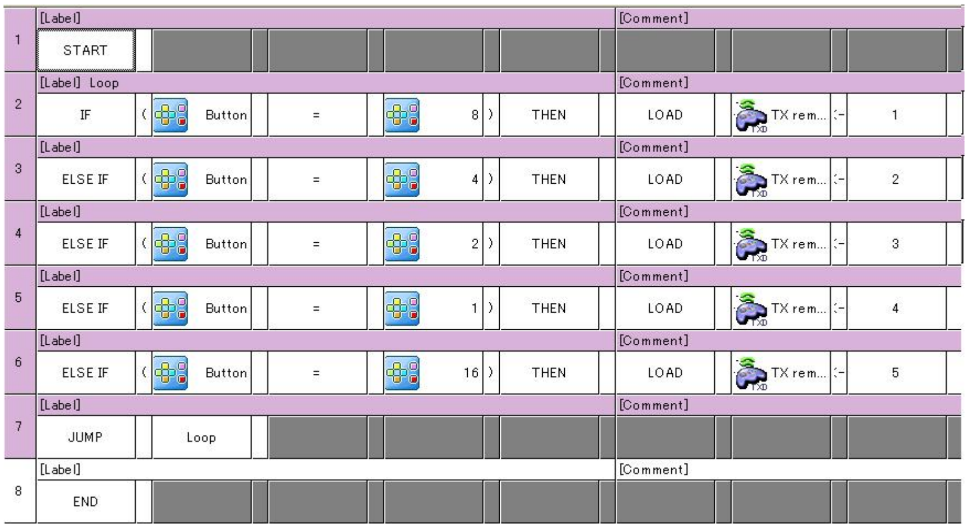


Before creating behavior control program, we have to set the IDs of ZIG-100s accordingly. By referring to "Examples\Example[Read my RF ID].bpg" and "Examples\Example[Set other RF ID].bpg" prepare ID setup beforehand. The CM-5 of robot that receives message and CM-5 that will be used as a remote control will each execute behavior control program.

Transmitter CM-5 behavior control program

- When the U button of CM-5 is pressed, it will send 1.
- When the D button of CM-5 is pressed, it will send 2.
- When the L button of CM-5 is pressed, it will send 3.
- When the R button of CM-5 is pressed, it will send 4.
- When the Start button of CM-5 is pressed, it will send 5.

The actual behavior control program is shown below. Refer to the "Examples\Example(RF remocon of Walking Droid).bpg" inside the CD.

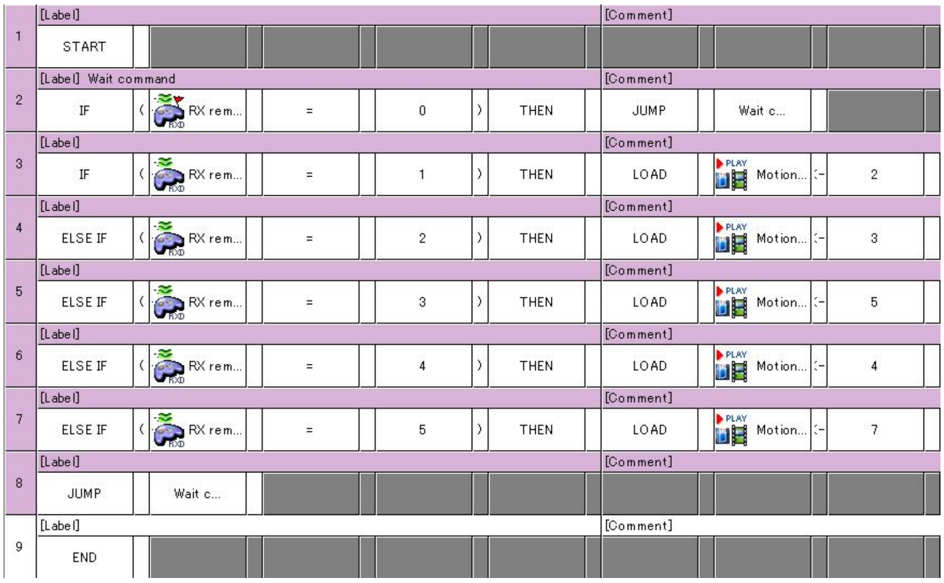


1	[Label] START						[Comment]			
2	[Label] Loop IF	(	Button	=	8	)	THEN	LOAD	TX rem...	<- 1
3	[Label] ELSE IF	(	Button	=	4	)	THEN	LOAD	TX rem...	<- 2
4	[Label] ELSE IF	(	Button	=	2	)	THEN	LOAD	TX rem...	<- 3
5	[Label] ELSE IF	(	Button	=	1	)	THEN	LOAD	TX rem...	<- 4
6	[Label] ELSE IF	(	Button	=	16	)	THEN	LOAD	TX rem...	<- 5
7	[Label] JUMP	Loop					[Comment]			
8	[Label] END						[Comment]			

Receiver CM-5 behavior control program

- When received wireless data is 1, it will take forward motion.
- When received wireless data is 2, it will take backward motion.
- When received wireless data is 3, it will take left turn motion.
- When received wireless data is 4, it will take right turn motion.
- When received wireless data is 5, it will take dancing motion.

The actual behavior control program is shown below. Refer to the “ “Examples\Example(RF control of Walking Droid).bpg” inside the CD. (Before running, download Examples\Example(Walking Droid).mtn, motion data from the CD.



	[Label]							[Comment]		
1	START									
	[Label] Wait command							[Comment]		
2	IF	(	RX rem...	=	0	)	THEN	JUMP	Wait c...	
	[Label]							[Comment]		
3	IF	(	RX rem...	=	1	)	THEN	LOAD	PLAY Motion...	<- 2
	[Label]							[Comment]		
4	ELSE IF	(	RX rem...	=	2	)	THEN	LOAD	PLAY Motion...	<- 3
	[Label]							[Comment]		
5	ELSE IF	(	RX rem...	=	3	)	THEN	LOAD	PLAY Motion...	<- 5
	[Label]							[Comment]		
6	ELSE IF	(	RX rem...	=	4	)	THEN	LOAD	PLAY Motion...	<- 4
	[Label]							[Comment]		
7	ELSE IF	(	RX rem...	=	5	)	THEN	LOAD	PLAY Motion...	<- 7
	[Label]							[Comment]		
8	JUMP	Wait c...								
	[Label]							[Comment]		
9	END									

If the program did not run properly, check the Zigbee ID. Also, for the “Examples\Example(RF control of Walking Droid).bpg,” it needs “Examples\Example(Walking Droid).mtn” motion data, so make sure that it has been downloaded.

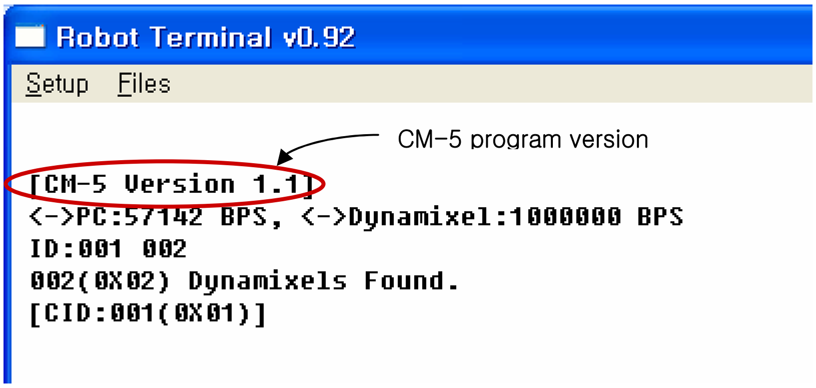
8. Management Mode

This chapter explains about using Manage Mode. In "manage mode" you can check the robot status and check or change the Dynamixel settings.

8-1. SETTING THE ID AND DYNAMIXEL SEARCH

Robot Terminal Run the Robot Terminal program. In the previous chapter, we have explained how the CM-5 and the PC is connected. As we have explained comport setup in detail in previous chapter, we will omit here.

Initial State When you execute the "manage mode" the following screen will show up in the "Robot Terminal" .



```
[CM-5 Version 1.1]
<->PC:57142 BPS, <->Dynamixel:1000000 BPS
ID:001 002
002(0X02) Dynamixels Found.
[CID:001(0X01)]
```

Here you can see the number and the IDs of the connected Dynamixels. If what appears on the screen is different from the actual configuration, check the following.

- Do all the Dynamixel have different IDs?
- Does the communication speeds between the Dynamixels and CM-5 agree?
- Are the cables connected properly and securely?

To check the wiring, turn the power off the CM-5 and then turn it back on. Check if the LEDs on the Dynamixels are blinking. If not, check the wiring again.

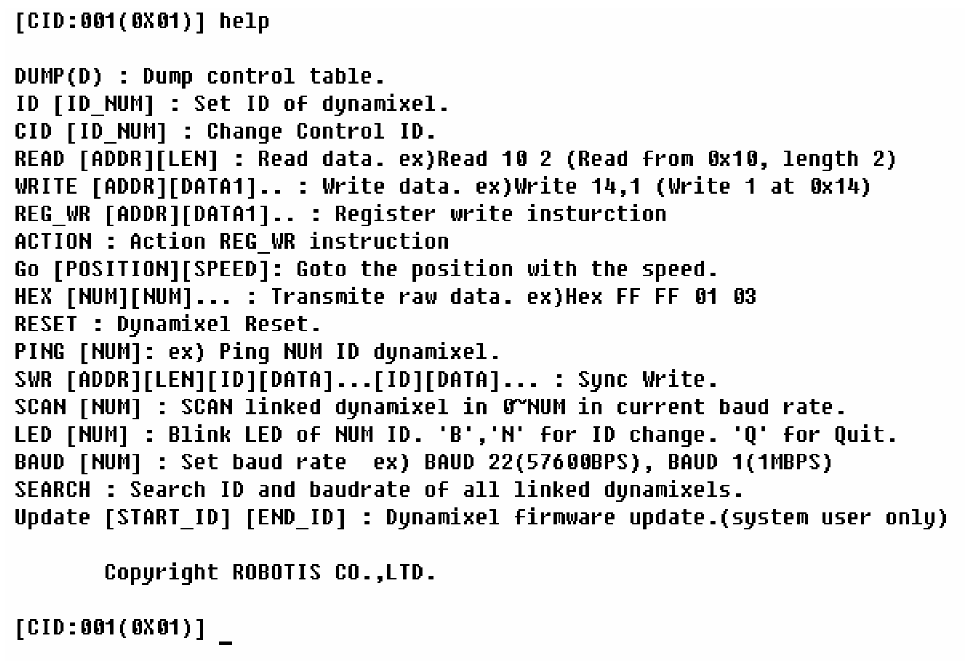
If the wires are connected properly you can check the communication speed and find the IDs by using the SEARCH command. This will be covered later.

Command Format

Type in a command followed by a number (parameter). The following are some examples.

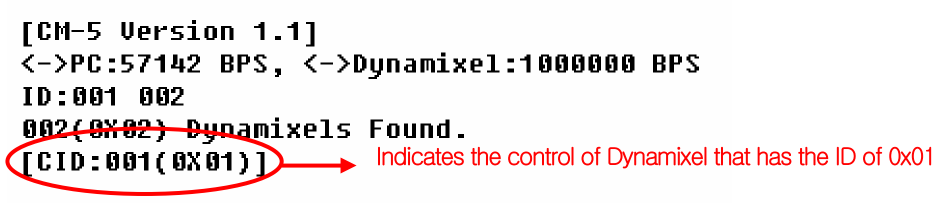
- ID 10
- Dump
- WR 10, 1
- RD 10, 2

HELP Type in help to see the available functions in manage mode.



```
[CID:001(0X01)] help

DUMP(D) : Dump control table.
ID [ID_NUM] : Set ID of dynamixel.
CID [ID_NUM] : Change Control ID.
READ [ADDR][LEN] : Read data. ex)Read 10 2 (Read from 0x10, length 2)
WRITE [ADDR][DATA1].. : Write data. ex)Write 14,1 (Write 1 at 0x14)
REG_WR [ADDR][DATA1].. : Register write insturction
ACTION : Action REG_WR instruction
Go [POSITION][SPEED]: Goto the position with the speed.
HEX [NUM][NUM]... : Transmite raw data. ex)Hex FF FF 01 03
RESET : Dynamixel Reset.
PING [NUM]: ex) Ping NUM ID dynamixel.
SWR [ADDR][LEN][ID][DATA]...[ID][DATA]... : Sync Write.
SCAN [NUM] : SCAN linked dynamixel in 0~NUM in current baud rate.
LED [NUM] : Blink LED of NUM ID. 'B','N' for ID change. 'Q' for Quit.
BAUD [NUM] : Set baud rate  ex) BAUD 22(57600BPS), BAUD 1(1MBPS)
SEARCH : Search ID and baudrate of all linked dynamixels.
Update [START_ID] [END_ID] : Dynamixel firmware update.(system user only)

       Copyright ROBOTIS CO.,LTD.

[CID:001(0X01)] _
```

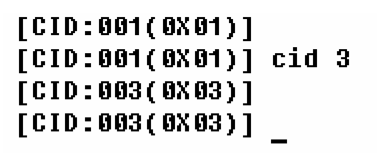
CID CID is the abbreviation of Control ID. This shows the ID of the Dynamixel that the CM-5 is controlling. CID is also used as a prompt character in manage mode.



```
[CM-5 Version 1.1]
<->PC:57142 BPS, <->Dynamixel:1000000 BPS
ID:001 002
002(0X02) Dynamixels Found.
[CID:001(0X01)]
```

Indicates the control of Dynamixel that has the ID of 0x01

To change the ID of the Dynamixel that the CM-5 controls to number 3, type in the following commands.



```
[CID:001(0X01)]
[CID:001(0X01)] cid 3
[CID:003(0X03)]
[CID:003(0X03)] _
```

After the command above, only the Dynamixel with an ID of 0x03 will react. Communication between the CM-5 and Dynamixel will not occur even if you run the CID command.

ID Use the ID command when you want to change the IDs of all the connected Dynamixels.

- Usage: ID [ID number]
- Example) ID 2 Set the connected Dynamixel’ s ID to 2.

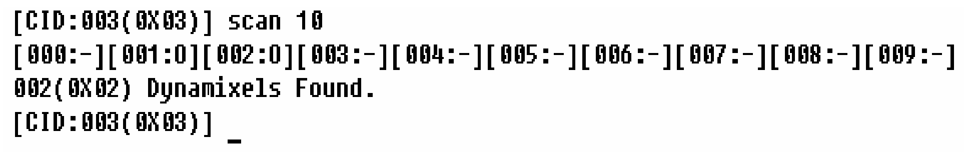
The ID command will change the IDs of all the connected Dynamixels regardless of the value of the CID. Therefore, when you use the ID command make sure that there is only one Dynamixel connected to the CM-5.

