

SONY®

4-659-404-11(1)



# ***AIBO Master Studio***

**ERF-PC03** *English Edition*

**Tutorial Guide**

Action Composer

Action

Motion

Sound

LED

AIBO  
Master  
Studio  
Setup

Behavior Arranger

Behavior

# Table of contents

## Introduction

<b>Before You Start</b> .....	<b>8</b>
About the Manuals .....	8
About the Online Manuals.....	8
Getting prepared .....	8
<b>Organization of the Tutorial Guide</b> .....	<b>9</b>
Let's Create Behaviors! .....	9
Let's Create Actions! .....	10
<b>Files in the Tutorial</b> .....	<b>12</b>
Files of Action Composer .....	12

## Let's Create Behaviors!

<b>Lesson 1 Touch AIBO's Sensor and AIBO stands up, sits down and lies down</b> .....	<b>14</b>
Starting Quick Behavior Arranger .....	14
Creating a sequential-execution program .....	15
Saving the program to the "Memory Stick" .....	16
Running the program on AIBO .....	17
<b>Lesson 2 Press AIBO's Head and It Sits; Say "Start" and It Walks</b> .....	<b>18</b>
Starting Quick Behavior Arranger .....	19
Creating a Stimulus-driven Execution program .....	19
Saving the program to the "Memory Stick" .....	21
Running the program on AIBO .....	21

<b>Lesson 3 Walk and Bow</b> .....	<b>22</b>
Getting ready .....	22
Creating a new behavior .....	23
Making action boxes .....	24
Making a “stand” action box .....	24
Making a “walk and bow” action box .....	26
Connecting action boxes .....	28
Saving the program and running it on AIBO .....	30
Using a wireless LAN .....	30
<b>Lesson 4 AIBO Bows If Head Is Pressed</b> .....	<b>33</b>
Opening the file created in Lesson 3 .....	34
Making boxes .....	35
Connecting the boxes to loop the program .....	38
Saving the program and running it on AIBO .....	41
<b>Lesson 5 Say “AIBO” and AIBO Bows and Greet You</b> .....	<b>42</b>
Preparing an action library .....	43
Collecting boxes together to make a group .....	45
Moving the TOUCH_HEAD? and WAIT_1SEC boxes into a Group .....	45
Editing a branch box and an action box .....	46
Embedding an action file into a program .....	51
Embedding the group and completing the program .....	52
Saving the program and running it on AIBO .....	55
<b>Lesson 6 Doing Several Actions Simultaneously</b> .....	<b>56</b>
Checking the current program .....	57
Executing several actions simultaneously .....	58
Saving the program and running it on AIBO .....	61
<b>Lesson 7 Mastering Advanced Techniques</b> .....	<b>63</b>
Preparing the file .....	63
Checking the main program .....	63
Checking the program in the “Ball Searching” group box .....	64
Changing the action depending on the number of the running command .....	66
Checking the “Kick” group program .....	67
Rearranging the terminals .....	68

## Creating Actions!

<b>Window Organization</b> .....	70
<b>Lesson 1 Let's Create an Action Library!</b> .....	72
Getting prepared .....	73
Creating an action library .....	73
Creating an action .....	74
Adding a supplied action to the library .....	76
Saving to the "Memory Stick" .....	77
Saving the library via a wireless LAN .....	78
<b>Lesson 2 Let's Preview a Created Action!</b> .....	80
Previewing an action and elements on the PC .....	81
Preview the motion .....	82
Preview the sound .....	83
Previewing LEDs .....	84
Testing an action on AIBO .....	85
Checking an action or elements not registered to the action library .....	86
Checking an action .....	86
Checking elements .....	87
<b>Lesson 3 Let's Edit Elements</b> .....	90
Preparing elements .....	91
Editing a motion .....	92
Editing a pose .....	92
Adding a key frame .....	93
Editing a sound .....	94
Editing a sound .....	94
Adjusting the starting point of the sound .....	96
Editing the LEDs .....	96
Adding LED data .....	96
Making the LEDs flash .....	98
Saving the action file .....	100

<b>Lesson 4 Let's Create Original Elements!</b> .....	<b>101</b>
Creating an action file .....	102
Creating a motion .....	102
Creating sounds .....	105
Combining elements .....	107
<b>Lesson 5 Let's Light up the LEDs to the Music</b> .....	<b>108</b>
<b>Lesson 6 Tips for creating actions</b> .....	<b>112</b>
General .....	112
Motion .....	112
Information about Each Model .....	116
Sound: WAVE .....	117
Sound: MIDI .....	117
LEDs .....	118





# Before You Start

The sections below describe the two manuals that are supplied with AIBO Master Studio and the preparations that are necessary prior to use.

## About the Manuals

Two manuals are supplied with AIBO Master Studio.

### Tutorial Guide (this manual)

The Tutorial Guide gives examples of AIBO Master Studio use. First-time users of AIBO Master Studio should try out all the operations by doing the tutorial in order to get acquainted with AIBO Master Studio. In the process, you will learn a variety of useful tips.

### User's Guide

The User's Guide provides an overview of AIBO Master Studio: installation, setup and other information required to operate AIBO, and information on the use of the programs that make up AIBO Master Studio.

#### Note

The illustrations and images in these manuals are mostly of AIBO ERS-210, with some of the AIBO ERS-220 and ERS-310 series.

## About the Online Manuals

The Tutorial Guide and the User's Guide are also available online as PDF files. To view these files, see page 4 of the User's Guide.

## Getting prepared

Have you finished installing AIBO Master Studio and the "Memory Stick"?

If you will be using a wireless LAN, you must set it up in advance. If you have not done the preparations, please refer to "Preparation" on page 17 of the User's Guide.



# Organization of the Tutorial Guide

This Tutorial Guide is divided into the following two parts.

## Let's Create Behaviors!

This part teaches you how to create programs with Quick Behavior Arranger and Behavior Arranger.

### ■ Beginner's Course

---

In Lesson 1 and 2, you will use Quick Behavior Arranger to create a program through the use of wizards.

In Lesson 3, you'll learn the basic operations of Behavior Arranger.

#### **Lesson 1** Touch AIBO's Sensor to Make AIBO Stand, Sit and Lie Down

Using Quick Behavior Arranger, you'll create a program in just a few simple steps that will make AIBO perform a series of specific actions!

#### **Lesson 2** Press AIBO's Head and It Sits; Say "Start" and It Walks

Using Quick Behavior Arranger, you'll create a program that will make AIBO perform different actions when you touch its sensors or say something to it. Since you'll be using wizards, it will be easy!

#### **Lesson 3** Walking and Bowing

In this lesson, you'll learn how to use Behavior Arranger to create a basic program and how to execute the program via a wireless LAN.

### ■ Intermediate Course

---

This course is for users who understand basic Behavior Arranger operations. By creating programs that make use of voice recognition or unique actions, you'll learn how to set branch conditions, create groups, and use parameters.

#### **Lesson 4** Make AIBO Bow when You Press Its Head

In this lesson, you'll edit the program that you made in Lesson 3 to create a program using branch boxes and sensor information. You will also learn how to loop the program so that it executes repeatedly.

#### **Lesson 5** Make AIBO Bow and Greet Your Guests when You Call Out to It

In this lesson, you'll learn how to create groups, use AIBO's voice recognition function, and embed actions made with Action Composer. This sounds a little complicated, but you'll learn how to do all of this step by step!

## ■ Advanced Course

---

By mastering the advanced techniques, you'll be able to create programs that allow AIBO to function as a pet or in games!

### Lesson 6 Executing Several Actions Simultaneously

You'll learn how to program AIBO like a real robot in order to execute several commands at once. You'll create a program in which AIBO changes its walking direction from forward to backward in response to sensory information.

### Lesson 7 Mastering Advanced Techniques

This lesson introduces advanced techniques for creating programs with Behavior Arranger. You'll learn programming methods by following a description of a program where AIBO searches for and plays with the pink ball.

## Let's Create Actions!

In this section, you'll learn how to use Action Composer to create an action library, action files and elements used with Behavior Arranger.

## ■ Beginner's Course

---

You'll learn the basics of Action Composer operations by using and combining supplied elements.

### Lesson 1 Creating an Action Library

In this lesson, you'll learn how to combine the supplied files to create action files and an action library for use with Behavior Arranger.

### Lesson 2 Previewing the Created Action

In this lesson, you'll preview the action file that you created by viewing it on the PC or by having AIBO actually perform the action. When creating an action, it's important to verify the sounds and actions with your own eyes and ears!

### Lesson 3 Editing Elements

You'll learn how to edit elements and action files by modifying the supplied action files.

## ■ Intermediate Course

---

This course introduces you to the creation of elements.

### Lesson 4 Creating Original Elements

You'll learn how to create elements from scratch. Once you've mastered these techniques, you will be a proficient AIBO content creator!

### Lesson 5 Lighting Up LEDs to Music

This lesson takes you further by allowing you to synchronize elements. For instance, have fun by synchronizing AIBO's winks and mouth movements with the sounds that it makes, or its dance movements with music.

## ■ Advanced Course

---

In this course, you'll learn techniques for creating actions with greater effectiveness and higher efficiency.

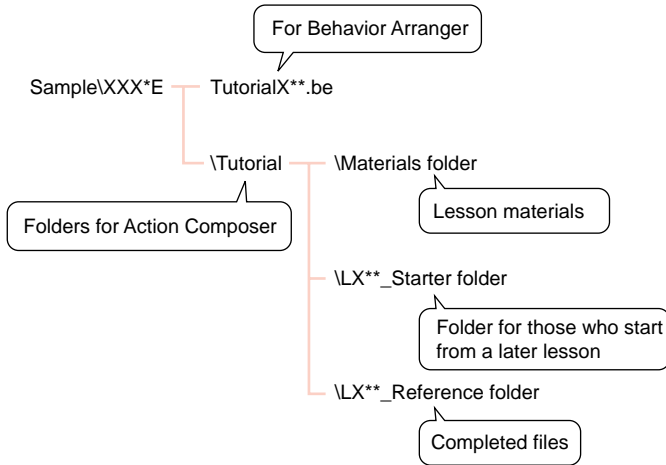
### Lesson 6 Tips on Creating Actions

In this lesson, you'll polish your skills by creating original actions. Tips are presented on how to further enhance the presentation of your AIBO programs.

# Files in the Tutorial

All the files you'll be using are in the "Sample" folder.

\AIBO Master Studio installed folder\Sample



Please do not overwrite the files used in this tutorial so that they can be used again in the future.

\* The folder has a name consisting of your AIBO's model and "E" (Separate folders are provided for each model). The E indicates that English words are recognized.

\*\* Lesson number

## Files of Action Composer

Before you start the tutorial, please copy the "Materials" folder to the My Documents folder, and then use the copied files.

### ■ For those who want to start from a later lesson

Copy the files in the "LX\*\*\_Starter" folder to the My Documents folder and start from the desired lesson.

For instance, if you want to start from Lesson 2, use the "L2\_Starter" file. (Please note that some file names may differ from those used in the lesson.)

### ■ To check what you've created

The finished files (with the same contents as the files that you create during a lesson) are placed in the "LX\*\*\_Reference" folder. Please refer to these files to check that results are correct or not.



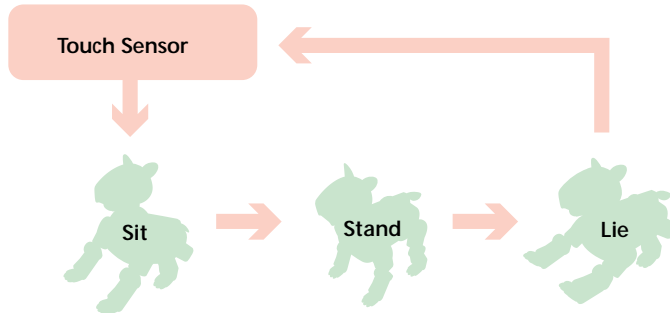
# Touch AIBO's Sensor and AIBO stands up, sits down and lies down

**GOAL:** To learn how to use Quick Behavior Arranger\* to create a sequential-execution program

\* If you use Quick Behavior Arranger, you can create programs easily through the use of wizards. See page 25 of the User's Guide.

## Overview of the program:

When you touch AIBO's sensor, AIBO executes specific actions in sequential order.



## Procedures

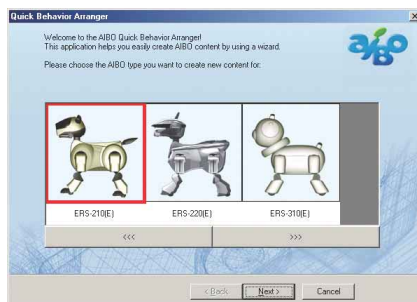
- ① Starting Quick Behavior Arranger (page 14)
- ② Creating a sequential-execution program (page 15)
- ③ Saving the program to the "Memory Stick" (page 16)
- ④ Running the program on AIBO (page 17)

## Starting Quick Behavior Arranger

- 1 Click the Start button in Windows, select Programs - AIBO Master Studio and click Quick Behavior Arranger.

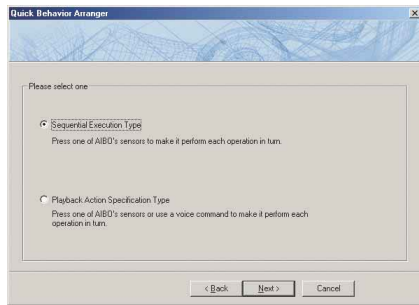
Quick Behavior Arranger starts up.

A window for selecting the AIBO model appears.



## 2 Select the model name and click Next.

A window for selecting the type of program appears.



The (E) indicates that English words are recognized.

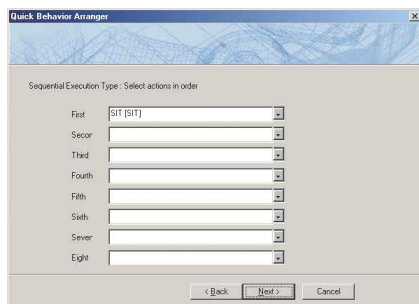
## Creating a sequential-execution program

### 1 Select Sequential Execution Type and click Next.

A window for selecting the actions appears.

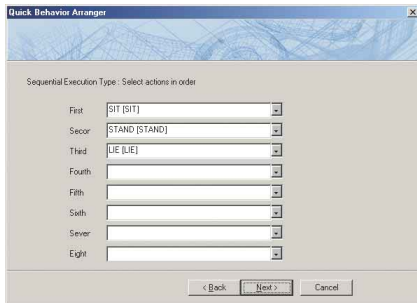
You will learn about stimulus-driven execution programs in Lesson 2.

### 2 Select "SIT" in the first box.



In this step, you've selected the first action that AIBO does when you touch its sensor.

- 3** Select "STAND" in the second box and "LIE" in the third.



In this step, you've selected the action that follows the first action set in step 2.

- 4** Click Next.  
"Program creation is complete!" appears.

## Saving the program to the "Memory Stick".

- 1** Insert the supplied "Memory Stick" into the drive and click Write.

The "Save with a new name" dialog box appears.

- 2** Name the file and save it.

Type "Lesson1.be" as a new file name and save it to the My Documents folder.

This file can be edited with Behavior Arranger.

- 3** Save the program to the "Memory Stick".

A confirmation dialog box appears. Click Yes.



You should make it habit to copy the necessary system files to the "Memory Stick" before you save programs to it. With Quick Behavior Arranger, when you copy your program to the "Memory Stick," the system files are also copied, if they have not been copied yet.

### CAUTION

If the "Memory Stick" drive has not been set, a dialog box appears asking you to set it. Type the drive name in the box. For details, see page 20 of the User's Guide.

When you save the program you've created to the "Memory Stick," it becomes possible to run it on AIBO.



Files saved to the “Memory Stick” cannot be edited with Behavior Arranger.  
After the copying has finished, the next dialog box appears.



#### 4 Click No.

The procedure for creating and saving the program is finished.

In this lesson, you will not edit the program with Behavior Arranger, so click No.

## Running the program on AIBO

Let's run the program you've created on AIBO.

#### 1 Insert the “Memory Stick” into AIBO and start up AIBO by pressing the pause button.

Wait until AIBO finishes booting and stretches his arms.

#### 2 Press any of AIBO's touch sensors.

AIBO sits, stands, and then lies down in the programmed sequence.  
AIBO stops when it has executed all the actions.

You can press any touch sensor.

#### 3 Press the touch sensor again.

The program runs again from the start.

If you press a sensor while AIBO is in motion, it stops. Touch the sensor again and AIBO executes the program from the beginning.

### Did everything go well?

Making a program with Quick Behavior Arranger is easier than you thought.

The Sequential-execution program allows you to make AIBO perform a series of consecutive actions whenever you touch its touch sensor.

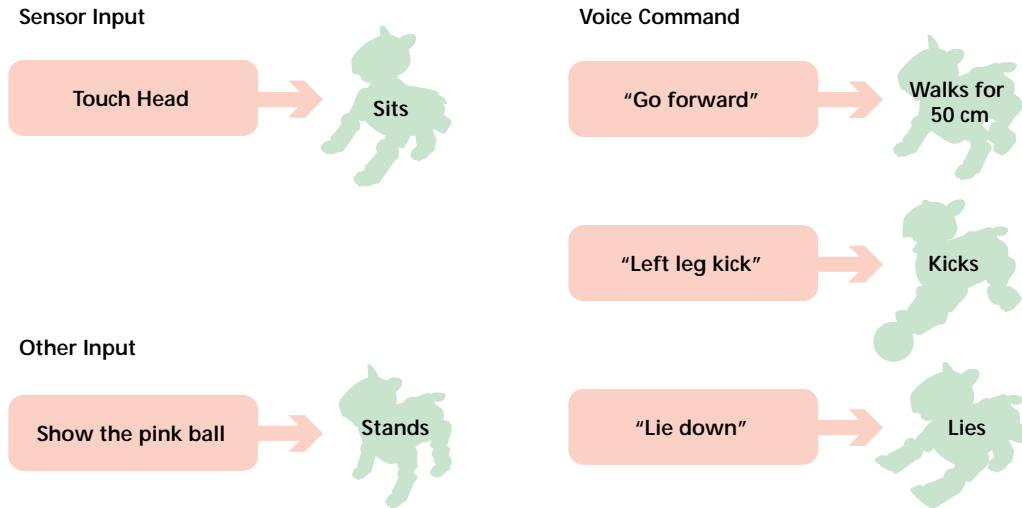
In the next lesson, you'll make a program using another Quick Behavior Arranger tool, the Stimulus-driven Execution program.

# Press AIBO's Head and It Sits; Say "Start" and It Walks

**GOAL:** To learn how to make a Stimulus-driven Execution program using Quick Behavior Arranger

## Overview of the program:

AIBO performs specific actions according to the sensory input or voice command.



## Procedure

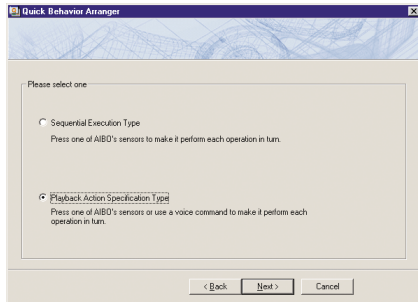
- ① Starting up Quick Behavior Arranger (page 19)
- ② Making a Stimulus-driven Execution program (page 19)
- ③ Saving the program to the "Memory Stick" (page 21)
- ④ Running the program on AIBO (page 21)

## Starting Quick Behavior Arranger

Start up Quick Behavior Arranger and select the AIBO model to be used.

A window for selecting the program type appears.

For details, see Lesson 1 (page 14).



## Creating a Stimulus-driven Execution program

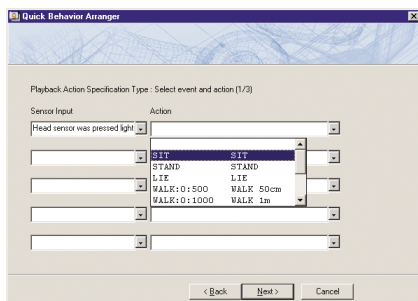
### 1 Select Stimulus-Driven Execution Type and click Next.

A window for selecting events and actions appears.

### 2 Select the items below and click Next.

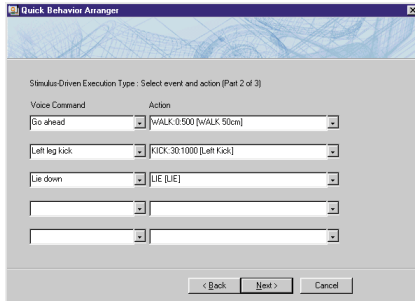
	Sensor Input	Action
ERS-210	Head sensor was pressed lightly	SIT[SIT]
ERS-220	Head sensor was pressed lightly	SIT[SIT]
ERS-310 series	Head is pushed downward	SIT[SIT]

When AIBO detects the input selected in the left box, it performs the action indicated in the right box.



