MicroWorlds 2.0

Logo Vocabulary

LCSI
November 21, 2000

Math Operators

number1 **-** *number2*
Reports the result of *number1* minus *number2*. See **difference**.
Example:
```
show 3 - 3
0
```

number1 ***** *number2*
Reports the product of *number1* multiplied by *number2*. See **product**.
Example:
```
show 3 * 3
9
```

number1 **/** *number2*
Reports the result of *number1* divided by *number2*. See **quotient**.
Example:
```
show 3 / 3
1
```

number1 + *number2*
Reports the sum of *number1* and *number2*. See **sum**.
Example:
```
show 3 + 3
6
```

number1 < *number2*
Reports **true** if *number1* is less than *number2*. See **less?.**
Example:
```
show 2.9 < 3
true
```

word-or-list1 = *word-or-list2*
Reports **true** if *word-or-list1* is equal to *word-or-list2*. See **equal?** and **identical?.**
Example:
```
show 3 = 3
true
```

number1 > *number2*
Reports **true** if *number1* is greater than *number2*. See **greater?.**
Examples:
```
show 3.1 > 3
true
show 5 > 6
false
```

abs

abs *number*
Stands for **abs**olute. Reports the absolute value of its input.
Example:
```
show abs -33
33
```

and

and *true-or-false1 true-or-false2*
(**and** *true-or-false1 true-or-false2 true-or-false3...*)
Reports **true** if all its inputs report **true**. If more than two inputs are used, **and** and its inputs must be enclosed in parentheses. See **or** and **not**.
Examples:
```
show and 2 = 2  3 = 5
false
cg
show and pos = [0 0] heading = 0
true
show (and 2 = 2  5 = 5  6 = 6)
true
```

announce

announce *word-or-list*
Displays the message in an alert box. Clicking OK closes the box. See **question** and **answer**.
To reposition the default alert box, use the **set** command. The position is in turtle coordinates; [0 0] is the middle of the page. The default position is [-200 50]. This is the position of the upper, left corner of the alert box. If the position accidentally places part of the box outside of the screen, use **Enter** to close the box.
If this variable is changed, you should have a **startup** procedure to reset it each time you load the project. See also **get**and *Startup Procedure* in *MicroWorlds Help Topics.*
Example:
```
announce [You win!!]
```

answer

answer
Reports the contents of the last answer typed in the question dialog box. Using **question** and **answer**, you can pick up the words typed at the keyboard to create an interactive program. If **answer** reports the empty list ([]), it means that the user has clicked on Cancel; if **answer** reports the empty word ("), it means that the user has entered nothing and clicked OK.
Example:
```
question [How old are you?]
```
Type the answer in the dialog box.
```
show answer
I'm 10        Your answer.
```

arctan

arctan *number*
Stands for **arc tan**gent. Reports the arc tangent (the inverse function of the tangent) of its input. See **tan** and **cos**.
Example:
```
show arctan 1
```

```
45
```

ascii

ascii *char*
Stands for American Standard Code for Information Interchange. Reports the ASCII number which represents the character. See **char**.
Example:

```
show ascii "a
97
```

ask

ask *who instruction-list*
Temporarily tells each element in the first input to run the instruction list. The first input can be the name(s) of one or many turtles or text box names. **Ask** does not change the current turtle or text box. The apostrophe can be used to ask a turtle to report something. Turtles have built-in properties: **pos**, **heading**, **color**, **size**, **pensize** and **shape**. Variables can be assigned to turtles using **turtlesown**.
Examples:
There are many turtles on the page.

```
ask [t1 t2 t3] [fd 50 rt 90 fd 50]
```

If you have two text boxes on the page and you want to keep Text1 current:

```
ask "text2 [print "hello]
```

If t1 is not the current turtle and you want to know its position:

```
show ask "t1 [pos]
```

Following are three examples of equivalent instructions:

```
show ask "t1 [pos]
0 0
show t1's "pos
0 0
t2, setpos ask "t1 [pos]
t2, setpos t1's "pos
turtlesown "gravity
t1, setgravity 10
show ask "t1 [gravity]
10
show t1's "gravity
10
```

back

back (bk) *number*
Moves the turtle backwards.
Examples:

```
pd bk 20
rt 90
pu bk 50
pd bk 10
```

bg

bg
Stands for **b**ack**g**round. Reports a number representing the color of the background. The background color is 0 (white) when MicroWorlds starts up. See **setbg.**
Example:

```
setbg 10
repeat 9 [setbg bg + 1]
```

Note: In order to use the full potential of MicroWorlds graphics, 16 bit color mode is recommended.

bottom

bottom
Puts the cursor (insertion point), in the current text box, at the end of the text. Try using this command in a button so that you can see the effect on the cursor. See **top**, **sol**, **eol**, and **eot?**.
Example:

```
text1,
repeat 5 [print "hello]
top
pr "say
bottom
pr "there
```

butfirst

butfirst (bf) *word-or-list*
Reports all but the first component of a word or list. See **butlast**, **first**, and **last**.
Examples:

```
show butfirst [0 1 2 3]
1 2 3
show butfirst [hello there]
there
```

butlast

butlast (bl) *word-or-list*
Reports all but the last component of a word or list. See **butfirst**, **first**, and **last**.
Examples:

```
show butlast [0 1 2 3]
0 1 2
show butlast "welcome
welcom
```

cancel

cancel *instruction-list*
Stops the process given as input. The process must have been launched using **launch**, **when**, **forever**, buttons, and clickable turtles. Corresponds to choosing Cancel from the Edit menu. The input must be the exact same instruction list that started the process.
Examples:

```
t1,
forever [fd 1]
forever [rt 1]
cancel [rt 1]
cancel [fd 1]
```

carefully

carefully *word-or-list-to-run1 word-or-list-to-run2*
Runs the first list of instructions. If the first list contains an error, **carefully** runs the second list of instructions and sets **errormessage** to the error that occurred. If there is no error in the first list, the second list is ignored. Since errors caught by **carefully** will not be accessible by the Help menu item, Last Message, use this command with caution.
Example:

```
carefully [fd 50]
          [announce [No turtle!]]
```

If you try this with a turtle on the page, it will go forward. If you try it without a turtle on the page, the message will be displayed in an alert box.

cb

cb
Stands for **c**ursor **b**ack. Moves the cursor (insertion point), in the current text box, to the previous character. Try using this command in a button so that you can see the effect on the cursor. See **cf**, **cd**, and **cu**.
Example:

```
text1,
ct insert "HELLO
repeat 5 [cb insert "x cb]
```

cc

cc
Stands for **c**lear the **C**ommand Center. Clears the text in the Command Center.
Example:

```
cc
```

cd

cd
Stands for **c**ursor **d**own. Moves the cursor (insertion point), in the current text box, to the next physical line. Try using this command in a button so that you can see the effect on the cursor. See **cu**, **cf**, and **cb**.
Example:

```
text1,
repeat 5 [print "hello]
top
repeat 5 [cd pr "there!]
```

cf

cf

Stands for **c**ursor **f**orward. Moves the cursor (insertion point), in the current text box, to the next character. Try using this command in a button so that you can see the effect on the cursor. See **cb**, **cd**, and **cu**.
Example:

```
text1,
insert "hello
top
repeat 5 [cf insert "x]
```

cg

cg
Stands for **c**lear **g**raphics. Clears the graphics on the page and returns the current turtle to its home position, facing up. See **clean.**
Example:

```
pd
fd 50 rt 90
stamp
fd 50
cg
```

char

char *number*
Stands for **char**acter. Reports the character represented by the ASCII number given as input. The number must be between 32 and 255. An exception is **char 9**, the tab character. See **ascii** and **print**.
Examples:

```
show char 97
a
show char 65
A
```

Use the command:

```
print "
```

to insert a carriage return and line feed sequence.

chdir

chdir *path*
Stands for **ch**ange **dir**ectory. Changes the current drive and/or subdirectory name to *path.* To return to the desktop, use the volume name alone as input to **chdir**. Backslashes are used to separate the names of directories. See **currentdir** and **directories**.
Examples:

```
show directories
Media My Work Projects
chdir "Media
```

If one of the elements of the path has spaces, vertical bars must enclose the whole path.

```
chdir "|C:\MicroWorlds\My Work|
```

Some special inputs can be used with **chdir**. **Chdir ".. ** returns to the parent directory. **Chdir "a:** (or any valid drive name) sets the current directory to the designated drive.

```
show currentdir
C:\MicroWorlds\My Work
chdir "..
show currentdir
C:\MicroWorlds
chdir "a:
show currentdir
A:\
```

clean

clean
Clears the graphics without changing any turtle's position. See **cg** and **freezebg**.
Example:

```
pd
setsh 12
setc "red
fd 50
clean
```

clearname

clearname *word*
Clears a global variable from memory. See **clearnames**, **names**, **make**, and **name**.
Example:

```
make "speed 5
make "direction "right
clearname "speed
show :speed
speed has no value
```

clearnames

clearnames
Clears all the global variables from memory. MicroWorlds doesn't clear the variables when you open or create a new project. Therefore, it is recommended to use **clearnames** each time you start a new project. See **names**, **clearname**, **make**, and **name**.
Example:

```
make "speed 5
make "direction "right
clearnames
show :direction
direction has no value
```

cleartext

cleartext (ct)
Clears the text in the current text box.
Example:

```
text1, print "hello
cleartext
```

clickoff

clickoff
Simulates a mouse click on the current turtle, turning it off if it was on. This command will only have an effect if the turtle is programmed to react to a mouse click. See **clickon**, **listen**, and *Synchronizing Processes* in *MicroWorlds Help Topics* for advanced features.
Note: If you used a list as input to **talkto** before running **clickoff**, **clickoff** will display an error message.
Example:
Type an instruction in the turtle's dialog box and set it to Many Times. Click on the turtle to make it run its instruction, then type in the Command Center:

```
clickoff
```

clickon

clickon

Simulates a mouse click on the current turtle, turning it on if it was off. This command will have an effect if the turtle is programmed to react to a mouse click. If used in a button's dialog box, it will change the current turtle. See **clickoff** and **listen**, and *Synchronizing Processes* in *MicroWorlds Help Topics* for advanced features.
Note: If you used a list as input to **talkto** before running **clickon**, **clickon** will display an error message.
Example:
Type an instruction in the turtle's dialog box. Then type in the Command Center:

```
clickon
```