When you use the ID command make sure that there is only one Dynamixel connected to the CM-5

ID 254 Using ID number 254 will send commands to all Dynamixels. A Dynamixel will only react to a command with its own ID or with ID 254, but it will not send back a packet for commands sent with ID 254. ID 254 is also called the “Broadcasting ID.”

SCAN You can find the IDs of the Dynamixels that are connected to the CM-5 by running the SCAN command. If you type in SCAN N, the program will scan Dynamixel number 0 to N. The SCAN command will only work if the communication speed between the CM-5 and the Dynamixels is set properly.

- Usage: Scan [number of IDs]



```
[CID:003(0X03)] scan 10
[000:-][001:O][002:O][003:-][004:-][005:-][006:-][007:-][008:-][009:-]
002(0X02) Dynamixels Found.
[CID:003(0X03)] _
```

SEARCH If you are not sure if the communication speed between the CM-5 and the Dynamixel is set properly, you can use the SEARCH command to search the Dynamixels.

```
[CID:003(0X03)] search
Found! BAUD_REG:001, ID:001
Found! BAUD_REG:001, ID:002

[CID:003(0X03)] _
```

The SEARCH command is slow and it could find duplicates if similar baud rates are used. This is because UART communication somewhat robust against baud rate error.

LED Sometimes you will want to check the ID of each Dynamixel connected to the CM-5. Type in LED ID and the LED of the Dynamixel of the selected ID will blink. Type B and N to change the ID. Typing Q will end the LED command.

```
[CID:003(0X03)] led 1
B(before),N(next),Q(Quit)
LED Blink ID:001  }
LED Blink ID:002  }  When N is typed in
LED Blink ID:003  }
LED Blink ID:004
LED Blink ID:004   When Q is typed in
[CID:003(0X03)]
```

8-2. Other Commands

READ This command is used to read the data values in the control table of a Dynamixel. The READ command is used as the following.

- Usage: READ [ADDRESS] [Data Length for Reading]

The example below shows a command that reads 1 byte from Address 25 of the Dynamixel with an ID of 1.

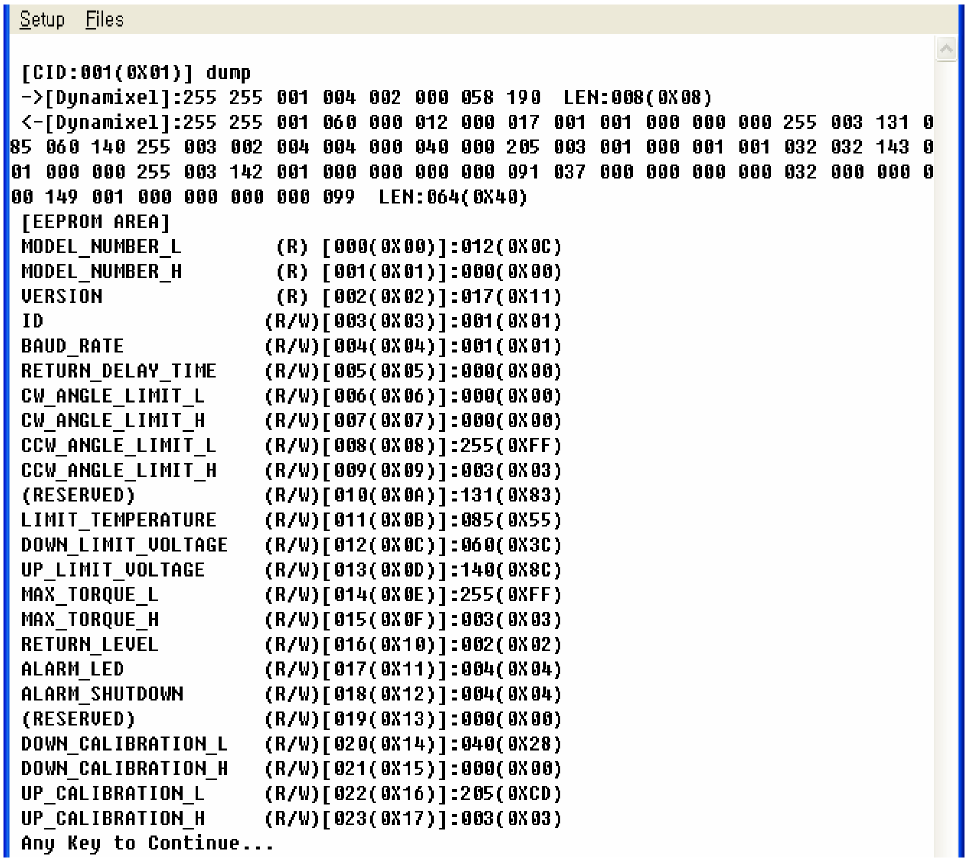
```
[CID:002(0X02)] rd 25 1
->[Dynamixel]:255 255 002 004 002 025 001 221  LEN:008(0X08)
<-[Dynamixel]:255 255 002 003 000 000 250  LEN:007(0X07)
[CID:002(0X02)]
```

WRITE This command is used to change a data value of the control table.
The WRITE command format as the follows.

- Usage: WRITE [Address] [Data] [Data] [Data]…
  In the example below, you can verify that the LED turns on and off when 1 and 0 is written to Address 25.

```
[CID:002(0X02)] wr 25 1
->[Dynamixel]:255 255 002 004 003 025 001 220  LEN:008(0X08)
<-[Dynamixel]:255 255 002 002 000 251  LEN:006(0X06)
[CID:002(0X02)] wr 25 0
->[Dynamixel]:255 255 002 004 003 025 000 221  LEN:008(0X08)
<-[Dynamixel]:255 255 002 002 000 251  LEN:006(0X06)
[CID:002(0X02)]
```

Dump This shows the control table values of the Dynamixel. The following shows the information that is dumped. Refer to the AX-12 manual for more information about control tables.

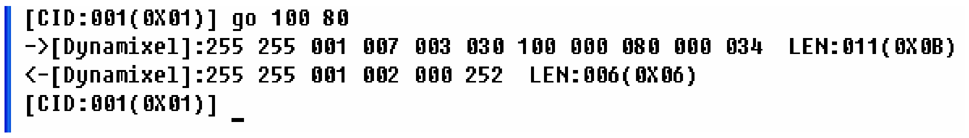


```
 [CID:001(0X01)] dump
 ->[Dynamixel]:255 255 001 004 002 000 058 190  LEN:008(0X08)
 <-[Dynamixel]:255 255 001 060 000 012 000 017 001 001 000 000 000 255 003 131 0
85 060 140 255 003 002 004 004 000 040 000 205 003 001 000 001 001 032 032 143 0
01 000 000 255 003 142 001 000 000 000 000 091 037 000 000 000 000 032 000 000 0
00 149 001 000 000 000 000 099  LEN:064(0X40)
 [EEPROM AREA]
 MODEL_NUMBER_L        (R) [000(0X00)]:012(0X0C)
 MODEL_NUMBER_H        (R) [001(0X01)]:000(0X00)
 VERSION               (R) [002(0X02)]:017(0X11)
 ID                  (R/W)[003(0X03)]:001(0X01)
 BAUD_RATE           (R/W)[004(0X04)]:001(0X01)
 RETURN_DELAY_TIME   (R/W)[005(0X05)]:000(0X00)
 CW_ANGLE_LIMIT_L    (R/W)[006(0X06)]:000(0X00)
 CW_ANGLE_LIMIT_H    (R/W)[007(0X07)]:000(0X00)
 CCW_ANGLE_LIMIT_L   (R/W)[008(0X08)]:255(0XFF)
 CCW_ANGLE_LIMIT_H   (R/W)[009(0X09)]:003(0X03)
 (RESERVED)          (R/W)[010(0X0A)]:131(0X83)
 LIMIT_TEMPERATURE   (R/W)[011(0X0B)]:085(0X55)
 DOWN_LIMIT_VOLTAGE  (R/W)[012(0X0C)]:060(0X3C)
 UP_LIMIT_VOLTAGE    (R/W)[013(0X0D)]:140(0X8C)
 MAX_TORQUE_L        (R/W)[014(0X0E)]:255(0XFF)
 MAX_TORQUE_H        (R/W)[015(0X0F)]:003(0X03)
 RETURN_LEVEL        (R/W)[016(0X10)]:002(0X02)
 ALARM_LED           (R/W)[017(0X11)]:004(0X04)
 ALARM_SHUTDOWN      (R/W)[018(0X12)]:004(0X04)
 (RESERVED)          (R/W)[019(0X13)]:000(0X00)
 DOWN_CALIBRATION_L  (R/W)[020(0X14)]:040(0X28)
 DOWN_CALIBRATION_H  (R/W)[021(0X15)]:000(0X00)
 UP_CALIBRATION_L    (R/W)[022(0X16)]:205(0XCD)
 UP_CALIBRATION_H    (R/W)[023(0X17)]:003(0X03)
 Any Key to Continue...
```

Press any key to continue dump.

GO This command moves the Dynamixel to the specified position. The GO command is used like the following.

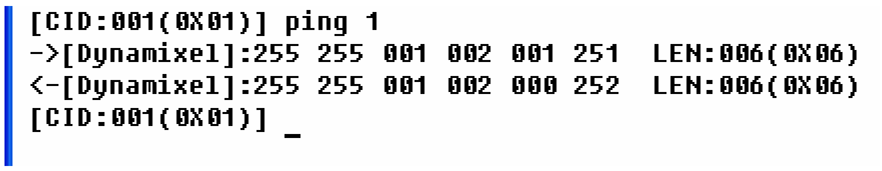
- Usage: GO [Position Value] [Speed Value]
  Here, the range of parameter values is from 0 to 1023. If you take a look at the packet, you can see that the WRITE command has been executed starting from Address 30 which corresponds to goal position and goal speed.



```
[CID:001(0X01)] go 100 80
->[Dynamixel]:255 255 001 007 003 030 100 000 080 000 034  LEN:011(0X0B)
<-[Dynamixel]:255 255 001 002 000 252  LEN:006(0X06)
[CID:001(0X01)] _
```

PING This command does not execute any special tasks, but is used to check to see if a Dynamixel is connected. The Dynamixel will return a packet even when it receives a Broadcasting ID with this command.

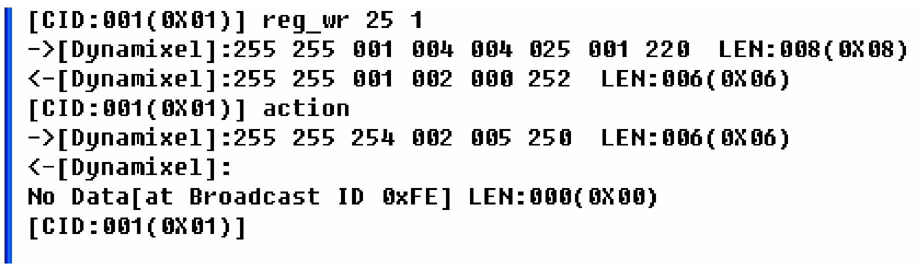
- Usage: PING [ID]



```
[CID:001(0X01)] ping 1
->[Dynamixel]:255 255 001 002 001 251  LEN:006(0X06)
<-[Dynamixel]:255 255 001 002 000 252  LEN:006(0X06)
[CID:001(0X01)] _
```

REG_WR This command registers the command WRITE. The command is only registered; not executed. The format is the same as the WRITE command. But it will only execute when the ACTION command is given.

ACTION This command executes the WRITE command that is registered by REG_WR.
The example below shows the process of turning on a LED using the REG_WR command. The LED will actually be turned on with the Action command.



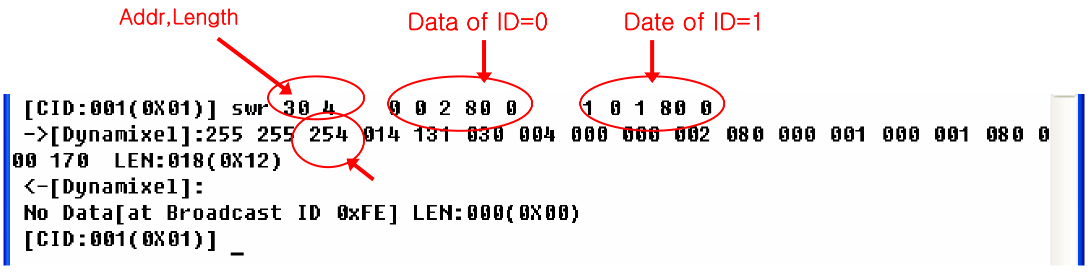
```
[CID:001(0X01)] reg_wr 25 1
->[Dynamixel]:255 255 001 004 004 025 001 220  LEN:008(0X08)
<-[Dynamixel]:255 255 001 002 000 252  LEN:006(0X06)
[CID:001(0X01)] action
->[Dynamixel]:255 255 254 002 005 250  LEN:006(0X06)
<-[Dynamixel]:
No Data[at Broadcast ID 0xFE] LEN:000(0X00)
[CID:001(0X01)]
```

The Action command is executed with the Broadcasting ID. The REG_WR and Action commands are useful when you want to actuate several Dynamixels starting at the same time.

SYNC_WR When you want to write to several Dynamixels and if the Write Addresses are all the same, you can use the SYNC_WR command to write to all of the Dynamixels at once. The format of a SYNC_WR command is as follows.

Usage: SWR [ADDRESS] [LENGTH] [ID] [DATA0] [DATA1] …[ID] [DATA0] [DATA1]…

The following example shows how to move a Dynamixel of ID = 0 to position 512(0x200) at a speed of 80(0x80) and a Dynamixel of ID = 1 to position 272(0x110) at a speed of 80(0x80). SYNC_WR is a broadcasting command.



```
 [CID:001(0X01)] swr 30 4   0 0 2 80 0    1 0 1 80 0
 ->[Dynamixel]:255 255 254 014 131 030 004 000 000 002 080 000 001 000 001 080 0
00 170  LEN:018(0X12)
 <-[Dynamixel]:
 No Data[at Broadcast ID 0xFE] LEN:000(0X00)
 [CID:001(0X01)] _
```

Baud This command is used to change the baud rate of the CM-5 Dynamixel controlling UART. The baud rate is calculated using the following equation.

Speed (BPS) = 2000000/(Parameter Value + 1)

Parameter Values for Important Baud Rates

Parameter	Set BPS	Goal BPS	Error
1	1000000.0	1000000.0	0.000%
3	500000.0	500000.0	0.000%
4	400000.0	400000.0	0.000%
7	250000.0	250000.0	0.000%
9	200000.0	200000.0	0.000%
16	117647.1	115200.0	-2.124%
34	57142.9	57600.0	0.794%
103	19230.8	19200.0	-0.160%
207	9615.4	9600.0	-0.160%

The Baud command changes the baud rate of the CM-5 itself and all the Dynamixels that are connected to the CM-5.

Usage: BAUD [Calculated parameter value]]

Note A maximum Baud Rate error of 3% is within the tolerance of UART communication.