### 3 Select the items below and click Next.

Voice Command	Action
Go forward	WALK:0:500 [walks 50cm]
Left leg kick	KICK:30:1000 [Left kick]
Lie down	LIE [Lies]

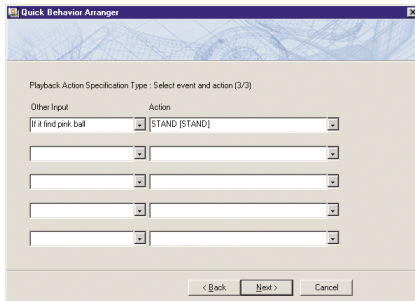


When you say the word indicated in the left box, AIBO performs the action indicated in the right box.

In this step, you've set the following commands.

- “Go forward” → Walks 50 cm
- “Left leg kick” → Kicks
- “Lie down” → Lies down

### 4 Set “if it finds a pink ball” in the “Other Input” box and “STAND” in the “Action” box, and then click Next.



If the condition in the left box are satisfied, AIBO performs the action indicated in the right box.

In this step, you've set the command “if you show a pink ball to AIBO, AIBO sits.”

## Saving the program to the “Memory Stick”

Save the program to the “Memory Stick” as in Lesson 1. Save the file with the file name “Lesson2.be”.

## Running the program on AIBO

Run the program on AIBO and verify that AIBO does what it was programmed to do!

- Press its head\* → sit
- Say “Go forward” → walks 50cm
- Say “Left leg Kick” → kicks
- Say “Lie down” → lies
- Show AIBO the pink ball → stands

With the above procedure, you can program AIBO to perform different actions by sensory input or voice command.

Now you have mastered using Quick Behavior Arranger!  
In the next lesson, you will make another program to learn the basic operations of Behavior Arranger.

See “Running the program on AIBO” in Lesson 1 (page 17).

\* For details on settings, see page 19.

AIBO will not hear your voice while it is making a sound. It is also difficult for AIBO to hear voices when it is moving.

# Walk and Bow

**GOAL:** To learn the following two things.

- Basic Behavior Arranger\* operations and how to create a program that causes AIBO to do a sequence of actions.
- Using a wireless LAN

## Overview of the Program:

When AIBO is activated, it stands up, walks 10 cm (4 inches) and bows.



## Procedure

- ① Preparation (page 22)
- ② Creating a new behavior (page 23)
- ③ Defining commands in action boxes\*\* (page 24)
  - 1 Stand command
  - 2 Walk and bow command
- ④ Connecting the actions (page 28)
- ⑤ Saving and running the program (page 30)
- ⑥ Running the program via a wireless LAN (page 30)

\* Behavior Arranger is a tool for programming AIBO actions by stringing together boxes containing commands. For details, see page 29 of the User's Guide.

\*\* For details on action boxes, see page 32 of the User's Guide.

## Getting ready

- Prepare the "Memory Stick" that you used in Lesson 1 and 2.
- If you use a wireless LAN, set it up in advance. For details, see page 23 of the User's Guide.

A wireless LAN cannot be used with Quick Behavior Arranger.

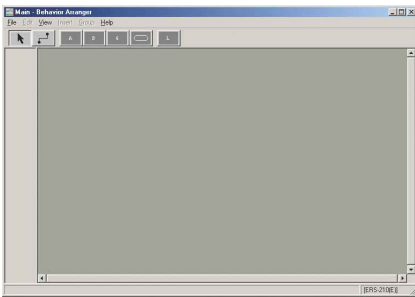
If you started this tutorial from Lesson 3, copy the relevant system files for your AIBO's model to the "Memory Stick."

## Creating a new behavior

In this lesson, you'll create a new behavior from scratch, and then use the behavior to create a program.

- 1 Click the Start button in Windows, select Programs-AIBO Master Studio and click Behavior Arranger.

Behavior Arranger starts up and an editing window appears.



- 2 From the menu, select File-New.

A window for selecting the AIBO model appears.

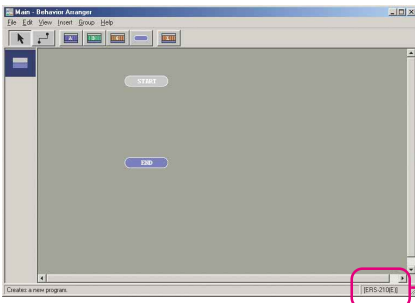
This selection is for creating a new behavior.

- 3 Select the AIBO model to be used.

The START and END terminal boxes\* appear in the window.

For details about the meaning of “(E),” see page 12.

\* For details on terminal boxes, see page 33 of the User's Guide.



The model name that you selected appears here.

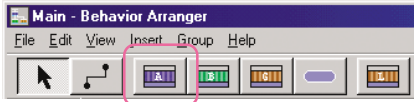
## Making action boxes

Commands are entered in action boxes.

First you'll make a "stand" action box, and then a "walk and bow" action box.

### Making a "stand" action box

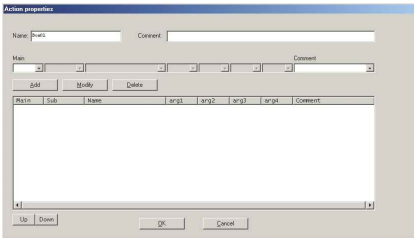
- 1 Click the Action box button .



- 2 Click between the START and END terminal boxes in the window.

The Action box appears where you clicked.

The Action Properties dialog box is what you use to define commands.



- 3 Type a name and comment for the action box.



To make AIBO walk, it must first be in a standing position. For this reason, you'll make a "stand" action box.

The name that you specified in the Name box will be displayed in the action box. (If you enter a name over 7 letters, it will be shortened) It is best to add a comment if the contents are not clearly understood by the name of the action box.



#### 4 Define a command to play back the supplied "STAND" system action.

Sub: ACTION (system action)

Main: PLAY (action)    Name: STAND

#### 5 Click Add to enter the command.

The "PLAY ACTION STAND" command defined in the previous step appears at the bottom.

Main	Sub	Name	arg1	arg2	arg3	arg4	Comment
PLAY	ACTION	STAND					

#### 6 Next, define the following command and click Add.

Main	Time#[ms]
------	-----------

WAIT    (blank)

The "WAIT" command appears at the bottom.

Main	Sub	Name	arg1	arg2	arg3	arg4	Comment
PLAY	ACTION	STAND					
WAIT	ACTION	STAND					

The above command causes the program to wait until AIBO finishes the PLAY-ACTION-STAND (stand) command defined in steps 4 and 5. This means that the next command is not executed until AIBO has finished standing up.

System actions are actions that are provided in AIBO Master Studio.

Selecting PLAY-ACTION allows you to select a system action.

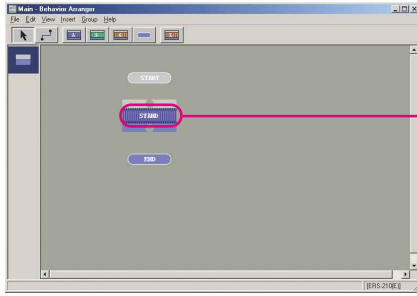
To change the command after clicking the Add button, select the command in the bottom window and enter another one.

Now you've defined a "stand" action.

AIBO sometimes executes several commands simultaneously. As a result, some of the commands are canceled mid-way or AIBO executes an unexpected action. If you want AIBO to postpone executing the next command until the previous one has completed, then make sure the Time#[ms] box of the WAIT command is blank. A blank box is the same as a "0" setting, which in programming means "do not execute the next command until the previous one has completed." It does not mean "wait 0 second" (a zero-second waiting time in programming is the same as immediate execution).

## 7 Click OK.

The Action Properties dialog box closes and a STAND action box appears between the START and END terminal boxes in the window.



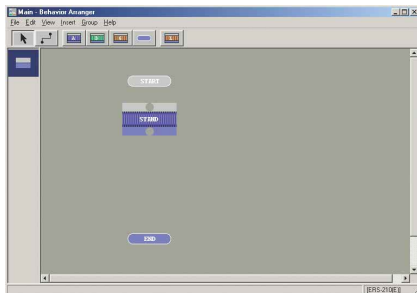
The box name specified in step 4

The STAND action box has now been completed!


## Making a “walk and bow” action box

Let's make another action box.

### 1 Click the END terminal box and drag it downward.



To make more space for another action box between a STAND action box and the END terminal box, move the END terminal box downward.

- 2** Click , and click between the STAND action box and the END terminal box in the window.

The Action Properties dialog box appears.

- 3** Enter a name and a comment for the action box.

Name: WALK\_BOW

Comment: Walks 10cm and bows.

- 4** Define a command for walking straight for 10 cm and click Add.

Main	Sub	Name	arg1	arg2
PLAY	ACTION	WALK	0	100

This commands causes AIBO to walk for 100 mm at the angle of 0° (straight forward).

- 5** Enter a WAIT command and click Add.

Main	Time#[ms]
WAIT	blank

This command causes AIBO postpone executing the next command until it has finished walking 10 cm.

- 6** Define a command for bowing the head (straight downward at an angle of -50°) and click Add.

Main	Sub	Name	arg1	arg2
PLAY	ACTION	MOVE_HEAD	0	-50*

\* Fifty is not included in the list, so type the number into the box.

This command causes AIBO to move its head at the horizontal angle of 0° and a vertical angle of -50°; i.e., AIBO lowers its head 50°.

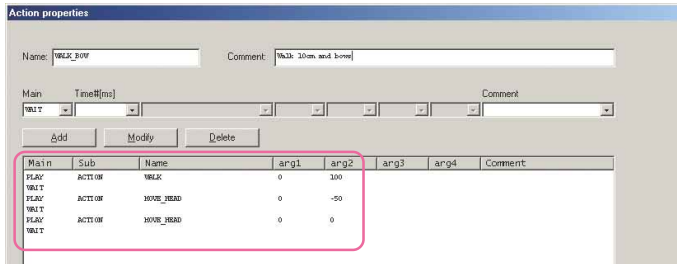
- 7** As in step 5, enter a WAIT command so AIBO does not execute the next command until it finishes lowering its head.

- 8** Define a command for resetting the head to its original position, and then click Add.

Main	Sub	Name	arg1	arg2
PLAY	ACTION	MOVE_HEAD	0	0

This command causes AIBO to move its head at a horizontal angle of 0° and a vertical angle of 0°; i.e., it returns its head back to the original position.

- 9** As in step 5, specify a WAIT command so that AIBO does not execute the next command until it finishes raising its head.



You have used system actions for lowering and raising AIBO's head and making AIBO bow. You could have also have made AIBO bow by creating the action and applying it with Action Composer.

- 10** Click OK.

The "walk for 10 cm and bow" action box has been completed.

Both action boxes are completed.

## Connecting action boxes

Let's connect all the action boxes to make it a series of actions consisting of stand, walk 10 cm, and bow.

- 1** Click the Link mode button .

The mouse changes to link mode.

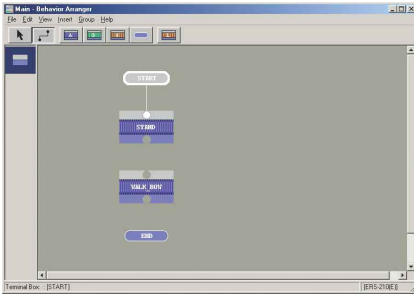


By connecting the boxes containing commands with lines, you link them as a series of consecutive actions.

Till now, you have used the mouse in Normal mode. For connecting boxes, you change the mouse to Link mode. At all other times, it should be kept in Normal mode.

- 
- 2** Click the **START** terminal box and drag it toward the **STAND** action box.

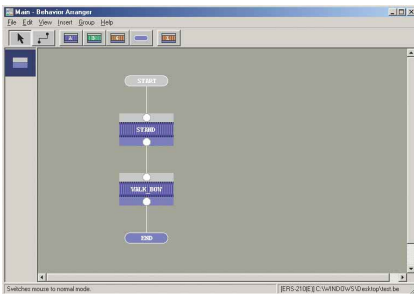
The two boxes are connected with a line.



- 
- 3** Connect the **STAND** and **WALK\_BOW** boxes, as in step 2.

- 
- 4** Connect the **WALK\_BOW** and **END** boxes.

All the boxes are now connected and have become a series of consecutive actions.



In Lesson 3, you created 2 boxes, but the same program can be made by placing all the commands into one action box.

## Saving the program and running it on AIBO

Save the program you've created to the hard disk and then save it to the "Memory Stick".

Afterwards, run the program on AIBO.

---

### 1 From the menu, select "File- Save As."

Save the file as "Lesson3.be".

---

### 2 Insert the "Memory Stick" into the "Memory Stick" drive.

---

### 3 From the menu, select "File-Save to the Memory Stick."

When the saving process is finished, a confirmation dialog box appears. Click OK. Insert the "Memory Stick" containing the program and start up AIBO. The program will run and AIBO will stand up.

To execute the program again, restart AIBO.

Saving the program to the hard disk is convenient since it allows you to use the program again when creating other behaviors.

If you do not use a wireless LAN, then this completes Lesson 3.

## Using a wireless LAN

If you use a wireless LAN, you can wireless transfer the program from the PC to AIBO for execution.

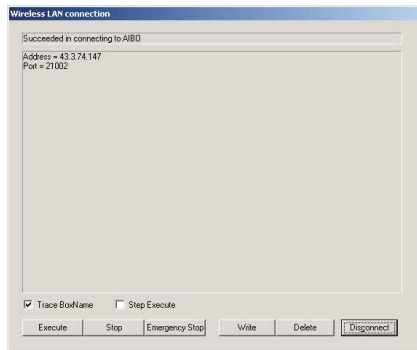
---

### 1 Insert the "Memory Stick" into AIBO, start up AIBO and place it within communication distance of the PC.

By using a wireless LAN, you can execute the program you created in Lesson 3 repeatedly without having to restart AIBO. Moreover, whenever you create or modify a program, you can upload the results to the "Memory Stick" from the PC, even while it is inserted in AIBO.

## 2 From the menu, select "File-Wireless LAN Connection."

The Wireless LAN Connection dialog box appears.

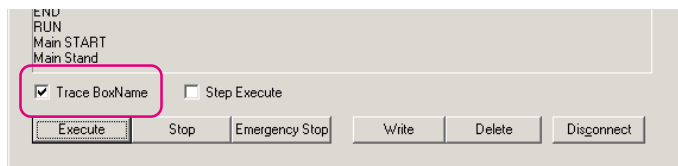


## 3 Click Write.

The program is saved to the "Memory Stick."

## 4 Click Execute.

AIBO performs the actions specified in the program. If you checked the box specifying "Trace the Box Name," the action boxes that have been executed are indicated on the screen.



## 5 When AIBO stops, click Disconnect.

Control returns to the editing window.

Although you have already saved the program to the "Memory Stick," please save it again via the wireless LAN for study purposes.

If you click the Disconnect button while AIBO is in motion, AIBO will continue executing the program until it ends. This is because the program has been downloaded already.

If you need to stop AIBO while it is still moving, click Normal Stop, and then Disconnect. AIBO will stop when it assumes a stable pose.

If you need to stop AIBO immediately, click Emergency Stop, and then Disconnect. AIBO will quit the program immediately. However, it may stop in an unstable pose and fall down.

### The programs you created in this lesson

are the same as those in the "Tutorial3.be" file in the "\Sample\model name" folder. Check these programs to verify that they were created correctly or not.

### Has everything gone well so far?

You may have discovered that creating programs with action boxes is easier than you had expected. You have just learned the basics of programming with Behavior Arranger. Try creating other programs by modifying some of the commands in the action boxes used in this lesson.

Next, you will go on to the Intermediate course. But don't worry! You'll just be editing the basic program that you made in this lesson.

See page 12.

Open the Tutorial3.be file with Behavior Arranger and double-click each box to display the Properties box and verify the command entered in the box.



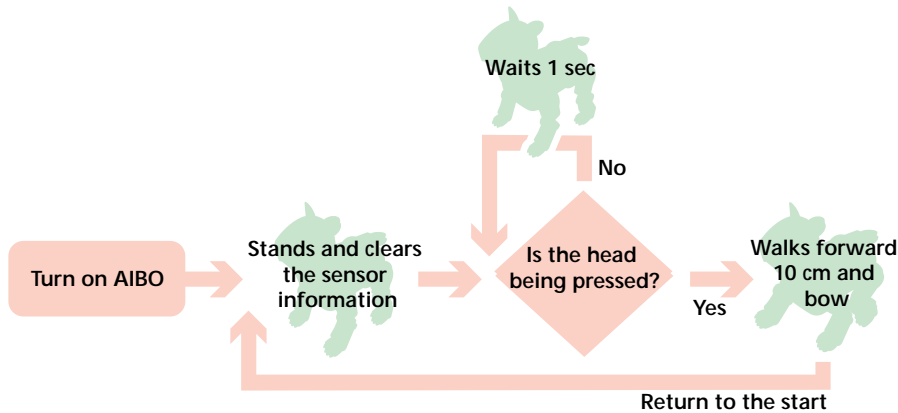
# AIBO Bows If Head Is Pressed

**GOAL:** To learn the following things:

- How to edit a sample program
- How to make a program with branch boxes that define branch conditions
- How to get sensory information
- How to make a loop in your program

**Overview of the Program:**

AIBO stands up, walks forward and bows each time its head is pressed.



**Procedure**

- ① Display the file to be edit (page 34)
- ② Making boxes (page 35)
  - 1 Add a command for initializing the sensors (edit an existing box)
  - 2 Make a branch box\* defining a branch condition
  - 3 Make a box that verifies whether AIBO's head is pressed or not
- ③ Connecting boxes to loop the program (page 38)
- ④ Saving and running the program (page 41)

\* For details on branch boxes, see page 32 of the User's Guide.

As in Lesson 3, prepare the "Memory Stick" and set up the wireless LAN.

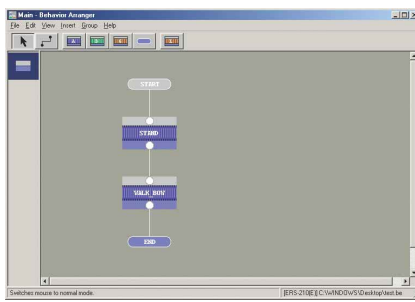
## Opening the file created in Lesson 3

1 Start up Behavior Arranger.

2 Load the program you created in Lesson 3.

Select File-Open from the menu, and then open the “Lesson3.be” file. If you are continuing from Lesson 3, click the Normal mode button.

If you are starting with this lesson, make a copy of the “Tutorial3.be” file in the “\Sample\model name” folder (see page 12).



## Making boxes

In this lesson, you will make and edit 3 boxes as follows.

- ① Add a command to the Stand box for initializing the sensors
- ② Make a branch box for the branch condition “if the head was pressed/if the head was not pressed.”
- ③ Make a box for the “wait for 1 second” command (for obtaining sensory information)

### ① Add a command for initializing the sensors

Add a command to the Stand action box for initializing the sensors.

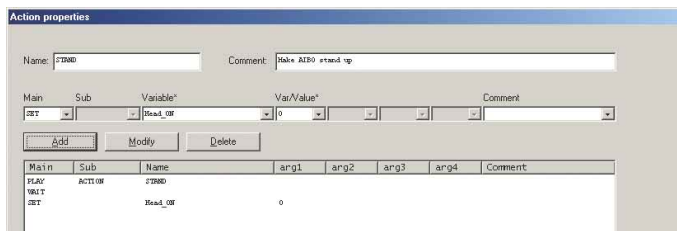
- 1 Click the line between the STAND and WALK\_BOW box, and delete it with the Delete key.

A dialog box asking “Delete immediately?” appears. Click Yes.

- 2 Double-click the STAND action box.

- 3 Add a command for initializing paw and head sensors, and then close the Properties box.

Main	Variable	Var/Value
SET	Head_ON*	0**



If a sensor parameter is set to “1” (the presence of sensory information means the sensor has been touched), it will maintain this value. For this reason, the sensor parameter must be restored to “0” when making a program that loops or when the sensor was previously used by another box. Restoring the sensor parameter to “0” is referred to as “initialization.”

Add a command after the STAND action box to initialize the value of the head sensor.

Adding a command that assigns a value of 0 to the “HEAD\_ON” parameter causes the the head touch sensor to be initialized.

\* The AIBO’s head sensor is referred to differently according to the model. It is the Head sensor on ERS-210, the Head touch sensor on ERS-220 and the Head switch on the ERS-310 series.

The parameter name for all of the above models is “Head\_ON.”

\*\* Head sensor parameter  
1: It was touched  
0: It was not touched

#### 4 Move the WALK\_BOW box and the END boxes downward.

#### ② Setting branch conditions in branch boxes

In this lesson, you'll create a box defining the following branch condition.

"If the head was pressed, AIBO bows. If not, the program jumps to another box."

