MicroWorlds EX™

ROBOTICS EDITION

Cricket Edition

LCSI®

MicroWorlds EX Robotics

Cricket Edition

by Alain Tougas and Susan Einhorn

Legal deposit, 2nd semester 2003

ISBN 2-89371-537-0

Printed 9-03

Table of Contents

Section 1: Getting Started

Introduction

The process of designing and building projects provides students with an opportunity to apply many curricular concepts in an authentic, real-life problem-solving activity. These projects are both hands-on and "minds-on", as students set goals, test hypotheses, and reflect on the questions that arise as they build computer-controlled inventions.

What You Need

You will need one Interface Cricket transmitter for each computer that you're using. Each group of students will require at least one Cricket. A maximum of two devices (motors and lights) and two sensors can be connected to each Cricket. So, for example, if there are six student groups, each developing one project, you should have, as a minimum:

- six Crickets and six Interface Crickets
- 12 motors and 12 lights
- six touch sensors and six light sensors

Other Materials

Here's a list of suggested materials you may want to have on hand for your building projects:

- Drinking straws and popsicle sticks
- Bottle caps
- Sturdy rubber bands of different sizes
- String or rope
- Moderately thick cardboard
- Boxes of different sizes
- Plastic cylindrical containers (for example, from candy)
- Tape
- Small wooden dowels
- Any other fun and interesting items you can find

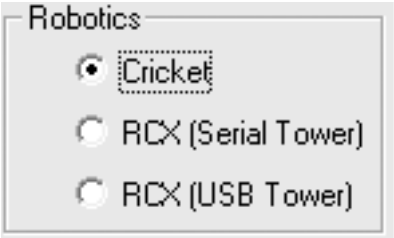
Installation

Simply connect the Interface Cricket to your PC's serial port following the instructions provided with your Cricket kit, or those from the Cricket's web site at www.handyboard.com/cricket.

On a Macintosh® equipped with USB ports, use a USB to Serial adapter, as recommended by the Cricket manufacturer.

Setup

Start MicroWorlds EX Robotics. When the Welcome Screen appears, select Free Mode. Choose **Preferences** in the File menu and select Cricket:



Then, use the Serial Port drop down menu to select the port to which the Interface Cricket is connected:

If you have made any change to the Preferences Panel, MicroWorlds EX Robotics asks you to reboot the application. You do not have to reboot the PC.

The Cricket and the Interface Cricket

This section describes briefly the features of the Cricket that are explicitly addressed by the MicroWorlds EX Robotics primitives. Refer to the documentation that came with your Cricket kit or the Cricket's web site for more information.

On the Cricket

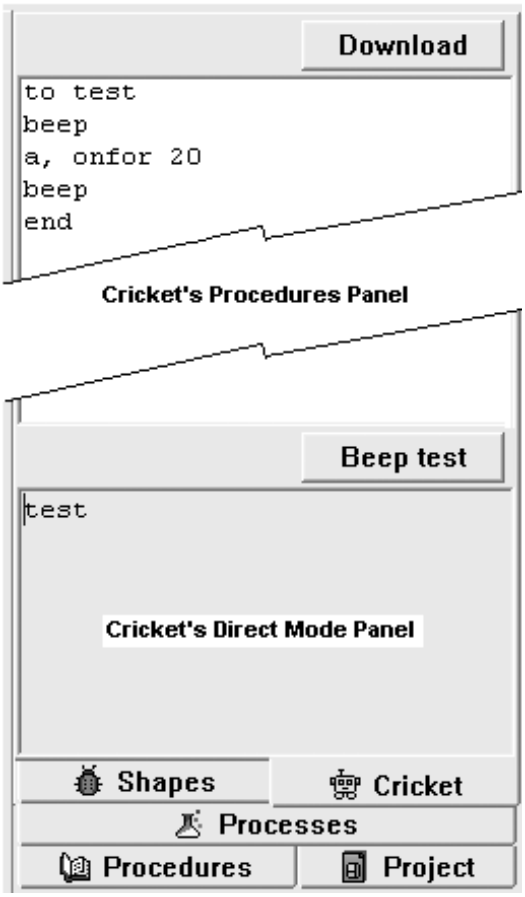
- Use the **On-Off** switch to turn the Cricket on and off;
- Use the **Run/Stop** switch to run either the most recently downloaded procedure named start, or, if there is no **start** procedure, the procedure appearing at the top of the Crickets Procedures Panel (the first procedure) in the set of procedures most recently downloaded to the Cricket;
- The motor ports A and B are used to connect motors and lights;
- The sensor ports A and B are used to connect the different sensors.

On the Interface Cricket

- Use the **On-Off** switch to turn the Interface Cricket on and off.

The Cricket Tab and the Beep Test

The Cricket Tab contains two areas: the Procedures Panel and the Direct Mode Panel. The **Download** button is used to download the procedures to the Cricket. The **Beep Test** button on the separator between the two panels is used to verify the communication path (line of sight) between the Interface Cricket and the Cricket.



To test the communication path between the Interface Cricket and the Cricket, turn on the Cricket and the Interface Cricket, place them facing one another and click on the **Beep Test** button. The Cricket should beep. This is exactly the same thing as using the beep command in the Direct Mode Panel.

Section 2: First Steps

Using the Direct Mode Panel

The direct mode allows you to send instructions to the Cricket line by line, just like you would send instructions to MicroWorlds EX using the MicroWorlds EX Command Center. Remember, however, that the instructions typed in the Cricket Direct Mode Panel are NOT executed by MicroWorlds EX. They are sent TO the Cricket and executed BY the Cricket. Therefore, you must maintain the Cricket in line of sight with the Interface Cricket (a Beep Test will confirm this) and you should expect a delay between the moment you press **Enter/Return** and the moment that the instruction is executed. Typing:

```
a, on
```
(press **Enter/Return**)

```
off
```
(press **Enter/Return**)

... does not have the same effect as:

```
a, on off
```
(press **Enter/Return**)

In fact, the instructions are executed so rapidly, one after the other in the last example, that the motor will not have time to spin noticeably.

The Direct Mode Panel has two main purposes:

- Sending plain robotics instructions to the Cricket (no procedures have been downloaded);
- Launching any downloaded procedure (assuming that some have been downloaded), including the **start** procedure or the first procedure in the Cricket Procedures Panel, without having to press the **Run/Stop** switch to launch it.

Use the Direct Mode Panel to test Instructions

Test your instructions to see if the Cricket behaves according to your expectations. For example, you may try different values as input for the command **onfor** or **setpower** to find out what is best for your purposes. If the device that you have constructed is likely to move (if you build some kind of vehicle), it is advisable that you put your vehicle on "blocks" so the wheels don't touch the ground when you test your instructions.

In-Sight Use versus Autonomous Devices

In-sight use corresponds to the cases when there is a direct line of sight (communication) between the Cricket and the Interface Cricket. Autonomous devices are those that can run without being in sight of the Interface Cricket. Their actions can be triggered by pressing the **Run/Stop** switch on the Cricket.

The in-sight usage method is what you should choose when the Cricket must communicate with MicroWorlds EX to send data such as sensor or timer values.

When creating autonomous devices, on the other hand, the **Run/Stop** switch on the Cricket is your only way to trigger the Cricket's actions. See the special procedure name **Start** in the section *Writing and Downloading Procedures*.

Sometimes the line of sight between the device and the Interface Cricket may have to be maintained only momentarily. If the Cricket is triggered by some event taking place in your MicroWorlds EX project (an instruction from the Command Center, a button or a programmed turtle or color), the Cricket must be in line of sight until this triggering event takes place, and longer if additional commands are going to be issued from MicroWorlds EX. See *Writing and Downloading Procedures*.

Motors, Lights and Sensors

This section describes the use of motors, lights and sensors. The examples for motors and lights commands are all executed in the Cricket Direct Mode Panel. The examples for sensors involve procedures that are downloaded.

When trying the examples below, make sure that the Cricket is on and it is in line of sight with the Interface Cricket. Type the instructions in the Cricket Direct Mode Panel. Perform a Beep Test before you start – when clicking on the **Beep Test** button above the Direct Mode Panel, the Cricket should beep. This is exactly the same thing as using the **beep** command in the Direct Mode Panel.

Motors and Lights

Connect motors or lights to ports A and B on the Cricket. Type the following commands in the Cricket Direct Mode Panel.

This instruction turns on the motor or light connected to port A:

```
a, on
```
(press **Enter/Return**)

Notice the **a**, at the beginning. This means that all the motor or lights commands that follow are addressed to whatever is connected to port A. The commands **b**, addressed port B in the same manner.

This instruction turns it off:

```
a, off
```

You don't have to type **a**, again if you are already talking to port A.

This instruction turns the motor or light on for a specific time (one second and five seconds in these examples). The input to **onfor** is in tenths of a second.

```
onfor 10
onfor 50
```

This turns on the motors or lights connected to ports A and B, both at the same time.

```
ab, on
```

Turn off both motors or lights.

```
off
```

The next instruction turns on the motor or light connected to port A for two seconds, then the one connected to port B for two seconds. **A, onfor** is completed before **b, onfor** is executed.

```
a, onfor 20 b, onfor 20
```

This turns on the motors or lights connected to ports A and B for two seconds, both at the same time (contrast with the previous instruction).