RESET The RESET command will change all the settings of the Dynamixel back to the factory initial settings.

Usage: reset [ID]

```
[CID:001(0X01)] reset 1
Are you sure ?(y/N)
->[Dynamixel]:255 255 001 002 006 246  LEN:006(0X06)
<-[Dynamixel]:255 255 001 002 000 252  LEN:006(0X06)
[CID:001(0X01)]
```

H The H command will send the numbers that are typed into the Robot Terminal as text to the Dynamixels in binary format. The H command is useful when testing the packet communication protocol.

Usage: H [Parameter] [Parameter] [Parameter]···

So far we have learned the functions of several manage mode commands.

9. Information for Advanced Users

This chapter is for advanced users who have experience with microprocessors. In order to understand the following material you will need to have knowledge of hexadecimal, binary numbers, and ASCII code. Finally, we briefly explain how to control the CM-5 using the C language.

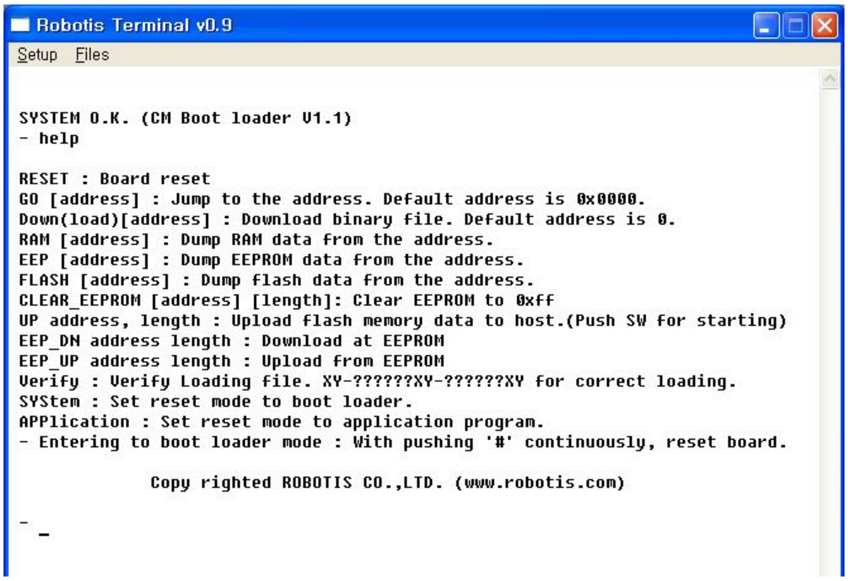
9-1. Boot Loader

Boot Loader When power is applied to the CM-5 unit, the "CM Boot Loader" program in Reset Vector is executed. The "CM Boot Loader" program does not have as many functions as a PC operating system, but it has the following basic features: uploading and executing user created programs to the CM-5 unit memory, verifying the data in the memory, and downloading programs back to a PC. All numbers are treated in hexadecimal.

Caution If you do not fully understand the system, do not use the Boot Loader.

Execution In the Robot Terminal, press and hold the # key and press the mode switch to go into the Boot Loader.

The figure below shows the Boot Loader screen. At this point, type "Help" and press Enter, and the following message will appear.



```
SYSTEM O.K. (CM Boot loader V1.1)
- help

RESET : Board reset
GO [address] : Jump to the address. Default address is 0x0000.
Down(load)[address] : Download binary file. Default address is 0.
RAM [address] : Dump RAM data from the address.
EEP [address] : Dump EEPROM data from the address.
FLASH [address] : Dump flash data from the address.
CLEAR_EEPROM [address] [length]: Clear EEPROM to 0xff
UP address, length : Upload flash memory data to host.(Push SW for starting)
EEP_DN address length : Download at EEPROM
EEP_UP address length : Upload from EEPROM
Verify : Verify Loading file. XY-??????XY-??????XY for correct loading.
SYStem : Set reset mode to boot loader.
APPlication : Set reset mode to application program.
- Entering to boot loader mode : With pushing '#' continuously, reset board.

             Copy righted ROBOTIS CO.,LTD. (www.robotis.com)

- _
```

This is a summary of the functions of the Boot Loader. Let' s take a closer look at them, one at a time.

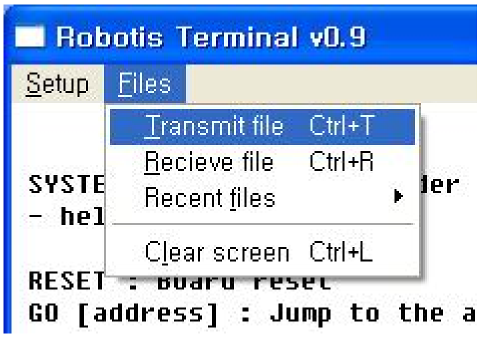
Download Let' s learn how to download a provided Firmware or a program you have created onto the CM-5 unit. Let' s try downloading a program called Bioloid.hex.

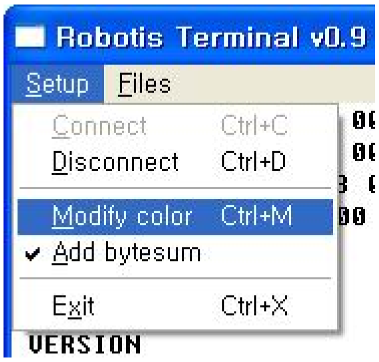
Type in the command "load." The following message should appear. This message indicates that the data is ready to be written to address 0.

```
- load
Write Address : 00000000
Ready..
```

Next, select "Transmit file" in the "Files" menu from the Robot Terminal program as shown below. It is recommended to check the "Add bytesum" menu item in the "Setup" menu as shown below.

Selecting a File for Transmission





Select "Examples\ROM file\Bioloid_VerXXX.hex" of the CD as the file to be transmitted. The selected file will be transmitted to the CM-5 through the serial cable.

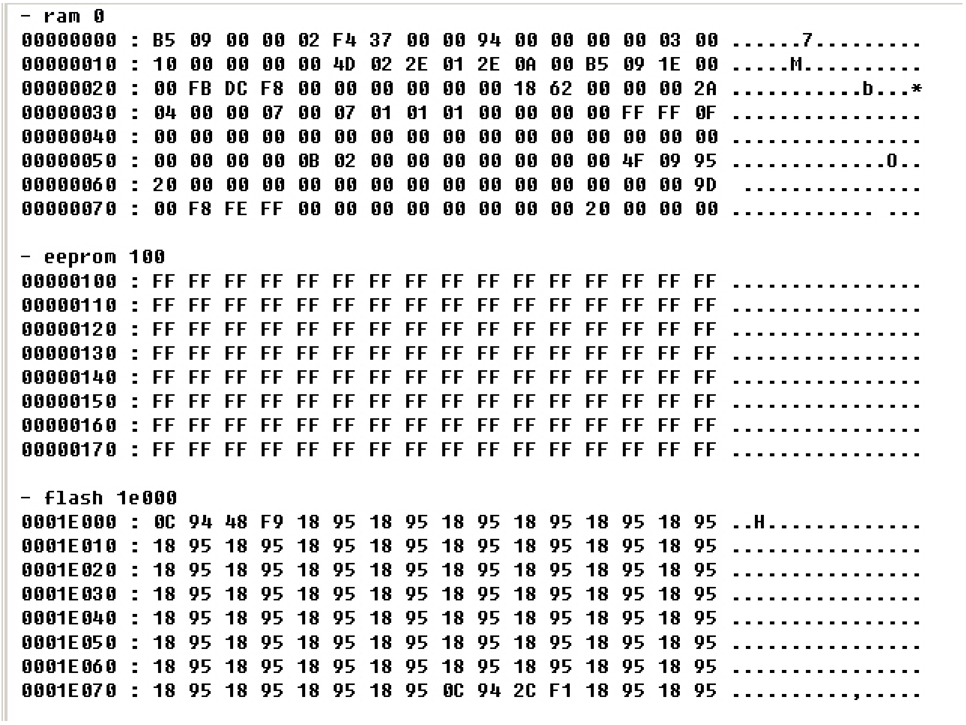
Transmission Complete

When the transmission is finished, a "Checksum:xx-xx" message will appear on the screen as shown below. If the two numbers match, this indicates that there were no errors during the transmission.

```
- ld
Write Address : 00000000
Ready..Success
Rewriting:0X0006
Size:0X000094A1  Checksum:91-91
-
```

If you press the mode change button, the downloaded program will be executed..

Memory Dump The CM-5 unit not only has 128 Kbytes of flash memory but also 4 Kbytes of RAM and 4 Kbytes EEPROM. There is a function in the CM-5 boot loader where you can check the contents in these memory spaces. Type in the memory type as the command followed by the address. The following figure is an example.



```
- ram 0
00000000 : B5 09 00 00 02 F4 37 00 00 94 00 00 00 00 03 00 ......7.........
00000010 : 10 00 00 00 00 4D 02 2E 01 2E 0A 00 B5 09 1E 00 .....M..........
00000020 : 00 FB DC F8 00 00 00 00 00 00 18 62 00 00 00 2A ...........b...*
00000030 : 04 00 00 07 00 07 01 01 01 00 00 00 00 FF FF 0F ................
00000040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000050 : 00 00 00 00 0B 02 00 00 00 00 00 00 00 4F 09 95 .............O..
00000060 : 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9D  ...............
00000070 : 00 F8 FE FF 00 00 00 00 00 00 00 00 20 00 00 00 ............ ...

- eeprom 100
00000100 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
00000110 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
00000120 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
00000130 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
00000140 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
00000150 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
00000160 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
00000170 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................

- flash 1e000
0001E000 : 0C 94 48 F9 18 95 18 95 18 95 18 95 18 95 18 95 ..H.............
0001E010 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 ................
0001E020 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 ................
0001E030 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 ................
0001E040 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 ................
0001E050 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 ................
0001E060 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 ................
0001E070 : 18 95 18 95 18 95 18 95 0C 94 2C F1 18 95 18 95 ..........,.....
```

9-2. USING THE C PROGRAM LANGUAGE

The program for the Bioloid was programmed in C and loaded with the Boot Loader. In order to write such a program, you will need to know how to program in C and you should also have some CPU hardware background. This is beyond the scope of this manual, thus we recommend you refer to other references for such information.
In this section we will learn about the Boot Loader and what part of the memory it is located at. We will also learn how much of the memory a user can use for programming.

Memory Map The CM-5 uses a CPU called the Atmega128. This CPU has 128 Kbytes of flash memory. The CM-5 divides the flash memory into several sections, as shown in the table below.

Address	Item	Function
0X00000 ~ 0X0BFFF	Bioloid program	Location of the program that operates the Bioloid
0X0A000 ~ 0X0DFFF	User area	Location of the user made behavior control program
0X0E000 ~ 0X1DFFF	Motion Data	Area for storing motion data for the robot
0X1E000 ~ 0X1FFFF	Boot Loader	Location of the "Boot Loader" program for verifying the download and memory status, etc.

When the power is applied, the "Boot Loader" located at address 0x1E000 executes. The file Bioloid_VerXXX.hex is loaded on to the 48 Kbyte user area, starting at Address 0X00000. You can see that the executable file of the user created C program has to be loaded at address of 0.

There is a compiler called AVR-GCC for creating the C program that you can use for free. This will be explained in more detail in chapter 9-3.

TIP Learning how to use C to operate the CM-5 is learning about microprocessors. Studying robotics and studying the microprocessor are two different things. Starting from using the IN, OUT commands in the microprocessor may not be an efficient way to operate a robot. A robot should be considered as a system, not from the level of a device or a board. When we make a homepage with a PC we usually don' t use ASM or C directly. We rather use a higher level tool to do the job. Similar to this, it would be more appropriate to use a higher level tool to concentrate more on the higher level behavior control of the robot.

9-3. Compiling(Compile)

This section talks about how to compile a CM-5 program. Before going through this section, we recommend studying the AX-12 manual.

Selecting a Compiler

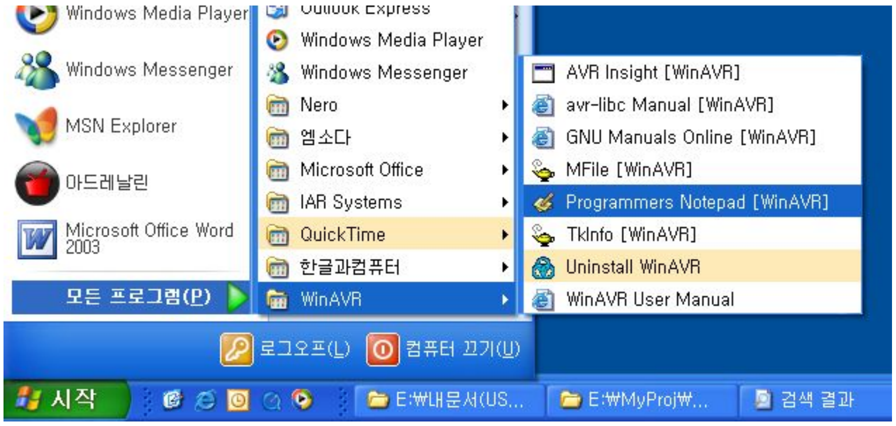
The CPU on the CM-5 unit is the Atmega128 of the AVR Series from Atmel. There are many different C compilers available for the Atmega128 but their prices are generally very high. One the other hand, a global organization called GNU is distributing their compiler called GCC for free of charge. For this reason, many research labs and institutions are using this compiler instead. The CM-5 unit also uses the AVR GCC compiler.

Compiler and Editor

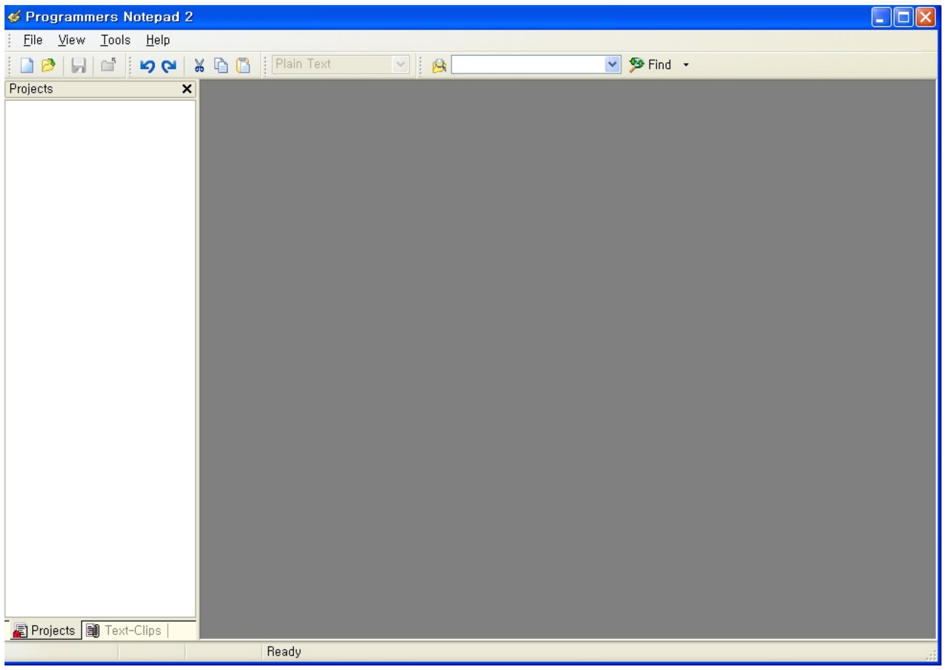
Windows OS users are familiar with compilers that have an editor function built in. But for compilers that run on text based OSes such as Linux, they usually have a separate compiler and an editor. The GCC is a compiler based on the command line interface and thus does not have a built in editor. Therefore, users have to use a separate editor to develop a program. The AVR-Edit and the WIN-AVR are two editors for GCC that are popular. We will be using the WIN-AVR editor in this tutorial. This editor runs the compiler by internally calling the AVR GCC.