#### 1 Click the box and place it between the STAND and WALK\_BOW boxes.

#### 2 Enter the box name and command, and then close the Properties box.

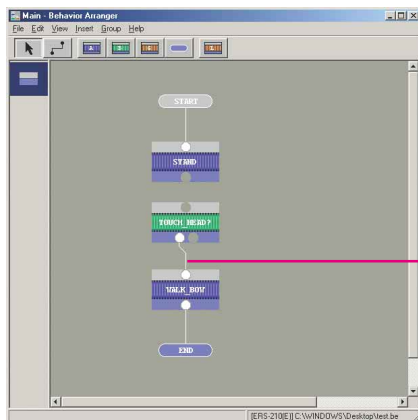
Name : TOUCH\_HEAD?

① Enter the command, "if the head was pressed, jump to the WALK\_BOW action box."

Type	Variable	Op	Var/Value	JumpTo
IF	Head_ON	=	1	WALK_BOW

② Enter the command, "If the head was not pressed."

Type	JumpTo
ELSE	null*



The jump destination is specified in step ①, so the boxes are connected with a line automatically.

To move 2 boxes at the same time, select them by dragging the mouse to enclose them. Then move them by dragging them to make a space between the STAND and WALK\_BOW boxes.

This creates a program that determines whether the head sensor was pressed or not, and then jumps to the next action.

If AIBO's head was pressed (HEAD\_ON=1), jump to the WALK\_BOW box.


If else, jump to null.

\* The box that the program jumps to will be created at a later stage, so select "null," which signifies "not yet determined." When you make a box and connect the two boxes with a line, the "null" will be automatically changed to the name of the connected box.

A branch box displays two terminals since it has two branch conditions.

### ③ Create a "Wait 1 second" action box

To check if the head was pressed or not at one-second intervals, you'll need to create an action box that causes AIBO to wait 1 second.

1 Click  and place a new action box beside the TOUCH\_HEAD? box.

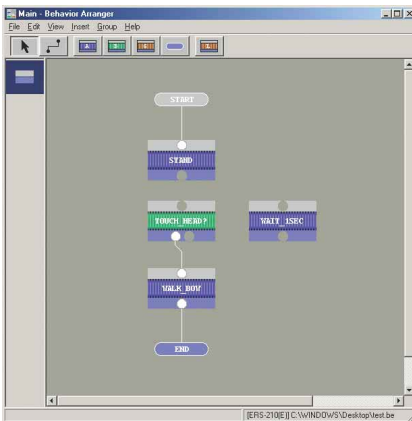
2 Enter a "Wait 1 second" command and close the Properties box.

Name: WAIT\_1SEC

Main	Time#[ms]
------	-----------

WAIT 1000

Wait 1000 ms (1 second).



## Connecting the boxes to loop the program

By connecting the boxes in a loop, you can make the program one continuous motion.

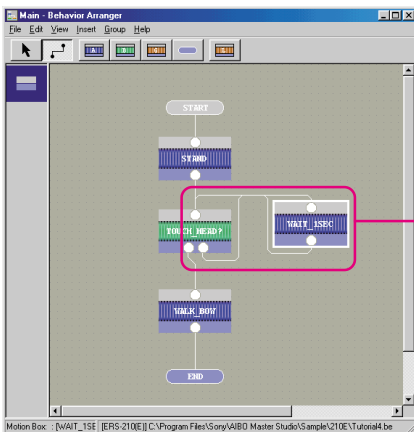
### ① Check the head sensor for input every second

**1** Set the mouse to link mode, and then connect the STAND and TOUCH\_HEAD? boxes.

**2** Connect the lines to make a loop as follows:

- ① Connect the free terminal of the TOUCH\_HEAD? box to the input terminal of the WAIT\_1SEC box.
- ② Connect the output terminal of the WAIT\_1SEC box to the input terminal of the TOUCH\_HEAD? box.

The WAIT\_1SEC action box loops and checks the input to the head sensor every second when AIBO has not received any input.



Checks for input to the head sensor every second.

If the mouse is in normal mode, you can also connect the boxes by clicking and dragging the mouse while pressing the shift key.

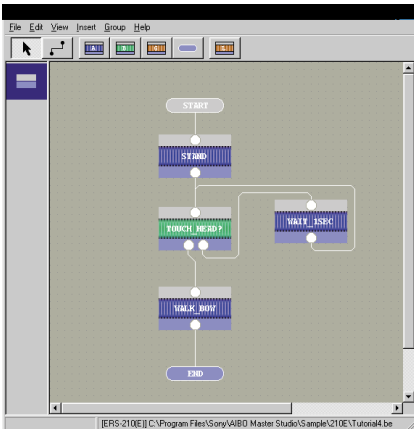
When connecting the branch box to the box that is the jump destination, confirm the condition of the terminals before connecting the box to the correct destination box. The branch condition is displayed when you place the mouse on the terminals. By connecting the empty terminal (ELSE) to WAIT\_1SEC, you've set the following condition: "If the head was not pressed" → Wait for 1 second (jump to the WAIT\_1SEC box)."

In the branch box, you've entered "ELSE-JumpTo:-null." If, however, you open the TOUCH\_HEAD? properties box, you'll find that the "null" has changed to "WAIT\_1SEC," since you connected the empty output terminal of the branch box to the WAIT\_1SEC box.

Type	Variable	Op	Value	JumpTo
IF	Head_ON	=	1	WALK_BOX
ELSE				WAIT_1SEC

### 3 Clarifying crossing lines.

If you cannot see the flow well because of crossing lines, set the mouse to normal mode and move the lines by dragging them.



If there are too many boxes and you cannot see all of them on the screen, select "View-Resize Edit Range" to resize the editing window.

You can select "Reduced View (50%)" or "Reduced View (25%)."

## ② Loop the entire program

Loop the program so that after AIBO walks and bows, the program returns to the start and AIBO bows again when you press its head.

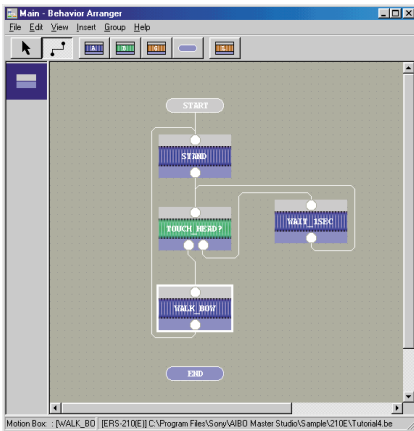
### 1 Disconnect the WALK\_BOW and the END boxes.

See page 35.

### 2 Connect the WALK\_BOW and STAND boxes.

To loop the program, do not connect the End box. Let it stand alone.

By returning to the STAND box after AIBO bows, the sensor is initialized and AIBO waits for input to the sensor again.





## Saving the program and running it on AIBO

Save the program you've created with the new file name "Lesson4.be," save it to the "Memory Stick" and run it on AIBO. When the program is executed, AIBO stands up and waits for its head to be pressed. If you press its head, AIBO walks forward 10 cm and bows. After that, it stands and waits for its head to be pressed again.

To end the program, support AIBO's body with your hands to prevent it from falling down, and press AIBO's pause button to turn it off.

If you are using a wireless LAN, click the Normal shutdown button to stop AIBO's motion and click the Disconnect button.

See page 30.

To stop AIBO immediately, see page 31.

### The program you created in this lesson is

the same as the program in the "Tutorial4.be" file in the "\Sample\model name" folder.

Confirm that the program was created correctly or not.

See page 12.

In this lesson, you created a slightly complicated program containing a branch condition. Change the commands in the branch box to set various conditions. You will have more fun with AIBO by programming it to do many behaviors!

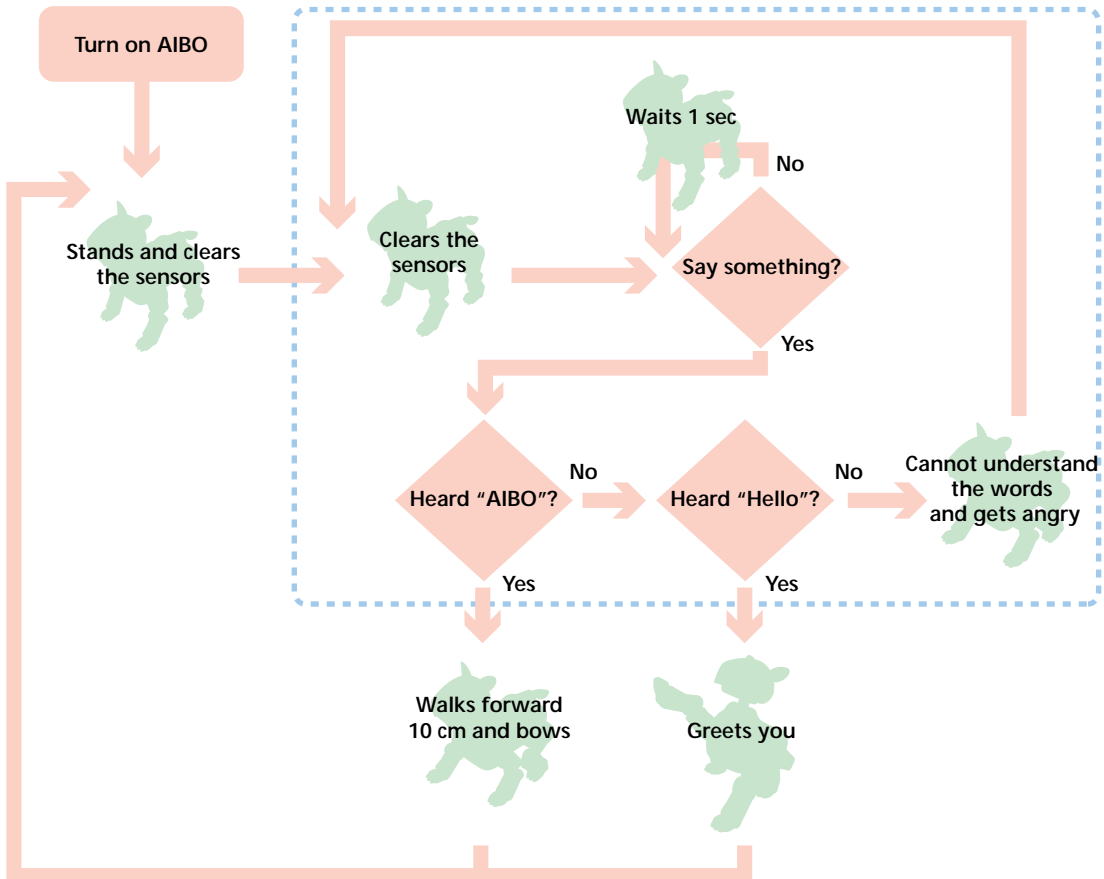
# Say “AIBO” and AIBO Bows and Greets You

**GOAL:** To learn the following points:

- How to make a group
- How to make a program with the voice recognition function
- How to embed an action made by Action Composer into a program

**Overview of the Program:**

After AIBO stands, it performs different actions depending on the words spoken to it.



**Procedure**

- ① Preparing an action library of actions made by Action Composer (page 43)
- ② Making a group (page 45)
  - Arranging boxes as one group
  - Editing an action box and a branch box
  - Adding an action file
- ③ Embedding a group into the main program (page 52)
- ④ Saving the program and running it on AIBO (page 55)

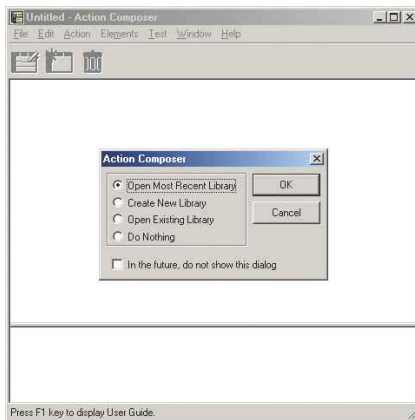
## Preparing an action library

To use an action library through Behavior Arranger, you will need to save the action library to the “Memory Stick” in advance. In this lesson, you’ll use the supplied action library.

To use new elements or action files that you’ve created with Action Composer, you will need to add them to an action library first, and then save the action library to the “Memory Stick.” For details, see page 38 of the User’s Guide.

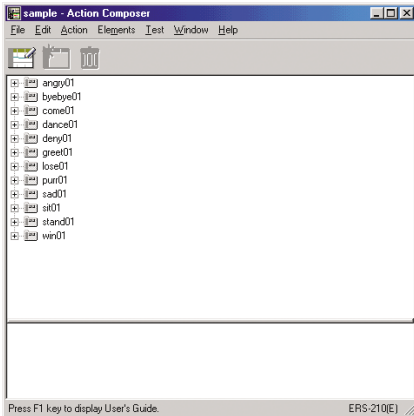
- 1 Click the Start button in Windows, select “Programs-AIBO Master Studio” and click “Action Composer.”

Action Composer starts up.



- 2 Select “Open Existing Library” and click OK.

- 
- 3** Open “Sample.alb” in the folder with the same model name and behavior.



The action library is stored in the “Sample\model name” folder (see page 12).

The model name and language for the behavior and the action library must be the same.

When you save an action library to the “Memory Stick,” a copy is saved to a certain directory in the hard disk to allow Behavior Arranger to reference it.

Behavior Arranger can reference only a single library, and only one library can be saved to the “Memory Stick”.

If another action library is already saved on the “Memory Stick,” the previous library will be overwritten.

- 
- 4** Insert the “Memory Stick” into the drive, and select “File-Save Library to Memory Stick.”

The Action library is saved to the “Memory Stick”.

**Note**

Behavior Arranger refers to the action library saved last. If you are using several “Memory Sticks,” make sure the action library in the “Memory Stick” you are using matches the one on the hard disk.

- 
- 5** Quit Action Composer.

- 
- 6** Start up Behavior Arranger and open the behavior file (Lesson4.be) that you created in Lesson 4.
- 

If you are starting from this lesson, make a copy of the following file for editing: “Tutorial4.be” file in the “Sample\model name” folder (see page 12).

## Collecting boxes together to make a group

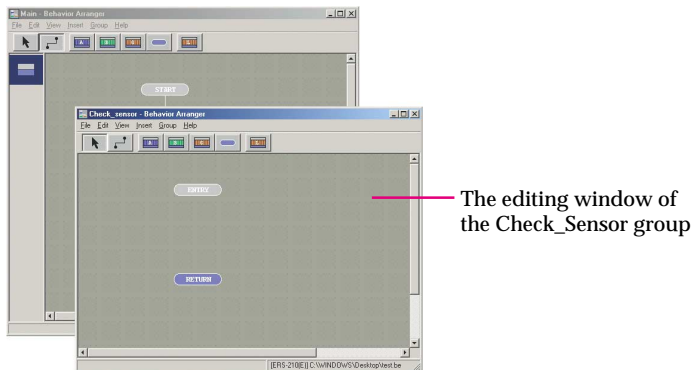
In this lesson, you'll learn how to use group boxes.\* Put the TOUCH\_HEAD? and WAIT\_1SEC action boxes together to make a group.

### Moving the TOUCH\_HEAD? and WAIT\_1SEC boxes into a Group

**1** Click "Group-Create New Group."

**2** Specify "Check\_sensor" in the Name box, and click OK.

A "Check\_Sensor" group is created and an editing window for the Check\_Sensor group appears.



**3** Drag the RETURN box downward.

Make space.

**4** In the Main window, while holding down the Ctrl key, click the TOUCH-HEAD? box, the WAIT\_1SEC box, and the two lines connecting the two boxes in order to select them.

**5** From the menu, select "Edit-Cut."

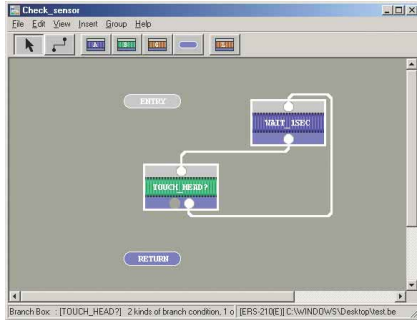
The selected boxes and lines are cut.

\* For details on group boxes, see page 43 of the User's Guide.

By putting several boxes into a group, you will have the convenience of using those boxes as a group in other programs.

## 6 In the Check\_Sensor window, select Edit-Paste.

The selected boxes and lines move to the Check\_Sensor group.



You cannot move the selected boxes or lines by dragging them to other windows.

## Editing a branch box and an action box

Let's edit the branch box and action box that you moved as indicated below.

- ① Change the command of the TOUCH\_HEAD branch box
- ② Add a command for initializing the voice recognition parameter
- ③ Make a branch box that branches to two boxes by voice recognition

### ① Change the command in the TOUCH\_HEAD branch box.

Let's change the command in the TOUCH\_HEAD branch box to one that causes control to exit from the loop when voice input is recognized.

Using the English system, AIBO recognizes 53 English words. You can make programs that use the voice recognition function by creating actions that react to these words. For a list of the words, see page 59 of the User's Guide.

## 1 Open the TOUCH\_HEAD branch box.

## 2 Change the name of the box and command in the Properties box.

- ① Change the name to "VOICE\_ID?"
- ② Select and change the command by clicking the displayed "IF Head\_ON = 1."

Variable	Op	Var/Value	JumpTo
AU_Voice*	=	1	null

- ③ Click Modify

The command, "IF Head\_ON = 1 WALK\_BOW" is changed to "IF AU\_Voice = 1".

\* "AU\_Voice" is a voice recognition parameter that indicates a word was recognized.

The command changes to "if AIBO recognizes any of the 53 words, jump to an undefined box." (For details on nulls, see page 36.)

If "auto modify" is checked, skip step ③.

---

### 3 Close the Branch Properties box by clicking OK.

---

#### ② Initializing the parameter for voice recognition

Before checking whether the voice input has been recognized or not in the VOICE\_ID? branch box, let's make another box that initializes the parameter.

When a parameter value of "1" has been detected, AIBO maintains the condition (i.e., voice recognition = 1 (voice recognition has occurred)).

### 1 Place a new action box above the VOICE\_ID? box.

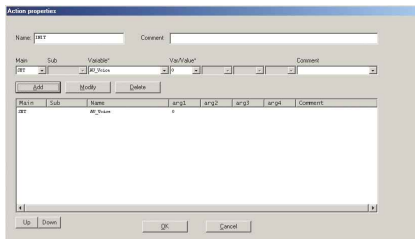
---

### 2 Enter a command for initializing the parameter, and then close the Properties box.

Name: INIT

Main	Variable	Var/Value
SET	AU_Voice	0

For details on initialization, see page 35.



The box for initializing the voice recognition parameter is completed.

---

### ③ Branching to two terminal boxes, depending on the voice recognition

In this lesson, you'll make an AIBO terminal box and a Hello terminal box. The program jumps to the AIBO terminal box when AIBO hears "AIBO," and to the Hello terminal box when AIBO hears "Hello." When AIBO hears another word, the program loops to the top of the group box.

#### 1 Double-click the RETURN box.

The Terminal Properties box appears.

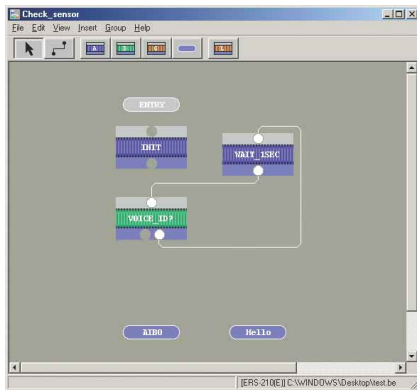
#### 2 Change the name to "AIBO" and close the Properties box.

The terminal box becomes the AIBO terminal box.

#### 3 Click the Terminal box button and place the box next to the AIBO box.

#### 4 Type "Hello" in the Name box and close the Properties box.

The Hello terminal box is created.



#### 5 Place a new branch box below the VOICE\_ID? box.

The program that you make will do the following:  
Check for voice recognition every 1 second



- Recognizes "AIBO" → jump to the AIBO box
- Recognizes "Hello" → jump to the Hello box
- Recognizes another word → loop (jump to the INIT box)

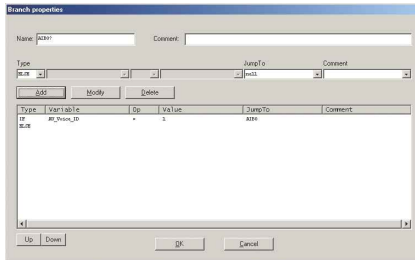
In this box, you'll enter the command, "if AIBO recognize "AIBO," the program jumps to the AIBO terminal box."



**6** Enter the name and command, and then close the Properties box.

Name: AIBO?

Type	Variable	Op	Var/Value	JumpTo
IF	AU_Voice_ID	=	1	AIBO
ELSE				null



Recognizes “AIBO” → jump to AIBO terminal box

“AU\_Voice\_ID=1” means “word detected by voice recognition=1 (AIBO)” (recognizes the word as “AIBO”).