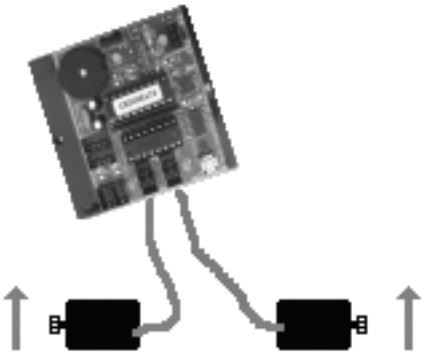
```
ab, onfor 20
```

The following sequence of instructions turns on the motor connected to port A, reverses its direction (**rd** stands for **r**everse **d**irection) and stops it. Execute these instructions as three separate lines to see the effect. **A, rd** has no effect on lights.

```
a, on
rd
off
```

The following sequence turns on the motors connected to ports A and B, sets the direction for motor A to one direction (called **thisway**) and sets the direction for motor B to the other direction (**thatway**). **Thisway** and **thatway** have no effect on lights. Notice that in these four instructions, we have to specify the port we are "talking to" each time because it is different each time.

```
ab, on
a, thisway
b, thatway
ab, off
```



This instruction sets the power level for the motor or light connected to port A

```
a, setpower 3
onfor 30
setpower 7
onfor 30
off
```

Sensors

Sensor commands cannot be used in the Direct Mode Panel because they are reporters (they return their values) and the Direct Mode Panel can only SEND commands to the Cricket. It cannot receive the values sent by the Cricket.

In fact, the sensors are used within robotics procedures either for control purposes (for example, if the switch is pressed, stop the motor) or to send values to MicroWorlds EX (for example, print the temperature). When the sensors are used by the Cricket to control its own action, the Cricket can be called an "autonomous device" because once the program is downloaded, the Cricket can "live on its own". When the sensors are used to transmit data to MicroWorlds EX, the Cricket must maintain a line of sight to the Interface Cricket in order to be able to interact with MicroWorlds EX.

To test the examples in this section:

- Create a text box;
- Import the ReadCricketMessage turtle from the Loadable Turtles directory;
- Click on the turtle to start its **OnClick** instruction.

Whenever the Cricket sends a message to MicroWorlds EX, the turtle prints the message (a numerical value) in the text box. This is a snapshot of the ReadCricketMessage turtle's Procedures Tab and Rules Tab:

Procedures Tab

Rules Tab

Note that the turtle communicates with the Cricket every few seconds to check for incoming messages. You must maintain the line of sight from the Cricket to the Interface Cricket while the procedure is running. The turtle spins to indicate that it's running its program. If the communication between the Cricket and the Interface Cricket is broken, MicroWorlds EX displays an error message and the turtle stops. Reset the communication path (line of sight) and click on the turtle again to start its instruction.

If you can't find the ReadCricketMessage turtle, you can make one following these instructions:

- Open a new project or go on a new page.
- Create a new text box.
- Create a new turtle.
- Open its backpack.
- Click on its Procedures Tab.
- Write the procedure in the illustration above.
- Click on its Rules Tab.
- Type ReadCricketMessage in the **OnClick** field, as in the illustration above. Set the mode to **Forever**.
- Close the backpack.
- Click on the turtle to start it. It should spin as an indication that it is currently running (and catching messages).

When trying the examples below, make sure that the ReadCricketMessage turtle is running and verify that the Cricket is turned on and in the Interface Cricket's line of sight. Perform a Beep Test before you start – when clicking on the **Beep Test** button above the Direct Mode Panel, the Cricket should beep. Also, make sure that you have a text box on your MicroWorlds EX page.

Connect a switch to the sensor port A and a different sensor (light, temperature...) to sensor port B on the Cricket.

In the Cricket Direct Mode Panel, type the following commands:

```
sendmessage 100
```

This is just a test. When you type this instruction in the Cricket Direct Mode Panel, the instruction is sent to the Cricket and executed BY the Cricket.

Therefore, the Cricket sends the message "100" to MicroWorlds EX. The turtle should catch the message and print the number in the text box. **Sendmessage** can only send numbers between –32768 and 32767 to MicroWorlds EX. It cannot send letters or words.

Try the following instructions. Always type them in the Cricket Direct Mode Panel.

```
sendmessage switcha
```

The value **0** (switch not pressed) or **1** (switch pressed) is printed in the text box. These 0 and **1** values can be used as input for the primitives **if**, **ifelse** and **waituntil**, as well as **and**, **or**, and **not.**

The command **switchb** addresses port B in the same manner.

Try this instruction in the Cricket Direct Mode Panel:

```
sendmessage sensorb
```

The value sent by the Cricket is printed by MicroWorlds EX. It is a raw value between 0 and 255. When using the primitive **sensora** and **sensorb**, you should calibrate your sensor: experiment with your device to determine the minimum and maximum values that you will get during your experiment. Use these values as threshold values in your procedures. The command **sensora** addresses port A in the same manner.

For example, you decide to make an electronic pet-plant and use a light sensor to figure out the lighting conditions. Set your "bright" environment – turn the light on, open the curtain, or do whatever you wish to do to make it bright. Then read the sensor's value in MicroWorlds EX (see *Sensors*, above.)

Next, do a reading for the "dark" environment.

Perhaps you recorded a value of 85 for the bright environment and 52 for the dark one. Use values slightly higher (for the dark value) and slightly lower (for the bright value) in your procedures as the threshold values:

```
to plant
if sensora > 75 [grow]
if sensora < 65 [ rest]
end
```

Or

```
to plant
waituntil [sensora > 75]
grow
waituntil [sensora < 60 ]
rest
end
```

Use **>** and **<** instead of **=** because you may never get the exact values.

Writing and Downloading Procedures

This section describes the process of writing and downloading procedures to the Cricket.

Writing Procedures

Procedures in the Cricket Procedures Panel follow almost the same rules as those of MicroWorlds EX. They must start with **to** followed by the name of the procedure, and they end with the word **end** on a line by itself. However, procedures cannot have inputs. You can use Cricket variables instead (see *Cricket Variables* below).

IMPORTANT: If there is a procedure named **start**, pressing the **Run/Stop** switch on the Cricket runs that procedure no matter where the procedure is in the list of procedures. If there is no procedure named **start,** pressing the **Run/Stop** button runs the *first* procedure listed in the Cricket Procedures Panel.

Downloading Procedures

When you are done writing one or more procedures, place the Cricket in front of the Interface Cricket and turn it on. Perform a Beep Test and, if this is successful, click the **Download** button. Check for error messages. Since the Cricket doesn't have a Command Center of its own, MicroWorlds EX verifies the procedures before downloading them... and refuses to download them if errors are found. For example, if you use a primitive that is not one of those that can run in the Cricket (for example, **print**), the following message is displayed:

```
Cannot download because I don't know how to print.
```

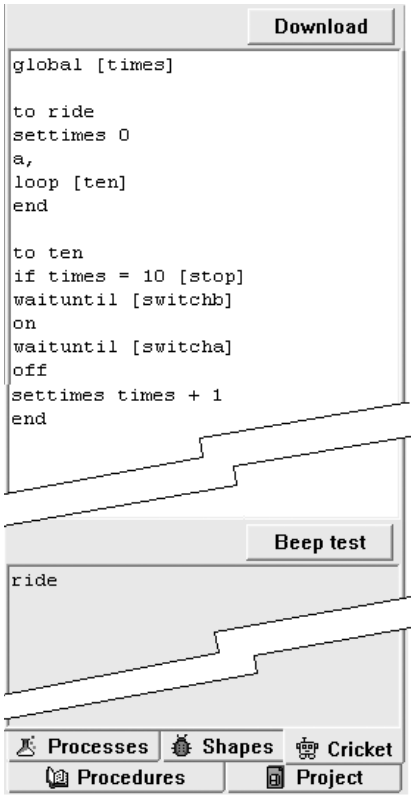
If no errors are encountered, the procedures are downloaded to the Cricket. When the **Download** button returns to its normal state, the process is finished. You can also check the orange LED behind the infrared sensors on the Cricket. Note that the procedures that you download *replace* existing procedures in the Cricket.

Once the procedures have been downloaded, you can take the Cricket out of sight of the Interface Cricket if you will, and trigger the **start** procedure or the first procedure found in the Procedures Panel by pressing the **Run/Stop** button.

Cricket Variables

The primitive **global** allows you to define a variable in the Cricket. One command and one reporter are associated with the variable name. The word **set** followed by the variable name is used to give a value to the variable. The variable name itself is used to report its value.

You must use the command **global** and its input in the Cricket Procedures Panel but outside of any of the procedures, as in the following example:



Global's input must be a list, even if it contains a single word. Use only one **global** instruction but you can define several variable names at once:

```
global [counter delay result]
```

Section 3: MicroWorlds EX Robotics Projects

The projects in this book are a starting point meant to fuel the idea generation process for both you and your students.

For the Teacher

Objectives and Standards Addressed

Students develop abilities of technological design and an understanding about science, math and technology as they:

- Identify appropriate problems for technological design.
- Design a solution or product.
- Implement a proposed design.
- Evaluate completed technological design or products.
- Communicate the process of technological design.

(National Science Education Standards)

The Design and Implementation Process

These activities highlight 4 main components of the design and implementation process:

- **D** Define the idea
- **C** Construct
- **P** Program
- **R** Reflect

Like any learning experience, design and implementation is a cyclical, not linear, process – reflection should occur after defining the idea, after construction, and after programming. This process of reflection may lead to new solutions and changes in any of the components, causing a cycle to be repeated.

Working in Groups

There's much to be gained by having students work in design groups or teams. Working as a group more closely reflects a real-world design and building projects. Among other benefits, collaboration leads to insights that the student working individually, may not have. Through this collaborative process, students not only build their own knowledge on a subject, they also learn from and contribute to the knowledge of others in the group.