Installing the Compiler

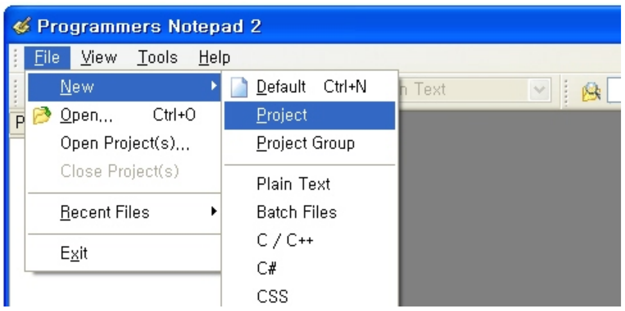
Let' s install and run the AVR GCC Editor, Win-AVR. Win-AVR can be downloaded from the Internet and you can find a link from the website www.robotis.com. Since Win-AVR already includes the AVR GCC, the user only needs to install the Win-AVR. The user can select the following menu after the installation is complete.



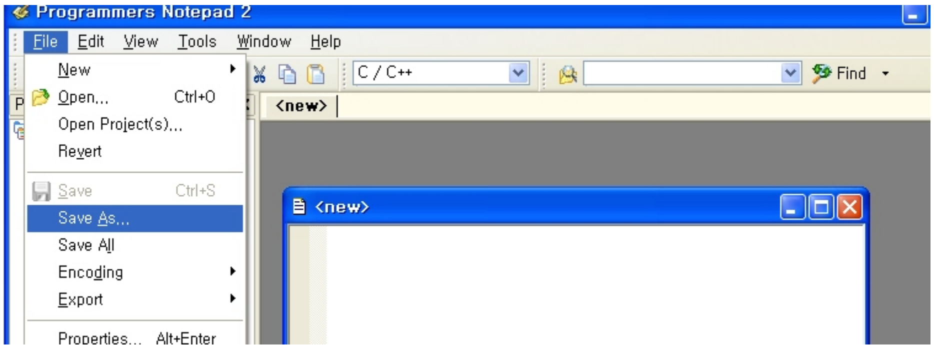
Win AVR The Win-AVR editor runs the GCC compiler by calling the AVR GCC internally. From the Win-AVR menu, select Programmer' s Notepad [Win AVR]. The following screen will appear.



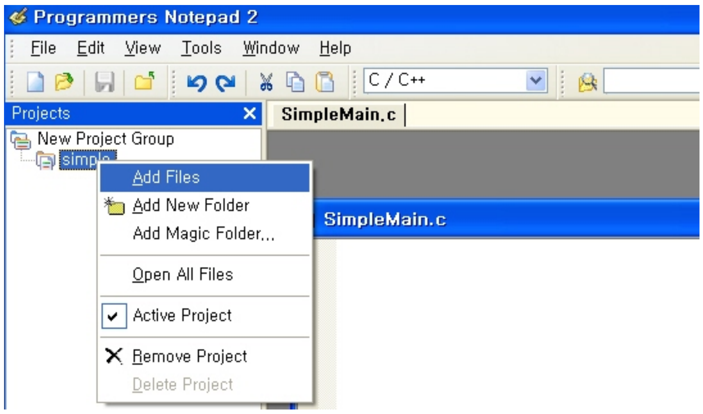
Project File When writing a large program, it is helpful to structure the program by dividing the source file into a number of smaller files by its contents. The Project File is a higher-level file that contains the list of the entire source files associated with the program that is being developed and includes all the compile options. Open a new Project File as shown below and give it any name as you wish. Here, the project is named "Simple." Select "Project" from the "New" menu item under the "File" menu.



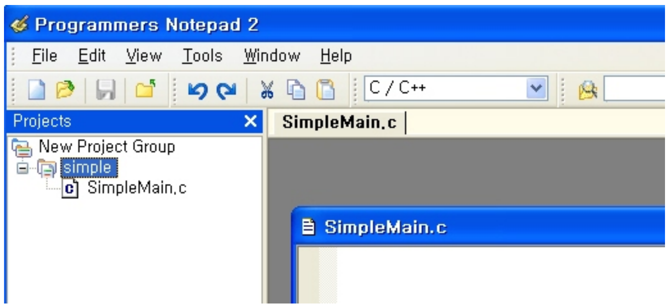
C Source File Next, we open the C source code file which is a lower-level file. Select "C/C++" from the "New" menu item under the "File" menu.



To assign a name, select "Save As…" from the "File" menu and give it any name as you wish. Here, the C source file is named "SimpleMain.c."
Next, the source file named "SimpleMain.c" needs to be added to the project file named "simple. On the left side of the project window, right click on "simple" then click "Add Files" to select "SimpleMain.c"



Now, a project named "simple" is created which includes a source file called "SimpleMain.c."



Main() Type the following in the “SimpleMain.c” source code.

```
void main(void)
{
}
```

The above program has no content and is in the form of the most basic structure of a C source code. Now the source code is completed, the next step is to compile it. To do so, the user has to select the necessary options for compiling it.

Makefile The Makefile contains the information for the compile options. Sometimes the project file may contain the information for these options. However, since the Win-VAR does not have an internal compiler, it needs a separate Makefile for the GCC. Just like creating a project, the Makefile needs to be created only once and then modified as needed.

Location of the Makefile

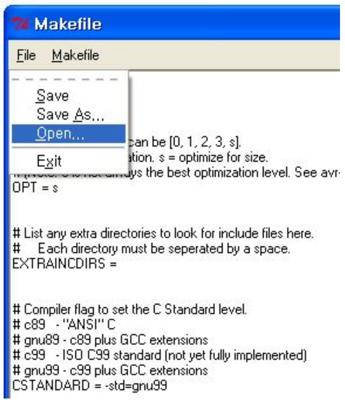
The Makefile has to be in the same directory (folder location) as the project file and the main source file that contains the Main function. The name of the file has to be Makefile without an extension and cannot be changed. Thus, the files “Makefile,” “Simple. pnproj,” and “SimpleMain.c” have to be in the same folder.

Editing the Makefile

Makefile contains information on opening and compiling the source file, and the name of the executable file. Makefile also contains other information but the important information that the user needs to deal with are the name of the resultant file, the name of the source file and its directory. This concept will become clearer once we go through the following tutorial. First, from the CD that came with the CM-5 unit, copy the Examples\CM-5\makefile to the current working directory folder.

Running the mfile

As the picture shown on the right, run the “mfile” program in Win-AVR. This file only contains the editing function for Makefile.

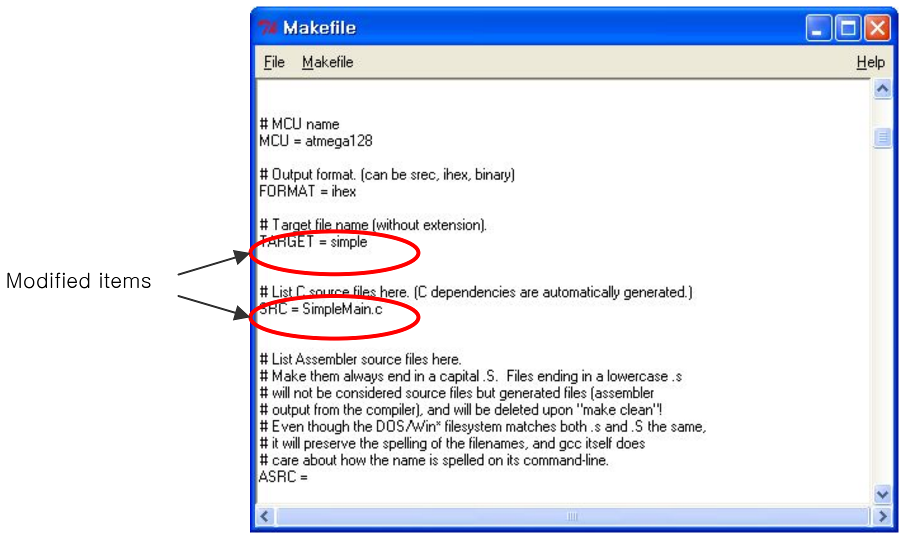


```
can be [0, 1, 2, 3, s].
ation. s = optimize for size.
ys the best optimization level. See avr-
OPT = s


# List any extra directories to look for include files here.
#     Each directory must be seperated by a space.
EXTRAINCDIRS =


# Compiler flag to set the C Standard level.
# c89   - "ANSI" C
# gnu89 - c89 plus GCC extensions
# c99   - ISO C99 standard (not yet fully implemented)
# gnu99 - c99 plus GCC extensions
CSTANDARD = -std=gnu99
```

First, open the file that you want to edit. There are two ways of editing the Makefile; the user can directly edit the contents of the Makefile, or the user can use the menu to edit it. To edit it using the menu, the user selects the "Makefile" menu on the right to change the options while the "mfile" is running. To directly edit the contents of the Makefile, the user selects the "Enable Editing of Makefile" under the "Makefile" menu to change options by using the keyboard..

Editing Using the Menu

When a new project is created, two sections have to be modified in the Makefile; one is the main file name section, and the other is the C/C++ Source file(s) section. First set the name of the main file name to "simple." This is used for the file names the compiler creates. Source codes can be added in the C/C++ Source file section. Edit the two sections of the Makefile as shown below.



```
# MCU name
MCU = atmega128

# Output format. (can be srec, ihex, binary)
FORMAT = ihex

# Target file name (without extension).
TARGET = simple

# List C source files here. (C dependencies are automatically generated.)
SRC = SimpleMain.c

# List Assembler source files here.
# Make them always end in a capital .S.  Files ending in a lowercase .s
# will not be considered source files but generated files (assembler
# output from the compiler), and will be deleted upon "make clean"!
# Even though the DOS/Win* filesystem matches both .s and .S the same,
# it will preserve the spelling of the filenames, and gcc itself does
# care about how the name is spelled on its command-line.
ASRC =
```

The Makefile can be edited by using the "Notepad" or any text editor.

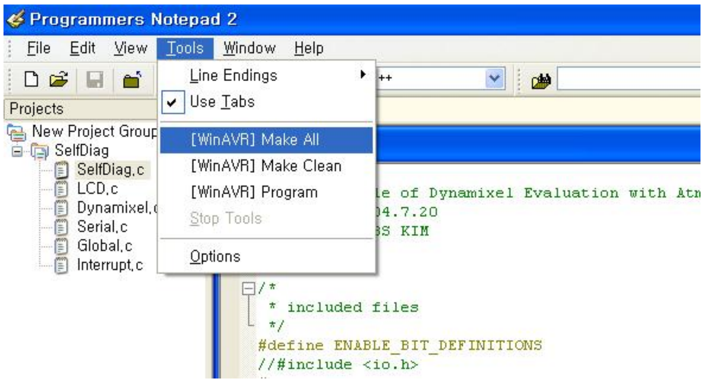
Summary of Makefile

The concept of Makefile can be tricky for those who use GCC for the first. The Makefile can be summarized with the following two concepts..

1. The Makefile has to be located in the same folder as the project file and source file. The name of the Makefile cannot be changed.
2. Within the Makefile, the source file section (SRC) and the resultant file section (TARGET) needs to be modified as needed.

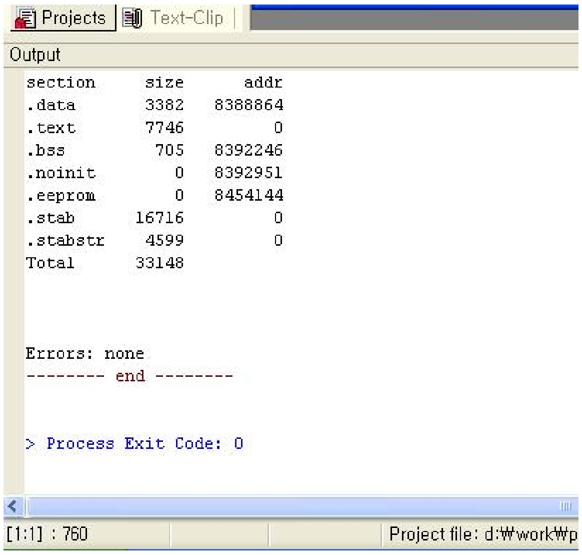
Executing Compile

Select “Make All” from the “Tools” menu of the Programmers Notepad 2 [WinAVR].



```
le of Dynamixel Evaluation with Atm
04.7.20
BS KIM

/*
 * included files
 */
#define ENABLE_BIT_DEFINITIONS
//#include <io.h>
```

The compile result message will appear at the bottom of the output window. If the compile was successful with no error, the “Errors: none” message will appear.