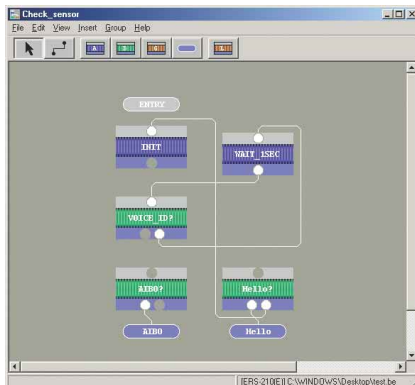
Another word → jump to the undefined box

**7** Place a new branch box next to the AIBO? box.

**8** Enter the name and command, and then close the Properties box.

Name: Hello?

Type	Variable	Op	Var/Value	JumpTo
IF	AU_Voice_ID	=	6	Hello
ELSE				INIT



In this box, you'll enter the command, “if AIBO recognize “Hello,” the program jumps to the “Hello” terminal box.”

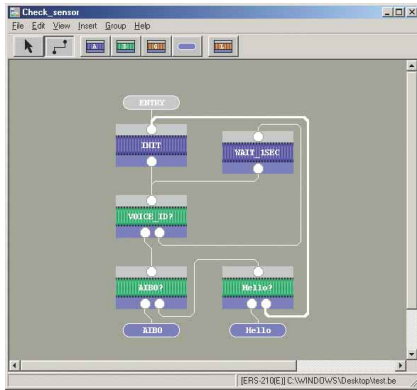
“AU\_Voice\_ID=6” means “word detected by voice recognition =6 (Hello)” (recognizes the word as “Hello”).

To loop the program when it recognizes a word other than “AIBO” or “Hello,” enter the command, “Another word → jump to INIT box.”

## 9 Connect the boxes.

Connect the boxes as follows:

ENTRY → INIT, INIT → VOICE\_ID?, the empty terminal of  
VOICE\_ID? → AIBO?, the empty output terminal of AIBO? →  
Hello?



To connect the AIBO? and Hello? boxes, you must connect the empty output terminal of AIBO? to the empty input terminal of Hello?. If you try to connect them by selecting the boxes, the Condition window appears. Click Cancel. Connect the terminals, not the boxes by clicking on the empty terminal of the AIBO box and dragging the mouse to the terminal of the Hello box.

You have now created a group box with the command “if the word “AIBO” or “Hello” is recognized, jump to the AIBO box or Hello box”.

If AIBO recognizes another word, it clears the “AU\_Voice” parameter and then checks the voice input every 1 second.

## Embedding an action file into a program

In this lesson, you'll learn how to embed an action file into a program using the action library that you previously saved (page 43). By doing so, you will give richer expression to AIBO's movements.

In the following program, you will make AIBO express anger whenever it recognizes a word other than "AIBO" or "Hello."

**1** Place a new action box to the lower right of the Hello? box.

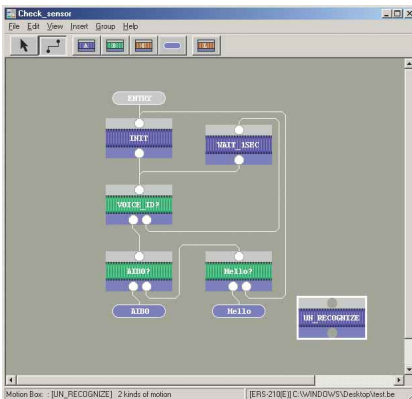
**2** Enter the name and command, and then close the Properties box.

Name: UN\_RECOGNIZE

Main	Sub	Name
PLAY	ACTION+*	angry01
Wait	blank**	

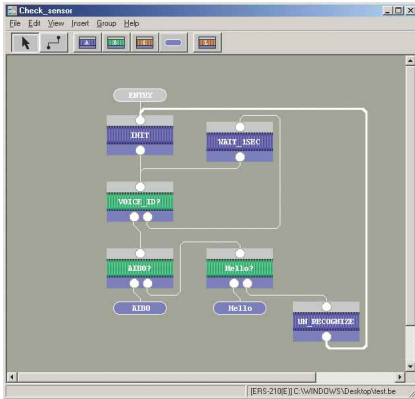
\* "ACTION+" indicates an action file in the action library. PLAY-ACTION+ allows actions in the action library to be used. In this step, you'll embed the "Angry01" action file.

\*\* The program waits until AIBO finishes performing the action (see page 25).



### 3 Reconnect the boxes and reorganize the lines.

Disconnect the “Hello?” and “INIT” boxes.  
Then connect the right output terminal of “Hello?” to  
“UN\_RECOG...,” and “UN\_RECOG...” to “INIT.”



ELSE → jump to INIT box  
Unrecognizable word → execute  
angry01 → jump to INIT box

### 4 Close the group window.

A group is not erased even after  
you have closed the window.

The Check Sensor group has been completed.

The group you just created is not  
displayed in the editing window.  
You'll embed the “Check\_Sensor”  
group into the program by using a  
group box in the next section.

## Embedding the group and completing the program

Return to the initial editing window to complete the program, then  
embed the group program that you created into the main program.

What you created in the beginning  
is also a kind of group, called the  
Main group. You'll embed the  
group you created into the Main  
group to complete the following  
action:

### 1 Click the Group box button and place it between the STAND and WALK\_BOW boxes.

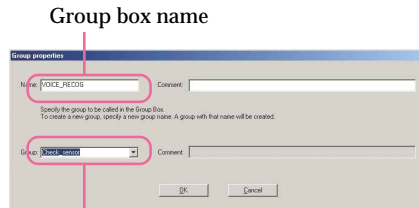
The Group Properties box appears.

“AIBO” → bows  
“Hello” → greets you by raising  
its right front hand  
Neither of the above words →  
AIBO cannot recognize word, and  
thus assumes an angry pose

↓  
Checks for word input every  
second

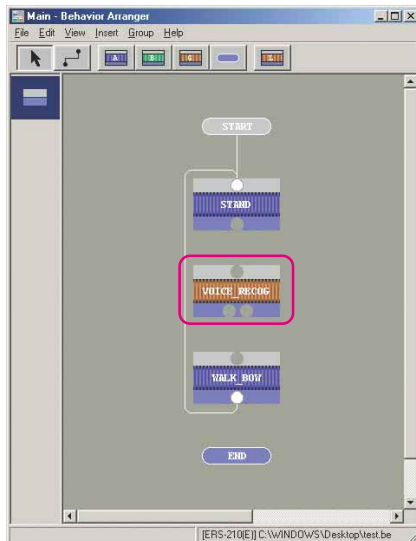
## 2 Enter the name of the group box, select the name for the group that the box invokes, and then close the Properties box.

Type “VOICE\_RECOG “ in the Name box and select “Check\_Sensor” (the name of the group you created) from the list in the Group box.



Name of group invoked by this box

The group box appears.



The “group box name” and “group name” are different. The “group box name” is the name displayed on the box, and the group name is the name of the group that the group box invokes.

The group box has two terminals since it invokes a group that has two terminals.

One is the output terminal of the AIBO terminal box, and the other the output terminal of the Hello terminal box.

Confirm that the names of the terminals are “AIBO” and “Hello” by positioning the mouse on the terminals.

### 3 Connect the AIBO terminal of the VOICE\_RECOG to the WALK\_BOW box.

You've now created a program that causes AIBO to walk and bow when it hears the word "AIBO."

### 4 Place a new action box to the right of the WALK\_BOW box.

Enter the "greet" action command (which is executed when AIBO hears and recognizes the word "Hello."). You'll use the "Greet01" action file in the action library (see page 43) for the greeting by AIBO.

### 5 Enter the box name and command, and then close the Properties box.

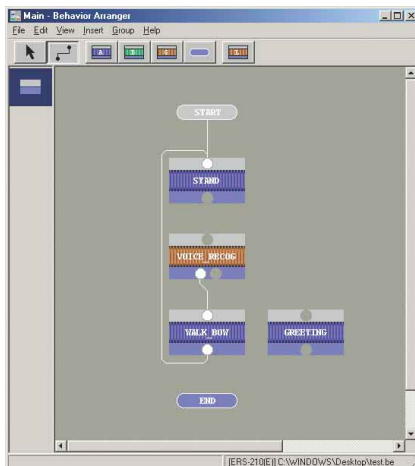
Name: GREETING

- ① Enter the "greeting" command on the first line.

Main	Sub	Name
PLAY	ACTION+	greet01

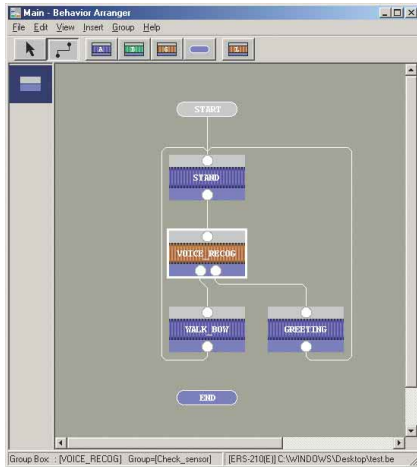
- ② Enter the "WAIT 0" command on the second line.

See page 25.



## 6 Connect the boxes and reorganize the lines.

Connect `STAND` → `VOICE_RECOG`, and the “Hello” terminal of `VOICE_RECOG` → `GREETING`, and `GREETING` → `STAND`.



When AIBO recognizes the word “Hello,” it greets you. The program then returns to the beginning and AIBO stands, waiting for another word input.

If the flowchart is hard to follow, rearrange the lines or boxes.

## Saving the program and running it on AIBO

Save the program that you created to the “Memory Stick” as “Lesson5.be” and run it on AIBO.

When the program is executed, AIBO repeats the following actions.

- Stand → recognizes “AIBO” → walks for 10 cm and bows
- recognizes “Hello” → greets you by raising its right front leg
- recognizes another word → gets angry! Poses

### The program you’ve created in this lesson is

The same as the program in the “Tutorial5.be” file in the “\Sample\model name” folder.

Confirm that the program was properly created or not.

### You’ve just completed a complex program!

It might have been difficult because of all the steps, but you learned many techniques that will allow you to make full use of Behavior Arranger. Make use of the program that you created by changing the embedded actions or walking distance.

See page 30.

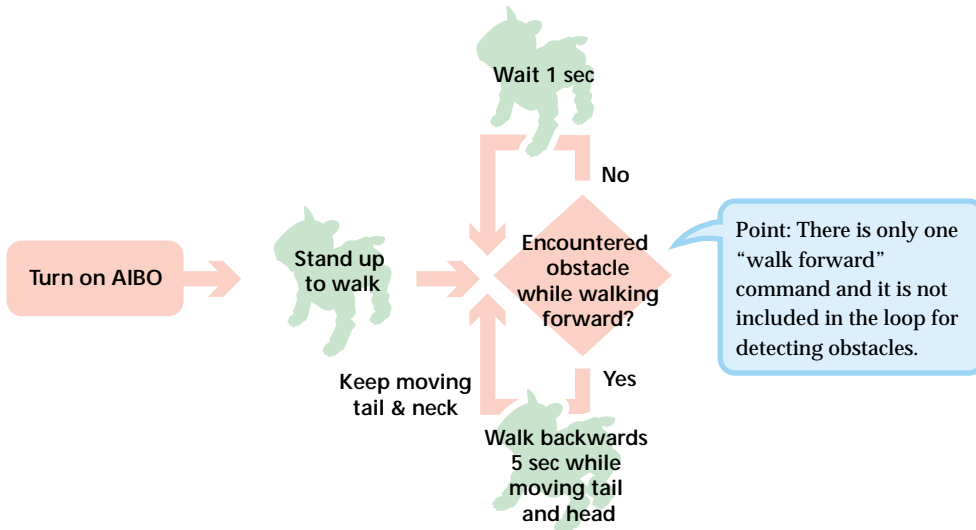
See page 12.

# Doing Several Actions Simultaneously

**GOAL:** To create a program that does several actions simultaneously.

**Overview of the program:**

When AIBO finds an obstacle, it walks backwards for 5 seconds while moving its tail and head. It then walks forward again.



**Procedure**

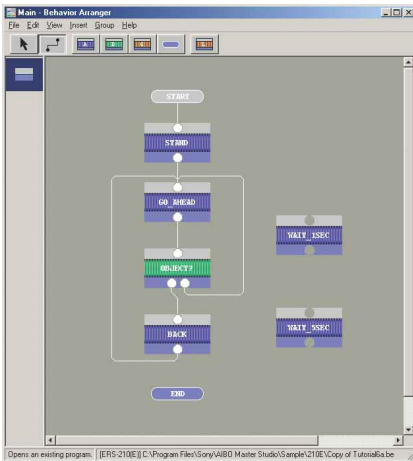
- ① Checking the current program (page 57)
- ② Creating a program that does several actions simultaneously (page 58)
  - Changing the action of the program from "go forward" to "search for obstacles while walking forward"
  - Looping the program
  - Adding two actions: twirling the tail (ERS-210 only) and moving the head sideways
- ③ Saving and running the program (page 61)



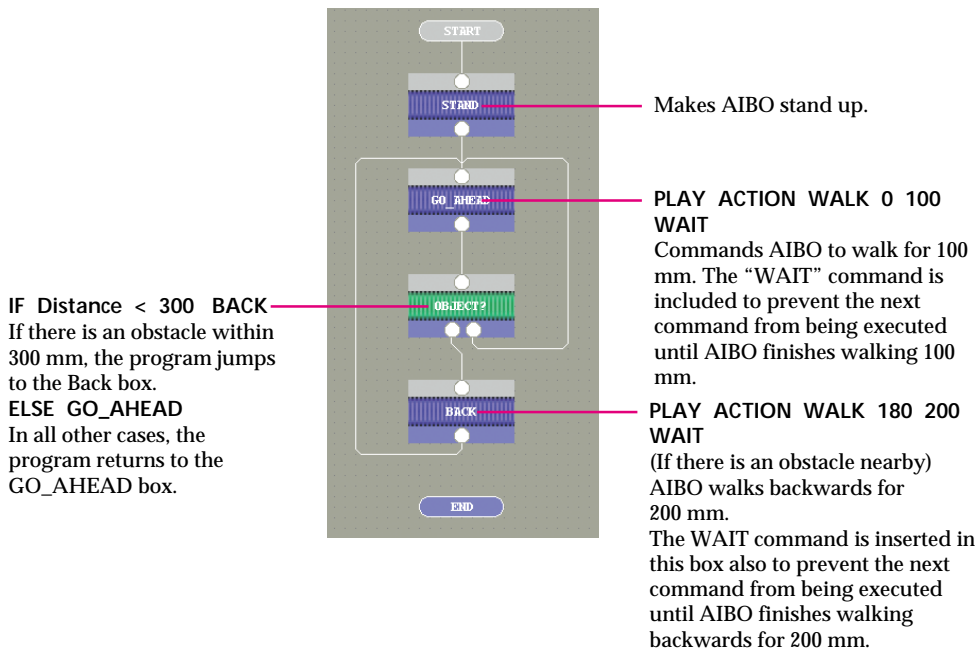
## Checking the current program

In this lesson, you'll edit the "Tutorial6a.be" file, so make a copy of the file and open it with Behavior Arranger.

The "Tutorial6a.be" file is stored in the "Sample\model name" folder (see page 12).



In your program, the commands will be executed sequentially (not simultaneously).



As shown above, the program is designed not to execute the next command until the previous command has been completed.

## Executing several actions simultaneously

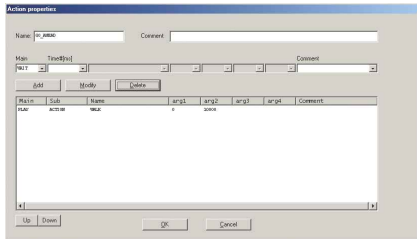
Let's modify the existing program so that several commands are executed simultaneously, such as AIBO searching for obstacles while walking.

**1** Open the "GO\_AHEAD Action Box" Properties box.

**2** Change the walking distance.

Change the distance in the PLAY-ACTION-WALK-0-100 command from 100 mm to 10,000 mm.

**3** Click WAIT, Delete, and then the OK button.



Next, we'll loop the program so that AIBO checks for obstacles every second when none has yet to be detected.

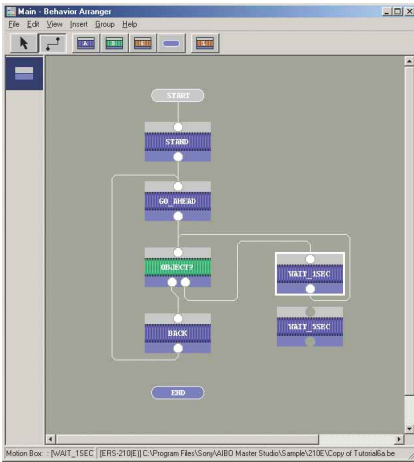
**4** In the "OBJECT? Action Box" Properties box, change "Else GO\_AHEAD" to "Else WAIT\_1SEC," and then close the Properties box.

AIBO continues walking since a long distance (10,000 mm) is specified.

Since the WAIT command has been deleted, the commands in "OBJECT?" box are executed right after the "walk forward for 10 cm" command is executed. AIBO then searches for obstacles as it walks forward.

In the box, the jump destination has been changed from "GO\_AHEAD" to "WAIT\_1SEC."

- 5** Connect the output terminal of the WAIT\_1SEC box with the input terminal of OBJECT? Box, and organize the lines.

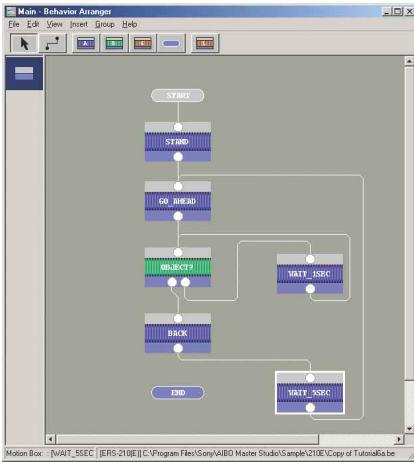


In the end, change the “distance” parameter to a “time” parameter with the condition that causes the program to execute the next command after AIBO has walked backwards.

- 6** Open the “BACK” Properties box, change the distance from “200 mm” to “10000 mm,” delete the WAIT command, and then close the Properties box.

AIBO will now search for obstacles every second while executing the “Go forward” command. The point of this step is to prevent the program from executing “go forward” again while the “OBJECT?” and “WAIT\_1SEC” boxes are still being executed in a loop.

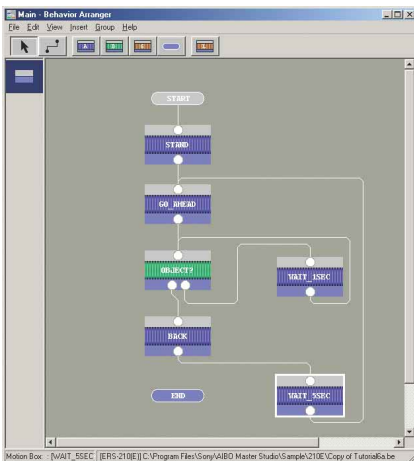
- 7** Disconnect the BACK and GO\_AHEAD boxes, and connect the BACK, WAIT\_5SEC and GO\_AHEAD boxes in sequential order.



Now add two more actions: twirling the tail (ERS-210 only) and moving the head sideways.

- 8** In the “Back Box” Properties box, add the commands indicated below and close the Properties box.

PLAY ACTION PALONE.AUTO.TAILROT (ERS-210 only)  
PLAY ACTION MOVE\_HEAD 50 0



Since the WAIT command has been deleted, the WAIT\_5SEC box is executed right after the command to go backwards. AIBO thus walks backwards for 5 seconds, and then walks forward afterwards.

Two commands, “AIBO twirling its tail” and “AIBO inclines its head 50°,” have been added.

## Saving the program and running it on AIBO

Save the program with the name “Lesson6.be” to the “Memory Stick” and run it on AIBO.

Let’s check the actions that you’ve altered.

When the Back box is executed, AIBO carries out the following four actions simultaneously.

- Waits 5 seconds
- Walks backwards
- Twirls the tail (ERS-210 only)
- Inclines the head 50° to the left

Afterwards, AIBO executes the GO\_AHEAD box,. Since the commands for twirling the tail and inclining the head continue to be valid after the execution of the GO\_AHEAD command, the execution of both commands continues thereafter.

In contrast to previous lessons, several commands can thus be executed simultaneously by not using the WAIT command (WAIT 0). The WAIT command should be used as required on a case by case basis.

### The program that you made in this lesson is

the same as the program in the “Tutorial6b.be” file in the “\Sample\model name”.

Verify that the program was properly created.

### HINT

- Of the system actions, the actions of waking and moving the head (WALK, MOVE...), searching (SEARCH...), moving the ears and tail (PALONE...) and lighting of the LED (SMESS...) are followed by the immediate execution of the next command when a WAIT command with no parameter is not used (overwriting of the command occurs).

If there is a need to interrupt the on-going execution of the current command in order to execute the next command, execution of the previous command is aborted and the new command is executed (e.g., switching from walking forward to walking backwards).