Type the following instruction in a button's dialog box:

```
t2, clickon
```

Hatch the t2 turtle. Then type in the Command Center:

```
t1, fd 45
```

Click the button with the **clickon** instruction for t2. Running **fd 45** in the Command Center now moves t2.

clipboard

```
clipboard
```

Reports the contents of the text Clipboard. The Clipboard contains the last text that has been cut or copied using the **cut** or **copy** command, or the equivalent Edit menu items. The Clear menu item and the **Delete** key do not affect the Clipboard. See also **select** and **paste**.
Example:
If you have copied or cut the words "My friend Kim":

```
show clipboard
My friend Kim
```

color

```
color
```

Reports the turtle's color as a number. **Color** reports a number, even if a name was used as input for **setc**.
Examples:

```
setc "red
show color
15
setc color + 1
show color
16
```

Note: In order to use the full potential of MicroWorlds graphics, 16 bit color mode is recommended.

colorunder

```
colorunder
```

Reports the color under the current turtle as a number. The portion of a turtle that recognizes a color is its center. **Colorunder** reports the background color as well as all the drawings.
Example:

```
t1, show colorunder
3
```

In a stop rule, always use the color *number*, not its name, to check the color under the turtle:

```
if colorunder = 15
   [announce [You win!]]
```

Colorunder reports not only integers but also decimal numbers. Note that when MicroWorlds reports decimal numbers, it may report a slightly different number than the one expected. This is due to the recalculation of the RGB color.

```
setc 105.6
fill
show colorunder
105.4
setc 17.2
```

```
fill
show colorunder
17.1
```
Note: In order to use the full potential of MicroWorlds graphics, 16 bit color mode is recommended.

copy

copy
Puts a copy of the selected text in the Clipboard. It unselects the current selection.
Example:
If you have selected the words "My friend Kim" in the text box:

```
copy
ct
paste paste
```
My friend Kim is pasted twice in the text box.

cos

cos *number*
Stands for **cos**ine. Reports the cosine of its input. See **sin** and **tan**.
Example:

```
show cos 60
0.5
```

count

count *word-or-list*
Reports the number of components in the word or the list. See **item** and **textcount**.
Examples:

```
show count "hello
5
show count [this is a list]
4
```

createprojectvar

createprojectvar *word*
Stands for **create project var**iable. Creates a project variable represented by a command and a reporter. For example, if the project variable "amount" is created, the command **setamount** sets its value, and **amount** reports its value. Project variables are saved with your project. See **projectvars**, **make**, **remove,** and **name**.
Example:

```
createprojectvar "amount
setamount 22
show amount
22
```

cu

cu
Stands for **c**ursor **u**p. Moves the cursor (insertion point), in the current text box, to the previous physical line. Try using this command in a button so that you can see the effect on the cursor. See **cd**, **cb**, and **cf**.
Example:

```
text1,
repeat 5 [print "there!]
repeat 5 [cu pr "hello cu]
```

currentdir

currentdir

Stands for **current dir**ectory. Reports the current directory that was set. See **chdir**.

Example:

```
show currentdir
C:\MicroWorlds\projects
```

cut

cut

Deletes the text selection in the current text box and puts a copy in the Clipboard. See **select, copy,** and **paste.**

Example:

If you have selected the words "My friend Kim" in the text box:

```
cut
paste paste
```

My friend Kim is pasted twice in the text box.

delete

delete
Deletes the character to the right of the insertion point, in the current text box.
Example:
Type some text into a text box and put the cursor in the middle of the text:
delete

difference

difference *number1 number2*
Reports the result of subtracting *number2* from *number1*. See **-** and **minus**.
Example:
```
show difference 7 3
4
```

directories

directories
Reports a list of subdirectory names. To change directories through a command, use **chdir**.
A name made up of more than one word will look like two words in the list. Use **item** to find the actual name. In this example, My Work is the name of one directory.
Example:
```
show directories
Media My Work Projects
```

distance

distance *turtle-name*
Reports the distance between the current turtle and the turtle indicated. See **towards.**
Example:
In this example, there are two turtles on the page.
```
t1,
show distance "t2
122
towards "t2
fd distance "t2
```
Your answer will be different.
T1 meets t2.
Set t1 to go Many Times and define the **go** procedure as follows. T1 will be "trapped" around t2:

```
to go
fd 1
if 100 < distance "t2 [towards "t2]
end
```

dolist

dolist *range instruction-list*
Runs the instruction list for each item in a list. The first input, *range*, is a list with a temporary variable name and a list of items. The second input is a list of instructions that uses the variable name included in the first input. In the following example, the instruction **remove :i** is run for each item of the first list. "**I**" successively takes the value t1, t2, and t3.
Example:
If there are three turtles on the page, the following instruction removes all three turtles.
```
dolist [i [t1 t2 t3]] [remove :i]
```
The following example displays **a**, **b**, **c**, and **d** in the Command Center.
```
dolist [i [a b c d]] [show :i]
```

done?

done? *instruction-list*

Reports **true** if the process indicated is completed. The process must have been launched using **launch** or **forever**. The input must be an exact copy of the instruction list that started the process. **Done?** can be used as an input to **waituntil** in order to synchronize events.

Example:

Circle and **square** are procedures, and there are two turtles on the page.

```
to circle
repeat 36 [fd 10 rt 36]
end

to square
repeat 4 [fd 50 rt 90]
end
```

In the following procedure, t1 makes a circle at the same time t2 makes a square. It takes longer to draw a circle, but MicroWorlds will wait for both shapes to be finished before telling the turtles to go elsewhere on the page to draw more circles and squares.

```
to sq-circ
t1, launch [circle]
t2, launch [square]
waituntil [done? [circle]]
t1, rt random 360 fd random 50
t2, rt random 360 fd random 50
sq-circ
end
```

dotimes

dotimes *range instruction-list*

Runs the instruction list for each value specified in the range. The first input is a list with a temporary variable name and a maximum number. The second input is a list of instructions that uses the variable name included in the first input. In the following example, the instruction **setc** sets the turtle color for each value of **i**, from 0 to 7.

Example:

```
dotimes [i 8] [setc :i wait 5]
```

The following example displays 0, 1, 2, 3, ... to 9 in the Command Center.

```
dotimes [i 10] [show :i]
0
1
2
...
9
```

Note: The name of the variable that you choose for **dotimes** is bound to this primitive. A change in its value (for example: make "i :i + 1) in the instruction list won't have an effect.

empty?

empty? *word-or-list*
Reports **true** if the input is an empty word or empty list.
Examples:

```
show empty? []
true
show empty? text1
false
```

This procedure can be used to get an answer to a question dialog box:

```
to insist
question [Your name please...]
if empty? answer [insist]
end
```

eol

eol
Stands for **e**nd **o**f **l**ine. Brings the cursor (insertion point), in the current text box, to the end of the current logical line. Try using this command in a button so that you can see the effect on the cursor. See **sol**.
Example:

```
pr "hello
top
eol
insert "!
```

eot?

eot?
Stands for **e**nd **o**f **t**ext. Reports **true** if the cursor (insertion point), in the current text box, is at the end of the text. See **bottom**.
Example:

```
text1,
bottom
show eot?
true
```

The following procedures can be used to number the lines in a text box. **Eot?** is generally used to stop a procedure that processes information in a text box using cursor (insertion point) commands like **cd**, **eol**, etc.

```
to numberlines
top
countup 1
end

to countup :n
if eot? [stop]
insert :n
insert char 32
sol cd
countup :n + 1
end
```

equal?

equal? *word-or-list1 word-or-list2*
Reports **true** if the two inputs are equal. The inputs may be words, numbers, or lists. See **identical?** and **=**.

Examples:

```
show equal? "a "A
true
show equal? "hello text1
true
show equal? [ ] "
false
```

erfile

erfile *path*

Stands for **er**ase **file**. Erases any type of file if it is not locked. The input must be the name of a file in the current directory or a full path. A full path starts with the name of the drive. Backslashes are used to separate the names of directories, subdirectories, and files.

Examples:

```
erfile "farm
erfile "C:\projects\farm
```

When there is more than one file with the same name, you need to add the extension.

```
erfile "quake.mw2
```

Otherwise, MicroWorlds will display the following error message:

There's more than one file named quake

If one of the elements of the path has spaces, vertical bars must enclose the whole path:

```
erfile "|C:\My projects\farm|
```

errormessage

errormessage

Reports the last error message trapped by **carefully**. If **errormessage** reports an empty word, it means that the last operation using **carefully** did not report an error.

Example:

There is no turtle on the page.

```
carefully [fd 50]
          [show errormessage]
No turtle found for forward
```

The following procedure asks a question and tries to play the answer. If the sound doesn't exist, the procedure displays a message and *continues to the next instruction.* Without **carefully**, an error message would be displayed and the procedure would stop.

```
to safe-play
question [What sound do you want?]
if empty? answer [stop]
carefully
      [run (list answer)]
      [announce [I don't have it.]]
safe-play
end
```

everyone

everyone *list-of-instructions*

Makes all the turtles on the current page run the instruction, one after the other. See **ask** and **talkto**.