As they work through their project, students should be encouraged to record their thoughts, ideas, questions, and concerns in a paper journal or digital journal such as Journal Zone™. Journal writing and sketching gives students an opportunity to make explicit normally covert processes. It helps students improve their communication skills in the area of science and technology. Journal writing gives the teacher an opportunity to not only view the student's final product, but to also understand the problem-solving strategies and thinking that the student used.

Project Contents

Each activity includes the following sections:

Activity Description	– What it is
Define	– Ideas on getting started including ideas for the onscreen component of the project.
Construct	– Some suggested materials and construction tips
Program	– Some programming hints or ideas for both off- and onscreen components
Reflection	– Sample journal entries related to each project
Stretch Your Thinking	– Project extension ideas
Going Further	– Further explorations

Projects and the Curriculum

The projects suggested here are very open-ended. There are many different designs that can be used and many ways to program them. Two projects are described in detail in the next section of the book, but many project ideas are also included in *Section 4: Technical Information* and in *Section 5: Cricket Vocabulary*.

The two projects are:

Project I – The Great Communicator – Communication Devices

In this project, students build a communication device and its "language."

Curricular links include:

Language Arts	– Students gain an understanding of language and communication
Social Studies	– History of Science and Technology, Impact of major discoveries and events on science and technology
Science and Technology	– Understanding the nature and process of technological design

Project 2 - Dancing On A String

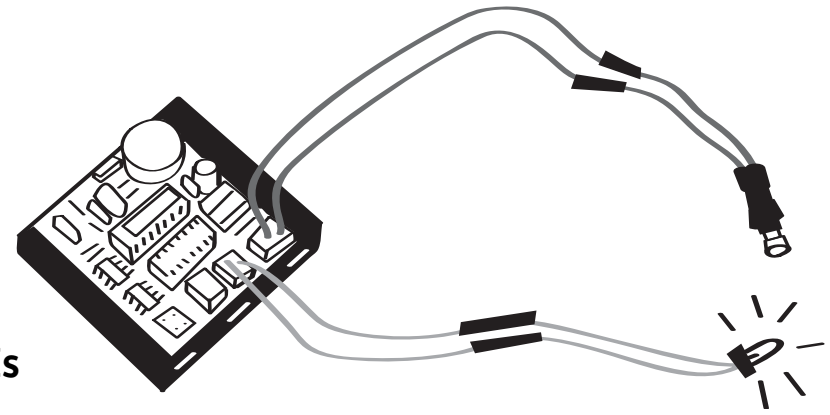
In this project, students build a puppet or other mechanical toy.

Curricular links include:

Science and Technology	– Motion of Objects: pulleys and gears
Social Studies	– History of Science and Technology
Anthropology	– Early Civilizations: how institutions influence culture
Language Arts	– Using visual language to communicate effectively for different purposes
Visual Arts	– Puppets; puppet theater

Project I

The Great Communicator-Communication Devices



What It Is

With instant messaging, email, and cell phones, it's easy to communicate with anyone, anywhere. But that wasn't always the case. The first email message appeared in the early 1970's. The telephone wasn't invented until 1876. And the first public display of Morse's electric telegraph was in 1844. Does that mean that before that time was no communication? Not at all! Paul Revere used his famous light code - One if by land, two if by sea – to warn fellow revolutionaries in the American colonies that the British soldiers were coming.
Smoke signals were very early "wireless" communications. And in Biblical times, rams' horns and other instruments were used to send messages over long distances.

D Define the Idea

Do some research on simple communication devices or methods. How do these devices send their signals? How does the recipient perceive the message – by seeing something, hearing something, touching something? Think of a communication device you can construct. For example, you could use sound (tapping with a mallet, beeping, or the whirring of a motor) or visual information (flashing lights, flags, movement). Figure out where you would use your device and to whom you would send your messages. Is your design suitable for these circumstances? For example, if you want to send a message to someone in another room, you may decide to use sound to convey your message and not movement or flashing lights. How will your listener know what the message means? What is your "language" like? Does it use an alphabet to spell words like English or Morse code or symbols/actions that stand for whole words like Chinese?

Onscreen, you can create a presentation explaining why you chose your design and who your expected audience is. You can also create a digital "dictionary" for the language used by your communication device. You can also design a way for other users to type in sentences and see their sentences communicated by your device. By clicking on "word buttons" on the screen, the information can be transmitted to your Cricket, which can then communicate your message via your device.

C Construct

The model pictured at the beginning of the project uses a light to convey a series of words. This is extremely simple to build but it can only communicate if the person receiving the message can see the device. You could also use a combination – a light and a motor driven device to convey a word. Create your own design for your device.

P Program

Decide on one or two words to define and communicate. Figure out a system for sending the information. For example, you may want to communicate the word "danger." Your code may be one light on for one second, then the other light on for half a second, then the first light on again for one second. You could define the following procedure in the Cricket Procedures Panel:

```
to danger
a, onfor 10
b, onfor 5
a, onfor 10
end
```

Download the procedure to the Cricket:

1- Make sure the Interface Cricket is communicating with the Cricket.

2- Click the download button on the Cricket Tab.

Test your procedure.

1- Make sure the Interface Cricket is communicating with the Cricket (in its line of sight).

2- In the Cricket Direct Mode Panel type:

```
danger
```

(You could also press the **Run/Stop** button on the Cricket.)

Next, you could create a procedure for a word indicating that the danger has passed - a **safe** procedure.

```
to safe
b, onfor 5
wait 5
onfor 10
end
```

Once you've finished creating new procedures, download all the contents of the Cricket Procedures Panel to the Cricket. To test each procedure, type the procedure name in the Direct Mode Panel.

(Remember, if there's no procedure named **start** in the Procedures Panel, the **Run/Stop** button runs the first procedure listed in the Procedures Panel. The only way to run any other procedure is to type the name of the procedure in the Direct Mode Panel.)

As you create your words, decide if each word will follow the same pattern. For example, will all words have three components to their signals? Or will different types of words have different numbers of components, for example, will nouns have three and verbs have two components? Will you have verbs at all or just nouns? Are there types of words you don't need?

Add a few more words – words that can be used to convey simple messages. For example, you could add these words:

```
to teacher
a, onfor 20
b, onfor 5
wait 5
b, onfor 5
wait 3
end

to test
a, onfor 10
wait 3
a, onfor 10
wait 3
a, onfor 10
wait 3
end
```

Once you've finished adding word procedures, download the contents of the Cricket Procedures Panel to the Cricket.

Create an onscreen transmitter to send messages. First, create a text box. Then create buttons that print the names of the procedures (words) you defined.

Click on the Button tool in the Toolbar and click on the page.

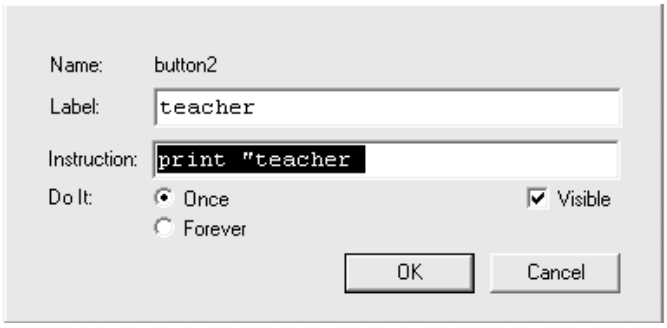
1- In the Instruction field, type:

```
print "teacher
```

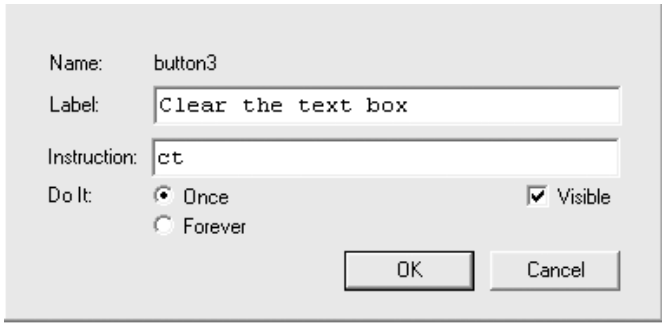
Use whatever word you defined.

2- Write the word in the Label field.

3- Keep Do It set to **Once** and click **OK**.



Follow these steps to create a button for each word that you defined. Also, add a button to clear the text box:

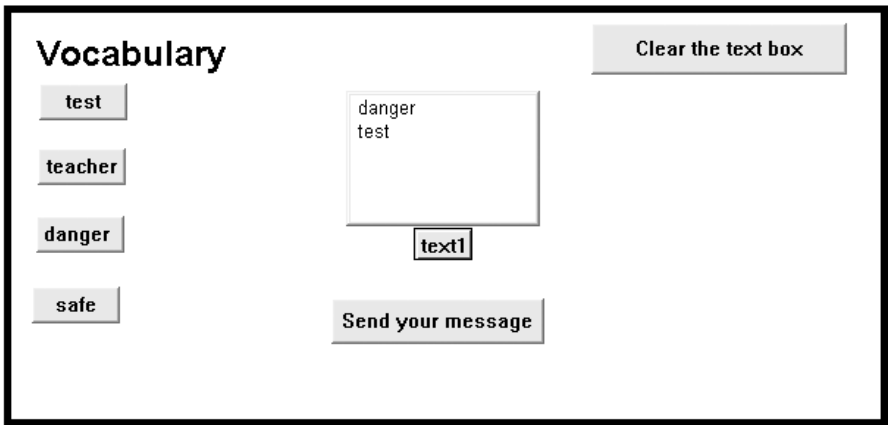


Finally, add a button that sends the communication command to the Cricket. It should look like this:

Sendremotecommand *instruction* sends a signal to the Cricket to run ***instruction***, which can be either a procedure that has been previously downloaded to the Cricket or a primitive. **Text1,** the name of the text box, reports whatever the text box contains. In this case, the words it contains are procedure names. **Sendremotecommand** tells the Cricket to run these procedures.