If there is no need to interrupt the execution of the current command, the execution of the current command continues (e.g., the commands for twirling the tail and inclining the head are still valid even after AIBO switches from walking forward to walking backward (or vice versa)).

Also, if a WAIT command with a no parameter is not used with system actions other than those mentioned above and user actions created with Action Composer, the next command is placed in the queue and is executed immediately after the current command.

If, however, you issue more commands that are not executed immediately, the queue may become full, resulting in an overflow that might cause the program to run improperly.

Do not overwrite the “Tutorial6a.be” file used in this lesson so that you can use it again in the future.

See page 12.

See page 55 of the User’s Guide.

- To change AIBO from walking forward to walking backwards immediately when AIBO detects an obstruction, delete “WAIT\_1SEC” box, and connect the ELSE terminal of the OBJECT? box to the input terminal of the OBJECT? box (looping within the box itself). This will minimize the time required for detecting an obstruction (not at one-second intervals).
- Specifying 0 ms in the WAIT command in the WAIT\_1SEC box is the same as a WAIT command with a null parameter. This command causes AIBO to wait until the completion of

**PLAY ACTION WALK 0 10000**

in the GO\_AHEAD box. Note that this means AIBO will not detect any obstructions until it finishes walking 10 meters (10,000 mm).

- If no obstructions are detected while AIBO walks forward 10 meters, the

**PLAY ACTION WALK 0 10000**

command will be completed and AIBO will walk no further. If you show an obstruction to AIBO when it has stopped after walking the 10 meters, the program will exit from the loop between the OBJECT? box and the WAIT\_1SEC box, and will then execute the BACK box, which causes AIBO to walk backward for 5 seconds, and then forward.

# Mastering Advanced Techniques

This lesson introduces advanced techniques for creating programs with Behavior Arranger. In it, you'll learn programming methods by studying a description of a "search for and kick the pink ball" program. In this lesson you will not edit the program.

## Preparing the file

Open the "Soccer.be" file with Behavior Arranger.

This file is stored in the "\\Sample\model name" folder (see page 12).

## Checking the main program

### Standing up

This program issues a "stand up" command and waits until AIBO finishes standing up.

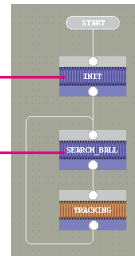
### Searching for the pink ball

```
PLAY ACTION MOVE_HEAD 0 0
WAIT
```

After the above command causes AIBO to face forward, the command below causes AIBO to search for the pink ball.

```
PLAY ACTION SEARCH.HEAD.LOWCENT PINK_BALL
WAIT
```

The SEARCH command can be overwritten when it is executed, but if there is a next command, the execution of that command may cause AIBO to stop searching for the ball. By adding a WAIT command, the execution of the next command can be delayed until AIBO finds the pink ball.



## Checking the program in the “Ball Searching” group box

### Starting the search for the ball

**PLAY ACTION TRACK\_HEAD PINK\_BALL**

The above command is included, but there is no WAIT command.

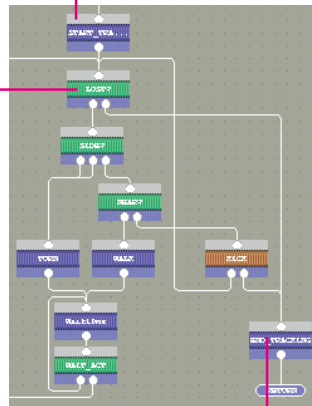
Note that the next command will be executed while AIBO is still searching for the pink ball.

### Determining whether AIBO has lost the ball or not

**IF Pink\_Ball = 1 SIDE?  
ELSE END\_TRACKING**

The above commands check whether AIBO is still able to see the pink ball or not. If AIBO sees the pink ball, a value of “1” is returned to the Pink\_Ball system parameter and the program jumps to the SIDE? Action box.

If AIBO has lost the ball, the program jumps to the END\_TRACKING box.



### Quit searching for the ball

At first, let’s check what the box specifies for AIBO when it has lost the ball.

In the END\_TRACKING box, only a “WAIT” command is defined. A “WAIT” command with no parameter makes the program wait until all commands have finished.

Here, the program waits for the following command in the START\_TRACKING box to complete:

**PLAY ACTION TRACK\_HEAD PINK\_BALL**

The system parameter of the Pink\_Ball command is not “1,” which means AIBO has lost the ball, but the TRACK\_HEAD command will not be completed immediately.

Since some time passes before the command completes, the above WAIT command is used to cause the program to wait until the completion of the TRACK\_HEAD command.



## Changing the direction of the body According to the location of the ball

The commands below are specified in the SIDE? action box which the program jumps to when AIBO sees the pink ball.

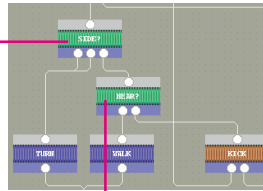
```
IF Head_Pan > 35
AND Wait = 1 TURN
IF Head_Pan < -35
AND Wait = 1 TURN
ELSE NEAR?
```

The Head\_Pan system parameter indicates the horizontal angle of AIBO's neck. If the angle is more than or less than 35° and the Wait value is "1," the program jumps to the TURN box which changes the direction of AIBO's body.

The Wait value indicates the number of on-going commands. When AIBO searches for the ball, the following command is executed, and so the value becomes "1."

```
PLAY ACTION TRACK_HEAD PINK_BALL
```

Otherwise, the ball is in front of AIBO, and so the program jumps to the "Near?" box, which calculates the distance to the ball.



## Finding the distance to the ball

```
IF Head_Tilt > -60
AND Wait = 1 WALK
ELSE KICK
```

The above command determines whether the ball is nearby or far away by the angle of AIBO's neck.

If the angle is more than -60°, i.e., if the head faces upward, it could be concluded that the ball is far away.

When the Wait value is "1," it means AIBO is still searching for the ball, and so the program jumps to the WALK box.

If the angle is less than -60°, i.e., if the head is facing downward, it can be concluded that the ball is near AIBO's legs. The program then jumps to the KICK group box.

### Turn and walk in the direction that AIBO is facing

The following command is entered into the TURN box:

**PLAY ACTION TURN Head\_Pan**

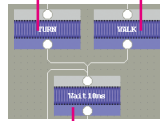
The following command is entered into the WALK box.

**PLAY ACTION WALK Head\_Pan 30**

The Head\_Pan represents the horizontal angle of AIBO's neck, i.e., the direction from which AIBO is viewing the ball. Note that there is no "WAIT" command after this command.

**PLAY-ACTION-TRACK\_HEAD-PINK\_BALL**

and either **PLAY ACTION TURN** or **PLAY ACTION WALK** are executed at the same time. In other words, two commands are executed simultaneously. If the values in the properties box are hard to read, adjust the width of the cell.



**Wait 0.01 second**

**WAIT 10**

The program waits for 10 ms (0.01 second).

In making this program, you can tell how many commands are in progress by the Wait system parameter.

## Changing the action depending on the number of the running command

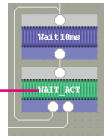
**IF Wait <> 1 Wait\_10ms  
ELSE LOST?**

The Wait system parameter represents the number of on-going commands.

When the TURN box or the WALK box completes, the **PLAY ACTION TRACK\_HEAD PINK\_BALL** command in the **START\_TRACKING** box and either the **PLAY ACTION TURN Head\_Pan** command in the TURN box or the **PLAY ACTION WALK Head\_Pan** command in the WALK box are simultaneously executed. The Wait system parameter is therefore "2".

In other words, until either TURN or WALK finishes, the "Wait" value is "not equal to 1," so the program loops every 10 ms to check if either program has finished or not.

When "PLAY ACTION TRACK\_HEAD PINK\_BALL" becomes the only on-going command, the Wait system value becomes "1" and the program returns to the LOST? branch box.



Changing the action depending on the number of on-going commands.

## Checking the “Kick” group program

### Maintaining system parameter values

SET pan Head\_Pan

WAIT 500

“Pan” is a user parameter where the value of the Head\_Pan system parameter is temporarily stored. Then, the program waits for 500 ms (0.5 seconds).

### Confirm that the ball has stopped

IF Head\_Pan = pan DO\_KICK

ELSE Continue

This program compares the “Head\_PAN” value stored 0.5 second ago with the present value. If the values are the same, the program jumps to the DO\_KICK box; if not, it jumps to the Continue box.

If the present value is the same as the one stored 0.5 second ago, the program concludes that the ball has not moved. AIBO thus kicks at that location. If the values are different, it concludes that the ball has moved. AIBO thus continues searching for the ball.

### Stop tracking the ball and kick it

STOP

PLAY ACTION KICK pan 1000

WAIT

The STOP command stops the execution of on-going commands.

Until now, execution of the command in the START\_TRACKING box has continued, but it is finally stopped by the STOP command.

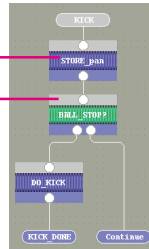
PLAY ACTION TRACK\_HEAD PINK\_BALL

PLAY ACTION KICK pan 1000

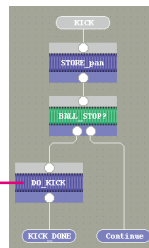
The above command causes AIBO to kick the ball 100 cm at the angle indicated by the “pan” user parameter.

To roll the ball a long distance, AIBO is commanded to kick the top part of the ball.

The WAIT command causes the program to wait until AIBO finishes kicking. The distance is measured from the center of AIBO’s body to the point of contact between the ball and the floor.



User parameters that start with a small letter can be defined and used whenever necessary.



## Rearranging the terminals

Let's now check the terminals of the KICK group box.

The KICK group box has two terminals, "Continue" on the left and "KICK\_DONE" on the right.

However, the Kick group that is invoked by the KICK group box has the KICK\_DONE terminal box placed on the left and the Continue terminal box placed on the right.

Terminal order is determined by the terminal list. From the menu, choose "Group-Set Order of Terminals," and confirm the terminal order. In the list, Continue and KICK\_DONE are listed at the top, and so the terminals for the KICK group box are listed in the same order from left to right.

If lines cross, rearrange the terminals order to uncross them.

You have now covered all the boxes in the "soccer.be" file.

This brings your lessons about Behavior Arranger to an end!

Enjoy AIBO by making original programs using the techniques you've learned.

For details on rearranging the terminals, see page 49 of the User's Guide.

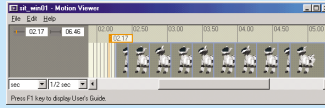
With the history bar on the left side of the window, you can go to higher and lower groups to confirm their contents.



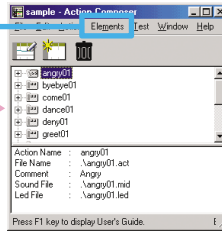
# Window Organization

Action Composer is made up of several windows, as shown below. Each of the lessons below will teach you how the windows are used.

**Motion Viewer\***  
Displays key frames of a motion. Use this function to copy key frames.



Lesson 2 Confirming key frames  
Lesson 6 Copying key frames

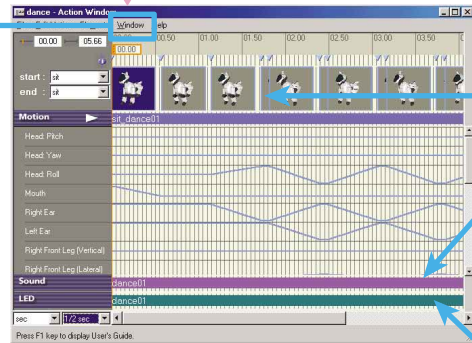


**Main Window**  
Elements and actions are managed in a library.

Lesson 1  
Creating action files

**Action Window\***  
In this window, you can adjust the timing of a motion and confirm all the elements of an action.

Lesson 1 Creating action files  
Lesson 4 Creating and combining elements  
Lesson 5 Synchronizing  
Lesson 6 Copying key frames



Pose Window

MIDI Window

WAVE Window

LED Window

**Preview Window**  
Open this window to preview on-screen an action that you've created.

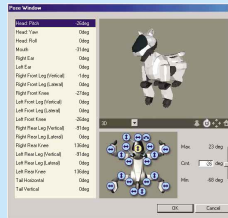
Lesson 2 Confirming motion/sound/LED



### Pose Window

Open this window to edit AIBO's poses (key frames of a motion).

- Lesson 3 Editing poses
- Lesson 4 Creating elements



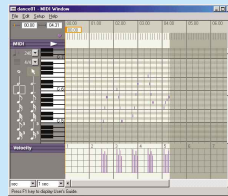
### Notes

\* These windows can also be opened by double-clicking the file name in the Main Window. You can also open them from the Start menu.

### MIDI Window\*

Open this window to create MIDI sounds for AIBO to play back. The piano roll interface allows you to visually confirm the movement and length of notes.

- Lesson 2 Checking MIDI sound
- Lesson 4 Creating elements
- Lesson 5 Synchronizing



### WAVE Window\*

Open this window to edit WAVE sounds for AIBO to play back.

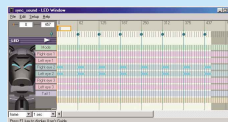
- Lesson 2 Checking WAVE sounds
- Lesson 3 Editing poses
- Lesson 4 Creating elements
- Lesson 5 Synchronizing



### LED Window\*

Open this window to create AIBO LED flashing patterns.

- Lesson 2 Checking the LED
- Lesson 3 Editing poses
- Lesson 4 Creating elements
- Lesson 5 Synchronizing



# Let's Create an Action Library!

**GOAL:** To learn how to make an action library and an action file

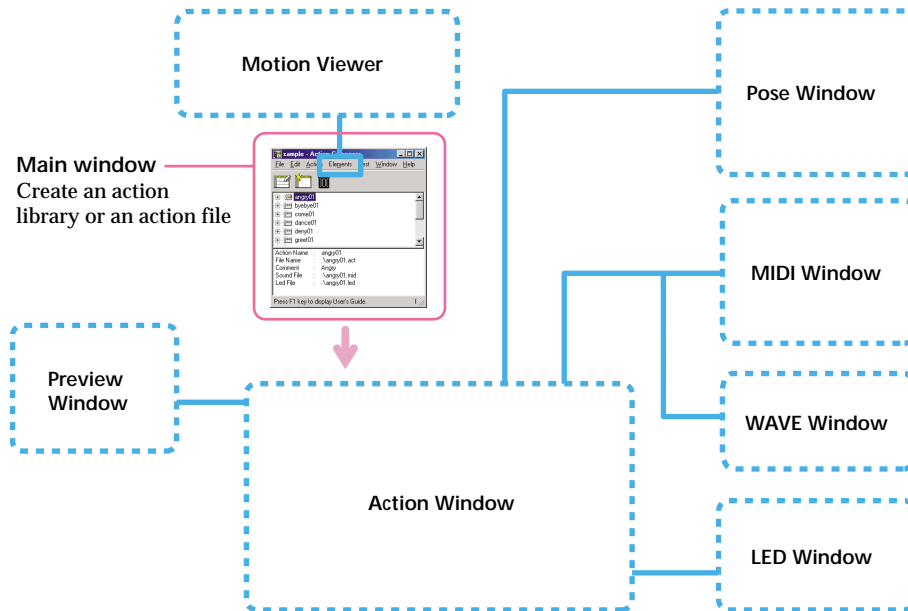
In Action Composer\*, you can make motions, sounds and LED elements, action files that combine these various materials, and action libraries that contain action files.

After creating the action libraries, you can combine them with programs in Behavior Arranger.

The number of action libraries used by each behavior is one. You can create an action library for each behavior, or create a library (for each model) to contain all the actions that you have created.

\* Action Composer is a tool for creating original actions. For details, see page 72 of the User's Guide.

## Windows Used in Lesson 1



Procedure:

- ① Preparations (page 73)
- ② Making an action library (page 73)
- ③ Making an action file (page 74)
- ④ Adding an action file to an action library (page 76)
- ⑤ Saving an action library (page 77)

For details on how to embed an action into a program with Behavior Arranger, see page 43.

You can create action libraries, action files and materials in any order that you like, but in this lesson, you'll create an action library first, and then an action file.



## Getting prepared

- Prepare a “Memory Stick” containing the system (E) suitable for your AIBO model.
- If you are using a wireless LAN, set it up in advance. For details, see page 23 of the User's Guide.
- The files you'll use in these lessons are stored in the “Materials” folder.

Copy the following “Materials” folder to your workspace (e.g., the “My Document” folder) in advance.

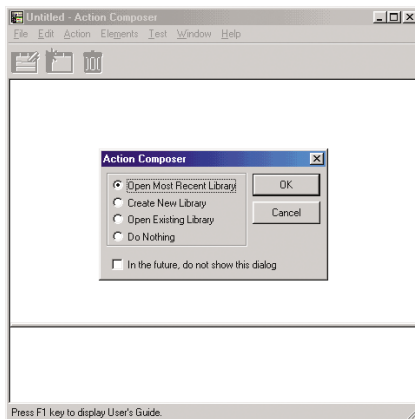
\<folder where AIBO Master Studio is installed>\Sample\<model name(E)>\Tutorial\Materials

For details on the files in the Tutorial folder, see “Files used in Tutorial” (page 12).

## Creating an action library

- 1 Click the Start button in Windows, select Programs-AIBO Master Studio and click Action Composer.

Action Composer starts up and a dialog box\* appears.

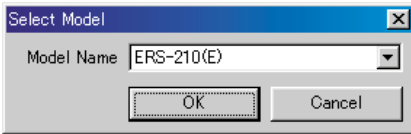


\* This dialog box is called the Navigation dialog box. If your system has been set not to display the box, you can display it by choosing Window-Option from the menu.

- 2 Select Create New Library and click OK.

If the Navigation dialog box is set not to be displayed, select File-New Library from the menu.

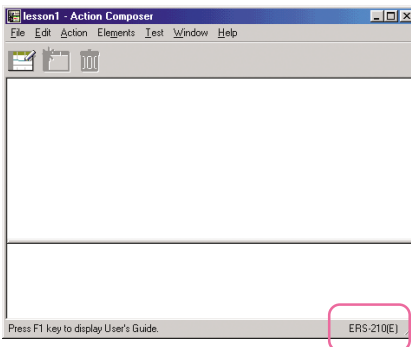
### 3 Select the AIBO model to be used and click OK.



The (E) indicates that English words are recognized. (The model name should correspond with the system on the "Memory Stick" and the behavior and action library.)

### 4 Type "lesson1.alb" as the file name and save the file in the copy of the Materials folder that you made.

The Main window appears.



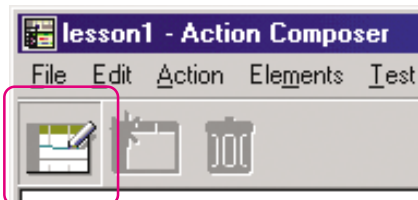
The model name (E) that you selected in step appears here.

In Action Composer, you create a file first, and then create the contents of the file.

## Creating an action

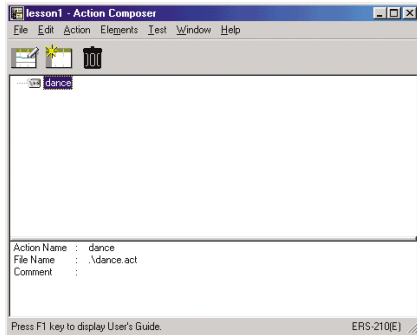
In this lesson, you'll make an action file in an action library using supplied elements.

### 1 Click the New Action button .



## 2 Save the file as “dance.act.”

The action file is created and the “dance” action file appears in the Main window.



## 3 Click the “dance” file created in step 2, and add elements from the copied Materials folder.

### Add a motion file

Select Elements-Add Motion, select “sit\_dance01.mtn” and click Open.

### Add a sound file

Select Elements-Add Sound, select “dance01.mid” and click Open.

### Add an LED file

Select Elements-Add LED, select “dance01.led” and click Open.

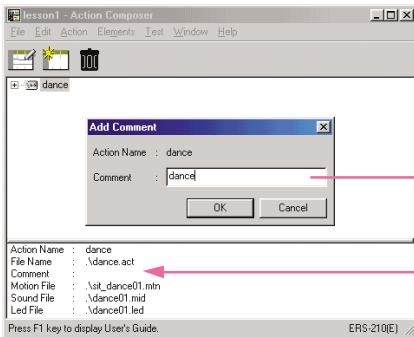
An action file made from the supplied elements is completed.

The elements are added to the selected action file.

Adding elements to an action file does not mean that the elements themselves are copied to the library or action files, but only that a reference is made to the element files. For this reason, if you edit the added elements (Lesson 3), this will cause the referenced elements themselves to be edited, so you must be careful. Make it a habit to make a copy of the element files in advance.

#### 4 Add comments to the action file.

With “dance” selected, select [Action]-[Add comment], type “dance”, and then click OK.