Examples:

Create many turtles on a page.

```
everyone [setsh 12]
everyone [repeat 4 [fd 50 rt 90]]
everyone [forever [fd 5]]
```

exp

exp *number*
Stands for **exp**onential. Reports the *number* to the power of the constant *e*.
Example:

```
show exp 1
2.71828182846
```

files

files *filetype*
Outputs a list of files of the given *filetype.*
Example:
```
show files "TXT
report summary
```

fill

fill
Fills a closed shape or the whole screen with the turtle's color. **Fill** will work regardless of the turtle's pen state (up or down). See **setc**.
Example:
```
pd repeat 5 [fd 50 rt 72]
```
Drag the turtle inside the area.
```
setc "blue
fill
```

first

first *word-or-list*
Reports the first component of the word or list. See **butfirst**, **butlast**, and **last**.
Examples:
```
show first "hello
h
show first [this is a list]
this
```

fontsize

fontsize
Reports the font size used at the insertion point in the current text box. If text that has more than one font size is selected, **fontsize** reports the first one.
Example:
```
text1,
show fontsize
12
```
The next instructions select all the text in the text box and double its size.
```
top
select
bottom
setfontsize fontsize * 2
```

forever

forever *word-or-list-to-run*
Runs the input repeatedly as an independent parallel process. Use **cancel**, the Cancel menu item, the Stop All menu item, or **Ctrl+Break** to stop the process. See also **launch**.
Examples:
```
forever [rt 1]
forever [fd 1]
cancel [rt 1]
```

forward

forward (fd) *number*

Moves the turtle forward.
Examples:

```
pd fd 20 rt 90
pu fd 50
pd fd 10
```

found?

found?
Reports **true** if the last **search** instruction was successful.
Example:
The following procedure will replace all occurrences of a word by another word in the current text box. Make sure you place the cursor at the top of the text box before running the procedure:

```
to replaceall :this :bythat
search :this
if not found? [stop]
insert :bythat
replaceall :this :bythat
end
```

fput

fput *word-or-list list*
Stands for **f**irst **put**. Reports the list created by adding the first input at the beginning of the second input. The second input has to be a list. See **lput**.
Examples:

```
show fput "a [b c d e f]
a b c d e f
show fput "a [bcdef]
a bcdef
```

freeze

freeze *word-or-list*
freeze *page-name*
Freezes objects so that they cannot be moved, resized, or removed with the mouse. The input is the name of an object on the page, or a list containing many names. A page name can also be used as input to freeze all the elements contained in that page.
For example, if you have "frozen" a text box, you can't move it or resize it. You can open its dialog box, type text in it, or use primitives such as **get** and **set** to make changes. If a turtle is frozen, you cannot use the mouse to change its shape, its size, stamp it, or remove it. **Unfreeze** undoes the effect of **freeze**.
Examples:

```
freeze "t1
freeze [text2 text3 text4]
freeze "page1
```

freezebg

freezebg
Stands for **freeze b**ack**g**round. Freezes the background graphics in their current state. You can still draw over the background and erase the new drawings, but the background that was present before freezing won't be erased. See **unfreezebg**.
Example:

```
pd
rt 11 fd 5000
freezebg
lt 22 fd 5000
```

cg

get

get *object property*
Reports a property of an object in the current project. The first input is the name of an object, a color, or a page. The second input is a property name.
Following is a list of the properties each object can have:
Page: turtles, texts, buttons, sliders, melodies, sounds, music, videos, audiocds, colordemons
Turtle: visible?, rule, on?, own
Button: pos, size, rule, on?
Slider: pos, showname?, limits, value
Text: visible?, pos, size, transparent?, showname?, text
Color: turtlerule, turtlemode, mouseclick
Melody: visible?, pos, on?, showname?, instrument, volume, tempo
Sound: visible?, pos, on?, showname?
Music: visible?, pos, on?, showname?
Video: visible?, pos, on?, showname?
AudioCD: visible?, pos, on?, showname?
announce: pos
question: pos
Note: Melody objects are those created by using the Melody Editor while music objects are those created by importing music in MIDI format.
See **set** and *Creating and Modifying Objects Under Program Control* in *MicroWorlds Help Topics* for more information.
The following examples assume that the objects are in the project.

```
show get "t1 "rule
forever [fd 1]
show get "t1 "on?
true
show get "button1 "rule
launch [page1]
show get "text1 "size
160 100
show get "t1 "own
speed 12
show get "question "pos
354 132
```

getpage

getpage *page-name*
Displays the page. The input must be the name of a page in the current project. Typing the page name alone has the same effect.
Example:
Menu is a page in the current project and this page is not currently showing.

```
getpage "menu
```

getproject

getproject *project-name*
Gets the project indicated (the current project is not saved). The input must be the name of a project in the current directory. See **saveproject**.
Example:

```
getproject "sunset
```

glide

glide *distance speed*
Makes the turtle glide over the distance indicated. The second input tells how fast the turtle will glide. The maximum *speed* for **glide** is 99.
Examples:

```
glide 100 1
glide 100 10
```

greater?

greater? *number1 number2*
Reports **true** if the first input is greater than the second input. See **less?** and ≥.
Examples:

```
show greater? 4 3
true
show greater? 3 slider1
false
```

heading

heading

Reports the current turtle's heading in degrees. The degrees correspond to those of a compass; 0 degrees is due North, 90 is East, 180 is South, and 270 is West. See **seth**.

Examples:

```
cg
show heading
0
seth 180
show heading
180
```

hidetext

hidetext

Hides the current text box. You cannot type in a hidden text box, but primitives such as **print**, **insert**, and **ct** still work. See **showtext**.

Example:

If you have a text box on the page, this instruction flashes the text box.

```
repeat 10 [hidetext wait 5
           showtext wait 5]
```

Text boxes can become invisible and visible through their dialog boxes. Use the eye tool to open the dialog box of a hidden text box.

home

home

Moves the turtle to the center of the page coordinates [0 0], pointing up.

Example:

```
cg
pd fd 50
rt 90
fd 50
home
```

ht

ht

Stands for **h**ide **t**urtle. Hides the current turtle. See **st**.

Example:

```
repeat 10 [ht wait 5 st wait 5]
```

identical?

identical? *word-or-list1 word-or-list2*
Reports **true** if both inputs are identical. A word and a list containing the same word are not identical. Corresponding characters of each input must match in terms of uppercase and lowercase letters. (Style, font, and text color are ignored.) See **equal?.**
Examples:

```
show identical? "a "A
false
show identical? "hello [hello]
false
```

if

if *true-or-false list-to-run*
Runs the instruction list only if the condition (first input) reports **true.** See **ifelse**.
Examples:

```
if colorunder = 15 [bk 50]
question [Are you ready?]
if answer = "yes [glide 100 5]
```

ifelse

ifelse *true-or-false instruction-list1 instruction-list2*
Runs the first instruction list if the condition is **true**. Runs the second instruction list if the condition is **false**. See **if**.
Example:

```
ifelse colorunder = 15
       [fd 50]
       [bk 50]
```

This instruction can be used to get an answer to a question dialog box. If the answer is not empty, it is displayed in a text box.

```
to insist
question [Your name please...]
ifelse empty? answer
      [insist]
      [settext1 answer]
end
```

insert

insert *word-or-list*
Prints the input at the cursor position, in the current text box. See **print**.
Examples:

```
text1,
insert "hi
insert char 32      Leaves a space.
pr "there
```

int

int *number*
Stands for **int**eger. Reports the integer portion of its input. See **round**.
Examples:

```
show int 9.9999
9
```

```
show int 2.5
2
```

item

item *number word-or-list*
Reports the specified element of a word or a list. The first input must be between 1 and the number of elements in the word or the list.
Examples:

```
show item 2 "hello
e
show item 3 [this is a list]
a
```

key?

key?

Reports **true** if a key is being pressed on the keyboard. You must click on the background of the page (outside of a text box, the Command Center, or Procedures page) for **key?** to work. Use **readchar** to reset **key?** to **false**.

Example:

First run the instruction and press a key after MicroWorlds starts displaying the word **false**.

```
repeat 10 [show key? wait 5]
```

Click on the page.

```
false
false
true      You pressed a key.
true
...
```

last

last *word-or-list*
Reports the last component of the word or list. See **first** and **butlast**.
Examples:

```
show last "hello
o
show last text1
m
show last parse text1
Kim
```

launch

launch *word-or-list-to-run*
Runs the input as an independent parallel process. If the process is launched from the Command Center, the cursor reappears immediately. Use **cancel**, the Cancel menu item, the Stop All menu item, or **Ctrl+Break** to stop the process. See also **forever**.
Example:

```
launch [glide 1000 1]
```

Type the next instructions while the turtle is gliding.

```
rt 90
lt 90
```

left

left (lt) *number*
Turns the turtle to the left.
Example:

```
repeat 10 [fd 40 bk 20 lt 36]
```

less?

less? *number1 number2*
Reports **true** if the first number is less than the second number. See **greater?** and ≤.
Example:

```
show less? 22 22.5
true
```

let

let *list-of-names-and-values*
Creates one or many temporary variables. The variables will exist only while the procedure containing the **let** instruction and procedures called by this procedure are running. The input is a list of paired variable names and values. **Let** can only be used in a procedure. See **local**.
Example:

```
to move
let [dist 100 head 90 delay 300]
right :head
wait :delay
fd :dist
end
```