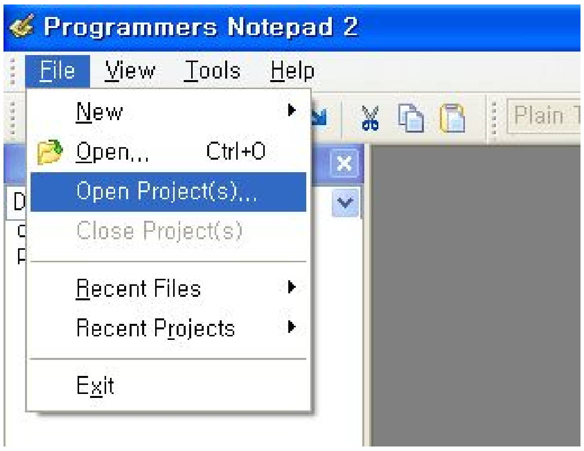
```
section      size       addr
.data        3382    8388864
.text        7746          0
.bss          705    8392246
.noinit         0    8392951
.eeprom         0    8454144
.stab       16716          0
.stabstr     4599          0
Total       33148


Errors: none
-------- end --------


> Process Exit Code: 0
```

Simple.hex Download

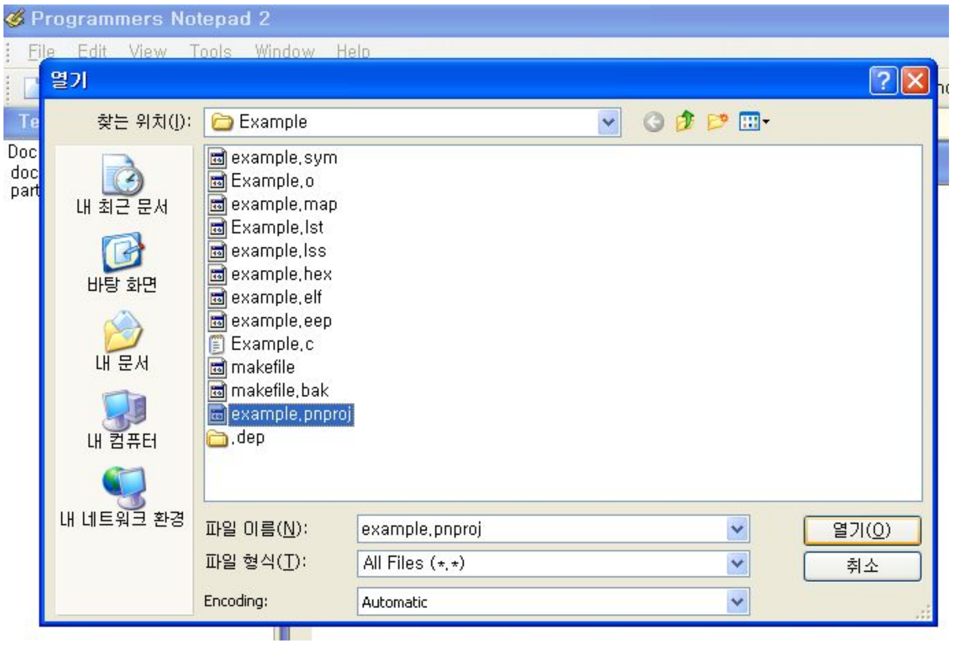
Now let’ s verify and download the file “simple.hex.” Use the Boot Loader to download the program. If you run it nothing will happen because the file does not contain any information.

9-4. Example.c

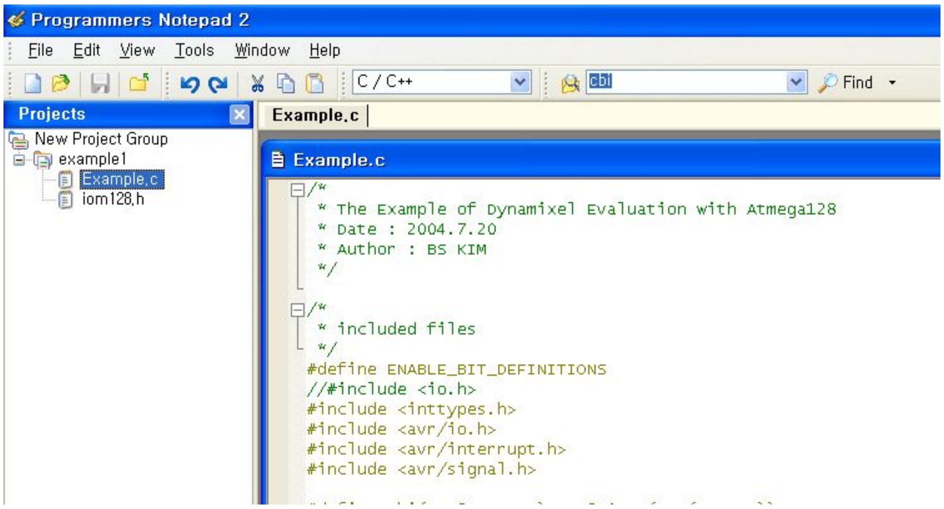
Example.c "Example.c" contains various routines for the CM-5 to directly control the Dynamixel actuators. Using these routines, one can easily develop a program for controlling them. Select "Open Project(s)" from the "File" menu on the MinAVR Programmers Notepad.



Project Open Open the "example.pnproj" project file in the Example folder.

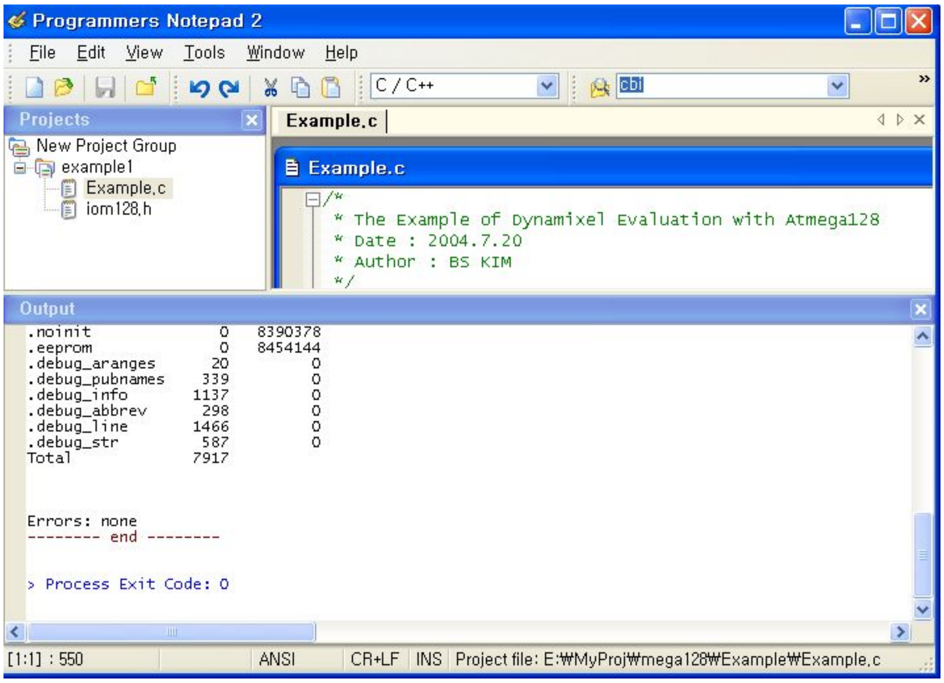


Files Open Double-click the "Example.c" on the left and the contents of it appear on the screen.



```
/*
 * The Example of Dynamixel Evaluation with Atmega128
 * Date : 2004.7.20
 * Author : BS KIM
 */

/*
 * included files
 */
#define ENABLE_BIT_DEFINITIONS
//#include <io.h>
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
```

Compile Select "Make All" from the "Tools" menu to compile. The output after the compile should look like the following.

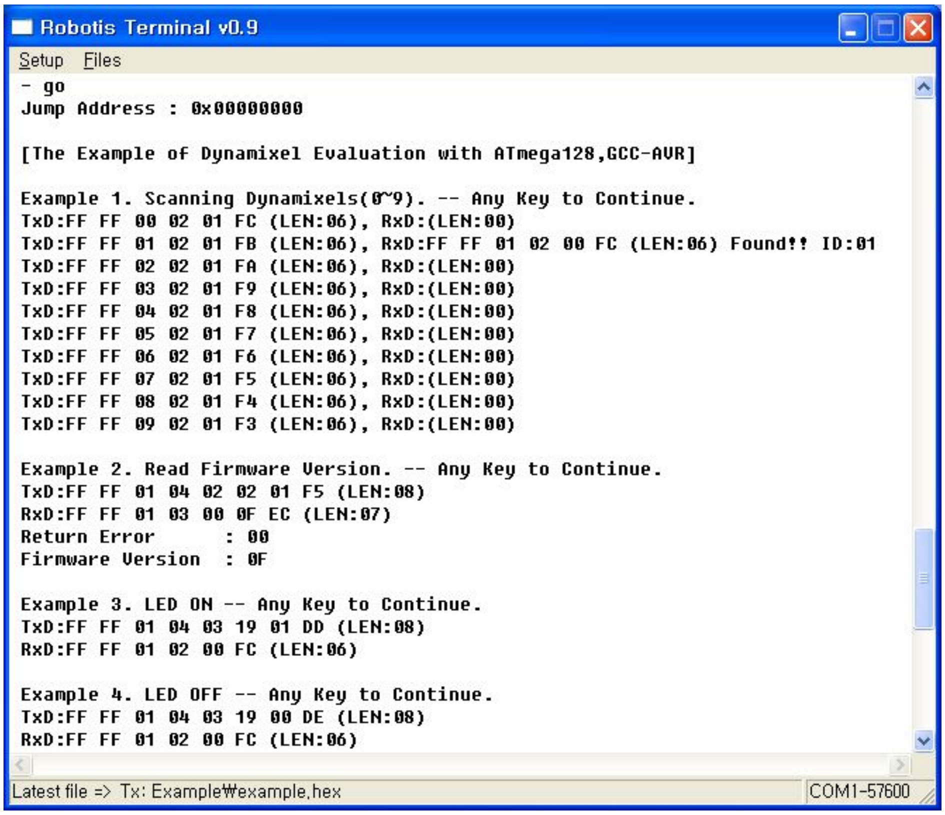


```
/*
 * The Example of Dynamixel Evaluation with Atmega128
 * Date : 2004.7.20
 * Author : BS KIM
 */
```

```
.noinit            0     8390378
.eeprom            0     8454144
.debug_aranges    20           0
.debug_pubnames  339           0
.debug_info     1137           0
.debug_abbrev    298           0
.debug_line     1466           0
.debug_str       587           0
Total           7917


Errors: none
-------- end --------


> Process Exit Code: 0
```

[1:1] : 550 ANSI CR+LF INS Project file: E:₩MyProj₩mega128₩Example₩Example.c

Download Now, let’ s use the Robot Terminal to download “example.hex” to the CM-5 unit. Please refer to Chapter 2 for downloading instructions. Use the “Go” command to execute “example.hex.” The screen shot of this is shown below. Pressing a key will make it proceed to the next example.



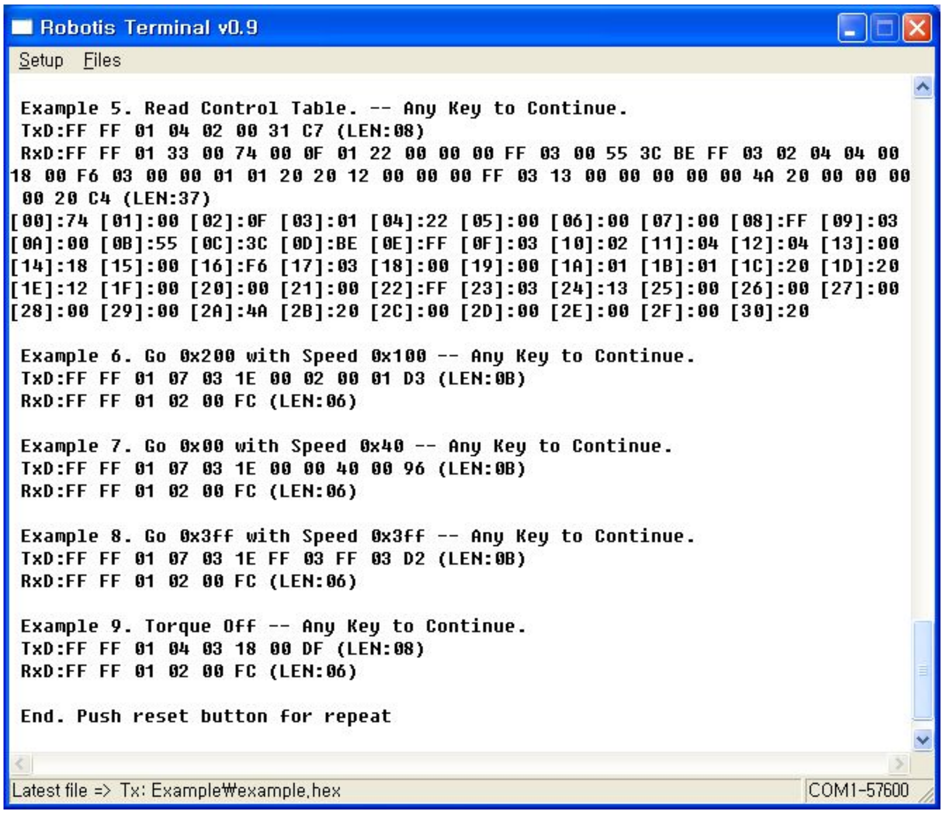
```
- go
Jump Address : 0x00000000

[The Example of Dynamixel Evaluation with ATmega128,GCC-AVR]

Example 1. Scanning Dynamixels(0~9). -- Any Key to Continue.
TxD:FF FF 00 02 01 FC (LEN:06), RxD:(LEN:00)
TxD:FF FF 01 02 01 FB (LEN:06), RxD:FF FF 01 02 00 FC (LEN:06) Found!! ID:01
TxD:FF FF 02 02 01 FA (LEN:06), RxD:(LEN:00)
TxD:FF FF 03 02 01 F9 (LEN:06), RxD:(LEN:00)
TxD:FF FF 04 02 01 F8 (LEN:06), RxD:(LEN:00)
TxD:FF FF 05 02 01 F7 (LEN:06), RxD:(LEN:00)
TxD:FF FF 06 02 01 F6 (LEN:06), RxD:(LEN:00)
TxD:FF FF 07 02 01 F5 (LEN:06), RxD:(LEN:00)
TxD:FF FF 08 02 01 F4 (LEN:06), RxD:(LEN:00)
TxD:FF FF 09 02 01 F3 (LEN:06), RxD:(LEN:00)

Example 2. Read Firmware Version. -- Any Key to Continue.
TxD:FF FF 01 04 02 02 01 F5 (LEN:08)
RxD:FF FF 01 03 00 0F EC (LEN:07)
Return Error     : 00
Firmware Version  : 0F

Example 3. LED ON -- Any Key to Continue.
TxD:FF FF 01 04 03 19 01 DD (LEN:08)
RxD:FF FF 01 02 00 FC (LEN:06)

Example 4. LED OFF -- Any Key to Continue.
TxD:FF FF 01 04 03 19 00 DE (LEN:08)
RxD:FF FF 01 02 00 FC (LEN:06)
```

Example 1 sends the “Ping” command to the Dynamixel actuators (ID from 0 to 9) and checks if there are any replies. The Baud rate for the CM-5 is set to 57,600 bps. From the results shown here, you can see that one Dynamixel actuator with the ID of 1 is connected to the CM-5 unit.

Example 2 demonstrates the use of the “Read” command. It reads the data from Address 2 of the Control Table of the Dynamixel actuator with the ID of 1. The data from Address 2 is the Firmware version and the results show that it currently has a firmware version of 0x0F. Please refer to the Dynamixel manual for information about the Control Table and for the structure of the packets

Example 3 turns on the LED of a Dynamixel actuator by writing 1 to address 0x19 of the Control Table. All actions for the Dynamixel actuators can be activated in this way by writing data to the corresponding address in the Control Table.