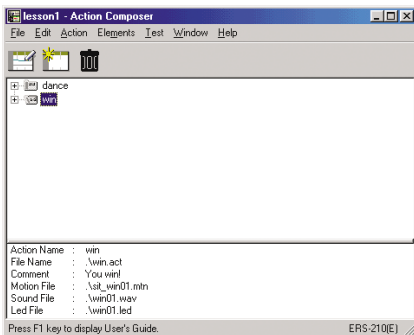
Comments are displayed here.

## Adding a supplied action to the library

You'll add supplied action files (in this case, sample files) to the action library.

#### 1 From the menu, select Action-Add Action, select “win.act” in the copy of the Materials folder, and then click Open.

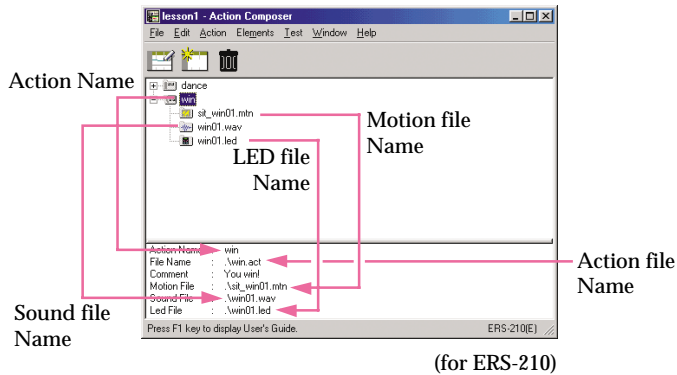
The action file is added to the Main window.



## 2 Click in front of "win".

You can check the elements added to the action file.  
Click an action file. Detailed information on the action file appears at the bottom of the window.

Action files or elements are shown in a tree structure in the Main window.



Note that in the case of the ERS-210/310, there is a LED file in the win folder.

## 3 From the menu, select File-Save Library to save the action library.

The contents of the library are updated.

If you do not overwrite the library when saving the action library, the library contents or the actions in the library will not be updated. To save what you've created in the Main window, you must overwrite the contents of the library.

## Saving to the "Memory Stick"

Using a "Memory Stick" that you have copied the system files on, save the action library you've created to the "Memory Stick." By saving the library to the "Memory Stick," the actions in the action library will be available for use in behaviors.

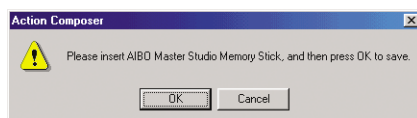
For details on copying the system files, see page 21 of the User's Guide.

If the AIBO model and the language of the action library are different from those of the system, it will not be possible to save the action library.

## 1 Insert the "Memory Stick" into the PC and select "File-Save Library to Memory Stick".

A confirmation dialog box appears.

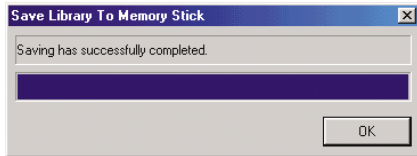
Only one action library can be saved to the "Memory Stick" (see page 44).



---

## 2 Click OK.

The library is saved to the “Memory Stick” and a dialog box appears when saving has finished. Click OK.



At the same time, the library is copied to the PC's hard drive. Quick Behavior Arranger and Behavior Arranger can now refer to the copied files, and the actions in the library can be used in behaviors (see page 51).

If you are not using a wireless LAN, then this completes Lesson 1.

---

## Saving the library via a wireless LAN

If you are using a wireless LAN, you can transmit the action library from the PC to AIBO.

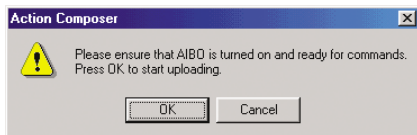
---

## 1 Insert the “Memory Stick” into AIBO and place the activated AIBO at a place where communication with the PC is possible.

---

## 2 From the menu, select File-Save Library To “Memory Stick”.

A confirmation dialog box appears.



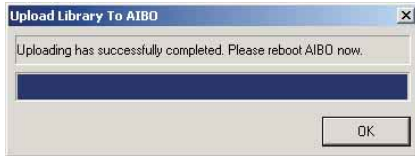
The action library that you created has already been saved to the “Memory Stick,” but save it again via the wireless LAN for study purposes.

---

### 3 Click OK.

The action library is sent to AIBO.

When the transmission finishes, a dialog box appears. Click OK.



You cannot make AIBO perform the action simply by saving it to the action library. This is where Action Composer's role ends and the baton is passed to Behavior Arranger. But in this tutorial, let's learn a little more about Action Composer.

The action library becomes available only after you've restarted AIBO. If you do not, the action will not play back in a behavior program, even if you start up Behavior Arranger and save the behavior program to the "Memory Stick."

---

The action library you've created is same as the "Tutorial1.alb" file in the L1\_Reference folder. Check to see if the file is correct.

Do you now understand how to create an action library by adding supplied actions and elements?

In Lesson 1, you created an action library without checking the contents of the elements or actions. In Lesson 2, you will check the contents that you add with your eyes and ears.

For details on the location of the L1\_Reference folder, see page 12 in "Files Used in the Tutorial."

# Let's Preview a Created Action!

**GOAL:** To learn how to preview an action.

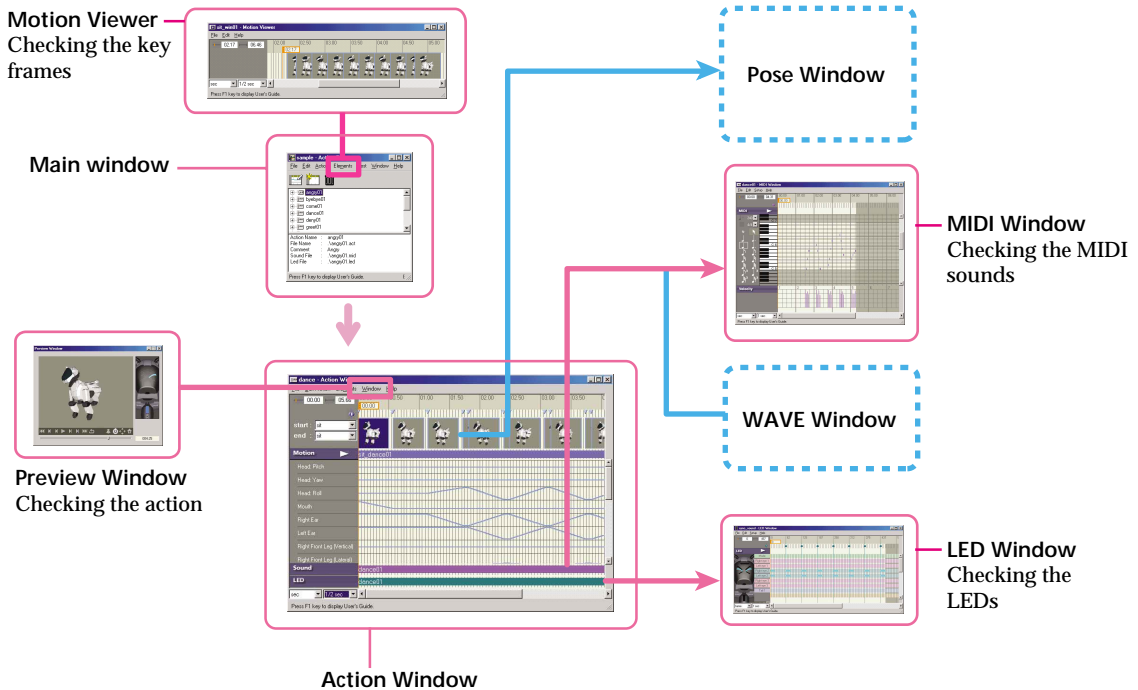
Let's find out how to view with your own eyes an action file that either you've created or has been supplied to you.

There are two following ways to preview a file:

- By viewing it on the PC (page 81)
- By having AIBO perform the action (requires a wireless LAN) (see page 85)

You will also learn how to preview an action file or elements that are not registered to an action library.

## Windows used in Lesson 2



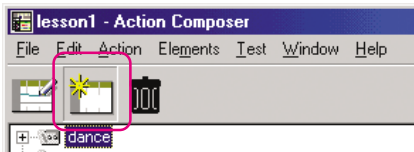


## Previewing an action and elements on the PC

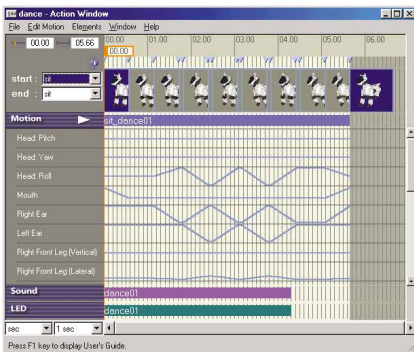
In this lesson, you will preview motions, sounds and LEDs in the Action Window.

**1** In the Main window, select the “dance” action file to open the action library that you used in Lesson 1.

**2** Click the Edit Action button .



The Action Window appears and the data for the selected action file is displayed.



If you started this tutorial from Lesson 2, open “Tutorial2.alb” in the L2\_Starter folder (see page 12). You won't edit the above action library, so there is no need to copy it.

You can also display the Action Window in step 1 by double clicking the action file.

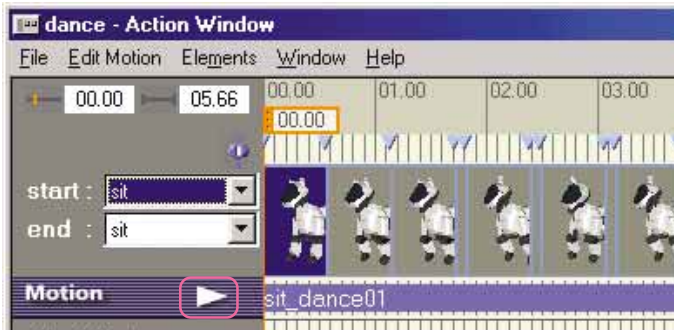
The elements that are combined in the action are listed in the Action Window. These include the file names and lengths of the key frames, sounds and LEDs in the motion.

By viewing this information in this window, you will gain a deeper understanding of what goes into a particular action.

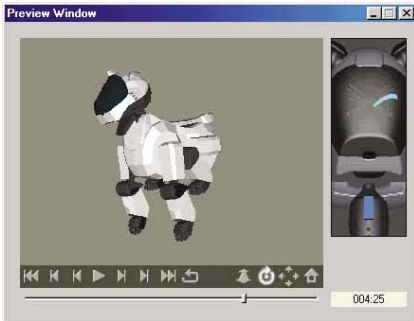
## Preview the motion

It's a little difficult to imagine an action simply by viewing the file names of its key frames, sounds and LEDs, so let's take a look at the action itself on the PC.

- 1 Click Replay button ▷ in the Action Window.



The Preview Window appears and the action starts to play back.



In the window, the elements that comprise the action file (motions, sounds and LEDs) are played back together.

To play the action again, click ▷ in the Preview Window.

### Note

Depending on the performance of your PC, some frames may be skipped due to extrapolation during the preview. We recommend that you to check the data for each key frame, sound and LED one by one.

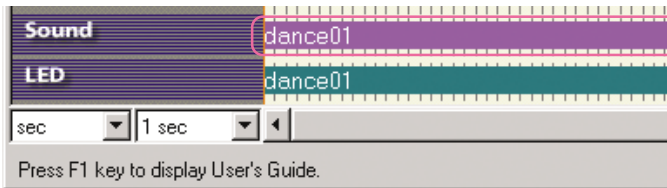
- 2 Click ✕ to close the window.

## Preview the sound

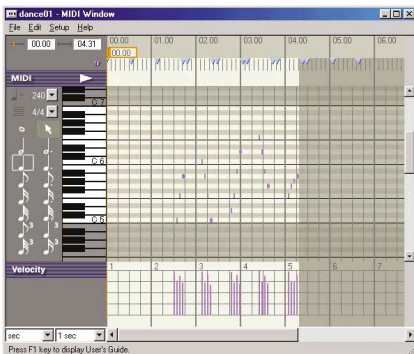
In the Preview Window, you can check the sound with your ears, but now let's check the sound with your eyes.

- 1 Double-click "dance01" that appears in the sound channel in the Action Window.

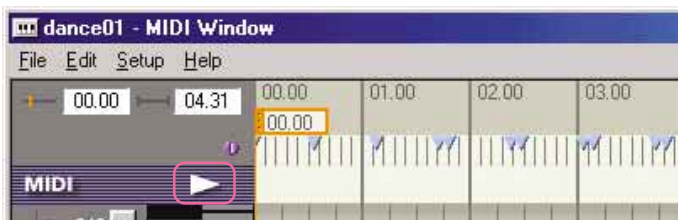
You can also open windows by double clicking a sound or LED file in the Main window.



The MIDI Window appears, showing the arrangement of the MIDI data.



- 2 Click the Playback button ▷ in the MIDI Window.



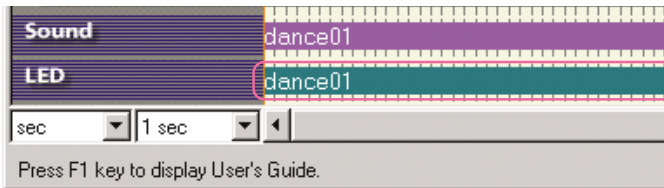
A preview of the sounds begins.

- 3 Close the window.

## Previewing LEDs

In the same way that you previewed the sound, let's check the LED data with your eyes.

- 1 Double-click "dance01" that appears in the LED channel in the Action Window.

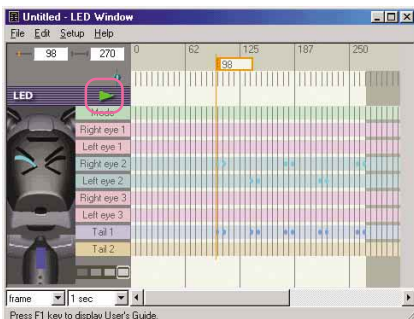


The LED Window appears, showing the arrangement of the LED data.



- 2 Click the Playback button ▷ in the LED Window.

You can check the LED lighting scheme on AIBO (shown on the left).



### 3 Close the window.

Close the Action Window.

If you are not using a wireless LAN, go to page 86, where you'll learn how to preview elements and actions that are not registered to an action library.

## Testing an action on AIBO

If you are using a wireless LAN, you can check an action by having AIBO perform it. Unlike checking an action through 3D modeling, this test allows you to see how AIBO moves in real life.

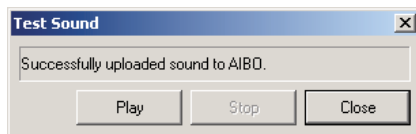
If you are not using a wireless LAN, you will have to add the action to a behavior in order to have AIBO do a test performance of the action.

### 1 Insert the "Memory Stick" into AIBO and place AIBO at a place where communication with the PC is possible.

### 2 Select an action or elements in the Main window and play them back with the Test menu as follows.

#### Testing an action

Select an action file and select Test-Test Action from the menu.



To have AIBO test a behavior using a wireless LAN, you must first register the file for the respective action or elements to the action library in advance.

Files are not be saved to the "Memory Stick" when doing a test performance.

#### Testing a motion

Select a motion file and select Test-Test Motion.

#### Testing a sound

Select a sound file and select Test-Test Sound.

#### Testing LEDs

Select an LED file and select Test-Test LED.

### 3 After the data has been transferred, click Play.

See how AIBO performs the action.

To stop AIBO immediately, click the Stop button. Note that AIBO may fall over if AIBO is stopped in an unstable position.

- 
- 4 When AIBO has stopped, click Close to close all the windows.
- 

You may be wondering if it is possible to check files that haven't been registered to the action library. No problem! Just do the following procedure.

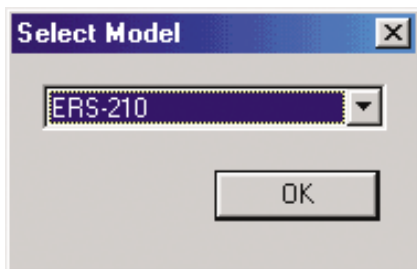
## Checking an action or elements not registered to the action library

In this procedure, you will open each window that comprises Action Composer, not from the Main window, but from the Action Composer Tool.

### Checking an action

---

- 1 Click the Start button in Windows, select Programs-AIBO Master Studio-Action Composer Tool and click Action Window.
- 2 Select AIBO mode, and then click OK.



- 3 From the menu, select File-Open, select "dance.act" (which you created in Lesson 1), and then click OPEN.

From here, do the procedure on page 82.  
Close the window when you finish the check.

For an action file or elements, the language specification (English or Japanese) is not significant (as it is for an action library).  
You need only to select the model.

If you started this tutorial from Lesson 2, copy the L2\_Starter folder and open the Tutorial2.act file (see page 12).

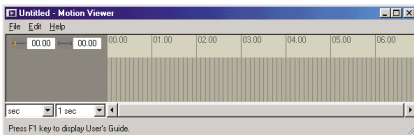
## Checking elements

### Checking a motion

As with an action file, you can check a motion from the Action Window by adding it to the window as an element. But in this section, we'll check it in another window for study purposes.

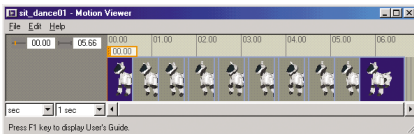
- 1 Click the Start button in Windows, select Programs-AIBO Master Studio-Action Composer Tool, and then click Motion Viewer.

The Motion Viewer appears.



- 2 From the menu, select File-Open, and then open the "sit\_dance01.mtn" file used in Lesson 1.

Key frames\* appear. Let's check the motion with the key frames.



In the Motion Viewer, key frames can only be displayed; they cannot be previewed or edited. You can preview and edit key frames in the Action Window. The Motion Viewer window is convenient for copying key frames for reuse.

You can also double click the motion file in the Main window to display this window.

\* For details on key frames, see page 72 of the User's Guide.

## Checking the sound

---

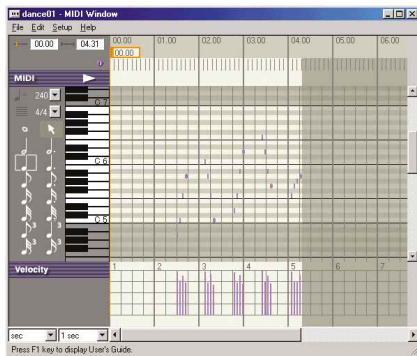
- 1 Click the Start button in Windows, select Programs-AIBO Master Studio-Action Composer Tool, and then click MIDI Window.

The MIDI Window appears.

---

- 2 From the menu, select File-Open, and then open the "dance01.mid" file used in Lesson 1.

Do the procedure on page 83.



If you click "WAVE Window" in the menu, the WAVE Window appears.

The sound format is the same for all three models, so you need not select the model. In other words, you can use the sample files supplied with other models.

## Checking the LEDs

---

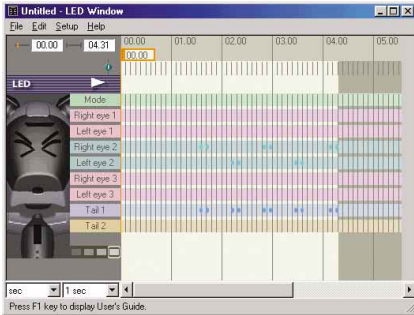
- 1 Click the Start button in Windows, select Programs-AIBO Master Studio-Action Composer Tool, and then click LED Window.
- 

- 2 Select the AIBO model to be used.

The LED Window appears.



- 
- 3** From the menu, select File-Open, and then open the “dance01.led” file used in Lesson 1.  
Do steps 2 to 3 on page 84.



---

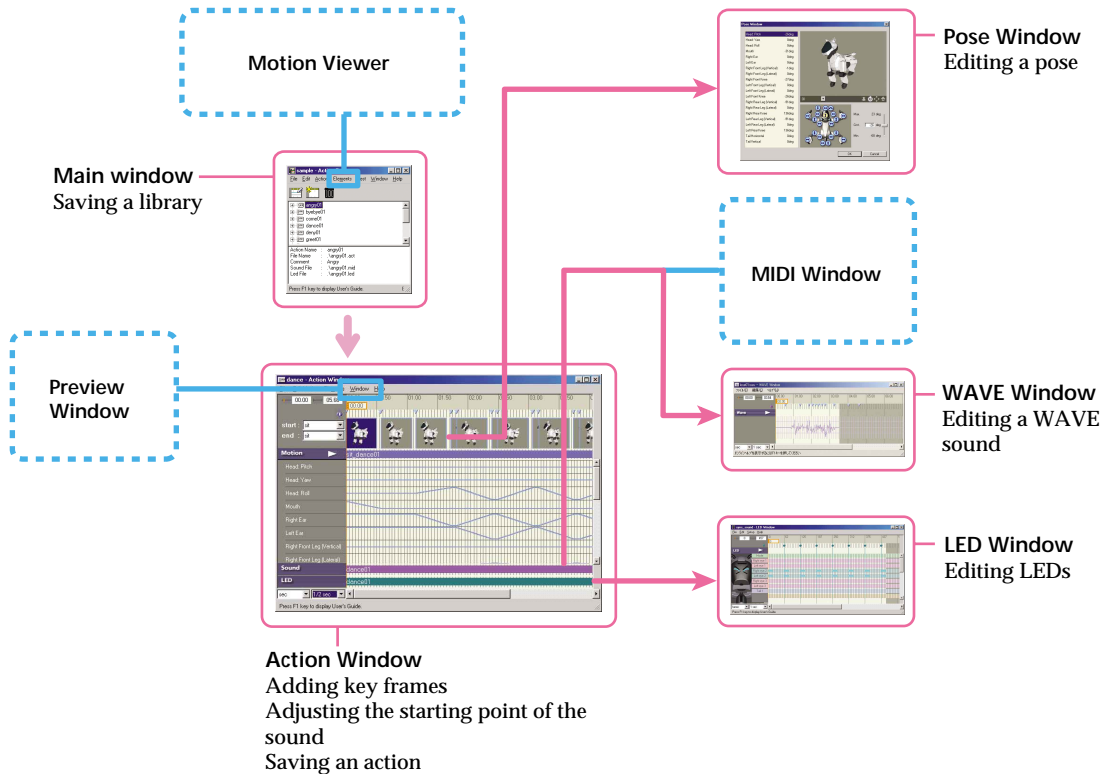
You have now learned how to check an action or elements.  
Now you can create an action library by combining the supplied elements and actions.  
Next, you will edit files and elements to create original actions.