Try the procedure in the Command Center. The turtle will move. When the procedure is over, check the value of the variables. Variables have lost their values. They don't even exist.

```
move
show :dist
```

dist has no value
The instruction **let [dist 100 head 90 delay 300]** is equivalent to:
```
local [dist head delay]
make "dist 100
make "head 90
make "delay 300
```

listen

listen
Sets the global "turtle who" (the turtle obeying instructions from the Command Center). This command allows you to change the global "turtle who" within a process that has been launched (e.g., from a button). For detailed information on how MicroWorlds handles processes, see *Local and Global Who* in *MicroWorlds Help Topics.* See also **talkto** and **ask**.
Note: If you used a list as input to **talkto** before running **listen**, **listen** will display an error message.
Example:
Create 2 turtles: t1 and t2. Click on t2 to make it active. Test that t2 is active:
```
fd 50
show who
t2
```
Create the following button:
{bmc graphics\vocab\t1but.bmp}
Click on this button. The other turtle, t1, goes back 50 steps. Repeat the instruction:
```
show who
t2
```
T2 is still the turtle that obeys instructions from the Command Center. The turtles addressed inside buttons (or other processes) are "local turtles" and don't affect the "global turtle." To change the "global turtle" while running a process (e.g., inside a button), use **listen**.
Change the button instruction to:
{bmc graphics\vocab\t1lis.bmp}
Click on this button. Write the following instructions in the Command Center:
```
fd 50
show who
t1        Now t1 is the "global turtle."
```

list

list *word-or-list1 word-or-list2*
(**list** *word-or-list1 word-or-list2 word-or-list3....*)
Reports one list made by combining the inputs (words or lists). If more than two inputs are used, **list** and its inputs must be enclosed in parentheses. See **sentence**.
Examples:
```
show list 2 3
2 3
show list "a [b]
a [b]
show (list "a "b "c "d)
a b c d
make "x 10 make "y 20
setpos list :x :y
```

list?

list? *word-or-list*
Reports **true** if the input is a list. See **word?.**
Example:

```
show list? [Hello there]
true
```

ln

ln *number*
Stands for **n**atural **l**ogarithm. Reports the natural logarithm (the logarithm in base *e*) of the *number*. Inverse of **exp**. See also **log**.
Example:

```
show ln 1
0
```

loadpict

loadpict *path*
Stands for **load pict**ure. Loads the picture on the current page. The input must be the name of a picture file that MicroWorlds supports in the current directory or a full or relative path. A full path starts with the name of the drive. Backslashes are used to separate the names of directories, subdirectories, and files. MicroWorlds supports the following formats: BMP, GIF, JPEG (extension jpg), PCX, and Targa (extension TGA). See **savepict** and *Importing Pictures* in *MicroWorlds Help Topics.*
Examples:

```
loadpict "scene
loadpict "c:\projects\scene
loadpict "media\tornado1
```

If one of the elements of the path has spaces, vertical bars must enclose the whole path:

```
loadpict "|c:\projects\my scene|
```

loadshape

loadshape *file shape-number*
Loads the specified picture file into that shape. The picture file name will become the name of the shape if there is no other shape with that name in the project.
The *file* must be the name of a picture file that MicroWorlds supports in the current directory or a full or relative path. A full path starts with the name of the drive. Backslashes are used to separate the names of directories, subdirectories, and files. MicroWorlds supports the following formats: BMP, GIF, JPEG (extension jpg), PCX, and Targa (extension TGA). The *shape-number* can be any number from 1 to 64. See **savepict** and *Importing Shapes* in *MicroWorlds Help Topics.*
Example:

```
loadshape "volcano 20
```

Loads the volcano picture into shape 20. The shape will be named volcano.

loadtext

loadtext *path*
Loads the text file in the current text box or on the Procedures page. The input must be the name of a text file in the current directory or a full or relative path. A full path starts with the name of the drive. Backslashes are used to separate the names of directories, subdirectories, and files. See **savetext** and *Importing Text* in *MicroWorlds Help Topics.*
Examples:

```
loadtext "story
loadtext "c:\projects\story
loadtext "media\quake-Italy
```

If one of the elements of the path has spaces, vertical bars must enclose the whole path:

```
loadtext "|c:\My projects\story|
```

Loadtext can load both TXT and RTF formats. The default is TXT. Note that text saved with TXT format will be plain (font, style, color, for example, will not be saved) while text saved under RTF will keep the current text font, style, and color.

```
loadtext "NiceText.rtf
```

local

local *word-or-list*
Makes the specified variable local to the procedure where **local** is used. **Local** can only be used in a procedure. See **let**, **make**, and **name**.
Example:

```
to move
local "dist
make "dist 100
fd :dist
end
```

Try the procedure in the Command Center. The turtle will move. When the procedure is over, check the value of the variables. Variables have lost their values. They don't even exist.

```
move
show :dist
dist has no value
```

log

log *number*
Stands for **log**arithm. Reports the logarithm of the number. See **ln** and **exp**.
Example:

```
show log 100
2
```

lput

lput *word-or-list list*
Stands for **l**ast **put**. Reports the list created by adding the first input at the end of the second. See **fput**.
Examples:

```
show lput "f [a b c d e]
a b c d e f
show lput "s [language]
language s
```

make

make *word word-or-list*
Creates a variable and gives it the value *word-or-list.* These variables keep their values as long as you don't clear them or quit MicroWorlds. They are not saved with your project. If you want your variables to have specific values each time the project is loaded, you should have a **startup** procedure. See *Startup Procedure* in *MicroWorlds Help Topics,* **name, thing, clearname, names,** and **createprojectvar.**
Example:

```
make "class [Peter Dennis Geni]
show :class
Peter Dennis Geni
```

Here is an example of a **startup** procedure that resets the global variables in a project.

```
to startup
make "count 0
make "list []
end
```

member?

member? *word-or-list1 word-or-list2*
Reports **true** if the first input is a component of the second.
Examples:

```
show member? "a [a b c]
true
show member? "Lucy text1
false
```

merge

merge *project-name word-or-list-of-types*
Copies pages, procedures, or shapes from another project into the current project. The first input must be the name of a project in the current folder. This is the project you want to copy from. To set the current folder, choose Save As from the File menu, find the directory that contains the source project, and click on Cancel instead of saving. You can also use **chdir.**
The second input can be the name of a specific page, a list of page names, the word **procedures** (to import the procedures and project variables), **pages** (to import all the pages), or **shapes** (to import the shapes that have been modified in the source project). If there are names in the new project that are the same as those in the current project, the new names will be renamed with a number.
Warning: There is a limit for the amount of pages that you can import. Pages take up a lot of memory space especially if they have many objects or complex drawings. Always save before using **merge**.
Examples:

```
merge "clocks "procedures
```

Imports the procedures and project variables.

```
merge "clocks "pages
```

Imports all the pages.

```
merge "clocks [page1 page3]
```

Imports individual pages.

```
merge "clocks "shapes
```

Imports all modified shapes.

minus

minus *number*
Reports the additive inverse of its input. **Minus** must be used to report the additive inverse of a variable (**minus :num** instead of **-:num**). See **-** and **difference**.
Examples:

```
show -5
-5
show minus 5
-5
show -ycor
I don't know how to -ycor
show minus ycor
-100
```

mousepos

mousepos

Stands for **mouse pos**ition. Reports the page coordinates representing the current mouse position on the screen. See **setpos**.

Examples:

```
show mousepos
60 63
show first mousepos
60
```

If there is a turtle on the page, the turtle will follow the mouse.

```
t1,
pd
forever [setpos mousepos]
```

Choose the Cancel menu item, the Stop All menu item, or **Ctrl+Break** to stop this process.

name

name *word-or-list word*
Creates a variable and gives it the value *word-or-list.* These variables keep their values as long as you don't clear them or quit MicroWorlds. They are not saved with your project. See **names**, **clearnames**, **make**, and **createprojectvar**.
Example:

```
name [Peter Anne Geni] "class
show :class
Peter Anne Geni
show first :class
Peter
```

namepage

namepage (np) *page-name*
Names the page being displayed with the input. **Namepage** corresponds to the Name Page item in the Pages menu.
Example:

```
namepage "presentation
```

name?

name? *word*
Reports **true** if the input is the name of a variable. See **make** and **name**.
Example:

```
make "age 10
show name? "age
true
```

names

names
Reports the names of all the variables with their values.
Examples:

```
make "friends [Joanne Lea]
make "age 12
show names
make "friends [Joanne Lea]
make "age 12
```

If you have a text box on the page:

```
pr names
```

will print the above list in a text box.

An interesting use of **names** in regard to files is the following:

```
make "filelist projectlist
pr names
```

will print all the file names with the vertical bars showing when a name is more than one word.

newbutton

newbutton *name [x y] instruction-list*
Creates a new button with the name and instruction specified, at the position *[x y]* indicated. The *name* cannot be more than 32 characters (including spaces). The position *[x y]* is the top, left corner of the button. The button is created in the Once mode. It will be sized to fit the *instruction-list.* See **set** to change the settings of the button.

Example:
newbutton "button1 [50 50] [fd 1]

newpage

newpage
Opens a new page called *Pagex* (*x* is the next available number).
Example:
newpage

newprojectsize

newprojectsize *list-of-numbers*
Sets the page size on the computer screen for new projects. The input is a list of two numbers: the width and the height of the page in turtle steps. Before using **newprojectsize**, there must be an empty project on the screen. The page size of a project is saved with the project. The minimum size is 300 by 300 and the maximum size is determined by the current display setting in the Control Panel. The standard project size is 744 by 426 when the display setting (in the Control Panel) is 800 x 600. The standard project size is 592 by 322 when the display setting (in the Control Panel) is 640 x 480. See **projectsize**.
Note: If you create a project that is smaller than the minimum size, there will be a black border around it to keep the menu display intact. See the Read Me file in the MicroWorlds Web Player on the LCSI web site (http://www.lcsi.ca).
Examples:
Choose New Project from the File menu so there is a perfectly empty project on the screen.
newprojectsize [300 300]
newprojectsize "standard Resets the project size to the default size.

newslider

newslider *name [x y] [min max current]*
Creates a new slider using the specified name at the position indicated. The position *[x y]* is the top, left corner of the slider. The last input is a list of three numbers representing the minimum, maximum, and current value of the slider. The minimum and maximum values are -9999 and 9999 respectively. See **set** to change the slider's settings.
Example:
newslider "step [100 100] [0 8 3]
show step
3
setstep 5 Sets the slider's value to 5.
setstep [10 20 15] Sets the slider to a minimum 10, a maximum 20, and a current value of 15.

newtext

newtext *name [x y] [xsize ysize]*
Creates a new text box using the name and size specified at the position *[x y]* indicated. The position is the top, left corner of the box. The maximum *[xsize ysize]* is the size of the page in the project. See **set** to change the text box's properties.
Example:
newtext "info [100 100] [50 50]
setinfo [Click on Next]

newturtle

newturtle *name*

Creates a new turtle with the name indicated. The new turtle appears at the position [0 0] and is hidden. Use **st** to make it visible.
Example:

```
newturtle "Shelly
st
glide 100 2
```

not

not *true-or-false*
Reports the logical inverse of its input. See **and** and **or**.
Example:

```
show empty? []
true
show not empty? []
false
```

note

note *number duration*

Plays a note using the current instrument. The first input is the MIDI note number and the second is the duration in tenths of a second. Middle C is 60. The maximum *number* for **note** is 127; the maximum *duration* is 255.