Example 4 turns off the LED of a Dynamixel actuator by writing 0 to address 0x19 of the Control Table.



```
 Example 5. Read Control Table. -- Any Key to Continue.
 TxD:FF FF 01 04 02 00 31 C7 (LEN:08)
 RxD:FF FF 01 33 00 74 00 0F 01 22 00 00 00 FF 03 00 55 3C BE FF 03 02 04 04 00
18 00 F6 03 00 00 01 01 20 20 12 00 00 00 FF 03 13 00 00 00 00 00 4A 20 00 00 00
 00 20 C4 (LEN:37)
[00]:74 [01]:00 [02]:0F [03]:01 [04]:22 [05]:00 [06]:00 [07]:00 [08]:FF [09]:03
[0A]:00 [0B]:55 [0C]:3C [0D]:BE [0E]:FF [0F]:03 [10]:02 [11]:04 [12]:04 [13]:00
[14]:18 [15]:00 [16]:F6 [17]:03 [18]:00 [19]:00 [1A]:01 [1B]:01 [1C]:20 [1D]:20
[1E]:12 [1F]:00 [20]:00 [21]:00 [22]:FF [23]:03 [24]:13 [25]:00 [26]:00 [27]:00
[28]:00 [29]:00 [2A]:4A [2B]:20 [2C]:00 [2D]:00 [2E]:00 [2F]:00 [30]:20

 Example 6. Go 0x200 with Speed 0x100 -- Any Key to Continue.
 TxD:FF FF 01 07 03 1E 00 02 00 01 D3 (LEN:0B)
 RxD:FF FF 01 02 00 FC (LEN:06)

 Example 7. Go 0x00 with Speed 0x40 -- Any Key to Continue.
 TxD:FF FF 01 07 03 1E 00 00 40 00 96 (LEN:0B)
 RxD:FF FF 01 02 00 FC (LEN:06)

 Example 8. Go 0x3ff with Speed 0x3ff -- Any Key to Continue.
 TxD:FF FF 01 07 03 1E FF 03 FF 03 D2 (LEN:0B)
 RxD:FF FF 01 02 00 FC (LEN:06)

 Example 9. Torque Off -- Any Key to Continue.
 TxD:FF FF 01 04 03 18 00 DF (LEN:08)
 RxD:FF FF 01 02 00 FC (LEN:06)

 End. Push reset button for repeat
```

Example 5 reads all the data from the Control Table by sending a packet to read data from address 0 to 0x31. The figure above shows the list of these 0x37 packets in the [Address]: Data form.

Example 6 demonstrates the command for moving the output of a Dynamixel actuator to a specified position. This is the most often used command. The Goal Position value of 0x200 (corresponding to the position at 180 degree) is written to address 0x1e of the Control Table. The Goal Speed value of 0x100 is written to address 0x20 of the Control Table as well. Note that both values (Goal Position and Goal Speed) can be written at the same time using only one packet.

Example 7 and Example 8 each demonstrate the command for moving the output of a Dynamixel actuator to a specified position, and they follow the same method as explain in Example 6.

The last example (Example 9) turns off the torque of the Dynamixel actuator by transmitting a packet to write a 0 to address 0x18 (address for Torque Enable) of the Control Table.

Example.c

```
/*
 * The Example of Dynamixel Evaluation with Atmega128
 * Date : 2005.7.11
 * Author : BS KIM
 */

/*
 * included files
 */
#define ENABLE_BIT_DEFINITIONS
//#include <io.h>
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>

#define cbi(REG8,BITNUM) REG8 &= ~(_BV(BITNUM))
#define sbi(REG8,BITNUM) REG8 |= _BV(BITNUM)


typedef unsigned char byte;
typedef unsigned int word;
#define ON 1
#define OFF 0
#define _ON 0
#define _OFF 1


//--- Control Table Address ---
//EEPROM AREA
#define P_MODEL_NUMBER_L      0
#define P_MODOEL_NUMBER_H     1
#define P_VERSION             2
#define P_ID                  3
#define P_BAUD_RATE           4
#define P_RETURN_DELAY_TIME   5
#define P_CW_ANGLE_LIMIT_L    6
#define P_CW_ANGLE_LIMIT_H    7
#define P_CCW_ANGLE_LIMIT_L   8
#define P_CCW_ANGLE_LIMIT_H   9
#define P_SYSTEM_DATA2        10
#define P_LIMIT_TEMPERATURE   11
#define P_DOWN_LIMIT_VOLTAGE  12
#define P_UP_LIMIT_VOLTAGE    13
#define P_MAX_TORQUE_L        14
#define P_MAX_TORQUE_H        15
#define P_RETURN_LEVEL        16
#define P_ALARM_LED           17
#define P_ALARM_SHUTDOWN      18
#define P_OPERATING_MODE      19
#define P_DOWN_CALIBRATION_L  20
#define P_DOWN_CALIBRATION_H  21
#define P_UP_CALIBRATION_L    22
#define P_UP_CALIBRATION_H    23

#define P_TORQUE_ENABLE          (24)
#define P_LED                    (25)
#define P_CW_COMPLIANCE_MARGIN   (26)
#define P_CCW_COMPLIANCE_MARGIN  (27)
#define P_CW_COMPLIANCE_SLOPE    (28)
#define P_CCW_COMPLIANCE_SLOPE   (29)
#define P_GOAL_POSITION_L        (30)
#define P_GOAL_POSITION_H        (31)
#define P_GOAL_SPEED_L           (32)
#define P_GOAL_SPEED_H           (33)
#define P_TORQUE_LIMIT_L         (34)
#define P_TORQUE_LIMIT_H         (35)
#define P_PRESENT_POSITION_L     (36)
#define P_PRESENT_POSITION_H     (37)
#define P_PRESENT_SPEED_L        (38)
#define P_PRESENT_SPEED_H        (39)
#define P_PRESENT_LOAD_L         (40)
#define P_PRESENT_LOAD_H         (41)
#define P_PRESENT_VOLTAGE        (42)
#define P_PRESENT_TEMPERATURE    (43)
#define P_REGISTERED_INSTRUCTION (44)
#define P_PAUSE_TIME             (45)
#define P_MOVING (46)
#define P_LOCK                   (47)
#define P_PUNCH_L                (48)
#define P_PUNCH_H                (49)

//--- Instruction ---
#define INST_PING           0x01
#define INST_READ           0x02
#define INST_WRITE          0x03
#define INST_REG_WRITE      0x04
#define INST_ACTION         0x05
#define INST_RESET          0x06
#define INST_DIGITAL_RESET  0x07
#define INST_SYSTEM_READ    0x0C
#define INST_SYSTEM_WRITE   0x0D
#define INST_SYNC_WRITE     0x83
#define INST_SYNC_REG_WRITE 0x84

#define CLEAR_BUFFER gbRxBufferReadPointer = gbRxBufferWritePointer
#define DEFAULT_RETURN_PACKET_SIZE 6
#define BROADCASTING_ID 0xfe

#define TxD8 TxD81
#define RxD8 RxD81

//Hardware Dependent Item
#define DEFAULT_BAUD_RATE 34   //57600bps at 16MHz

////// For CM-5
#define   RS485_TXD   PORTE   &=   ~_BV(PE3),PORTE   |=   _BV(PE2)
                      //_485_DIRECTION = 1
#define   RS485_RXD   PORTE   &=   ~_BV(PE2),PORTE   |=   _BV(PE3)
                      //PORT_485_DIRECTION = 0
/*
////// For CM-2
#define RS485_TXD PORTE |= _BV(PE2); //_485_DIRECTION = 1
#define RS485_RXD PORTE &= ~_BV(PE2);//PORT_485_DIRECTION = 0
*/
//#define  TXD0_FINISH  UCSR0A,6   //This bit is for checking TxD
                      Buffer in CPU is empty or not.
//#define TXD1_FINISH  UCSR1A,6

#define SET_TxD0_FINISH   sbi(UCSR0A,6)
#define RESET_TXD0_FINISH cbi(UCSR0A,6)
#define CHECK_TXD0_FINISH bit_is_set(UCSR0A,6)
#define SET_TxD1_FINISH   sbi(UCSR1A,6)
#define RESET_TXD1_FINISH cbi(UCSR1A,6)
#define CHECK_TXD1_FINISH bit_is_set(UCSR1A,6)

#define RX_INTERRUPT 0x01
#define TX_INTERRUPT 0x02
#define OVERFLOW_INTERRUPT 0x01
#define SERIAL_PORT0 0
#define SERIAL_PORT1 1
#define BIT_RS485_DIRECTION0  0x08  //Port E
#define BIT_RS485_DIRECTION1  0x04  //Port E

#define BIT_ZIGBEE_RESET                PD4   //out : default 1
                      //PORTD
#define BIT_ENABLE_RXD_LINK_PC       PD5  //out : default 1
#define BIT_ENABLE_RXD_LINK_ZIGBEE   PD6  //out : default 0
#define BIT_LINK_PLUGIN              PD7  //in, no pull up

void TxD81(byte bTxdData);
void TxD80(byte bTxdData);
void TxDString(byte *bData);
void TxD8Hex(byte bSentData);
void TxD32Dec(long lLong);
byte RxD81(void);
void MiliSec(word wDelayTime);
void PortInitialize(void);
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt);
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength);
byte RxPacket(byte bRxLength);
void PrintBuffer(byte *bpPrintBuffer, byte bLength);

// --- Gloval Variable Number ---
volatile byte gbpRxInterruptBuffer[256];
byte gbpParameter[128];
byte gbRxBufferReadPointer;
byte gbpRxBuffer[128];
byte gbpTxBuffer[128];
volatile byte gbRxBufferWritePointer;

int main(void)
{
  byte bCount,bID, bTxPacketLength,bRxPacketLength;

  PortInitialize(); //Port In/Out Direction Definition
  RS485_RXD; //Set RS485 Direction to Input State.

                     SerialInitialize(SERIAL_PORT0,DEFAULT_BAUD_RAT
                     E,RX_INTERRUPT);//RS485
                     Initializing(RxInterrupt)
  SerialInitialize(SERIAL_PORT1,DEFAULT_BAUD_RATE,0);         //RS232
                     Initializing(None Interrupt)

  gbRxBufferReadPointer  =  gbRxBufferWritePointer  =  0;    //RS485
                     RxBuffer Clearing.

  sei();  //Enable Interrupt -- Compiler Function
  TxDString("¥r¥n  [The  Example  of  Dynamixel  Evaluation  with
                     ATmega128,GCC-AVR]");

//Dynamixel Communication Function Execution Step.
//  Step  1.  Parameter  Setting  (gbpParameter[]).  In  case  of  no
                     parameter  instruction(Ex.  INST_PING),  this
                     step is not needed.
//   Step   2.   TxPacket(ID,INSTRUCTION,LengthOfParameter);   --Total
                     TxPacket Length is returned
// Step  3.  RxPacket(ExpectedReturnPacketLength);  --  Real  RxPacket
                     Length is returned
// Step 4 PrintBuffer(BufferStartPointer,LengthForPrinting);

  bID = 1;
  TxDString("¥r¥n¥n Example 1. Scanning Dynamixels(0~9). -- Any Key
                     to Continue."); RxD8();
  for(bCount = 0; bCount < 0x0A; bCount++)
  {
    bTxPacketLength = TxPacket(bCount,INST_PING,0);
    bRxPacketLength = RxPacket(255);
    TxDString("¥r¥n                                          TxD:");
                     PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString(",                                             RxD:");
                     PrintBuffer(gbpRxBuffer,bRxPacketLength);
    if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE)
    {
      TxDString(" Found!! ID:");TxD8Hex(bCount);
      bID = bCount;
    }
  }

  TxDString("¥r¥n¥n Example 2. Read Firmware Version. -- Any Key to
                     Continue."); RxD8();
  gbpParameter[0] = P_VERSION; //Address of Firmware Version
  gbpParameter[1] = 1; //Read Length
  bTxPacketLength = TxPacket(bID,INST_READ,2);
  bRxPacketLength                                                  =
                     RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpParamet
                     er[1]);
TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);
if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE+gbpParameter[1])
{
  TxDString("¥r¥n Return Error      : ");TxD8Hex(gbpRxBuffer[4]);
  TxDString("¥r¥n Firmware Version  : ");TxD8Hex(gbpRxBuffer[5]);
}

TxDString("¥r¥n¥n  Example  3.  LED  ON  --  Any  Key  to  Continue.");
                     RxD8();
gbpParameter[0] = P_LED; //Address of LED
gbpParameter[1] = 1; //Writing Data
bTxPacketLength = TxPacket(bID,INST_WRITE,2);
bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

TxDString("¥r¥n¥n  Example  4.  LED  OFF  --  Any  Key  to  Continue.");
                     RxD8();
gbpParameter[0] = P_LED; //Address of LED
gbpParameter[1] = 0; //Writing Data
bTxPacketLength = TxPacket(bID,INST_WRITE,2);
bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

TxDString("¥r¥n¥n  Example  5.  Read  Control  Table.  --  Any  Key  to
                     Continue."); RxD8();
gbpParameter[0] = 0; //Reading Address
gbpParameter[1] = 49; //Read Length
bTxPacketLength = TxPacket(bID,INST_READ,2);
bRxPacketLength                                                    =
                     RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpParamet
                     er[1]);

TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);
if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE+gbpParameter[1])
{
  TxDString("¥r¥n");
  for(bCount = 0; bCount < 49; bCount++)
  {
    TxD8('[');TxD8Hex(bCount);TxDString("]:");
                     TxD8Hex(gbpRxBuffer[bCount+5]);TxD8(' ');
  }
}

TxDString("¥r¥n¥n Example 6. Go 0x200 with Speed 0x100 -- Any Key
                     to Continue."); RxD8();
gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
gbpParameter[1] = 0x00; //Writing Data P_GOAL_POSITION_L
gbpParameter[2] = 0x02; //Writing Data P_GOAL_POSITION_H
gbpParameter[3] = 0x00; //Writing Data P_GOAL_SPEED_L
gbpParameter[4] = 0x01; //Writing Data P_GOAL_SPEED_H
bTxPacketLength = TxPacket(bID,INST_WRITE,5);
bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

TxDString("¥r¥n¥n Example 7. Go 0x00 with Speed 0x40 -- Any Key to
                     Continue."); RxD8();
gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
gbpParameter[1] = 0x00; //Writing Data P_GOAL_POSITION_L
gbpParameter[2] = 0x00; //Writing Data P_GOAL_POSITION_H
gbpParameter[3] = 0x40; //Writing Data P_GOAL_SPEED_L
gbpParameter[4] = 0x00; //Writing Data P_GOAL_SPEED_H
bTxPacketLength = TxPacket(bID,INST_WRITE,5);
bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

TxDString("¥r¥n¥n Example 8. Go 0x3ff with Speed 0x3ff -- Any Key
                     to Continue."); RxD8();
  gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
  gbpParameter[1] = 0xff; //Writing Data P_GOAL_POSITION_L
  gbpParameter[2] = 0x03; //Writing Data P_GOAL_POSITION_H
  gbpParameter[3] = 0xff; //Writing Data P_GOAL_SPEED_L
  gbpParameter[4] = 0x03; //Writing Data P_GOAL_SPEED_H
  bTxPacketLength = TxPacket(bID,INST_WRITE,5);
  bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

  TxDString("¥r¥n¥n Example 9. Torque Off -- Any Key to Continue.");
                      RxD8();
  gbpParameter[0] = P_TORQUE_ENABLE; //Address of LED
  gbpParameter[1] = 0; //Writing Data
  bTxPacketLength = TxPacket(bID,INST_WRITE,2);
  bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

  TxDString("¥r¥n¥n End. Push reset button for repeat");
  while(1);
}

void PortInitialize(void)
{
  DDRA = DDRB = DDRC = DDRD = DDRE = DDRF = 0;  //Set all port to
                      input direction first.
  PORTB = PORTC = PORTD = PORTE = PORTF = PORTG = 0x00; //PortData
                      initialize to 0
  cbi(SFIOR,2); //All Port Pull Up ready
  DDRE |= (BIT_RS485_DIRECTION0|BIT_RS485_DIRECTION1); //set output
                      the bit RS485direction

  DDRD                                                            |=
                      (BIT_ZIGBEE_RESET|BIT_ENABLE_RXD_LINK_PC|BIT_E
                      NABLE_RXD_LINK_ZIGBEE);

  PORTD &= ~_BV(BIT_LINK_PLUGIN); // no pull up
  PORTD |= _BV(BIT_ZIGBEE_RESET);
  PORTD |= _BV(BIT_ENABLE_RXD_LINK_PC);
  PORTD |= _BV(BIT_ENABLE_RXD_LINK_ZIGBEE);
}

/*
TxPacket() send data to RS485.
TxPacket() needs 3 parameter; ID of Dynamixel, Instruction byte,
                      Length of parameters.
TxPacket() return length of Return packet from Dynamixel.
*/
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength)
{
    byte bCount,bCheckSum,bPacketLength;

    gbpTxBuffer[0] = 0xff;
    gbpTxBuffer[1] = 0xff;
    gbpTxBuffer[2] = bID;
    gbpTxBuffer[3]              =              bParameterLength+2;
                      //Length(Paramter,Instruction,Checksum)
    gbpTxBuffer[4] = bInstruction;
    for(bCount = 0; bCount < bParameterLength; bCount++)
    {
        gbpTxBuffer[bCount+5] = gbpParameter[bCount];
    }
    bCheckSum = 0;
    bPacketLength = bParameterLength+4+2;
    for(bCount = 2; bCount < bPacketLength-1; bCount++) //except
                      0xff,checksum
    {
        bCheckSum += gbpTxBuffer[bCount];
    }
    gbpTxBuffer[bCount] = ~bCheckSum; //Writing Checksum with Bit
                      Inversion

    RS485_TXD;
    for(bCount = 0; bCount < bPacketLength; bCount++)
    {
        sbi(UCSR0A,6);//SET_TXD0_FINISH;
        TxD80(gbpTxBuffer[bCount]);
    }
    while(!CHECK_TXD0_FINISH); //Wait until TXD Shift register empty
    RS485_RXD;
    return(bPacketLength);
}

/*
RxPacket() read data from buffer.
RxPacket() need a Parameter; Total length of Return Packet.
RxPacket() return Length of Return Packet.
*/

byte RxPacket(byte bRxPacketLength)
{
#define RX_TIMEOUT_COUNT2   3000L
#define RX_TIMEOUT_COUNT1  (RX_TIMEOUT_COUNT2*10L)
  unsigned long ulCounter;
  byte bCount, bLength, bChecksum;
  byte bTimeout;

  bTimeout = 0;
  for(bCount = 0; bCount < bRxPacketLength; bCount++)
  {
    ulCounter = 0;
    while(gbRxBufferReadPointer == gbRxBufferWritePointer)
    {
      if(ulCounter++ > RX_TIMEOUT_COUNT1)
      {
        bTimeout = 1;
        break;
      }
    }
    if(bTimeout) break;
    gbpRxBuffer[bCount]                                         =
                      gbpRxInterruptBuffer[gbRxBufferReadPointer++];
  }
  bLength = bCount;
  bChecksum = 0;

  if(gbpTxBuffer[2] != BROADCASTING_ID)
  {
    if(bTimeout && bRxPacketLength != 255)
    {
      TxDString("¥r¥n [Error:RxD Timeout]");
      CLEAR_BUFFER;
    }

    if(bLength > 3) //checking is available.
    {
      if(gbpRxBuffer[0] != 0xff || gbpRxBuffer[1] != 0xff )
      {
        TxDString("¥r¥n [Error:Wrong Header]");
        CLEAR_BUFFER;
        return 0;
      }
      if(gbpRxBuffer[2] != gbpTxBuffer[2] )
      {
        TxDString("¥r¥n [Error:TxID != RxID]");
        CLEAR_BUFFER;
        return 0;
      }
      if(gbpRxBuffer[3] != bLength-4)
      {
        TxDString("¥r¥n [Error:Wrong Length]");
        CLEAR_BUFFER;
        return 0;
      }
      for(bCount = 2; bCount < bLength; bCount++) bChecksum +=
                      gbpRxBuffer[bCount];
      if(bChecksum != 0xff)
```

```
        {
          TxDString("¥r¥n [Error:Wrong CheckSum]");
          CLEAR_BUFFER;
          return 0;
        }
      }
    }
  return bLength;
}


/*
PrintBuffer() print data in Hex code.
PrintBuffer()  needs  two  parameter;  name  of  Pointer(gbpTxBuffer,
                      gbpRxBuffer)
*/
void PrintBuffer(byte *bpPrintBuffer, byte bLength)
{
    byte bCount;
    for(bCount = 0; bCount < bLength; bCount++)
    {
        TxD8Hex(bpPrintBuffer[bCount]);
        TxD8(' ');
    }
    TxDString("(LEN:");TxD8Hex(bLength);TxD8(')');
}

/*
Print value of Baud Rate.
*/
void PrintBaudrate(void)
{
  TxDString("¥r¥n
                      RS232:");TxD32Dec((16000000L/8L)/((long)UBRR1L
                      +1L) );  TxDString(" BPS,");
  TxDString("  RS485:");TxD32Dec((16000000L/8L)/((long)UBRR0L+1L) );
                      TxDString(" BPS");
}


/*Hardware Dependent Item*/
#define TXD1_READY                  bit_is_set(UCSR1A,5)
                      //(UCSR1A_Bit5)
#define TXD1_DATA                   (UDR1)
#define RXD1_READY                  bit_is_set(UCSR1A,7)
#define RXD1_DATA                   (UDR1)

#define TXD0_READY                  bit_is_set(UCSR0A,5)
#define TXD0_DATA                   (UDR0)
#define RXD0_READY                  bit_is_set(UCSR0A,7)
#define RXD0_DATA                   (UDR0)

/*
SerialInitialize() set Serial Port to initial state.
Vide Mega128 Data sheet about Setting bit of register.
SerialInitialize() needs port, Baud rate, Interrupt value.

*/
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt)
{
  if(bPort == SERIAL_PORT0)
  {
    UBRR0H = 0; UBRR0L = bBaudrate;
    UCSR0A = 0x02;  UCSR0B = 0x18;
    if(bInterrupt&RX_INTERRUPT)  sbi(UCSR0B,7);  //  RxD  interrupt
                      enable
    UCSR0C = 0x06; UDR0 = 0xFF;
    sbi(UCSR0A,6);//SET_TXD0_FINISH; // Note. set 1, then 0 is read
  }
  else if(bPort == SERIAL_PORT1)
  {
    UBRR1H = 0; UBRR1L = bBaudrate;
    UCSR1A = 0x02;  UCSR1B = 0x18;
    if(bInterrupt&RX_INTERRUPT)  sbi(UCSR1B,7);  //  RxD  interrupt
                      enable
    UCSR1C = 0x06; UDR1 = 0xFF;
    sbi(UCSR1A,6);//SET_TXD1_FINISH; // Note. set 1, then 0 is read
  }
}

/*
TxD8Hex() print data seperatly.
ex> 0x1a -> '1' 'a'.
*/
void TxD8Hex(byte bSentData)
{
  byte bTmp;

  bTmp =((byte)(bSentData>>4)&0x0f) + (byte)'0';
  if(bTmp > '9') bTmp += 7;
  TxD8(bTmp);
  bTmp =(byte)(bSentData & 0x0f) + (byte)'0';
  if(bTmp > '9') bTmp += 7;
  TxD8(bTmp);
}

/*
TxD80() send data to USART 0.
*/
void TxD80(byte bTxdData)
{
  while(!TXD0_READY);
  TXD0_DATA = bTxdData;
}

/*
TXD81() send data to USART 1.
*/
void TxD81(byte bTxdData)
{
  while(!TXD1_READY);
  TXD1_DATA = bTxdData;
}

/*
TXD32Dex() change data to decimal number system
*/
void TxD32Dec(long lLong)
{
  byte bCount, bPrinted;
  long lTmp,lDigit;
  bPrinted = 0;
  if(lLong < 0)
  {
    lLong = -lLong;
    TxD8('-');
  }
  lDigit = 1000000000L;
  for(bCount = 0; bCount < 9; bCount++)
  {
    lTmp = (byte)(lLong/lDigit);
    if(lTmp)
    {
      TxD8(((byte)lTmp)+'0');
      bPrinted = 1;
    }
    else if(bPrinted) TxD8(((byte)lTmp)+'0');
    lLong -= ((long)lTmp)*lDigit;
    lDigit = lDigit/10;
  }
  lTmp = (byte)(lLong/lDigit);
  /*if(lTmp)*/ TxD8(((byte)lTmp)+'0');
}

/*
TxDString() prints data in ACSII code.
*/
void TxDString(byte *bData)
```

```
{
  while(*bData)
  {
    TxD8(*bData++);
  }
}

/*
RxD81() read data from UART1.
RxD81() return Read data.
*/
byte RxD81(void)
{
  while(!RXD1_READY);
  return(RXD1_DATA);
}

/*
SIGNAL() UART0 Rx Interrupt - write data to buffer
*/
SIGNAL (SIG_UART0_RECV)
{
  gbpRxInterruptBuffer[(gbRxBufferWritePointer++)] = RXD0_DATA;
}
```