# Let's Edit Elements

**GOAL:** Learning how to edit elements

In this lesson, you'll learn how to edit action files by modifying the supplied action files.

## Windows used in Lesson 3

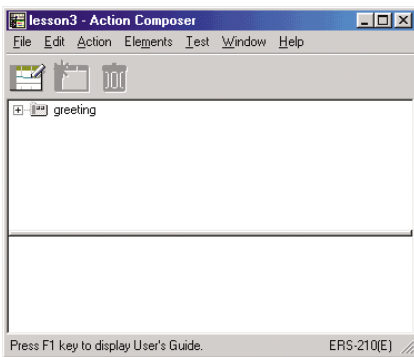



## Procedures

- ① Preparing elements (page 91)
- ② Editing a motion file (page 92)
  - Change the "raise head slowly, lower head somewhat fast" motion to a "raise and lower head fast" motion.
- ③ Editing a sound file (page 94)
  - Delete the silent parts and adjust the starting point of the sound
- ④ Editing an LED file (page 96)
  - Create a light flashing sequence that matches the motion
- ⑤ Saving an action file (page 100)

## Preparing elements

- 1 Start up Action Composer.
- 2 Select Open Existing Library in the Navigation dialog box, open "lesson3.alb" in the copy of the Materials folder.



- 3 Select the "greeting" action file.
- 4 Click the Edit Action  button.

The Action Window appears.

Play back the motion and check the "raise head slowly, say Hello, and lower head somewhat fast" motion.

If you started this tutorial from Lesson 3, start the lesson after copying the Materials folder (see page 73).

If the Navigation dialog box is set not to be displayed, select File-Open Library in the menu.

You can also open the Action Window by double clicking the "greeting" action file.

You'll edit elements in this Action Window.

## Editing a motion

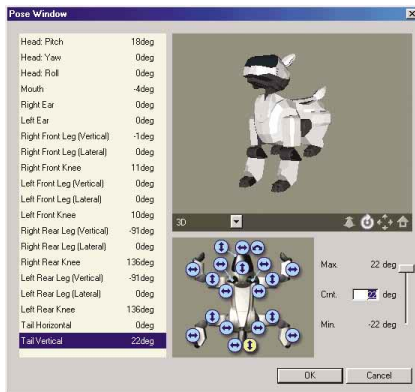
Let's create the "raise head fast" motion.

### Editing a pose

#### 1 Double click the second key frame.

The Pose Window appears.

#### 2 Select the part to be modified from the Parts list, enter the degree in the Current setting box, and then press the Enter key on the keyboard.



ERS-210/220 Head: Pitch 18 deg

Tail Vertical 22 deg

ERS-310 Head: Pitch 20 deg

3D AIBO has moved!

#### 3 Click OK to close the Pose Window.

Confirm that the second key frame in the Action Window has changed.

In this window, you'll edit the pose in the key frame.

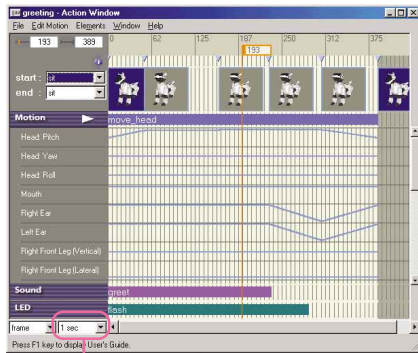
You can select any part indicated by  $\updownarrow \leftrightarrow$  in the AIBO diagram at the bottom of the window.

Enter the degree by typing the number or moving the right slider up or down.

## Adding a key frame

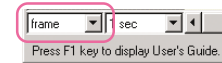
- 1 Click "193rd frame (3.10 seconds)" on the Time/frame channel.

The current bar moves and the current position changes.



(Time interval\*: 1 sec)

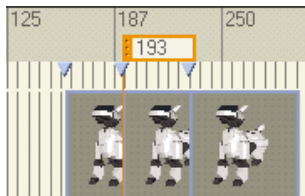
To change the display unit of the Time/frame channel, select the unit in the sec/frame box at the bottom of the Action Window.



\* The time interval is the interval between the numbers displayed on the Time/frame channel. Set a longer interval to make it easier to enter data or select a frame.

- 2 From the menu, select Edit Motion-Change To Key Frame.

A key frame is created for the pose that AIBO is currently in.



A motion is made by creating a key frame, and then editing the pose.

### 3 Set the pose in the key frame.

Do steps 1 to 3 on page 92, making the following settings for each part.

ERS-210	Head: Roll	19 deg
ERS-220	Head: Yaw	30 deg
ERS-310 series	Head: Yaw	30 deg

You have now modified the motion.

From the menu, select Elements-Save Motion As, and then save the file with the name “move\_head\_2.mtn”. The name of the motion in the motion channel also changes.

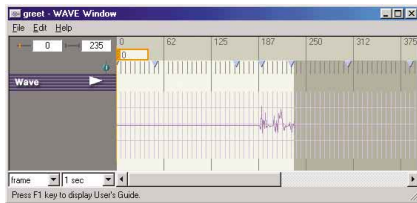
## Editing a sound

### Editing a sound

Let's delete the silent part in a sound.

#### 1 Double-click the “greet” file on the sound channel bar in the Action Window.

The WAVE Window appears.



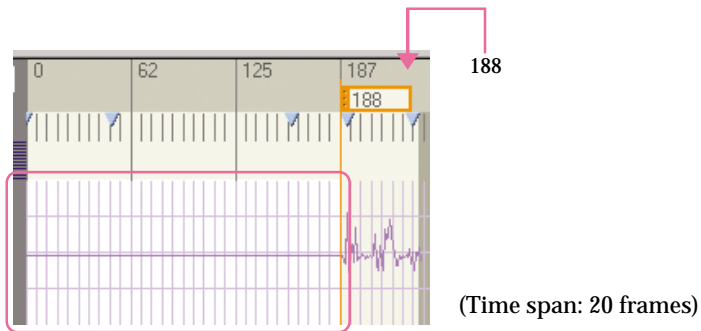
There is the phrase “Nice to meet you,” but also a silent part at the beginning.

The WAVE file is a large one, so the silent parts should be removed to the extent possible. Silent parts should be used to draw attention to something.

AIBO can play WAVE files in the 8-kHz, 8-bit, monaural PCM format. Audio files recorded using the Windows Sound Recorder or free music files should be converted to the proper format before use.

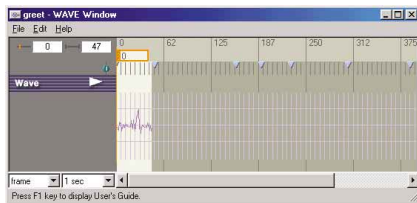
- 2** Click the 0th frame (0 second) and Shift-click the 188th frame (3 seconds).

The silent part is selected and highlighted in white.



- 3** From the menu, select Edit-Cut.

The selected part is cut.



Selecting Edit-Delete deletes the sound, but leaves the length unchanged.

- 4** From the menu, select File-Save As and save the file with the name "greet\_2.wav".

A sound file without the silent part has been created. Close the window.

- 5** Delete the sound file in the Action Window, and then add the "greet\_2.wav" file to the action.

Select Elements-Delete Sound from the menu to delete the file, and then select Elements-Add Sound to add the above-mentioned file.

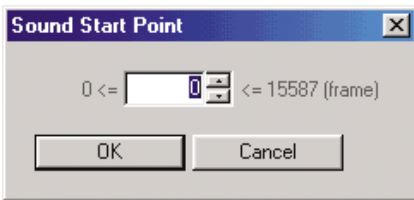
When you edit sound or LED files that have already been combined, you have to combine them again manually to update the action.

## Adjusting the starting point of the sound

Since you've cut out the silent part, the sound has been moved forward. Let's adjust the starting point of the sound.

### 1 In Action Window, select Elements-Set Sound Start Point.

The Sound Start Point dialog box appears.



### 2 Enter 188, the number of frames that you've deleted, and then click OK.

The position of the file moves and the starting point of the sound returns to the original position.

The 188th frame is the starting point. In a large file like the a WAVE file, the starting point of the sound should be positioned for maximum effect.

## Editing the LEDs

The LED file is short compared to the motion file. Let's edit the LED file.

### Adding LED data

Add LED data to create a flashing pattern.

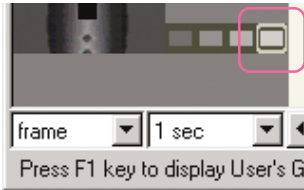
### 1 Double-click "flash" on the LED channel bar in the Action Window.

The LED Window appears.




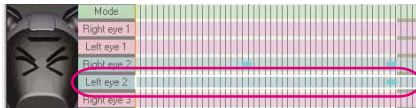
## 2 Click the right-most Brightness button.

This selects the brightest level.



## 3 Click LED on AIBO.

The cursor changes to .

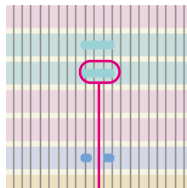


The mouse is set to the data-input mode. By clicking, you can see the location within the LED channels that corresponds with the selected LED.

## 4 Position the LED data as shown below.

### ERS-210

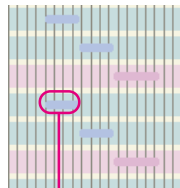
Add data to "Left eye 2" to make both eyes flash.



Enter here

### ERS-220

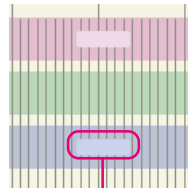
Specify the highest brightness for "Face L1" to make the lights flash in sequence.



Enter here

### ERS-310

Add blue to make it pink.

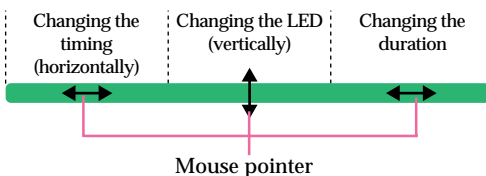


Enter here

(Time interval: 1/2 sec)

### To move or change the length of data

You can move or change the length of the data by pointing and dragging the cursor on the data. What you can do depends on where you place the cursor, as illustrated below.



If you've entered data in the wrong place

Select Edit-Undo and clear the misplaced data.

If you try to create new LED data where data already exists, you will not be able to overwrite the existing data. To create new data, first change the length of the LED data with blank area, and then place the new data at the desired location.

## Making the LEDs flash

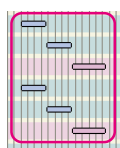
Let's learn how to make the LEDs flash and to match the flashing duration to a motion.

- 1 From the menu, select Setup>Select Mode and select LED data by dragging the mouse.

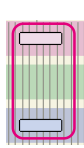
ERS-210




ERS-220



ERS-310 series



Input mode is used to enter data to a channel. To change the mouse to Input mode, click one of AIBO's LEDs in the LED Window or click a musical note in the MIDI Window.

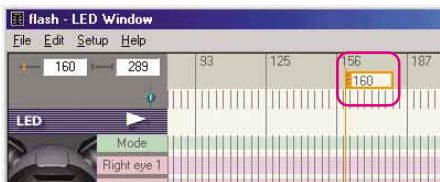
Select mode is used to edit data in a channel. You can select or move data in the channel. (To change the mouse to Select mode, click  in the MIDI Window.)

- 2 From the menu, select Edit-Copy.

- 3 Click the 165th frame (2.56 seconds) on the Time/frame channel.

The current bar moves.

The LEDs light again from this frame.

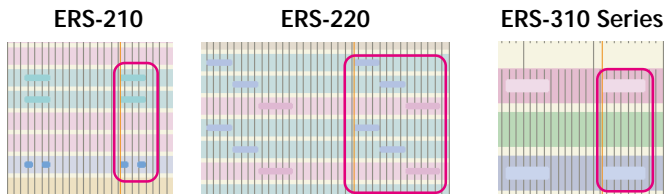


(Time interval: 1/2 sec)

---

**4 From the menu, select Edit-Paste.**

The first lighting pattern is copied.



(Time interval: 1/2 sec)

Click ▷ to check the flashing.

---

**5 Save the file with the name "flash\_2".**

---

**6 Combine the LED file again in Action Window, and then click ▷.**

Let's check the lighting sequence and motion in the Preview Window.

---

See step 5 on page 95.

## Saving the action file

- 1 Save the file with **File-Save in the Action Window**, and then close the window.
- 2 Save the action library with **File-Save Library in the Main window**.

Elements comprising the action are replaced with the newly created elements.

The files you've created are identical to the files in the L3\_Reference folder (see page 12). Check to see that the files are correct.

You have now mastered the basic operations in Action Composer and the way to make files for use in Behavior Arranger!

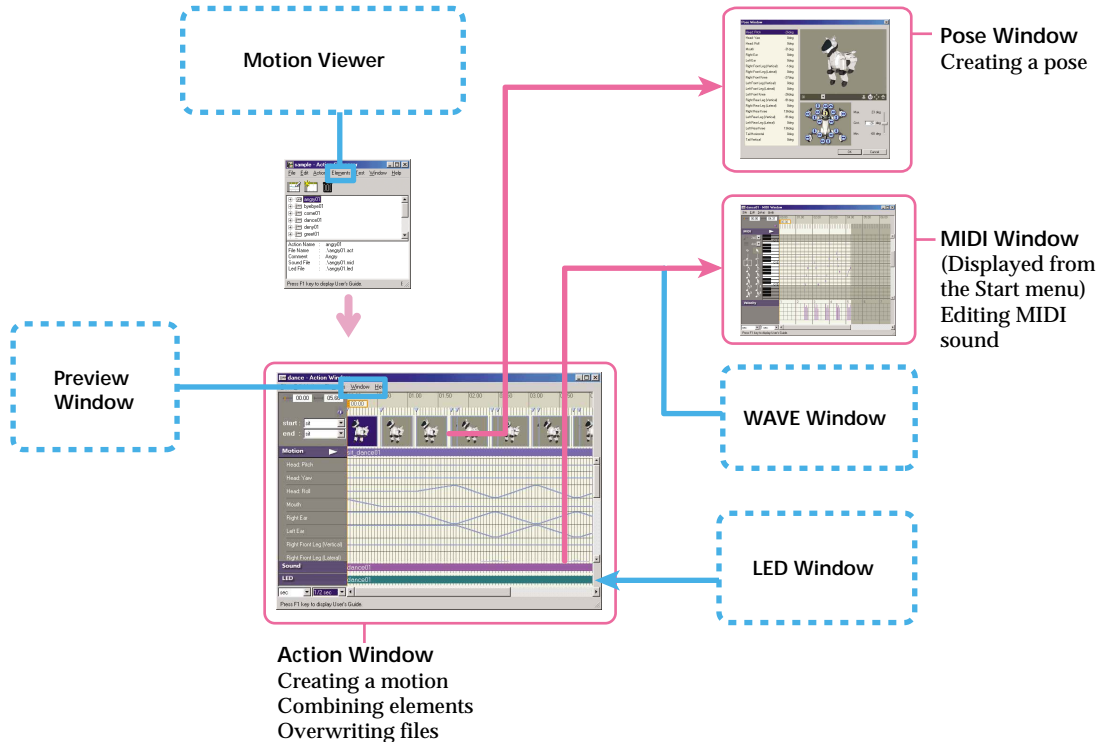
You can now create your own original actions by modifying the details in the lessons that you've studied.

# Let's Create Original Elements!

**GOAL:** Learning how to make elements from scratch.

Now that you've learned how to create an action library, the next step is to learn how to create original works of your own.

## Windows used in Lesson 4



Procedures:

- ① Creating an action file (page 102)  
In Lesson 4, you'll start with the making of an action file, unlike what you did in the beginner's course.
- ② Creating a motion (page 102)  
You'll make a simple motion from scratch.
- ③ Creating sounds (page 105)  
You'll learn how to input MIDI sound data.
- ④ Combining elements (page 107)

## Creating an action file

In creating an action, you can start by creating the action library first, then create the action to be contained in the library. Another way, however, is to create the action file first, and then add it to the action library afterwards.

Create a new action file; do not use a file from the action library.

**1** Click the Start button in Windows, select Programs-AIBO Master Studio-Action Composer Tool, and then click Action Window.

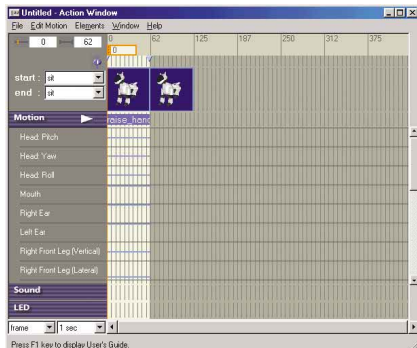
**2** Select the AIBO model to be used.  
The Action Window appears.

## Creating a motion

As a lesson in motion creation, let's make a simple motion consisting of AIBO sitting down and then raising its head at the same time as its front leg.

**1** Select Elements-New Motion.  
A dialog box appears. Save the file as "raise\_hands".  
The starting and ending poses appear.

Create the file first before creating its contents.



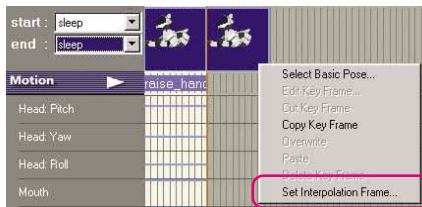
## 2 Specify "sleep" for both standing and ending poses.

The starting and ending poses are changed.



Selecting starting and ending poses from the basic poses eliminates the need to specify the intervening poses in the action. This is done automatically by AIBO, from starting pose to ending pose.

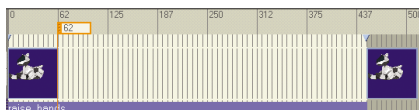
## 3 Right-click the end key frame, and select Set Interpolation Frame.



By lengthening the interpolation frame, the entire motion becomes longer.

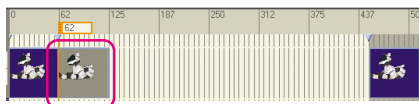
## 4 Enter 450 and click OK.

The interpolation section is lengthened and the entire motion becomes 451 frames long.



## 5 Click the 62nd frame (1 second) on the Time/ frame channel to create a key frame.

A new key frame is inserted.

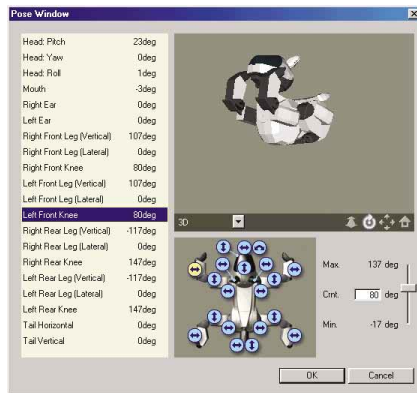


See page 93.