Before using this button, make sure the Cricket is in the Interface Cricket's line of sight (they're communicating). If you're not sure, click on the **Beep Test** button on the Crickets Tab.

Set up your page.



Test out some different word combinations. You may find that you need to add a "separator" at the end of each word procedure so that there's a little pause between words (you don't want any run-on words!). The word separator can be a wait instruction or an audio signal (created with **beep** or **note**). Once you change the procedures in the Cricket Procedures Panel, download all the procedures once again to the Cricket.

Is it easy to remember what each word is? You may need to rethink your "language" so that the patterns are easier to remember.

Remember, if you change your word procedures or add more procedures, you must download all the procedures to the Cricket again!

R Reflection

At first we made up any combinations of actions for words, but we couldn't remember what each one meant. Then we decided to create specific types of patterns. That made it a little easier.

We wanted to find a way to communicate without looking at the Cricket (no lights!). We needed to be able to create two different sounds with the motors and they needed to be loud enough to hear in another room (that was our test) and strong enough to withstand many messages. Agreeing on the ideas took more time than we expected.

Stretch Your Thinking

Develop a language and vocabulary with the rest of your class. Then create communicators you can wear. Each Cricket should have a start procedure that combines several of these words into a meaningful "sentence". Wear your messages and see if others understand them.

Going Further

You may want to find out more about:

- languages that use ideograms, such as Chinese.
- sign language.
- how new languages develop. You may want to consider the unique language of electronic messaging (lol, cya, ttys, etc.).
- non-human forms of communicate. How do computers communicate? How do animals communicate? You may want to check out Karl von Frisch. He conducted research on how bees communicate through their "dances." Is this real communication?

Final Thought

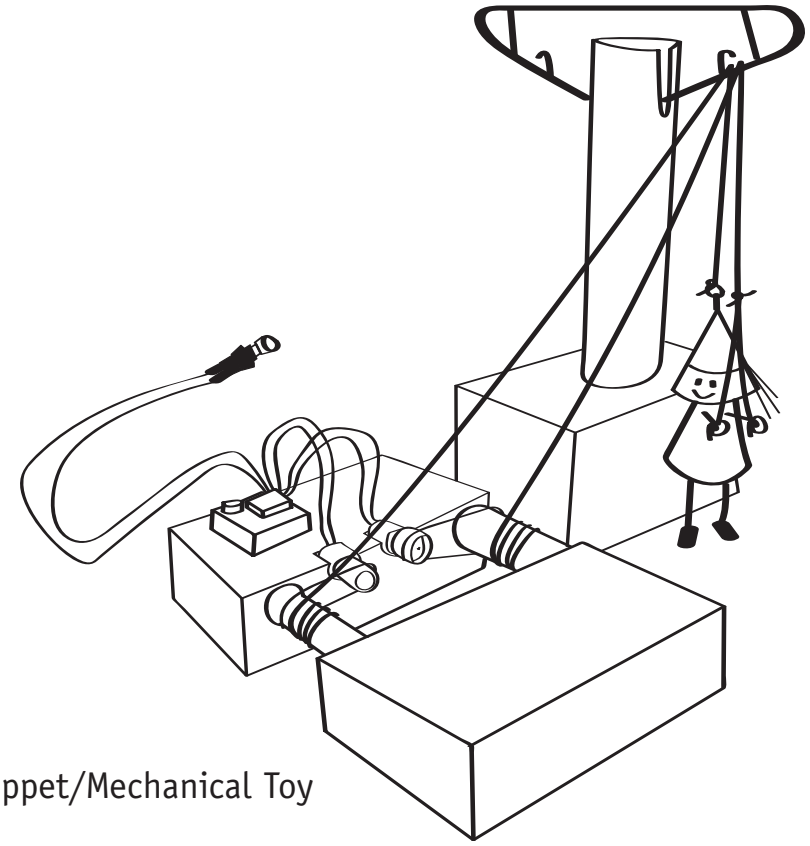
According to the National Science Education Standards,
"Technology influences society through its products and processes." *

Think about this statement in respect to this activity.
What are your ideas?

* *National Science Education Standards,*
Chapter 6: Content Standards: 5-8, Science in Personal and Social Perspectives,
National Academy of Sciences, 1995.

Project 2

Dancing On A String



What It Is – Puppet/Mechanical Toy

String-operated puppet-like toys have been found dating from 2000 BC in Egypt and from 2400 BC in the Indus Valley civilization of Pakistan. It's believed that the earliest puppets were used in religious rituals. Nowadays, puppets are used in different types of performances as well as for play.

D Define the Idea

Design a simple puppet that can be moved using motors. Your puppet may be a character from a well-known story, someone from history, or it may be a character you create. Write a description of your puppet's personality and its likes and dislikes. How can you design your puppet so that other people know what your puppet is like?

Onscreen, create a turtle with a similar personality. How can you show the personality with turtle movements? Other ideas:

- Have each group member create a different onscreen turtle-character from history or from a story. Import all of the turtles into one project and get them to interact or "act" in a scene.
- Create a colourful background and an animation starring your turtle-character. When the turtle reaches a certain part of the "scene" have the puppet begin to move. Press a touch sensor to have the puppet stop and the onscreen action continue.

C Construct

The picture above is of a puppet that "dances" by moving its arms and body up and down. Some suggested materials for this model are listed below.

For the control mechanism:

- Plastic or thick cardboard cylinders for the larger pulleys.
- Plastic cylinders – like the tops of glue sticks - for the smaller pulleys
- Metal axles. These could be made from thick metal paper clips, for example
- Elastic bands (thick and thin ones)
- 1 or 2 motors, depending on your design.
- 1 Cricket for every two motors.

For the puppet:

- Paper or thin plastic cups
- Colored construction paper
- Tape
- Some equal-sized cardboard boxes as support
- Some sort of support above the puppet
- Yarn

Some additional construction notes:

- Balance is key in this project. You may need to tape the motor down on one of the cardboard boxes.
- The gear for the yarn should be long enought to have room to wind the yarn.
- Use fairly long pieces of yarn so the puppet has room to move.

P Program

The design pictured has two motors using the same Cricket and program – one motor moves the hands up and down and one moves the whole body. You may want to start by getting one part to work before working on the second.

Sample procedures:

```
to handsup
a, thisway onfor 3
end

to handsdown
a, thatway onfor 3
end
```

Use these procedures to create dance programs. For example, your puppet could lift its hands up, put its hands down, lift its body up and then lower its body down. What other actions can your puppet do? Create a **start** procedure to run all the other procedures. It may look like this:

```
to start
repeat 10 [handsup handsdown]
end
```

Once you've created all your procedures, download them to the Cricket:

1- Make sure the Interface Cricket is communicating with the Cricket.

2- Click the **Download** button on the Cricket Tab.

You may decide to start your puppet at a specific time during an onscreen presentation that you create.

For example, you decide to create a procedure that makes the turtle-character move. You want the procedure to start when you click on the turtle.

1- Open the Rules Tab in the turtle's backpack.

2- Write the procedure's name in the **OnClick** instruction line.

3- Set the mode to **Forever**.

Now, when you click on the turtle, the turtle-character moves across the screen.

Next, you decide that when the turtle reaches a specific part of the background scene, the turtle stops moving and the puppet begins dancing. To do this, you should:

1- Make sure the Cricket is lined up with the Interface Cricket in order to receive the signal to start.

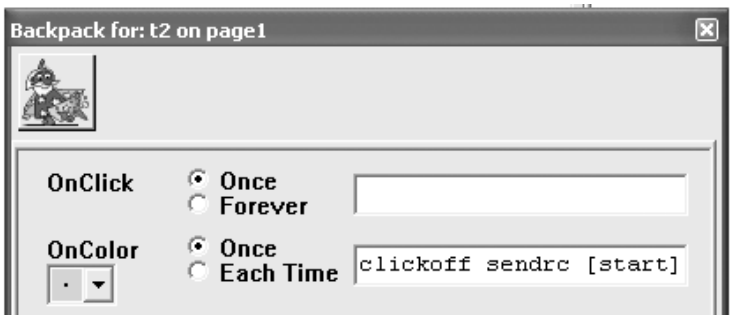
2- Select a color for that specific part of the screen – a color that isn't used in any other part of the screen, for example, lime (your color may be different). Careful! This means any shade of lime will be able to trigger the turtle to stop, so don't use it in other parts of the screen.

3- Open the Rules Tab of the turtle's backpack and select the lime color box (or the box corresponding to the color you used) in the dropdown menu beneath the word **OnColor**.

4- In the **OnColor** instruction line, type:

```
clickoff sendremotecommand [start]
```

5- Leave the mode set to **Once**.



When the turtle is over any shade of the lime color you programmed, first, the turtle-character stops moving (clicks off). Then MicroWorlds EX sends an instruction to the Cricket to run the **start** procedure that you have previously downloaded to the Cricket.

If you want the turtle to begin moving again once the puppet stops dancing, you must add an instruction to the **start** procedure that is downloaded to the Cricket:

1- Change the **start** procedure in the Cricket Tab so it looks like this:

```
to start
repeat 10 [handsup handsdown]
sendmessage 1
ab, off
end
```

Sendmessage sends a message to MicroWorlds EX. In this case, it doesn't matter what the message is as long as it's a number.