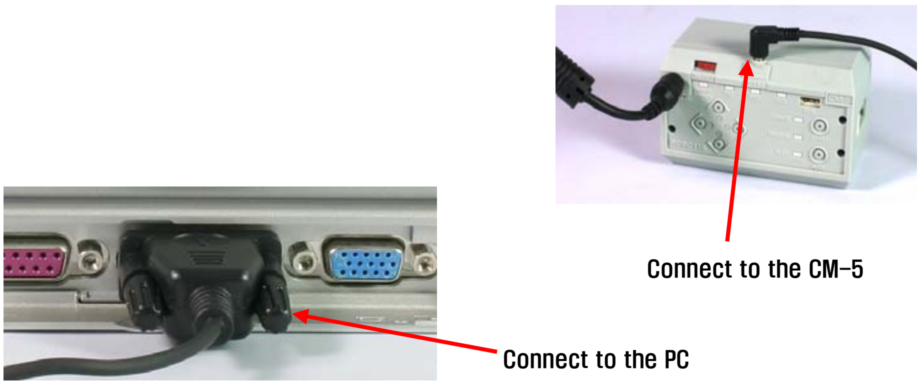
10. Bioloid Program Update

This chapter is about the Bioloid program update. We are going to introduce a way to maintain Bioloid in latest version by showing how to update firmware for CM-5, main controller, and Dynamixel AX-12. We recommend that you visit Robotis site, www.robotis.com, and download the latest version.

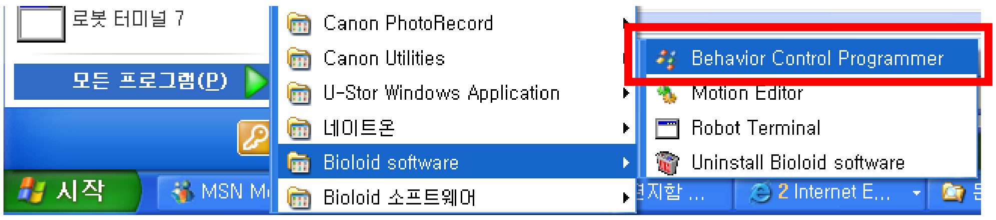
10-1. CM-5 Program Update

CM-5 program is updated through behavior control programmer. Follow the process below to update to the latest version.

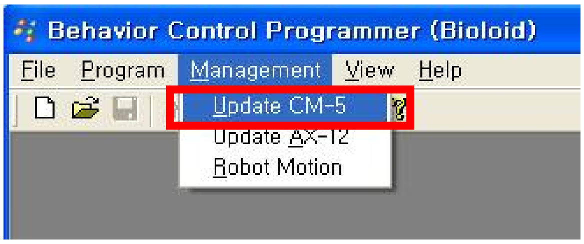
Step 1 As shown below, connect to the PC and CM-5 and turn on the power.



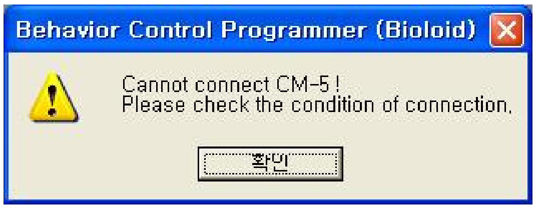
Step 2 Execute the behavior control programmer.



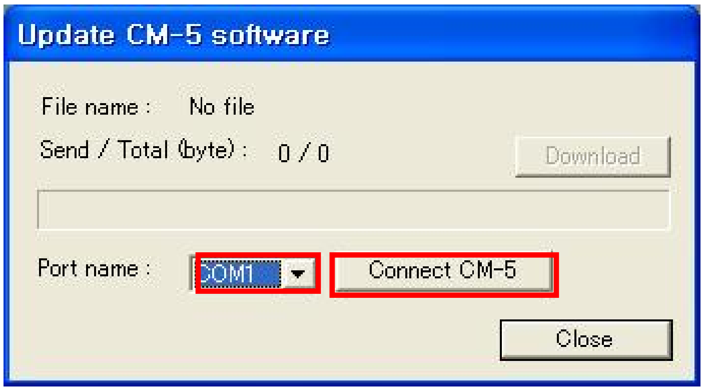
Step 3 From "Manage" menu, select" CM-5 update" as shown below.


Behavior Control Programmer (Bioloid)
File Program Management View Help
Update CM-5
Update AX-12
Robot Motion

Step 4 If you see "Can not connect to CM-5!" message, set "Com port" properly and click "CM-5 connect" button.