Following is a list of MIDI values. Each line corresponds to one interval.

A	A#	B	C	C#	D	D#	E	F	F#	G	G#
			1	2	3	4	5	6	7	8	
9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64	65	66	67	68
69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92
93	94	95	96	97	98	99	100	101	102	103	104
105	106	107	108	109	110	111	112	113	114	115	116
117	118	119	120	121	122	123	124	125	126	127	

Examples:

```
note 60 2     Plays middle C.
note 64 5
```

number?

number? *word-or-list*
Reports **true** if the input is a number.
Example:

```
show number? 3.1416
true
```

Note that the decimal symbol can be a point or a comma depending on the configuration of your Windows computer. You can change the symbol using Regional Settings of Number in the Control Panel. If you intend to run your project with different regional settings (such as through the Internet), we suggest using fractions instead of decimal numbers.

opaque

opaque *text-box-name*
Makes the designated text box opaque. This is equivalent to unchecking Transparent in the text box's dialog box. See **transparent**.
Example:

```
transparent "text1
opaque "text1
```

or

or *true-or-false1 true-or-false2*
(**or** *true-or-false1 true-or-false2 true-or-false3...*)
Reports **true** if any of its inputs report **true**. If more than two inputs are used, **or** and its inputs must be enclosed in parentheses. See **and** and **not**.
Examples:

```
show or (2 = 2) (3 = 5)
true
cg
show or (2 * 4) = 8 pos = [0 0]
true
show (or (2 = 2) (3 = 5) (8 = 9))
true
```

output

output (op) *word-or-list*
Stops the procedure and reports a word or list.
Examples:

```
to firstnumber :word
if empty? :word [output "none]
if number? first :word
   [output first :word]
output firstnumber bf :word
end

show firstnumber "abcde5fgh
5
```

pagelist

pagelist

Reports a list containing the names of all the pages in the current project. The first name reported by **pagelist** is always the current page.

Examples:

```
show pagelist
menu page2 practice
show member? "menu pagelist
true
```

parse

parse *word*

Turns character strings into plain lists. Spaces and carriage returns, and line feed sequences contained in the character string become separators in the list. **Parse** can be used to turn the long word (a sequence of characters including spaces) reported by text boxes into lists of words.

Example:

```
show text1
Hi out there
show count text1
12              A word of 12 characters, including spaces.
show count parse text1
3               A list of three words.
```

paste

paste

Pastes a copy of the Clipboard in the current text box. The Clipboard contains the last text that has been cut or copied using the **cut** or **copy** command, or the equivalent Edit menu items. See also **select**.

Example:

Select some text and choose Copy from the Edit menu. Then go to the Command Center and type:

```
paste
```

pd

pd

Stands for **p**en **d**own. Puts down the pen of the current turtle. The turtle will then leave a trace when it moves, but not when it is dragged. See **pu.**

Example:

```
repeat 10 [pu fd 10 pd fd 10]
```

pensize

pensize

Reports a number representing the pen size of the current turtle. The original pen size is 1. The maximum is 100. See **setpensize.**

Example:

```
t1, setpensize 10
setc "pink
pd fd 50
show pensize
10
```

pi

pi

Reports the constant pi.
Example:
```
show pi
3.14159265359
```

pick

pick *word-or-list*
Reports an element chosen randomly from the word or the list. Picking from a word reports a character, picking from a list reports an element of the list (a word or a list). See **textpick**.
Examples:
```
show pick "hello
e
show pick [to all my friends]
my
```

pictlist

pictlist
Stands for **pict**ure **list**. Reports a list containing the names of picture files in the current directory. Only the files with picture formats that MicroWorlds supports (created using **savepict** or other applications) are shown. See **textlist, projectlist,** and **files.**
Example:
```
show pictlist
mybackground.bmp logo.gif
```
A file name made up of more than one word will look like two files in the list. Use **item** to find the actual name. In this example, autumn scene is the name of one file.
```
show pictlist
mybackground.bmp autumn scene.gif
show item 2 pictlist
autumn scene.gif
```

placepict

placepict *pict-file-name position size*
Stands for **place pict**ure. Imports a picture file, places it at the designated position, and adjusts its size to fit the size indicated. The first input is the file name in the current directory or a full or relative path; the second input is the position for the top, left corner of the picture, and the third input is the size of the image on the page, in *x* and *y* turtle steps.
Example:
```
placepict "balloon [0 0] [100 100]
```

pos

pos
Stands for **pos**ition. Reports the position of the turtle. The result is a list of two numbers. [0 0] is the position at the center of the page. See **setpos**.
Example:
```
show pos
50 50
```

power

power *number1 number2*
Reports *number1* raised to the power of *number2*.
Examples:
```
show power 2 10
```

1024

presentationmode

presentationmode
Hides the Command Center, Tool Palette, the project's title bar, and MicroWorlds' menus. The project is centered on the screen and the background is filled in. **Presentationmode** is used to display completed projects. To return to MicroWorlds' regular mode, use **presentationmode** again or click outside the MicroWorlds project window. This command corresponds to the Presentation Mode item in the Gadgets menu.
Example:
presentationmode

print

print (pr) *word-or-list*
Prints a word or list in the current text box. The text is followed by a carriage return and line feed sequence. See **insert**.
Example:
repeat 5 [print "hello]

printtext

printtext
Prints out, on the printer, the contents of the current text box or the Procedures page, depending on what is currently showing. All text is printed, even text that's not visible. **Printtext** opens the printing dialog box.
Example:
Try this with a text box showing or with the Procedures page showing.
printtext

procedures

procedures
Opens the Procedures page. This is equivalent to selecting the Procedures item from the Pages menu.
Note: Each page name in the current project can also be used as a command.
Example:
procedures Opens the Procedures page.
printext
page1

product

product *number1 number2*
(**product** *number1 number2 number3...*)
Reports the result of multiplying its inputs. If more than two inputs are used, **product** and its inputs must be enclosed in parentheses. See *.
Examples:

```
show product 3 3
9
show (product 3 3 3)
27
show product 1000000 1000000
1e+012
```

projectlist

projectlist

Reports a list containing the names of MicroWorlds projects in the directory. See **textlist, pictlist,** and **files.**
Example:

```
show projectlist
maze myadventure
```

A project name made up of more than one word will look like two projects in the list. Use **item** to find the actual name. In this example, my adventure is the name of one project.

```
show projectlist
maze my adventure
show item 2 projectlist
my adventure
```

projectsize

projectsize
Reports the current project size, in turtle steps. See **newprojectsize**.
Example:

```
show projectsize
744 426
```

projectvars

projectvars
Stands for **project var**iable**s**. Reports the list of currently defined project variables. See **createprojectvar**.
Example:

```
show projectvars
amount speed inventory
```

pu

pu
Stands for **p**en **u**p. Lifts up the pen of the current turtle. The turtle will not leave a trace when it moves. See **pd**.
Example:

```
repeat 10 [pu fd 10 pd fd 10]
```

question

question *word-or-list*
Opens a dialog box displaying the question and an area to type the answer. **Answer** reports what was typed in the dialog box. If you write a very long question, only the part that fits will be displayed.
To reposition the default alert box, use the **set** command. The position is in turtle coordinates: the default position is [-200 50]. This is the position of the upper, left corner of the alert box. If the position accidentally places part of the box outside of the screen, use **Enter** or **Esc** to close the box.
If this variable is changed, you should have a **startup** procedure to reset it each time you load the project.
See **get** and **set**.
Example:

```
question [How old are you?]
```

Type the answer in the dialog box. In the Command Center, type:

```
show answer
I'm 10          Your answer.
```

This procedure can be used to verify that the user has actually answered a question dialog box:

```
to insist
question [Your name please...]
if empty? answer [insist]
end
```

quotient

quotient *number1 number2*
Reports the result of dividing *number1* by *number2*. See /.
Example:

```
show quotient 3 3
1
```

random

random *number*
Reports a random non-negative integer less than *number*.
Example:
```
show random 100
22
```
This procedure will simulate the roll of a die:
```
to die
show 1 + random 6
end
```

readchar

readchar
Pauses the execution and waits for a character to be typed on the keyboard. You must click on the background of the page (outside of a text box, the Command Center, or Procedures page) for **readchar** to recognize the character typed. See **key?.**
Example:
```
to getchoice
show [Type your choice, A, B, or C]
make "answer readchar
show :answer
end
```

recycle

recycle
Frees up unused Logo memory space. See **space**.
Example:
```
show space recycle show space
251036    Your results will differ.
433004
```

remainder

remainder *number1 number2*
Reports the remainder after *number1* is divided by *number2*. The remainder of a negative number will be negative. If *number1* and *number2* are non-integers, they are rounded to the nearest integers.
Example:
```
show remainder 100 3
1
```

remove

remove *name*
Deletes an object, page, project variable created with **createprojectvar,** or a turtle variable created with **turtlesown**. If the named object is not on the current page, MicroWorlds will look for it on other pages in the project. If the input to **remove** is **procedures**, the Procedures page in the current project will be cleared. If you remove the only page of a project, a new page called Page1 will be created.
Examples:
```
remove "text
remove "t1
remove "page1
```

repeat

repeat *number list-to-run*
Runs the list of instructions the specified number of times. See **dotimes** and **dolist** for more advanced features.
Example:

```
repeat 10 [setsh "bird1 wait 5
           setsh "bird2 wait 5]
```

rerandom

rerandom
Reproduces the same sequence of numbers generated by **random**. After running **rerandom**, **random** with the same input generates the same sequence of numbers the next time.
Examples:

```
rerandom
repeat 2 [show random 10]
8
5
rerandom
repeat 2 [show random 10]
8
5
```

resetvideo

resetvideo *word*
Resets the video to the beginning.
Example:

```
resetvideo "Video1
```

resett

resett
Stands for **reset t**imer. Resets the timer to 0. The timer starts when you start up MicroWorlds. See **timer**.
Example:

```
resett show timer
0
```