2- Download all the procedures from the Cricket Procedures Panel to the Cricket once again by clicking the **Download** button.

Next, you must add another instruction to the **OnColor** instruction line for lime:

3- In the Rules Tab of the turtle's backpack, change the **OnColor** instruction line for lime to:

```
clickoff clearbuffer sendremotecommand
[start] waituntil [message?]clickon
```

First, MicroWorlds EX uses **clearbuffer** to tell the Cricket to remove any previous message from the Cricket's message buffer. After sending the remote command, MicroWorlds EX waits to receive the new message. Once it gets it, the turtle clicks on again.

Note: It's a good idea to put these instructions into a procedure in the Procedures Tab in the turtle's backpack. Then all you need to write in the **OnColor** instruction line for lime is the name of the procedure.
For example:

```
to puppetdance
clickoff
clearbuffer
sendremotecommand [start]
waituntil [remotemessage?]
clickon
end
```

R Reflection

- We discovered that the cylinders we used for pulleys needed to have edges. Otherwise, the elastic bands would fall off.

- If the elastic bands weren't stretched tightly, they wouldn't work.

- It was kind of tricky to build pulleys out of the different things we found. You need to be very patient!

- We had to make sure the Interface Cricket and the Cricket were lined up to get the puppet started at the right time and the turtle then moving at the right time.

Stretch Your Thinking

- Add sensors to control your puppets dance.
- If you made a puppet with strings, try to create one that moves with rods.
- Get multiple puppets to dance together to music.
- Try to make a different type of simple mechanical toy.
- Import the character-turtles from all your group members into one project and use **OnTouching** instructions to get then to interact according to their personalities.

Going Further

You may want to find out more about:

- how puppets were used in ancient cultures.
- different types of puppets, for example rod-operated puppets or shadow puppets– their construction and their history.
- other types of mechanical toys, automata (mechanical sculptures), animatronics, or coocoo clocks.

Final Thought

"Puppetry is at once an art, a form of communication, and a technology."*

What does this mean? Do you agree?

* *www.puppetools.com,* 2001.

Section 4: Technical Information

Communication (Data) Between the Cricket and MicroWorlds EX

This section describes how to use the primitives that let you transfer information between MicroWorlds EX and the Cricket. Use these techniques in projects that combine onscreen action and robotic devices.

One way to use this type of interaction is to have an onscreen event trigger an action by the Cricket. For example:

- A turtle winning a race on screen triggers a light show executed by the Cricket;
- Information about a robotics project is presented onscreen and the robotic device is triggered by a button or turtle in the MicroWorlds EX project;
- An onscreen quiz can animate a robotic device to make a "game-show" action based on the right and wrong answers provided by the player;
- MicroWorlds EX can transmit numerical values to the Cricket.

Note, however, that infrared communication is slow and you should not attempt to create projects where the synchronization between the onscreen and the robotics actions is critical.

A second way to integrate onscreen action and robotic devices is to have MicroWorlds EX Robotics and the Cricket communicate data in one direction or the other. For example:

- The Cricket sends sensor values (light, temperature, etc.) to MicroWorlds EX, where they can be recorded in a text box and/or in a graph.
- A MicroWorlds EX project (onscreen and interactive) sends data to the Cricket to control its behavior.

Some primitives are dedicated to communication between MicroWorlds EX and the Cricket. When using these primitives, the Cricket must maintain a line of sight to the Interface Cricket

When this is executed within MicroWorlds EX Robotics...	The Cricket must use or respond with...	Example
download *word*	Nothing. After the **download** process, the procedures can be run by the Cricket. After a download instruction, you can either press the **Run/Stop** switch on the Cricket to launch the procedure named **start**, or, if there is none, the first procedure in the Cricket Procedures Panel, or you can issue a **sendremotecommand** instruction from MicroWorlds EX. The input for **download** is a word. The most common input for **download** is the contents of a text box. You can take advantage of the fact that the name of a text box reports the text box' contents as a "long word". Therefore, the name of a text box can be used as input for **download**. In the instruction **download text1** **text1** reports its contents as a long word. This long word is taken by **download** as input. When the text "arrives" in the Cricket, it is interpreted as normal procedures, as if they originated from the Procedures Panel in the Cricket Tab.	**Text1** contains the procedure **goforit**. **Text1** reports its contents to **download** which sends it to the Cricket. The next instruction, **sendremotecommand "goforit,** starts the procedure in the Cricket. These instructions can be typed in the MicroWorlds EX Command Center or executed from a button, a programmed color or turtle. `download text1` `sendrc "goforit` There are actually three ways to start the procedure in the Cricket: • Press the **Run/Stop** switch on the Cricket, • Type **goforit** in the Cricket Direct Mode Panel; or, • Use **sendremotecommand "goforit** from within MicroWorlds EX.

When this is executed within MicroWorlds EX Robotics...	The Cricket must use or respond with...	Example
sendremotecommand *word-or-list* **sendrc** *word-or-list*	Nothing. The input is a word or a list that contains instructions that can be run in the Cricket (procedures that have been downloaded or primitives). The *word-or-list* instruction is run in the Cricket and it is linked to the **Run/Stop** button.	A motor or a light is connected to motor port A on the Cricket. These instructions are typed in the MicroWorlds EX Command Center. `sendremotecommand "on` `sendrc [a, onfor 10]`
sendremotemessage *number* **sendrm** *number*	Use **message?** and **message** to "read" the message in the Cricket. The input must be a number between 0 and 255.	There is a light or a motor connected to motor port A and the following procedures must be downloaded to the Cricket. `to start` `loop [getinputfrommw]` `end` `to getinputfrommw` `if message? [a, onfor` `message]` `end` In the Cricket Direct Mode Panel, type: **start** Type, in the MicroWorlds EX Command Center (they can also be run from buttons, clickable turtles, etc.) while the Cricket is in the Interface Cricket's line of sight. `sendremotemessage 10` `sendrm 30`
requestremotemessage *word-or-list* **requestrm** *word-or-list*	Nothing. The command is executed in the Cricket and the result is sent back to MicroWorlds EX. The input is a word or a list that reports a number when executed inside the Cricket. **Requestremotemessage** is a MicroWorlds EX reporter. You do not need to use **remotemessage?** and **remotemessage** to "read" the Cricket's reply.	There is a text box on the page and a light sensor is connected to port A on the Cricket. The following commands are executed within MicroWorlds EX (in the Command Center, by buttons, clickable turtles, etc.). The results are just examples. `print requestrm "timer` `2351` `print requestrm` `[sensora / 10]` `24`

When this is executed in the Cricket	MicroWorlds EX Robotics must use or respond with...	Example
sendmessage *number* **sm** *number*	**clearbuffer, remotemessage?** and **remotemessage.** In the example, **reflex** is executed from a button in MicroWorlds EX. Click on the button, get a few feet away from the switch and rush to press the switch when you hear the motor or see the light. The Cricket will tell MicroWorlds EX how much time it took you to react. The **reflex** procedure waits between 10 and 20 seconds and sends the instruction to run the **howfast** procedure to the Cricket. The **reflex** procedure then waits for a message to arrive from the Cricket. In the Cricket, the **howfast** procedure resets the timer, turns on the light or motor, waits for the switch to be pressed, and then sends a message containing the value of the timer (this is the time elapsed between **resett** and pressing the switch). Finally, the motor is turned off. In MicroWorlds EX (in the **reflex** procedure), **remotemessage?** reports **true** when the message from the Cricket arrives, ending the process started by the instruction **waituntil[remotemessage?].** **Reflex** then prints the contents of the message, divided by 1000 (the number of seconds).	There is a light or a motor connected to motor port A and a switch connected to sensor port A on the Cricket. The following procedure is downloaded to the Cricket. `to howfast` `resett` `a, on` `waituntil [switcha]` `sendmessage timer` `off` `end` The following procedure is defined in the MicroWorlds EX project Procedures Tab. `to reflex` `clearbuffer` `wait 100 + random 100` `sendremotecommand` `[howfast]` `waituntil` `[remotemessage?]` `pr remotemessage / 1000` `end`

Additional Information

Some robotics primitives have the same name as MicroWorlds EX primitives but work slightly differently. This section describes the differences between MicroWorlds EX and the Cricket for the primitives **if**, **ifelse**, **waituntil**, **and**, **or** and **random**.

True and False are 1 and 0 in the Cricket
If you are familiar with MicroWorlds EX, note the differences in the way the following primitives work when executed in the Cricket.

If
When executed in the Cricket, the first input of the primitive must report **1** or **0** instead of **true** or **false**.

Ifelse
When executed in the Cricket, the first input of the primitive must report **1** or **0** instead of **true** or **false**.

Waituntil
When executed in the Cricket, the first input of the primitive must report **1** or **0** instead of **true** or **false**.

Math and logic operators report **1** and **0** instead of **true** and **false**. This approach suits the requirements of **if**, **ifelse** and **waituntil**, which require **1** or **0** as input instead of **true** or **false**.

Examples**:**

```
if switcha [a, onfor 20]
ifelse switchb [a, thisway] [a, thatway]
if timer > 10000 [a, off]
waituntil [and switcha switchb]
```

In the last example, **switcha** and **switchb** report **1** or **0** depending on their state (pressed or not). **And** takes these values (**0**-**0**, **0**-**1**, **1**-**0** or **1**-**1**) as input and reports **1** if both switches reported **1**, **0** otherwise. **Waituntil** takes its input (**0** or **1**) from **and**.