## 6 Double-click the key frame and create the following pose in the Pose Window.

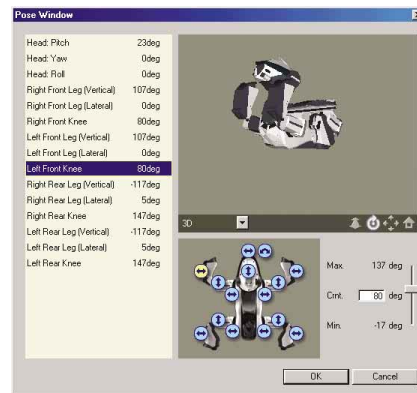
### ERS-210

Head: Pitch	23 deg
Right Front Leg (Vertical)	107 deg
Right Front Leg Knee	80 deg
Left Front Leg (Vertical)	107 deg
Left Front Leg Knee	80 deg



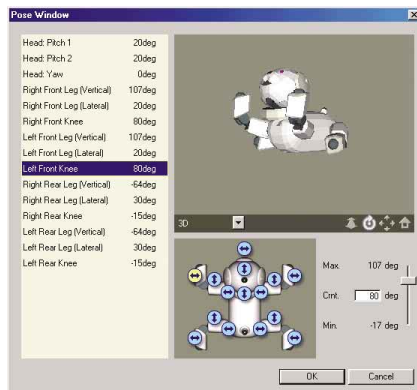
### ERS-220

Head: Pitch	23 deg
Right Front Leg (Vertical)	107 deg
Right Front Leg Knee	80 deg
Left Front Leg (Vertical)	107 deg
Left Front Leg Knee	80 deg



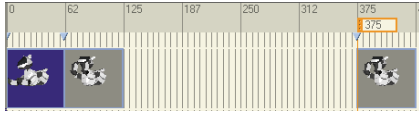
### ERS-310 series

Head: Pitch	20 deg
Head: Pitch2	20 deg
Right Front Leg (Vertical)	107 deg
Right Front Leg Knee	80 deg
Left Front Leg (Vertical)	107 deg
Left Front Leg Knee	80 deg





- 
- 7** Copy the key frame you created in step 6 and paste into the 375th frame (6.00 seconds), overwriting the data there.



- 
- 8** Create a new key frame at the 230th frame (3.68 seconds) and set the pose below.

(The settings are the same for all models.)

Right Front Leg (Vertical)    60deg  
 Left Front Leg (Vertical)    60deg

- 
- 9** Select “Elements-Save Motion As” to save to the “raise\_hands” file, overwriting the current contents.

Close the Action Window without saving the action.

---

After creating a new motion file in the Action Window, you can save the file without creating an action file by selecting Elements-Save Motion As, and then the file name that you specified in step 1.

## Creating sounds

It is not possible to create a new sound file or a new LED file from the Action Window. Display the edit windows for each window from the Action Composer Tool.

After creating files, combine them in an action file.

In this section, you’ll learn how to use the MIDI window to input sounds from a supplied file. After mastering the procedure, you’ll be able to create MIDI files by inputting the main melody line from scores.

It is possible to use sounds created by commercial MIDI editors, but we recommend that you use the MIDI Window of Action Composer for its ease of use in creating sounds suited to AIBO.

- 
- 1** Click the Start button in Windows, select Programs-AIBO Master Studio-Action Composer Tool, and then click MIDI Window.

The MIDI Window appears.

---

## 2 Select File-Open and open “jinglebells.mid”.

---

The copied materials are inside the folder.

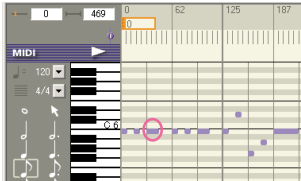
## 3 Click to play the file.

It sounds like the Jingle Bells, only a little funny.  
Add sounds where they are lacking, and make it sound more in tune.

---


## 4 Click note and add sounds as shown below.

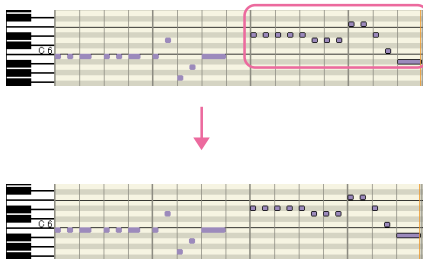
Add sound data where it is lacking.  
If you put data in the wrong place, drag it to the correct place.



Clicking a note changes the mouse to input mode, allowing you to add sounds to the Score channel. For details on adjusting the data, see page 98.

## 5 Click and select the music data as shown below, and then drag it downward.

Position the mouse so that it becomes , allowing you to drag the data.



## 6 Click .

Confirm that you’ve positioned the data correctly!

---

## 7 Save the MIDI file with the name “jinglebells\_2”.

Close the window.

---

## Combining elements

By adding the files that you created into the Action Window, you can combine the elements.

- 
- 1** Select the AIBO model, start the Action Window, and then add the elements that you created.

Motion: raise\_hands.mtn

Sound: jinglebells\_2.mid

- 
- 2** Save the file as “sing\_jinglebells.act” and close all windows.
- 

The file that you created in this lesson is identical with the “Tutorial4.alb” file in the L4\_Reference folder (page 12). Check the file to confirm that it was properly made.

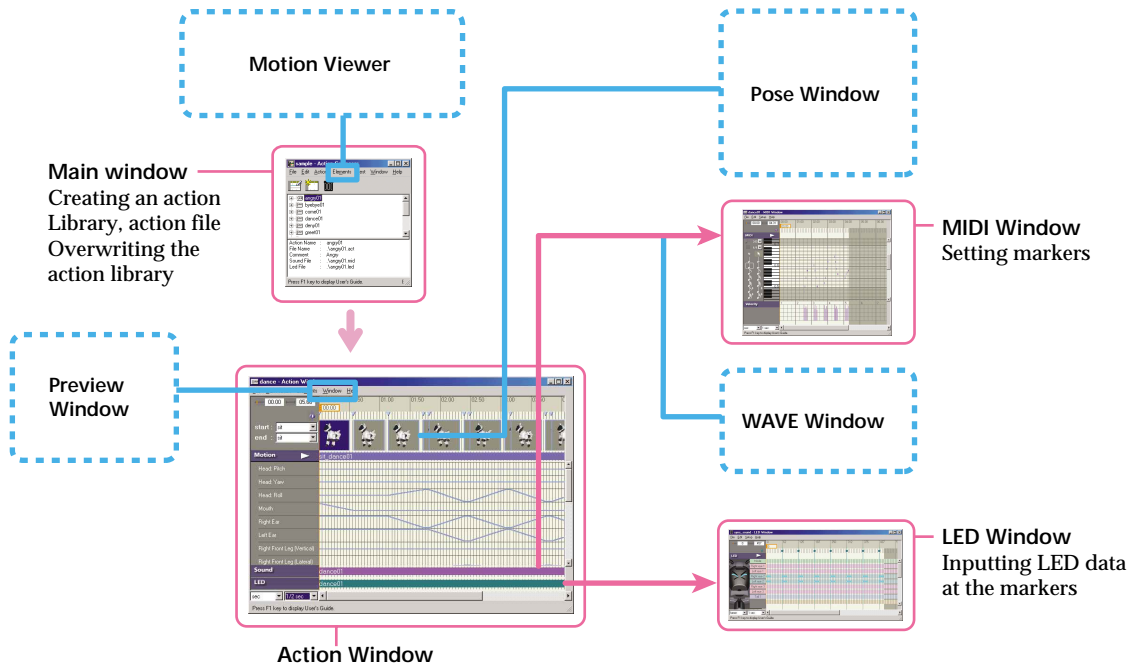
You now know how to create action files and elements. What you make from now is up to your imagination!

# Let's Light up the LEDs to the Music

**Goal:** Learning how to synchronize motion, sound and LED

Now that you can create your own elements, let's move on to the synchronizing of elements. For instance, you can make AIBO wink and move its mouth in sync with the sounds it emits. Making AIBO dance to the music can also be great fun.

## Windows used in Lesson 5



### Procedure:

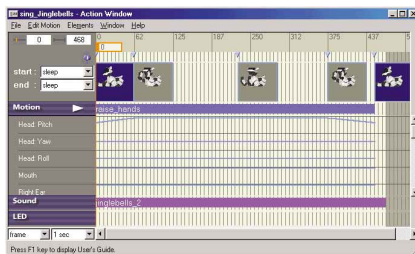
Set markers in the MIDI window opened from the Action Window.  
Then make an action file by creating an LED light sequence by aligning it with the MIDI markers.

The marker function in the action file is convenient for synchronizing elements. In this lesson, you'll create an action in which AIBO lights up its LEDs to music.

**1** In the Action Window, open the “sing\_Jinglebells” action file created in Lesson 4.

**2** From the menu, select Elements-Add LED, and then select the “sync\_sound” file.

This LED file does not contain any LED data, so nothing appears in the LED channel.



**3** Open the edit windows for the sounds and LEDs.  
Open the LED window by selecting Elements-Edit LED.

If you started from Lesson 5, first copy the L5\_Starter folder, and then open the “Tutorial5.act” file (see page 12).

The LED and sound files cannot be created in the Action Window. In this lesson, use the supplied files in the Materials folder. You cannot set markers in MIDI, WAVE and LED windows activated directly from the Main window. Open the Action Window first, and then open the edit window for each element.

#### 4 Set markers in the MIDI Window.

Press the Alt key and click the marker channel to set markers at the following frames.

(These settings are the same for all models.)  
 Frame 62, 125, 187, 250, 312, 375, 437



Make sure that the purple markers set in the MIDI Window are also set in the Action Window and LED window.

If you misplace a marker You can delete it with Alt+click.

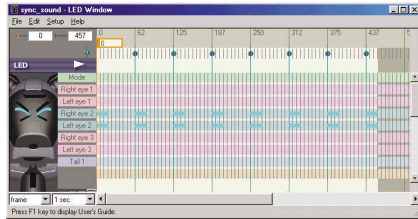
#### 5 Input the LED data at the markers.

Create data that repeats a 20-frame long LED illumination, as shown below.

For details, see page 96.

##### ERS-210

Light up Right Eye 2 and Left Eye 2 with high brightness



##### ERS-220

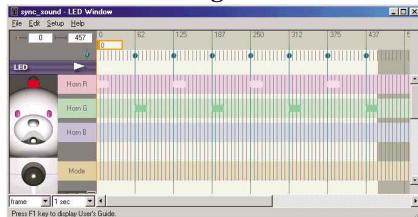
Alternates blue and red

Blue : Face RL1      Red: Face CR3  
 Face RL2              Center Tail  
 Right Tail  
 Left Tail



##### ERS-310 Series

Alternates red and green

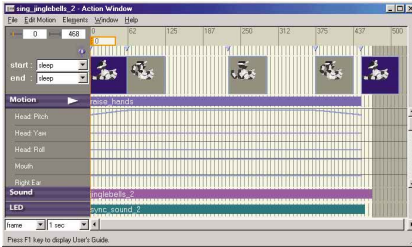


(Time interval: 1 sec)

## 6 Save the LED file as “sync\_sound\_2”.

Overwrite the LED file “sync\_sound\_2” in the Action Window.  
Now click ▶ to preview the file.

For details on inputting data, see page 95.



## 7 Save the action file as “sing\_Jinglebells\_2”.

When you open this action file next time, you'll see the markers set in each window.

In the same way, setting markers in the LED Window is convenient for aligning the position of key frames and sound data to the LEDs.

The files that you created in this lesson are identical to the files in the L5\_Reference folder (page 12). Check these files to see if they were properly made.

You have now finished creating an action file which synchronizes motion, sound and LEDs! Did you see how easy it is to synchronize elements by using markers?

In the next lesson, we give you tips that will help you make better actions.

# Tips for creating actions

In this course, we give you hints to help you create more attractive AIBO actions!

A thorough knowledge of Action Composer use is naturally essential, but in the end, the key to creating good works will be your knowledge of AIBO's features and your personal sense of design.

## General

**Motions, sounds, and LEDs are the three parts that make up the whole!**

- A good action uses AIBO's motions, sounds and LEDs at the same time.

A simple action can be made more attractive merely by the addition of LEDs or sounds.

**Real-life testing!**

- It is best to try out an action on AIBO, rather than just previewing it on a PC.

## Motion

**Keeping balance is the key!**

- Even after you've created an action and previewed it on the PC, AIBO will, in many cases, fall down when it actually performs the action. This is because balance has not been calculated. If, for example, you incline AIBO's body to the left before it raises its right leg, AIBO will remain stable. As you create motions, it is recommended that you to copy parts from the sample files and verify motions on AIBO via a wireless LAN.
- The "sleep" pose is AIBO's most stable posture, so it's easier to create motions based on it.

*Example: Motions using the sleep pose*

It would be fun to have AIBO stretch out its back legs and kick them as if swimming!

Before stretching out the legs, you have to raise the paws off the ground and then stretch the legs out backward. If you try to stretch legs without raising the paws first, the paws will drag along the ground, making it hard for AIBO to stretch out its legs.

The file "lesson6.mtn" is provided in the Materials folder as a sample, so please refer to it.





- In the 3D window, AIBO's spatial center is located at the center of its body, so the motion in the display will differ somewhat from the real motion. It's important to take this difference into account. For instance, when you create a motion in which AIBO shakes its body while standing with its paws set on the floor, shoulder and hip joint" should be set at a symmetrical angle to give AIBO a smooth motion like that of a speed skater. This will cause the real AIBO to perform a shaking motion. When creating a motion with AIBO's paws remaining stationary, you can view the 3D AIBO from below in the Preview window as you create the motion.

### No floor?

- There is no floor in the 3D window. For this reason, the direction of the body in 3D might differ from the direction in real life. So you should create motions that assumes a floor in the 3D window. If you create a motion that doesn't allow AIBO to contact the floor, AIBO will fall down.

#### *Example: In a motion in which AIBO lies on its back...*

If you create a motion in which AIBO lies on its back with its 4 legs straight up in the air, its back may not necessarily make contact with the floor. In reality, this motion results with AIBO not on its back, but rather with its four feet on the floor.

### Collision?

- There is no function for checking if any of AIBO's parts have collided with each other. In real life, pay attention to this possibility and stop AIBO immediately when there is sign that a motion may result in collided parts.

### Position and Pose!

- An important tip is to create the main poses first, and then create the in-between key frames afterwards. If you start with the small parts first, like the ears or tail, you will have a hard time later on.
- AIBO interpolates automatically between key frames, so all you need to do is specify the key frames that determine the basic motion, and then let AIBO do the rest.

**Example: The front leg turns**

In a movement in which AIBO moves its right front leg all the way to the back from above, if you add no key frames in between the starting and ending poses, AIBO may try to move its leg backward from below.

Though AIBO will be able to do this in the 3D window, in real life, the front leg will collide with the floor or another part, causing damage or a malfunction to occur.

**If it's usable, reuse it!**

- Reuse anything that you've created. It is convenient to put all often-used key frames and motions into one motion file and name it "KEY FRAMES." Copy the key frames in this file when creating new actions.
- Standing up, sitting down and other motions that change AIBO's position are difficult to program, so let's leave it to AIBO to change its position. If you feel the need to create such motions, make it easier to do by copying and referencing the motions from the Sit01 or Stand01 files in the Sample library. The standing-up motion is more difficult than sitting down since it defies gravity.

For details on copying key frames, see page 84 of the User's Guide.

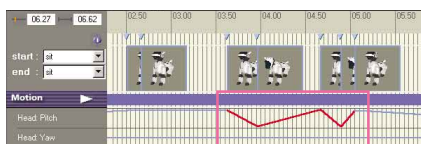
**Slow down or speed up the motion**

- Moving AIBO's legs and head at the same speed or at a slower or faster rate produces different effects. You can also produce different effects by slowing down or speeding up the entire motion. A variety of effects can be produced merely by changing the speed of the sample files!
- You can also insert identical key frames to cause a pause in the motion for greater interest.

**Take care with the speed!**

- AIBO's joints move faster than you might imagine, so you shouldn't create motions that are too fast. If the speed is excessive, the motion channel will appear in red.

Increase the number of interval frames or reduce the hinge degree to adjust the speed.



Area of excessive speed

- Inserting a key frame just before AIBO's leg touches the floor will change the speed of the motion, causing the leg to touch down slowly. If you let a part touch down at a fast speed, inertia will add to the force with which the part collides with the floor. Try to create motions that are gentle on AIBO.

### **Be gentle on the neck!**

- The neck is the most delicate part. Avoid motions that twirl the head excessively.

### **The paws are AIBO's stabilizers!**

- AIBO uses its paws to stabilize itself, so it is important that they make contact with the floor to keep AIBO from falling down. This contact, however, has the opposing effect of hindering movement by putting pressure on the bottom leg joints. When moving a leg, you should first lift the paw off the floor.

### **Your line of sight!**

- The line of sight when you create a motion is different from your line of sight when you are actually watching AIBO perform the motion. It is therefore important to test the motion on AIBO to verify that it will be performed as you imagined.

## Information about Each Model

By understanding the strengths and weaknesses of your particular model, you will be able to create a world of AIBO motions that is unique in its own special way.

### ERS-210/220

- It's relatively easy for AIBO to raise one of its front legs while in the sitting position, but raising both legs may be a little difficult. Suspend and spread the back legs and place its paws on the floor.
- On the ERS-210, use the ears and tail to create various expressions.
- On the ERS-220, frequent vertical and horizontal movements or consecutive motions of the same speed will make AIBO appear robotic.
- A rolling of the head makes AIBO look cute. This should be avoided if you want your AIBO to behave like a robot.

### ERS-310/311 series

- The head is heavy, so it is more difficult to maintain balance than with the ERS-210/220.

#### *Example: Raising the front leg*

To raise the right front leg, shift the center of gravity backward by bending the left back leg sharply. In this case, you should try to avoid lowering the head.

To raise the front leg while AIBO is sitting, shift center of gravity backward and by spreading the back legs, and then raise the front leg.

It is very difficult to raise both front legs.

- The neck motion differs from that of the ERS-210/220. To change the position of the neck, change the degree of "Head: Pitch2". The position of the neck when AIBO is sitting also differs from that of the ERS-210/220.
- Slow-moving motions are more appropriate for the ERS-310 series.
- The body and legs are wide, so it is easy for these parts to collide with each other.

## Sound: WAVE

### Using the Normalize function

- Whenever you feel the sound is small, use the Normalize function. This raises the volume to the highest level without distortion.

## Sound: MIDI

### Change the tempo!

- Changing the tempo within the same piece of music will produce varied impressions, such as slow and calm or restless.

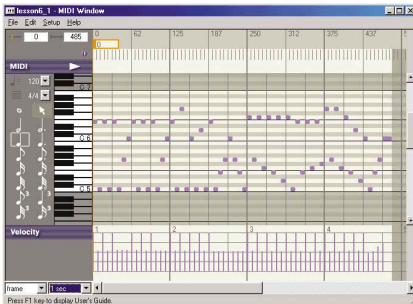
### Advanced techniques

This course introduces you to advanced techniques for creating MIDI data.

The lessons will make use of the two following examples.

- *Example 1: Shorten the melody and put in accompaniment in between.*

Shorten the main melody and add accompaniment that will make the music more complex. As an alternative to shortening the melody, you can increase the velocity and shorten the accompaniment, thereby allowing the main melody to be more clearly heard. AIBO can only emit one tone at a time, so the sound may be more music-like if you spread the chords out within the accompaniment (broken chords).

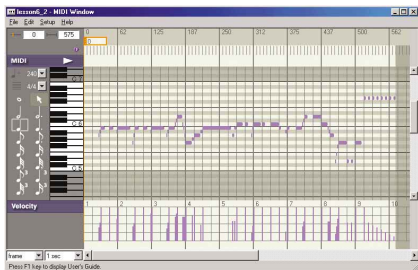


(Materials\Lesson6\_1.mid)

- **Example 2: Adding decorative effects in front of a sound to create a different expression**

Add decorative effects using similar sounds.

If you put too many decorative effects, the results may be noisy or active; if too little, the results may be calm.



(Materials\Lesson6\_2.mid)

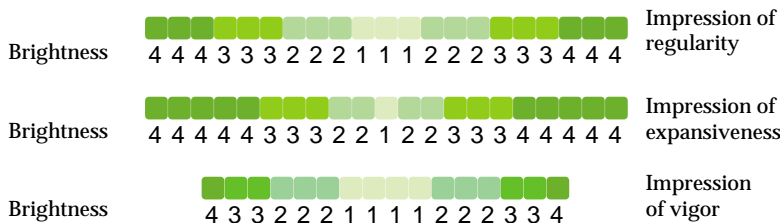
## LEDs

### Blending colors

- Multicolor LEDs are provided on the tail of the ERS-210 or the horn of the ERS-310 series. If you light up several LEDs together, you can produce different colors through blending.

### Using Brightness

- Using the four LED brightness levels (if provided on your model), you can make different impressions by changing the speed at which the LEDs light up. For instance, you can intensify the brightness of the LEDs in successive, equal intervals, gradually, or instantaneously, as follows:



**Adding light to the sound!**

- Synchronizing the lighting of the LEDs to the sound can be quite effective.

**Frequent light-up!**

- Lighting up the LEDs at a high rate can produce a vigorous impression, but it can sometimes be annoying, so be careful!

This ends the lessons for Action Composer.

Did you understand the techniques for creating elements and motions?

Create many original elements and motions and have a lot of fun with AIBO!

Even if you are not good at programming, you can make programs easily by saving the action library that you created and then use it with Quick Behavior Arranger. Putting your energies into the creation of actions is one of the enjoyments provided by AIBO Master Studio!

In the Sample folder, you will find an extra action library with samples. Make use of them in your own actions.

Printed on recycled paper

Printed in Japan