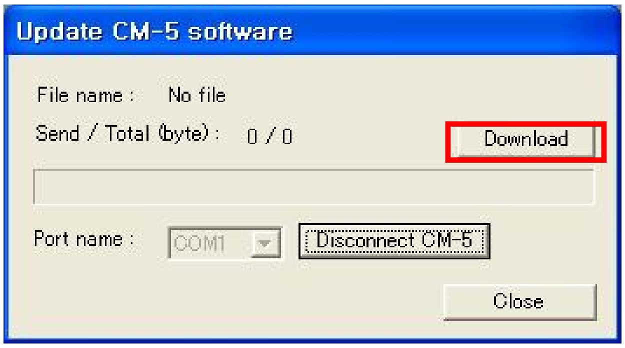

Behavior Control Programmer (Bioloid)

Cannot connect CM-5 !
Please check the condition of connection.


Update CM-5 software

File name : No file
Send / Total (byte) : 0 / 0

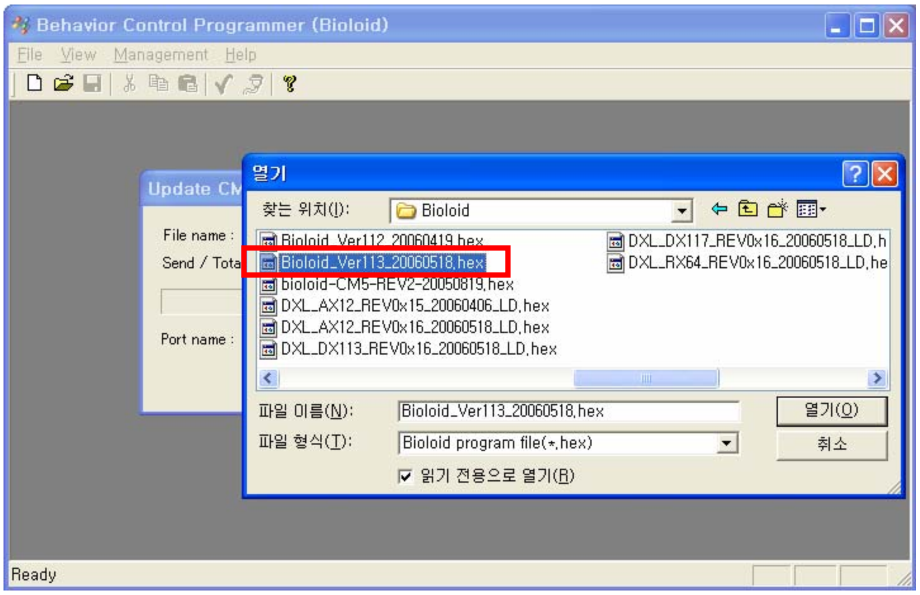
Step 5 After connection, click "Download" button.


Update CM-5 software

File name : No file
Send / Total (byte) : 0 / 0

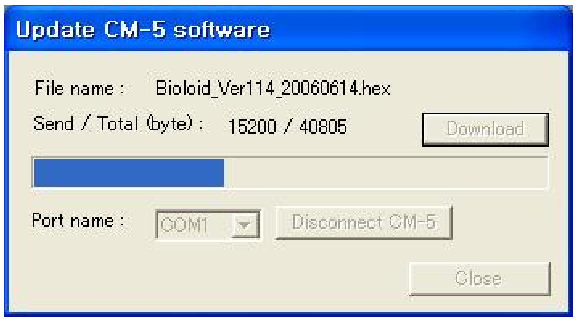
Step 6 Select CM-5 program file.

Go to Robotis homepage, www.robotis.com and download the latest version. We recommend that you periodically check out the homepage for the latest updates.


Behavior Control Programmer (Bioloid)
File View Management Help
Update CM
File name :
Send / Tota
Port name :
열기
찾는 위치(I): Bioloid
Bioloid_Ver112_20060419.hex
Bioloid_Ver113_20060518.hex
bioloid-CM5-REV2-20050819.hex
DXL_AX12_REV0x15_20060406_LD.hex
DXL_AX12_REV0x16_20060518_LD.hex
DXL_DX113_REV0x16_20060518_LD.hex
DXL_DX117_REV0x16_20060518_LD.h
DXL_RX64_REV0x16_20060518_LD.he
파일 이름(N): Bioloid_Ver113_20060518.hex
파일 형식(T): Bioloid program file(*.hex)
열기(O)
취소
☑ 읽기 전용으로 열기(R)
Ready

Step 7 You will see the update progress bar.

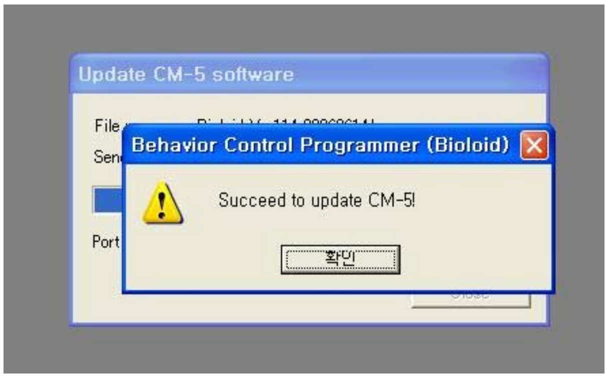
While updating make sure that CM-5 power is not turned off.


Update CM-5 software

File name : Bioloid_Ver114_20060614.hex

Send / Total (byte) : 15200 / 40805

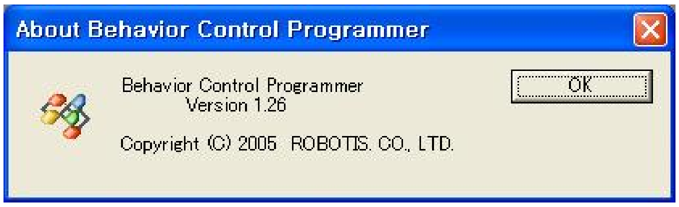
Step 8 If update is successful, you will get the message indicating so, if not, start once again from the beginning. IF CM-5 does not operate properly as a result of update problem, refer to the "9-1. Boot Loader of "9. Information for Advanced Users and update it manually.


Behavior Control Programmer (Bioloid)

Succeed to update CM-5!

10-2. Dynamixel AX-12 Program Update

AX-12 program update is a function that was added from the 1.26 version of "Behavior Control Programmer." Thus, users who are using versions below 1.26 should visit www.robotis.com and download the latest version and install it. To see what version is installed, go to "help -> Behavior Control Programmer Information."


About Behavior Control Programmer

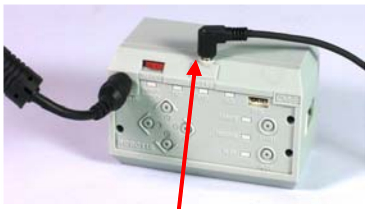
Behavior Control Programmer
Version 1.26

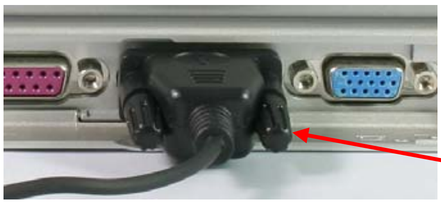
Copyright (C) 2005 ROBOTIS. CO., LTD.

OK

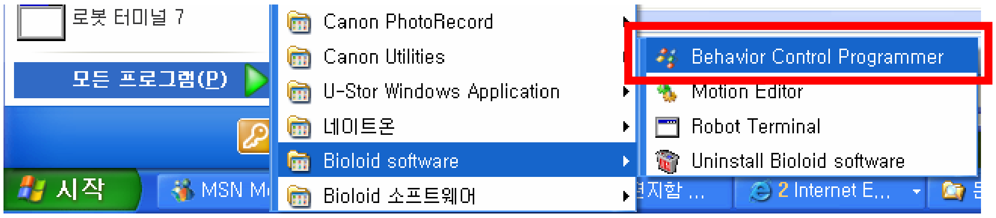
<Note: Make sure that CM-5 is at least Ver. 1.13>
AX-12 program is updated in behavior control programmer. Only versions above Ver 1.13 are applicable. Thus, if you have versions below Ver 1.13, go to www.robotis.com and download the latest CM-5 program and run the CM-5 upgrade by referring to the "10-1. CM-5 Program Upgrade." To see the version of CM-5, refer to the "Robot Terminal" of "8. 1. Setting the ID and Dynamixel Search."

Step 1 As shown below, connect to the PC and CM-5 and turn on the power. At this point, AX-12 must be connected to the CM-5 to update. Also, pre-assembled robot can be connected at this time. AX-12 that will be updated must have ID between 1 and 19. If there is ID that exceed this range, correct it before running update. Also, keep in mind that AX-12 that has redundant ID will not update properly.

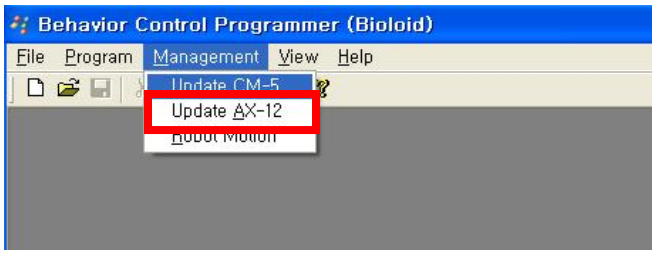




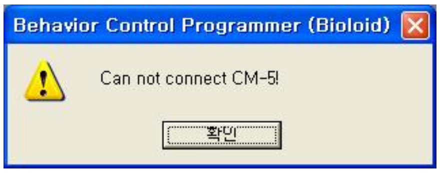
Step 2 Execute the behavior control programmer.



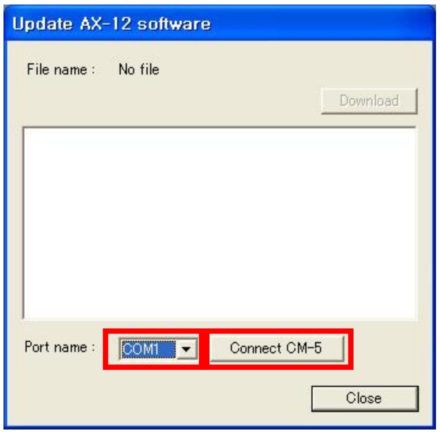
Step 3 From "Manage" menu, select "AX-12 update" as shown below.



Step 4 If you see "Can not connect to CM-5!" message, set "Com port" properly and click "CM-5 connect" button.


Behavior Control Programmer (Bioloid)

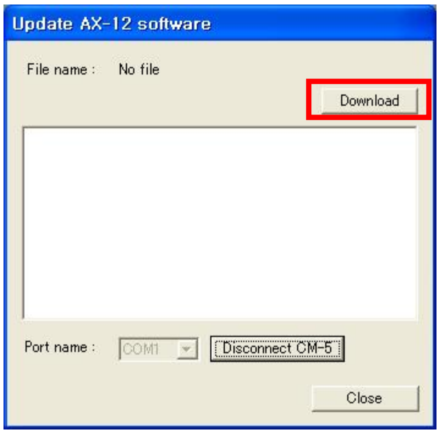
Can not connect CM-5!


Update AX-12 software

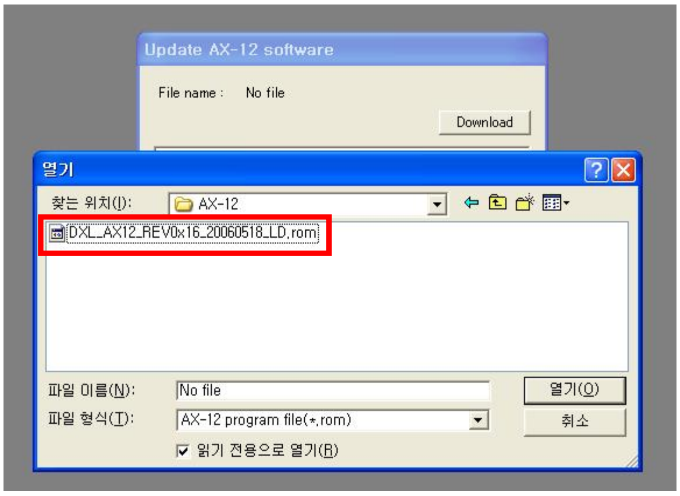
File name : No file

Port name : COM1

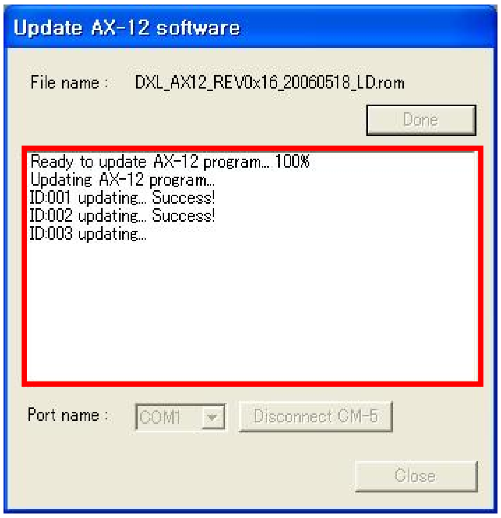
Step 5 After connection, click "Download" button.



Step 6 Select AX-12 program file.
Go to Robotis homepage, www.robotis.com and download the latest version. We recommend that you periodically check out the homepage for the latest updates.



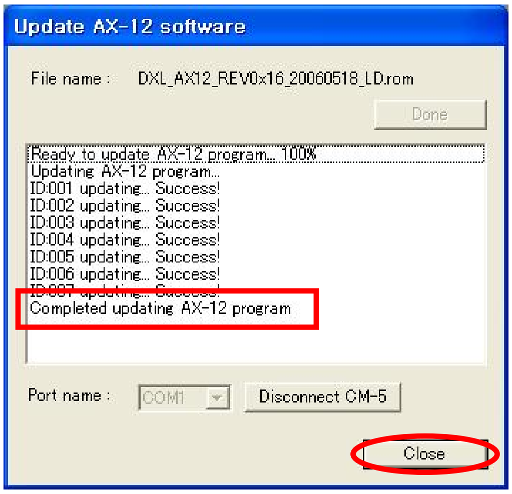
Step 7 You will see the update progress on the print screen window
While updating make sure that CM-5 power is not turned off.


Update AX-12 software

File name : DXL_AX12_REV0x16_20060518_LD.rom

```
Ready to update AX-12 program... 100%
Updating AX-12 program...
ID:001 updating... Success!
ID:002 updating... Success!
ID:003 updating...
```

Step 8 If you get the message indicating update completion, click "close" button to close the AX-12 program update.


Update AX-12 software

File name : DXL_AX12_REV0x16_20060518_LD.rom

```
Ready to update AX-12 program... 100%
Updating AX-12 program...
ID:001 updating... Success!
ID:002 updating... Success!
ID:003 updating... Success!
ID:004 updating... Success!
ID:005 updating... Success!
ID:006 updating... Success!
ID:007 updating... Success!
Completed updating AX-12 program
```