This is in contrast to a similar instruction running in MicroWorlds EX (not in the Cricket):

```
waituntil [and shape = 0 heading = 1]
```

In this example, **shape = 0** reports **true** if it is the case, **false** otherwise. The same goes for **heading**. Then **and** takes these values **(true-true, false-false, true-false** or **false-true**) as input and reports **true** if both its inputs are **true**. Finally, **waituntil** takes its input (**true** or **false**) from **and**.

Inputs for And and Or

Contrary to MicroWorlds EX, the primitives **and** and **or**, when running in the Cricket, can only have two inputs. In MicroWorlds EX, these primitives accept more than two inputs if the primitive and their inputs are enclosed in parentheses.

In MicroWorlds EX:

```
if (and heading = 0 shape = 0 pos = [0 0])
[announce [I'm home!]]
```

In the Cricket

```
if and switcha sensorb > 20 [off]
```

No Input for Random

Contrary to MicroWorlds EX, the primitives **random**, when running in the Cricket, takes no input. It reports a number between 0 and 32767.

The MicroWorlds EX instruction:

```
fd random 32767
```

…is similar to the Cricket instruction:

```
onfor random
```

In order to get a smaller random number, you can use the remainder of the large number divided by your maximum desired value. The MicroWorlds EX instruction

```
fd random 20
```

...is similar to the Cricket instruction

```
onfor remainder random 20
```

In the case of MicroWorlds EX, **random 20** reports a value between **0** and **19**. In the case of the Cricket, **random** reports a number between 0 and 32767, and **remainder** reports the remainder of 32767 divided by 20, which is a random number between 0 and 19. Try this in MicroWorlds EX using different large numbers. The results are just examples.

```
show remainder 4343 20
16
show remainder 12943 20
8
```

Remember that the number 20, in the previous examples, is the second input of **remainder**, not that of **random**.

Finally, if you want a random number not starting with 0, the MicroWorlds EX instruction:

```
fd 10 + random 20
```

...is similar to the Cricket instruction

```
onfor 10 + remainder random 20
```

In both cases, the input to **fd** or **onfor** will be a number between 10 and 29.

This Cricket procedure outputs random numbers in the same manner as MicroWorlds EX:

```
to ran20
op remainder random 20
end
```

Section 5: Cricket Vocabulary

This section describes the primitives that can run in the Cricket and some that run in MicroWorlds EX to communicate with the Cricket.

Motors and Sensors

These primitives can be used to write procedures to be downloaded into the Cricket.

Primitive *Input(s)*	Description	Example
a, b, ab,	Designates the current motor or light port. Any motor or light command (**on off onfor rd setpower thisway thatway**) following this designation addresses the designated port. This type of instruction (letter(s) followed by a comma) can be typed on a line by itself or it can be followed by other instructions. It doesn't have to be typed for each instruction as long as you don't change the designated port. The comma is part of this instruction. There is no space between the letter(s) and the comma. When starting the Cricket, the default port is A.	There is a motor or light connected to ports A and B. These instructions are typed in the Cricket Direct Mode Panel. `a, on` `b, on` `rd` These instructions are `off` still for port B. `a,` `off` `ab, on wait 30 off`
on	Turns on the motor or light connected to the designated port(s) (A or B or both).	There is a motor or light connected to ports A and B. This instruction is typed in the Cricket Direct Mode Panel. `ab, on wait 20 off`

Primitive *Input(s)*	Description	Example
off	Turns off the motor or light connected to the designated port(s) (A or B or both).	There is a motor or light connected to ports A and B. This instruction is typed in the Cricket Direct Mode Panel. `ab, on wait 20 off`
onfor *number*	Turns on the motor or light connected to the designated port(s) (A or B or both) for the duration of time indicated (*number*). The duration is measured in tenths of a second. The maximum duration is 65353 (6,553.5 seconds).	There is a motor or light connected to ports A and C. This instruction is typed in the Cricket Direct Mode Panel. `a, onfor 20` `b, onfor 20` `ab, onfor 20`
rd	**R**everses the **d**irection of the motor connected to the designated port (A, B or both) so it goes in the opposite direction.	There is a motor connected to port A. These instructions are typed in the Cricket Direct Mode Panel. `a, on` `repeat 6 [rd wait 20]` `off`
setpower *number*	Sets the power level for the designated port(s) (A or B or both). The maximum power level is 8. If you use an input greater than 8, the power level is set at the remainder of the input divided by 8 (**setpower 10** is equivalent to **setpower 2**).	There is a motor or light connected to port A. This instruction is typed in the Cricket Direct Mode Panel. `a, on` `setpower 2` `setpower 8` `off` In this example, **switcha** is used to raise the power level of port A and **switchb** decreases it. If both switches are used at the same time, the motor stops. The following is typed in the Cricket Procedures Panel and downloaded to the Cricket: `global [level]` `to variablespeed` `setlevel 3` `a, setpower 3` `on` `loop[control]` `end`

(continued)

Primitive *Input(s)*	Description	Example
setpower *number*		`to control` `if switcha [setlevel level + 1]` `if switchb [setlevel level - 1]` `if and switcha switchb [off]` `setpower level` `end` The following is typed in the Cricket Direct Mode Panel: `variablespeed` Press one switch or the other a few times to see the effect.
thisway	Sets the motor connected to the designated port(s) (A or B or both) to go "thisway". The actual direction of rotation depends on how the connector is placed on the Cricket but if two connectors are placed in the same manner, setting one to **thisway** and the other one to **thatway** will make motors run in opposite directions.	There is a motor connected to ports A and B. These instructions are typed in the Cricket Direct Mode Panel. `a, thisway` `b, thatway` `ab, on` `ab, off`
thatway	Sets the motor connected to the designated port(s) (A or B or both) to go "thatway". The actual direction of rotation depends on how the connector is placed on the Cricket but if two connectors are placed in the same manner, setting one to **thisway** and the other one to **thatway** will make motors run in opposite directions.	There is a motor connected to ports A and B. These instructions are typed in the Cricket Direct Mode Panel. `a, thisway` `b, thatway` `ab, on` `ab, off`

Primitive *Input(s)*	Description	Example
switcha **switchb**	Reports **1** (**true**) if a switch connected to the designated sensor port (A or B) is pressed. Otherwise, reports **0** (**false**).	There is a switch connected to sensor port A and a motor connected to motor port A. This procedure is downloaded to the Cricket: `to ride` `a, on` `waituntil [switcha]` `a, off` `waituntil [not switcha]` `end` This instruction is typed in the Cricket Direct Mode Panel. `loop [ride]`
sensora **sensorb**	Reports the raw value of a sensor connected to the designated sensor port (A or B). The value depends on the type of sensor being used but is between 0 and 255. When using this primitive, you should "calibrate" your sensor: experiment with your device to determine the minimum and maximum values that you get during your experiment. Use these values as thresholds in your procedures.	There is a motor connected to motor port A and a light sensor connected to sensor port A. This procedure is downloaded to the Cricket: `to autodoor` `a,` `rd onfor 20` `waituntil` `  [sensora > 125]` `rd onfor 20` `waituntil` `  [sensora < 120]` `end` This instruction is typed in the Cricket Direct Mode Panel: `loop [autodoor]`

Control

These primitives can be used to write procedures to be downloaded into the Cricket. Except for **stop** and **output**, they can also be used in the Cricket Direct Mode Panel.

Primitive *Input(s)*	Description	Example
if ***1 or 0*** *(true or false)* ***list-of-instructions***	Runs the *list-of-instructions* if the first input (a condition that is either true or false) is **true** (reports **1**). Note that **=**, **<**, **>**, **and**, **not**, **or**, as well as **switcha** and **switchb** report **1** or **0**.	There is a switch connected to sensor port A and a motor connected to motor port A. This procedure is downloaded to the Cricket: `to maybereverse` `if switcha [rd]` `onfor 20` `end` This instruction is typed in the Cricket Direct Mode Panel. `loop [maybereverse]`
ifelse ***1 or 0*** *(true or false)* ***list-of-instructions1*** ***list-of-instructions2***	Runs the first *list-of-instructions* if the the first input (a condition that is either true or false) is **true** (reports **1**). Runs the second *list-of-instructions* if the condition is **false** (reports **0**). Note that **=**, **<**, **>**, **and, not**, **or**, as well as **switcha** and **switchb** report **1** or **0**.	There is a switch connected to sensor port A and a motor connected to motor port A. This procedure is downloaded to the Cricket: `to setdirandrun` `ifelse switcha` `[thisway]` `[thatway]` `onfor 20` `end` This instruction is typed in the Cricket Direct Mode Panel. `loop [setdirandrun]`

Primitive *Input(s)*	Description	Example
loop ***list-of-instructions***	Keeps running *list-of-instructions* indefinitely.	There is a switch connected to sensor port A and a motor connected to motor port A. This procedure is downloaded to the Cricket: `to ride` `on` `waituntil [switcha]` `a, off` `waituntil [not switcha]` `end` This instruction is typed in the Cricket Direct Mode Panel. `loop [ride]`
repeat ***number*** ***list-of-instructions***	Runs the *list-of-instructions* the specified number of times.	There is a motor connected to motor port A. This procedure is downloaded to the Cricket: `to washcycle` `a, on` `repeat 10 [wait 10 rd]` `off` `end` This instruction is typed in the Cricket Direct Mode Panel. `washcycle`
timer	Reports the value of the internal timer in thousandths of a second. The timer is set to 0 when the Cricket is turned on, when the primitive **resett** is executed, and when the timer reaches 32767 (32.767 seconds). It should therefore be use to time events that take place within this limit. See **resett**.	There is a switch connected to sensor port A and a light connected to motor port A. This procedure is downloaded to the Cricket: `to reflex` `wait 50 + remainder` `random 50` `resett` `a, on` `waituntil [switcha]` `sendmessage timer` `off` `end`