The next procedure displays a question just after resetting the timer. If you got the right answer, it tells you how fast you were at typing it. The time is in tenths of a second.

```
to reflex
resett
let [num1 1 + random 10
     num2 1 + random 10]
question (se [What is] :num1 [times] :num2 [?])
ifelse answer = :num1 * :num2
  [announce se timer / 10 "sec]
  [announce [Wrong answer]]
end
```

rest

rest *duration*
Inserts a rest in a sequence of notes. The duration is in tenths of a second, and has a maximum of 255. See **note**.
Example:

```
note 60 5 note 62 5 note 64 5
```

```
rest 5 note 60 5 note 62 5
```
Or, as a procedure:
```
to song
note 60 5
note 62 5
note 64 5
rest 5
note 60 5
note 62 5
end
```

restore

restore
Restores the background to the way it was the last time a **snapshot** command was issued. The turtle's position does not change. Everything else remains intact.
Example:
```
pd
fd 50 rt 90
snapshot
clean
restore
```
You can use **restore** to "transport" an image to another page or to another project. Just use **snapshot** on the page that you want to keep a copy, then go to another page or open another project and use **restore**.

right

right (rt) *number*
Turns the turtle to the right.
Example:
```
pd repeat 10 [fd 40 bk 20 rt 36]
```

round

round *number*
Reports the number given as input rounded to the nearest integer. Numbers ending with .5 are rounded to the higher integer.
Examples:
```
show round 3.3333
3
show round 2.5
3
show round 3.5
4
```

run

run *word-or-list-to-run*
Runs a word or an instruction list.
Example:
The next instruction runs the contents of a text box.
```
run text1
```

savehtml

savehtml *directory*
Saves the current project as a series of HTML pages in a new *directory* created in the current folder. This primitive saves your project into a Web page (or pages) with limited functionality. You must use buttons or turtles to change pages in the project. Each page is a snapshot of the screen: animation, text, video, sound, and music icons are part of the background and do not react to clicks. For further information about this feature, see *Creating Web Pages* in *MicroWorlds Help Topics.*
Example:
Suppose you have a project with two pages named Room and Lake:

```
savehtml "Champlain
```

On the desktop, go to the current folder. You should now have a directory named Champlain with twice the number of files as pages. In this example, you will have the following files:
Lake.html
lake.gif
Room.html
room.gif

If you have a procedure called **links** on the Procedures page of your project, you can link to other Web pages outside of your project. You can make as many links as you like inside the **links** procedure. Here is an example of a **links** procedure:

```
to links
make "lcsi "http://www.lcsi.ca
end
```

One of the pages in your project should have a button or turtle running the instruction **lcsi** to link to the Web site.
For more information about how to test your Web pages locally (running from your computer and not from the server), see *Creating Web Pages* in *MicroWorlds Help Topics.*

savepict

savepict *path*
Stands for **save pict**ure. Saves the background of the current page as a picture file. MicroWorlds supports the following picture formats: BMP, GIF, JPEG (extension jpg), PCX, and Targa (extension TGA). The default is BMP.
When you use the **savepict** command, the turtles, buttons, text boxes, and other objects are not part of the background. Stamped images and stamped text are part of the background. Type in the Command Center:
savepict "mypict Use a file name of your choice.
By default it will save it in BMP format. If you want to save it in a different format (that MicroWorlds supports), add the appropriate extension to the file name:
savepict "mypict.jpg Use a file name of your choice.
You can also include the full or relative path when you want to save that picture in a directory other than the current one. Backslashes are used to separate the names of directories, subdirectories, and files. A file name cannot be longer than 32 characters including spaces. See *Exporting Pictures* in *MicroWorlds Help Topics.*
Examples:

```
savepict "scene
savepict "media\mytornado
savepict "C:\projects\scene
savepict "scene.jpg
savepict "C:\projects\scene.jpg
```

If one of the elements of the path has spaces, vertical bars must enclose the whole path:

```
savepict "|C:\My projects\scene|
```

saveproject

saveproject
Saves the current project without closing it. The project must have a name in order for this command to work. This is equivalent to the Save item in the File menu. Use this primitive if you want to save your project before getting another project with **getproject**.
Example:
Name the current project first. Add some turtles.

```
saveproject
```

saveshape

saveshape *file-name number*
Saves the specified shape as a picture file.
Examples:

```
saveshape "moon 1
```

By default it will save it in BMP format. If you want to save it in a different format (that MicroWorlds supports), add the appropriate extension to the file name:

```
saveshape "moon.jpg 1
```

Use a file name of your choice.

savetext

savetext *path*
Saves the text found in the current text box or on the Procedures page in a text file format. MicroWorlds supports TXT and RTF (Rich Text Format) formats. The default is TXT. Note that text saved under TXT format will save plain text (font, style, color, for example, will not be saved) while text saved under RTF will keep the current text font, style, and color.
Examples:

```
savetext "story
```

Will save in TXT format.

```
savetext "C:\proj\story
```

Will save in TXT format.

```
savetext "story.rtf
```

Will save in RTF format.

```
savetext "C:\proj\story.rtf
```

Will save in RTF format.

The text can be loaded with **loadtext**. The input can be a file name with or without an extension (and the file will be saved in the current directory) or it can be a full or relative path. Backslashes are used to separate the names of directories, subdirectories, and files. See *Exporting Text* in *MicroWorlds Help Topics.*

```
savetext "story
savetext "projects\story
savetext "C:\projects\story.rtf
```

If one of the elements of the path has spaces, vertical bars must enclose the whole path:

```
savetext "|C:\My projects\story|
savetext "|C:\My projects\story.rtf|
```

search

search *word*
Tells MicroWorlds to search and select (highlight) the word in the current text box. Nothing happens if the word is not found. **Search** starts searching at the insertion point. Use **unselect** to undo the highlighting effect of **search**. See also **found?**.
Example:
The following procedure will replace all occurrences of a word by another word in the current text box. Make sure you place the cursor at the top of the text box before trying the procedure:

```
to replaceall :this :bythat
search :this
if not found? [stop]
insert :bythat
```

```
replaceall :this :bythat
end
```

select

select
Tells MicroWorlds to start selecting text in the current text box. Any cursor motion (**top**, **bottom**, **cu**, **cd**, **cf**, and **cb**) will select text.
Example:

```
text1,
top
select
cd
cut
bottom
paste paste
```

sentence

sentence (se) *word-or-list1 word-or-list2*
(**sentence** *word-or-list1 word-or-list2 word-or-list3...*)
Reports a list which is made up of its inputs (words or lists). **Sentence** can take more than 2 inputs when **sentence** and the inputs are enclosed in parentheses. See **list**.
Examples:

```
show sentence "a "b
a b
show (sentence "hi "there [Bill])
hi there Bill
```

set

set *object property value*
Sets a property for an object to the specified value. The first input is the name of an object. The second input is a property name and the last one is the value. See **get** and *Creating and Modifying Objects Under Program Control* in *MicroWorlds Help Topics.*
Following is a list of the properties for each object that can be modified by **set**:

Turtle:	rule, on?
Button:	pos, size, rule, on?
Slider:	pos, showname?, limits, value
Text:	visible?, pos, size, transparent?, showname?, text
Color:	turtlerule, turtlemode, mouseclick
Melody:	visible?, pos, on?, showname?
Music:	visible?, pos, on?, showname?
Sound:	visible?, pos, on?, showname?
Video:	visible?, pos, on?, showname?
AudioCD:	visible?, pos, on?, showname?
announce:	pos
question:	pos

Note: Melody objects are those created by using the Melody Editor while music objects are those created by importing music in MIDI format.
Examples:

```
set "text1 "visible? "false
set "t1 "rule [launch [seth random 360 fd 50]]
set "slider1 "showname? "true
set "red "turtlerule [silly-sound]
set "red "mouseclick [announce [You win!]]
set "announce "pos [0 0]
```

setbg

setbg *name-or-number*
Stands for **set b**ack**g**round. Sets the background color for the page. The input can be the name of a color or a number. If the input is a name, a quotation mark must precede it. Use **setbg 0** or **setbg "white** to reset the original background color to white. You can't use **setbg** on a frozen background. See **bg, unfreeze,** and **loadpict.**
Note: If you set the background color to the same color as portions of your drawings, the drawings will be integrated with the background (and lost).
Examples:

```
setbg "red
setbg 22
setbg 0
```

Resets the original background color.
Note: In order to use the full potential of MicroWorlds graphics, 16 bit color mode is recommended.

setc

setc *name-or-number*
Stands for **set c**olor. Sets the color of the turtle's pen. If the turtle has its original shape, it will change color to show the pen color. The input can be the name of a color or a number. If the input is a name, a quotation mark must precede it. The original pen color is black, or color number 9. The input can be an integer or a one place decimal. See **color**.
Examples:

```
setc "red
setc 12
```

When using decimal numbers, MicroWorlds tries to find the best color match. For example:

```
setc 19.9
```

You may assume that this should be a very dark red. Yet it becomes black.