(continued)

Primitive *Input(s)*	Description	Example
timer		On the MicroWorlds EX page, there is also a text box and a button running the following instruction in **Forever** mode. The instruction "catches" messages sent by the Cricket and prints them in the text box (**rm** stands for **remotemessage)** `if rm? [pr rm]` This instruction is typed in the Cricket Direct Mode Panel. `reflex`
resett	Resets the value of the Cricket's internal timer to 0.	See the example for **timer**.
stop	Stops the procedure that is running. This primitive can only be used in a procedure. In this example, the power is increased until the switch connected to port A is pressed.	There is a motor connected to motor port A and a switch connected to sensor port A. This procedure is downloaded to the Cricket: `to calibrate` `a, on` `setpower 2 wait 20` `if switcha [stop]` `setpower 4 wait 20` `if switcha [stop]` `setpower 6 wait 20` `if switcha [stop]` `setpower 8 wait 20` `if switcha [stop]` `end` This instruction is typed in the Cricket Direct Mode Panel. Press the switch when the motor speed seems adequate. `calibrate`

Primitive *Input(s)*	Description	Example
stopme	Stops all the procedures running in the Cricket, not just the procedure containing the **stopme** command. **Stopme** does not stop everything: it does not stop any processes started with a **when** instruction.	There are switches connected to sensor ports A and B and a motor connected to motor port A. These procedures are downloaded to the Cricket Procedures Panel. `to ride` `goright` `finish?` `goleft finish?` `end` `to goright` `thisway` `loop [onfor 2 wait 2` `if switcha [stop]]` `end` `to goleft` `thatway` `loop [onfor 2 wait 2` `if switcha [stop]]` `end` `to finish?` `if switchb` `[off stopme]` `end` This instruction is typed in the Cricket Direct Mode Panel: `ride` **Stop** stops only the procedure containing the **stop** command (**goright** or **goleft**) and the **ride** procedure continues with the next instruction. **Stopme** stops all the procedures.

Primitive *Input(s)*	Description	Example
wait ***number***	Causes a pause in the execution of a program or instruction. The time (*number*) is measured in tenths of a second.	There is a switch connected to sensor port A and a light connected to motor port A. This procedure is downloaded to the Cricket. `to reflex` `wait 50 + remainder` `random 50` `resett` `a, on` `waituntil [switcha]` `sendmessage timer` `off` `end` Create a text box and a button on the MicroWorlds EX page. In the button's instruction field, type: `if rm? [pr rm]` Set the button to **Forever** mode. The instruction "catches" messages sent by the Cricket and prints them in the text box (**rm** stands for **remotemessage**) This instruction is typed in the Cricket Direct Mode Panel. `reflex`
waituntil ***list-of-instructions*** *(reports 1 or* 0)	Runs the *list-of-instructions* repeatedly until it reports **true (1)**. When this happens, **waituntil** stops running the instruction and the procedure proceeds to the next instruction.	See the example for **wait**.

Primitive *Input(s)*	Description	Example
when ***list-of-instructions*** *(reports 1 or* 0) ***list-of-instructions***	Runs the first *list-of-instructions* repeatedly and executes the second *list-of-instructions* whenever the first list reports **1 (true).** The second instruction is triggered when the first instruction switches from being false (reporting **0**) to being true (reporting **1**). In the example, pressing and maintaining the switch will only run **[onfor 3]** once. Pressing the switch several times will run **[onfor 3]** as many times as it is pressed. The Cricket can run only one "when process" at a time. You cannot send additional instructions from the Cricket Direct Mode Panel to the Cricket while it is busy running a **when** instruction. See **whenoff**.	There is a switch connected to sensor port A and a motor connected to motor port A. This procedure is downloaded to the Cricket. `to trigger` `when [switcha]` `[onfor 3]` `wait 200` `whenoff` `end` This instruction is typed in the Cricket Direct Mode Panel. `trigger` For the next 20 seconds, the motor will run each time you press on the switch. After 20 seconds, the command **whenoff** stops the **when** process.
whenoff ***list-of-instructions***	Turns off a **when** process. See **when**.	See the example for **when**.
output ***word-or-list*** **op** ***word-or-list***	Stops the procedure and reports a value. This primitive can only be used in a procedure.	There is a switch connected to sensor port A and a thermal sensor connected to sensor port B. These procedures are downloaded to the Cricket: `to senddata` `sendmessage temperature` `end` `to temperature` `waituntil [switcha]` `output sensorb` `end` On the MicroWorlds EX page, there is a text box and a button running the following instruction in **Forever** mode. The instruction "catches" messages sent by the Cricket and prints them in the text box. (**Rm** stands for **remotemessage**) `if rm? [pr rm]` This instruction is typed in the Cricket Direct Mode Panel. `loop [senddata]`

Audio-Display

These primitives can be used to write procedures to be downloaded into the Cricket.

Primitive *Input(s)*	Description	Example
beep	Plays a short beep.	This instruction is typed in the Cricket Direct Mode Panel. `repeat 5 [beep wait 5]`
note ***pitch*** ***duration***	Plays a note using the specified *pitch* and *duration*. *Duration* is measured in tenths of a second.	There is a motor connected to motor port A. These instructions are typed in the Cricket Direct Mode Panel. `a, on note 65 10 off`

Math and logic

These primitives can be used to write procedures to be downloaded into the Cricket.

Primitive *Input(s)*	Description	Example
( )	Parentheses are used in the Cricket to determine the order of mathematical operations. In the example, without the parentheses, only **sensorb** would be divided by 2 to get the average and by 10 to get the value in Celsius (divided by 20 in the example to simplify the instruction), and the result would be added to **sensora**. The ***** and **/** operations are normally executed before the **+** and **-** operations. The parentheses are used here to force a different sequence.	There is a motor connected to motor port A and two temperature sensors connected to sensor ports A and B. The following instruction triggers the motor connected to port A if the average temperature reported by the two sensors is above 25 degrees Celsius: `if ((sensora + sensorb) / 20) > 25 [a, on]`
+ - * /	These mathematical operators follow the same rules as those in MicroWorlds EX.	See example for **( )** above.

Primitive *Input(s)*	Description	Example
= > <	These mathematical operators follow the same rules as those in MicroWorlds EX except they report **1** and **0** instead of **true** or **false**. They are generally used in **if, ifelse** and **waituntil** instructions because these primitives require a **1** or a **0** as their first input. See the examples.	There is a motor or light connected to motor port A. These instructions are typed in the Cricket Direct Mode Panel. `if timer > 2000 [on] resett` `on waituntil [timer > 10000] off`
and ***1 or 0*** *(true-or-false)* ***1 or 0*** *(true-or-false)*	Reports **1** (true) if all its inputs report **1** (true).	There are motors connected to motor ports A and B and switches connected to sensor ports A and B. `to drive` `ab, on` `ifelse switcha` `[a, thisway]` `[a, thatway]` `ifelse switchb` `[b, thisway]` `[b, thatway]` `if and switcha switchb` `[ab, off]` `end` This instruction is typed in the Cricket Direct Mode Panel. `loop [drive]`
not ***1 or 0*** *(true-or-false)*	Reports the logical inverse of its input.	There is a motor connected to motor port A and a switch connected to sensor port A. This instruction is typed in the Cricket Direct Mode Panel. Press the switch before you send the instructions to the Cricket. Release the button to stop the motor. `a, on` `waituntil [not switcha]` `off`

Primitive *Input(s)*	Description	Example
or ***1 or 0*** *(true-or-false)* ***1 or 0*** *(true-or-false)*	Reports **1** (**true**) if any of its inputs report **1** (**true**). In the example, the motors are turned off if either of the switches (one or both) is pressed.	There are motors connected to motor ports A and B and switches connected to sensor ports A and B. `to drive` `ab, on` `if or switcha switchb` `  [ab, off]` `end` This instruction is typed in the Cricket Direct Mode Panel. `loop [drive]`
random	Reports a random number between 0 and 32767. See *Additional Information* in *Section 4, Technical Information*.	See the example for **remainder**
remainder ***number1*** ***number2***	Reports the remainder when *number1* is divided by *number2*. The remainder of a negative number is negative. If *number1* and *number2* are non-integers, they are truncated.	There is a switch connected to sensor port A and a light connected to motor port A. This procedure is downloaded to the Cricket. `to reflex` `wait 50 +` `  remainder random 50` `resett` `a, on` `waituntil [switcha]` `sendmessage timer` `off` `end` On the MicroWorlds EX page, there is also a text box and a button running the following instruction in **Forever** mode. The instruction "catches" messages sent by the Cricket and prints them in the text box. (**Rm** stands for **remotemessage**) `if rm? [pr rm]` This instruction is typed in the Cricket Direct Mode Panel. `reflex`

Variables

This primitive can be used in the Cricket Procedures Panel but not in the Direct Mode Panel.

Primitive *Input(s)*	Description	Example
global *list*	Assigns a state variable to the Cricket and creates the following command-reporter pair of primitives that can be used in the procedures downloaded into the Cricket: the word **set** followed by the name of the variable (for example, **global "level** creates a **setlevel** command); and,the name of the variable (in this example, the reporter **level**). The command **global** and its input must be typed in the Cricket Procedures Panel, but not inside a procedure (before, between or after procedures).	There is a motor or light connected to motor port A and two switches connected to sensor ports A and B. In this example, **switcha** is used to raise the power level of the motor or light connected to motor port A and **switchb** decreases it. If both switches are used at the same time, the motor stops. The following is typed in the Cricket Procedures Panel and downloaded to the Cricket: `global [level]` `to variablespeed` `setlevel 3` `a, setpower 3` `on` `loop [control]` `end` `to control` `if switcha` `  [setlevel level + 1]` `if switchb` `  [setlevel level - 1]` `if and switcha switchb` `  [off]` `setpower level` `end` The following is typed in the Cricket Direct Mode Panel: `variablespeed` Press one switch or the other a few times to see the effect.

Messaging Primitives Running In the Cricket

These primitives can be used to write procedures to be downloaded to the Cricket. The Cricket must be in the Interface Cricket's line of sight when these primitives are used.

Primitive *Input(s)*	Description	Example
sendmessage *number* **sm** *number*	Sends a value between -32767 and 32767to MicroWorlds EX. It is commonly used to send the value of the timer or that of a sensor. If the input is not an integer, it is truncated. The primitives **remotemessage?** and **remotemessage** must be used within a MicroWorlds EX project to receive and "read" the message sent by the Cricket. In the example, **reflex** is executed from a button in MicroWorlds EX. Click on the button, get a few feet away from the switch and rush to press it when you hear the motor or see the light. The Cricket will tell MicroWorlds EX how much time it took you to react. The **reflex** procedure waits between 10 and 20 seconds and sends the instruction to run the **howfast** procedure to the Cricket. The **reflex** procedure then waits for a message to arrive from the Cricket. On the Cricket side, the **howfast** procedure resets the timer, starts the light or motor, waits for the switch to be pressed, and then sends a message containing the value of the timer (this is the time elapsed between **resett** and pressing the switch). Back in MicroWorlds EX (in the **reflex** procedure), **remotemessage?** reports **true** when the message from the Cricket arrives and this ends the process started by the instruction: **waituntil [remotemessage?]** **Reflex** then prints the contents of the message, divided by 1000 (the number of seconds). Finally, the motor is turned off.	There is a light or a motor connected to motor port A and a switch connected to sensor port A of the Cricket. The following procedure is downloaded to the Cricket. `to howfast` `resett` `a, on` `waituntil [switcha]` `sendmessage timer` `wait 20` `end` The following procedure is defined in the MicroWorlds EX project Procedures Tab. `to reflex` `clearbuffer` `wait 100 + random 100` `sendremotecommand` `[howfast]` `waituntil[rm?]` `pr remotemessage / 1000` `sendrc [off]` `end`

Primitive *Input(s)*	Description	Example
message?	This primitive is executed in the Cricket and it reports **1** (**true**) if a message was sent by MicroWorlds EX using the command **sendremotemessage**. Use **message** to read the message.	There is a light or a motor connected to motor port A. The following procedure is downloaded to the Cricket. `to start` `loop [getinputfrommw]` `end` `to getinputfrommw` `if message?` `[onfor message]` `end` The procedure **start** is started in the Cricket by pressing the **Run/Stop** switch. The following commands are executed within MicroWorlds EX (in the Command Center, by buttons, clickable turtles, etc.) while the Cricket is in the Interface Cricket's line of sight. `sendremotemessage 10` `sendrm 30`
message	Reports the message sent by MicroWorlds EX. Reports **0** if there is no message to be read. It is best to use **message?** to find out if there is a message waiting to be read before using **message**.	See the example for **message?**

Messaging Primitives Running Within MicroWorlds EX

These primitives are used within MicroWorlds EX (from the Command Center, by buttons, turtles, etc.) to download procedures, instructions or messages to the Cricket. They cannot be used in the Cricket Tab. The Cricket must be in the Interface Cricket's line of sight when these primitives are used.

Primitive *Input(s)*	Description	Example
download ***word***	Downloads its input to the Cricket. This is equivalent to clicking the **Download** button with the same text in the Crickets Procedures Panel. The input for **download** is a word. The most common input for **download** is the contents of a text box. You can take advantage of the fact that the name of a text box reports the text box' contents as a "long word". Therefore, the name of a text box can be used as input for download. In the instruction **download text1** **text1** reports its contents as a long word. This long word is used by **download** as input.	**Text1** contains the procedure **goforit**. **Text1** reports its contents to **download**, which sends it to the Cricket. The next instruction, **goforit**, starts the procedure in the Cricket. These instructions can be typed in the MicroWorlds EX Command Center or executed from a button, a programmed color or turtle. `download text1` `sendrc "goforit`
sendremotecommand ***word*** **or** ***list-of-instructions*** **sendrc** ***word*** **or** ***list-of-instructions***	The instruction is sent to and executed by the Cricket. The input is a word or a list that contains instructions that can be run in the Cricket (either procedures that have been downloaded to the Cricket or primitives).	There is a motor or a light connected to motor port A and a switch connected to sensor port A. These instructions are typed in the MicroWorlds EX Command Center. `sendremotecommand "on` `sendrc [b, onfor 10]`

Primitive *Input(s)*	Description	Example
sendremotemessage *number* **sendrm** *number*	Sends a number as a message to the Cricket. The Cricket must use **message?** to determine if a message was sent and/or **message** to find out what the message is. The input must be a number between 0 and 255.	There is a light or a motor connected to motor port A. The following procedure is downloaded to the Cricket. `to start` `loop [getinputfrommw]` `end` `to getinputfrommw` `if message? [onfor message]` `end` The procedure start is started in the Cricket by pressing the **Run/Stop** button. The following commands are executed within MicroWorlds EX (in the Command Center, by buttons, clickable turtles, etc.) while the Cricket is in the Interface Cricket's line of sight. `sendremotemessage 10` `endremotemessage 30`

These primitives are used within MicroWorlds EX in order to detect and use incoming messages sent by the Cricket. The Cricket must be in the Interface Cricket's line of sight when these primitives are used.

Primitive *Input(s)*	Description	Example
clearbuffer	Empties the message buffer in the Cricket before starting a communication session between the Cricket and MicroWorlds EX. The message buffer contains messages that have been sent by the Cricket (using the primitive **sendmessage**) but that have not yet been "read" by MicroWorlds EX. The message stays in the Cricket's buffer until **remotemessage** is used in MicroWorlds EX to receive and "read" the message. In the example, **reflex** is executed from a button in MicroWorlds EX. Click on the button, get a few feet away from the switch and rush to press it when you hear the motor or see the light. The Cricket will tell MicroWorlds EX how much time it took you to react. The **reflex** procedure waits between 10 and 20 seconds and sends the instruction to run the **howfast** procedure to the Cricket. The **reflex** procedure then waits for a message to arrive from the Cricket. On the Cricket side, the **howfast** procedure resets the timer, starts the light or motor, waits for the switch to be pressed, and then sends a message containing the value of the timer (this is the time elapsed between resett and pressing the switch). Back in MicroWorlds EX (in the **reflex** procedure), **remotemessage?** reports **true** when the message from the Cricket arrives and this terminates the instruction **waituntil [remotemessage?].** **Reflex** then prints the contents of the message, divided by 10 (the number of seconds). Finally, the motor is turned off.	There is a light or a motor connected to motor port A and a switch connected to sensor port A of the Cricket. The following procedure is downloaded to the Cricket. `to howfast` `resett` `a, on` `waituntil [switcha]` `sendmessage timer` `end` The following procedure is defined in the MicroWorlds EX project Procedures Tab. `to reflex` `clearbuffer` `wait 100 + random 100` `sendrc [howfast]` `waituntil [rm?]` `pr remotemessage / 1000` `sendrc [off]` `end` Create a text box on the MicroWorlds EX page. To test your reflexes, first, make sure the Cricket is in the Interface Cricket's line of sight. Then, in the MicroWorlds EX Command Center, type: `reflex`

Primitive *Input(s)*	Description	Example
remotemessage? **rm?**	Reports **1** (**true**) if there is a message to be "read" in the message buffer. The message would have been sent by the Cricket using the command **sendmessage**. Reports **0** (**false**) otherwise. Use **remotemessage** to read the message.	There is a text box on the page. There is a switch connected to sensor port A and a light sensor connected to sensor port B. Create this procedure in the MicroWorlds EX project Procedures Tab: `to trackdata` `if remotemessage?` `  [pr remotemessage]` `end` Download the following procedures to the Cricket: `to start` `loop[sendonclick]` `end` `to sendonclick` `if switcha` `  [sendmessage sensorb]` `end` Create a button on the page. In the instruction field, type **trackdata**. Set the mode to **Forever**. Each time the Cricket sends a message, it is read by MicroWorlds EX and printed in the text box. Click on the button in MicroWorlds EX. Press the **Run/Stop** switch on the Cricket. Press on the switch to send the light sensor's value to MicroWorlds EX.
remotemessage **rm**	Reports the message sent by the Cricket. Reports **0** if there is no message to be read. It is best to use **remotemessage?** to find out if there is a message waiting to be read before using **remotemessage**.	See the example for **remotemessage?**

Primitive *Input(s)*	Description	Example
requestremotemessage ***word*** **or** ***list-of-instructions*** **requestrm** ***word*** **or** ***list-of-instructions***	Sends the *word* or *list-of-instructions*, each of which must report a value, to the Cricket, waits for the value reported by the Cricket and reports that value in MicroWorlds EX.	There is a text box on the page. This instruction is executed in the MicroWorlds EX Command Center. The values are just examples: `requestremotemessage "timer 25365` `requestrm "sensorb 1021` The following **delay** procedure is downloaded to the Cricket and a switch is connected to sensor port A. `to delay` `resett` `waituntil [switcha]` `output timer` `end` This instruction is executed in the MicroWorlds EX Command Center: `pr requestrm "delay` Press the switch (the value is just an example). `2134`