Try this example in 16 bit color mode:

```
seth 45
pd
repeat 100 [setc color + 0.1 fd 20]
```

Note: In order to use the full potential of MicroWorlds graphics, 16 bit color mode is recommended.

setfont

setfont *word*
Sets the font for the selected text in the current text box. If no text is selected, **setfont** sets the cursor to use that font for typing. The input must be the name of a font in your system. You can also set the text font by choosing Font from the Text menu. We recommend using "True Type Fonts" (those with the symbol in the font dialog box). True type fonts allow many font sizes and style settings. See **settc, setfontsize,** and **setstyle.**
Example:

```
setfont "Arial
print [Hello there]
```

The next instructions select all the text in the text box and set its font.

```
top
select
bottom
setfont "Impact
unselect
```

If the name of the font has spaces, vertical bars must enclose the whole name:

```
setfont "|Times New Roman|
```

setfontsize

setfontsize *number*
Sets the font size for the current text box. If no text is selected, **setfontsize** sets the cursor to use that font size for typing. The maximum *number* for font size is 1638. You can also set the text size by choosing Font from the Text menu. We recommend using "True Type Fonts" (those with the symbol in the font dialog box). True type fonts allow many font sizes. See **setfont, settc,** and **setstyle.**
Example:

```
ct
setfontsize 12
pr [To be or not to be]
```

The next instructions select all the text in the text box and double its size.

```
top
select
bottom
setfontsize fontsize * 2
unselect
```

setfooter

setfooter *word-or-list*
Sets the contents of the footer on printouts. Use the empty list as input if you do not want a footer. The default footer is: MicroWorlds from LCSI.
Examples:

```
setfooter [Logo Lovers, Grade 4]
setfooter []
```

seth

seth *number*
Stands for **set h**eading. Sets the turtle's heading to the specified direction (in degrees). The degrees correspond to those of a compass: 0 degrees is due North, 90 is East, 180 is South, and 270 is West. **Right** and **left** turn a turtle a number of degrees from its current heading. **Seth** makes a turtle point to a specific direction, regardless of its previous heading. See **heading**.
Examples:

```
seth 0
seth 90
```

setinstruction

setinstruction *instruction-list*
Sets the instruction for the current turtle. This is equivalent to typing the instructions in the turtle's dialog box. If the list includes the word **forever** or **launch**, the instruction's mode is set to Many Times or Once accordingly. If these words are not included, the mode is unchanged.
Note: If you used a list as input to **talkto** before running **setinstruction**, **setinstruction** will display an error message.
Examples:

```
setinstruction [fd 50]
clickon
clickoff
setinstruction [forever [fd 1]]
clickon
```

setinstrument

setinstrument *name-or-number*

Sets the instrument for the next **note** commands. There are 7 instrument names: piano, harpsichord vibraphone, guitar, violin, clarinet, and kalimba. You can also use any number from 1 to 128.
Examples:

```
setinstrument "violin
note 65 10
setinstrument 116
note 65 10
```

setpensize

setpensize *number*
Sets the turtle's pen size which determines the thickness of the lines it will draw. The pen size can also be set by picking a pen size and the pencil in the Drawing Center and clicking on a turtle. The original pen size is 1. The maximum pen size is 100.
Examples:

```
pd
setpensize 20 fd 50
setpensize 30 fd 50
```

setpos

setpos *[x y]*
Stands for **set pos**ition. Moves the turtle to the designated *x y* coordinates. The center of the page is [0 0]. The maximum number for *x* and *y* is 9999. Note that if a turtle lands on a programmed color after a **setpos** command, the color won't react: only **fd** and **bk** activate programmed colors. See **pos.**
Example:

```
setpos [50 50]
```

setshape

setshape (setsh) *name-or-number*
setshape (setsh) *list-of-names-or-numbers*
Gives a shape to the turtle. If the input is a name, a quotation mark must precede it. The maximum number for **setshape** is 64. When a list of shape names or numbers is given as input, each **fd** and **bk** command makes the turtle cycle through the list of shapes (the maximum number of items in the list is 64). The shape can also be set by picking a shape from the Shapes Center and clicking on a turtle. See **shape** and *Animation Techniques* in *MicroWorlds Help Topics.*
Examples:

```
setshape 12
setshape "moon
repeat 25 [setsh "bird1 fd 2 setsh "bird2 fd 2]
setsh [dog1 dog2]
repeat 10 [fd 5]
glide 100 5
```

setsize

setsize *number*
Sets the size of the turtle. The original size is 40 and the maximum size is 160. Turtle shapes look nice when they are multiples of 20. You can also change the size of the turtle with the magnifiers.
Examples:

```
setsize 20
setsize 40
```

setstyle

setstyle *word-or-list*
Sets the font style in the current text box. The input must be the name of a style (regular, bold, italic, underline). If no text is selected, **setstyle** sets the cursor to use that style for typing. Multiple styles can be applied by inserting styles in a list. You can also set the text style by choosing Font from the Text menu. We recommend using "True Type Fonts" (those with the symbol in the font dialog box). True type fonts allow many style settings. See **setfont, setfontsize,** and **settc.**
Examples:

```
setfont "Courier
setstyle "bold
insert "tic
setstyle "italic
insert "tac
setstyle [bold italic]
print "toe
```

settc

settc *name-or-number*
Stands for **set t**ext **c**olor. Sets the color of the text in the current text box. The input can be the name of a color or a number. If the input is a name, a quotation mark must precede it. The original text color is black, or color number 9. If no text is selected, **settc** sets the text color for the next characters typed. You can also set the text color by choosing Color from the Text menu. See **setfont, setfontsize,** and **setstyle**.
Examples:

```
settc "red
insert "O
settc 104
print "K
```

setx

setx *number*
Sets the x coordinate of the current turtle. The y coordinate remains unchanged.
Examples:

```
setx 100
setx -100
setx xcor - 10
```

sety

sety *number*
Sets the y coordinate of the current turtle. The x coordinate remains unchanged.
Examples:

```
sety 100
sety -100
sety ycor - 10
```

shape

shape
Reports the shape name or number, or a list of shape names or numbers of the current turtle. See **setshape**.
Examples:

```
t1,
setsh "heart
show shape
heart
```

```
setsh 6
setsh shape + 1
```

show

show *word-or-list*
Prints a word or a list in the Command Center.
Examples:

```
show "hello
hello
show [hello there]
hello there
```

showtext

showtext
Makes the current text box visible. See **hidetext**.
Example:
If you have a text box on the page, this instruction flashes the text box.

```
repeat 10 [hidetext wait 5
           showtext wait 5]
```

sin

sin *number*
Stands for **sin**e. Reports the sine of *number* degrees. See **cos**.
Example:

```
show sin 90
1
```

size

size
Reports the size of the current turtle. See **setsize**.
Example:

```
show size
40
```

snaparea

snaparea *shape-number [x y] [xsize ysize]*
Copies the graphics in the defined area and pastes it in a turtle shape. *[x y]* is the top, left starting point, and *[xsize ysize]* determines the size of the rectangle that is copied into the shape. See **snapshape**.
Example:

```
pd rt 11 fd 5000
snaparea 1 [20 20] [60 60]
setsh 1
```

snapshape

snapshape
Copies the background behind the turtle into the current shape of the turtle. You cannot use **snapshape** if the turtle has its original turtle shape. The turtle must be completely visible on the page. **Snapshape** resets the turtle's size to 40 (its original size). If you want to copy the shape into an empty number in the Shapes Center, set the turtle to this number before using **snapshape**.
Example:
Place the turtle on the background that you want to copy.

```
setsh 16
```

snapshape
Now move the turtle.

snapshot

snapshot
Takes a snapshot of the background. The next time a **restore** command is used, the background will be restored to what it was at the moment the snapshot was taken. Note that there is only one snapshot per project, and it isn't saved with it.
Example:

```
pd
fd 50 rt 90
snapshot
clean
restore
```

sol

sol
Stands for **s**tart **o**f **l**ine. Brings the cursor (insertion point), in the current text box, to the beginning of the current logical line. Try using this command in a button so that you can see the effect on the cursor. See **eol**.
Example:

```
text1,
repeat 5 [print "hello]
top
repeat 5 [sol cd pr "there!]
```

space

space
Reports the amount of free Logo space in bytes. See **recycle**.
Example:

```
show space recycle show space
251036        Your results will differ.
433004
```

sqrt

sqrt *number*
Stands for **sq**uare **r**oo**t**. Reports the square root of its input.
Example:

```
show sqrt 9
3
```

st

st
Stands for **s**how **t**urtle. Shows the current turtle. See **ht**.
Example:

```
repeat 10 [ht wait 5 st wait 5]
```

stamp

stamp
Stamps a copy of the turtle on the background. The pen does not have to be down to stamp. You can also use the stamper tool in the Tool Palette to stamp the turtle's shape.
Example:

```
setsh "tree
pu
repeat 10 [stamp fd 40]
```

stamptext

stamptext *text-box-name*
Leaves a copy of the designated transparent text box on the background. This is equivalent to clicking with the stamper on a transparent text box. See **transparent**.
Example:

```
stamptext "text1
```

Drag the text box elsewhere.

stop

stop
Stops the procedure that is running. **Stop** can only be used in a procedure. See **stopall** and **stopme**.
Example:
The second line of this procedure is called a stop rule.

```
to countup :number
if :number > 100 [stop]
print :number
countup :number + 5
end
```

Try:

```
text1,
countup 0
```

Other examples of stop rules.

```
if colorunder = 15 [stop]
t1, if (distance "t2) > 100 [stop]
if slider1 < 1 [stop]
if empty? answer [stop]
if empty? text1 [stop]
```

stopall

stopall
Stops all running procedures and processes including turtles and buttons. **Stopall** can be used as a button, from the Command Center, or in a stop rule in a procedure. See **stop** and **stopme.**
Example:

```
forever [fd 1]
stopall
```

As a stop rule, **stopall** stops not only the procedure that contains it, but also all running procedures.

stopme

stopme
Stops the process in which this command was run. **Stopme** cannot be used to stop a turtle inside a color instruction. In this case, use **clickoff** instead. See **stop** and **stopall.**
Example:
The action will stop when turtle 1 will be more than 100 steps away from turtle 2.

```
t1, forever [fd 1 if (distance "t2) > 100
                    [stopme]]
```

sum

sum *number1 number2*

(**sum** *number1 number2 number3...*)
Reports the sum of its inputs. If more than two inputs are used, **sum** and its inputs must be enclosed in parentheses. See +.
Examples:

```
show sum 3 3
6
show (sum 3 3 3)
9
```

talkto

talkto (tto) *turtle-or-list-of-turtles*
talkto (tto) *text-box*
Makes the turtle(s) or text box current. This command has the same effect as typing the name of a turtle or text box followed by a comma. This is the only way of making many turtles do the same thing at the same time. See **ask,** **listen,** and *Local and Global Who* in *MicroWorlds Help Topics.*
Examples:

```
talkto "t1
fd 50
talkto [t1 t2]
bk 50          Makes both turtles go back.
talkto "text1
print "hello
```

tan

tan *number*
Stands for **tan**gent. Reports the tangent of its input. See **sin** and **cos**.
Example:

```
show tan 45
1
```

tc

tc
Stands for **t**ext **c**olor. Reports the number of the text color used in the current text box, at the insertion point. If text that has more than one color is selected, **tc** reports an empty list. See **settc.**
Example:

```
ct
settc "red
print "Eureka!
show tc
15
```

textcount

textcount *text-box-name*
Reports the number of lines in the current text box. Lines are delimited by carriage returns and line feed sequences (they are logical lines, not physical lines). Empty lines are taken into account. The number that **textcount** reports is the maximum number that can be used with **textitem**.
Example:
If you have this text box on the page:

```
show textcount "text1
3          Text1 contains 3 lines.
```

The following example takes a column of numbers in Text1 and prints the square of each number in Text2.

```
to square
text2,
do.one 1
end

to do.one :n
local "number
if :n > textcount "text1 [stop]
make "number textitem :n "text1
```

```
print :number * :number
do.one :n + 1
end
```

textitem

textitem *line-number text-box-name*
Reports the designated "line" of the named text box. Lines are delimited by carriage returns and line feed sequences (they are logical lines, not physical lines). Empty lines are taken into account. The first input must be between 1 and the number of lines in the text box. The line reported by **textitem** is a long word (a sequence of characters including spaces). Use **parse** to turn a long word into a list. See **textcount** and *Long Words* in *MicroWorlds Help Topics.*
Examples:
If you have this text box on the page:

```
show textitem 2 "text1
Kim
show textitem 4 "text1
Plato's cat
```

textlist

textlist
Reports a list containing the names of text files in the current directory. Only TEXT type files (created by **savetext** or other applications) are shown.
Example:

```
show textlist
mytext.txt listoffriends.rtf
```

A file name made up of more than one word will look like two or more files in the list. Use **item** to find the actual name. In this example, telephone numbers is the name of one file.

```
show textlist
mytext.txt telephone numbers.txt
show item 2 textlist
telephone numbers.txt
```

textpick

textpick *text-box-name*
Reports the text in a randomly-chosen line from the named text box. Lines are delimited by carriage returns and line feed sequences (they are logical lines, not physical lines). **Textpick** can pick empty lines. The line reported by **textpick** is a long word (a character string including spaces). Use **parse** to turn a long word into a list. See *Long Words* in *MicroWorlds Help Topics.*
Example:
If you have the following text box:

```
show textpick "text1
Plato's cat is very friendly.
```

Textpick reports one of the "lines."

textwho

textwho
Reports the name of the current text box.
Examples:

```
text1,
show textwho
```

```
text1
if textwho = "text1 [ct]
```

thing

thing *name*
Reports the value of the named variable. Corresponds to the use of a colon (:) preceding a word. See **make** and **name**.
Example:

```
make "age 10
show thing "age
10
show :age
10
```

timer

timer
Reports a number representing the time elapsed since the program started, or since the last **resett** command was run. The number is in tenths of a second.
Example:

```
resett
```

Wait a little.

```
show timer
22
```

The next procedure displays a question just after resetting the timer. If you got the right answer, it tells you how fast you were at typing it. The value of the timer is divided by ten in order to get the value in seconds.

```
to reflex
resett
question [What is 12 times 12?]
ifelse answer = 144
  [announce se timer / 10 "sec]
  [announce [Wrong answer]]
end
```

top

top
Moves the cursor (insertion point) to the beginning of the text in the current text box. Try using this command in a button so that you can see the effect on the cursor. See **bottom**.
Example:

```
pr "hello
top
pr "there
```

touching?

touching? *turtle-name turtle-name*
Reports **true** if the two turtles are touching each other. Reports **false** if they are not touching; always reports **false** if one of them is invisible.
Examples:

```
show touching? "t1 "t2
true
waituntil [touching? "t1 "t2]
when [touching? "t1 "t2] [do-this]
```

towards

towards *turtle-name*
Sets the heading of the current turtle to aim towards the one whose name is given as input. See **distance**.
Example:

```
t1,
towards "t2          T1 faces t2.
fd distance "t2      T1 meets t2.
```

transparent

transparent *text-box-name*
Makes the designated text box transparent. This is equivalent to checking Transparent in the text box's dialog box. See **opaque** and **stamptext**.
Example:

```
transparent "text1
opaque "text1
```

turtlesown

turtlesown *word*
Assigns a variable to all the turtles in the current project. This variable can then be set to a specific value for each turtle. This command also creates a new primitive made of the word **set** followed by the name of the variable (e.g., **turtlesown "speed** creates a **setspeed** command as in **t1, setspeed 12**).
There are two ways to get the value of a given turtle variable: you can talk to a turtle and use the variable name to report the value (e.g., **t1, show speed** displays **12** in this example) or you can use the turtle name followed by 's (e.g., **show t1's "speed** displays **12**).
Use **remove** to remove a turtle variable. This removes the named variable for all the turtles in the project. After a **turtlesown** instruction, the value of the variable is set to the empty list (see the first three lines in the example below).
Example:

```
turtlesown "speed
t1, show speed
      (empty list)
t1, setspeed 10
t2, setspeed 20
t3, setspeed 5
t1, show speed
10
show t2's "speed
20
everyone [fd speed]
everyone [forever [fd speed]]
```

Choose Stop All from the Edit menu.

```
remove "speed
```

unfreeze

unfreeze *word-or-list*
unfreeze *page-name*

Unfreezes the button, text box, turtle, slider, or any object on a page, so they can be changed using the mouse. A page name can also be used as input to unfreeze all the elements contained in that page. Use the eye tool to find out the name of the element. See **freeze.**
Examples:

```
unfreeze "button1
unfreeze "page1
```

unfreezebg

unfreezebg
Stands for **unfreeze b**ack**g**round. Unfreezes the background that was frozen by **freezebg.**
Example:

```
rt 11 fd 500
freezebg
lt 22 fd 500
cg
unfreezebg
cg
```

unselect

unselect
Undoes the highlighting effect of **select** or **search**.
Example:

```
text1, print "hello
top
select
cf cf
unselect
cf cf
```

wait

wait *number*
Causes a pause in the execution of a program or instruction. The time is measured in 10ths of a second.
Example:

```
repeat 5 [ht wait 10 st wait 10]
```

waituntil

waituntil *true-or-false-list-to-run*
Tells MicroWorlds to wait until *true-or-false-list-to-run* is **true** before running another instruction. The input must be an instruction list that reports either **true** or **false** when it is run. See **done?**.
Example:
In the following procedure, t1 makes a circle at the same time t2 makes a square. It takes longer to draw a circle, but MicroWorlds will wait for both shapes to be finished before telling the turtles to go elsewhere on the page to draw more circles and squares.

```
to sq-circ
t1, launch [repeat 36 [fd 10 rt 10]]
t2, launch [repeat 4 [fd 50 rt 90]]
waituntil [done? [repeat 36 [fd 10 rt 10]]]
t1, rt random 360 fd random 50
t2, rt random 360 fd random 50
sq-circ
end
```

when

when *true-or-false-instruction-list instruction-list*
Starts a parallel process that repeatedly tests whether the first instruction list reports **true** or **false**. If it reports **true**, the second instruction list is run. To stop a **when**, use **cancel** (only on the *true-or-false-instruction-list*), the Cancel menu item, or the Stop All menu item, or press **Ctrl+Break**.
Examples:

```
when [ycor > 50][bk 20]
repeat 1000 [fd 1]
```

Draw a red spot in front of the turtle

```
forever [fd 1]
when [colorunder = 15][cancel [fd 1]]
```

who

who
Reports the name of the current turtle.
Examples:

```
t1,
show who
t1
if who = "t1 [remove "t2]
```

word

word *word-or-list1 word-or-list2*
(**word** *word-or-list1 word-or-list2 word-or-list3...*)
Combines its inputs into one word and reports the word. **Word** can be used to make a list into a word. **Word** can take more than 2 inputs when **word** and the inputs are enclosed in parentheses.
Examples:

```
show word "hello "there
```

```
hellothere
show (word "hello char 32 "there)
hello there
```

word?

word? *word-or-list*
Reports **true** if the input is a word.
Examples:

```
show word? "hello
true
show word? 3.5
true
```

xcor

xcor

Stands for **x co**o**r**dinate. Reports the x coordinate of the current turtle.

Examples:

```
show xcor
50
setx xcor + 10
```

ycor

ycor
Stands for **y co**o**r**dinate. Reports the y coordinate of the current turtle.
Examples:

```
show ycor
50
sety ycor + 10
```