



# Handy Instruction Manual

## Table of Contents

9	About SmileBASIC
10	Using SmileBASIC
11	About BASIC
12	About the TOP MENU
13	Projects in the Cloud
14	Managing Projects / Files
15	Options
16	Starting BASIC
17	Using the Keyboard
18	What is DIRECT Mode?
19	Writing in EDIT Mode
20	Features in EDIT Mode
21	Managing Programs
22	About Sample Programs
23	Using the HELP Tool
24	Using the SMILE Tool
25	"PRINT" and Variables
26	Using Variables
27	Conditional Judgment
28	Computer Colors (RGB)
29	Graphic Instructions
30	Sound Instructions
32	3D Effects
33	Screen Layout
34	BG (Backgrounds)
35	Sprites

SmileBASIC is a tool that allows you to easily write programs on a Nintendo 3DS system. As it is also compatible with the system's 3D mode, you can create programs that utilize the 3D feature.

### User Agreement

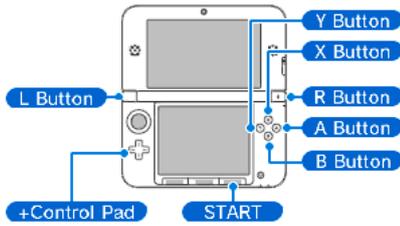
- Programs and resources such as images created using this product can be made open to the public using the Publish feature, allowing large numbers of people to view and run them. Please be sure not to publish any content that other people may find offensive, or content that reveals personally identifiable information or violates the rights of others (including portrait rights, privacy rights, and copyrights).
- Anyone who performs improper acts that may cause public nuisance, or who publishes obscene or libelous images, risks punishment in accordance with the applicable laws and regulations.
- SmileBoom Co.Ltd. assumes no responsibility for any issues resulting from information or programs published by its customers.
- Please note that if we receive a report from a customer that they are offended by a published project, we may delete said project unconditionally, without any prior procedure such as confirming with the relevant creator. We thank you for your understanding.

### Precautions regarding BASIC compatibility

Please also refer to the ["Standard BASIC Specification"](#) page.

- The BASIC language used in this product is not compatible with pre-existing versions of BASIC. Please also note that this product is not compatible with our Petit Computer and Petit Computer mkII products. Please make sure to pay attention to the differences in syntax when attempting to port programs from these products.
- This product uses double-precision real-type numbers or integers to represent values internally. This may cause errors in binary number calculations, and therefore the product should not be used for applications that require precise calculations.
- Branch instructions giving line numbers cannot be used. Instead of the line number, you should first assign a label (name tag) beginning with @ to the branch target line, and then specify that label.
- Programs that perform complex calculations or display a large amount of visual data on-screen can lead to slowdowns.
- If instructions that write to files, such as SAVE and DELETE, are used repeatedly, it may take longer to read or write these files.
- Although the program edit feature does not have a restriction on the length of a single line, input will no longer be possible once the memory reaches its limit.
- Array variables must be declared with the DIM instruction before they are used. Omitting the declaration will cause an error.
- For array variable parentheses, [] should be used instead of ().
- () should be used for specifying the priority of calculations, and for specifying the arguments of functions.
- The assignment instruction LET has been abandoned in this product. You should only use the = symbol in the format "Variable name=Expression."
- The exponentiation calculation symbols ^ and ↑ cannot be used. Please use the POW() function.
- As in other programming languages, conditional expressions in IF statements are represented using == for "equals" and != for "not equals." Please pay particular attention to the differences from conventional BASIC usage, where = is often used for "equals" and <> for "not equals." For assignment, = should be used.
- In FOR ... NEXT instructions, this product determines the conditional statement first. In cases such as FOR I=0 TO -1, if the conditional statement is not satisfied in STEP1, the program will skip the content of FOR and carry on executing the instructions after NEXT. Please be aware that, unlike in conventional BASIC usage, the instructions in a FOR ... NEXT loop are not guaranteed to be executed at least once.
- Variable names such as "NEXT I" cannot be specified in the NEXT instruction. Such specifications will not cause errors, but will work in the same way as when simply specifying "NEXT."
- Control variables in ON ... GOTO and ON ... GOSUB should be started from 0, not 1.
- The INT() function in conventional BASIC usage corresponds to the FLOOR() function in this product.
- When you require only the integer part in values gained from calculations involving divisions, you should use the FLOOR() function to obtain the integer part. Accumulated errors in calculations such as coordinate calculations will cause a subtle deviation.
- The RND() function returns integer values. If you require real-type values, the RNDF() function should be used.
- The graphics instructions in this product do not use () or -.  
Example for other products: LINE(0,0)-(639,399) Example for this product: GLINE 0,0,399,239

## 10 Using SmileBASIC



As this product is a development tool, the controls are used differently from regular games.

<b>+Control Pad</b>	Moves the cursor when in EDIT mode. - While pressing the L button, push the +Control Pad upward or downward to move to the next or previous page, and leftward or rightward to move to the beginning or end of the current line. (You can also move the cursor by pushing the Circle Pad upward/downward or leftward/rightward.)
<b>A Button</b> <b>B Button</b> <b>X Button</b> <b>Y Button</b>	The user can use these buttons when a program is running. When in EDIT mode, the A button functions as the ENTER key, and the Y button as the BS key.
<b>L Button</b> <b>R Button</b>	These buttons function in the same way as the SHIFT key on a keyboard. Use them when you want to input the purple characters above each key. The user can use these buttons when a program is running.
<b>START</b>	Runs a program. If used when a program is running, this button suspends the program.

Petit Computer 3 is a tool that allows you to write programs to make a computer do stuff.

### What is a Program?

A program is a set of instructions arranged sequentially that tells a computer what to do. The computer will execute the program exactly as it is told to.

Games are, of course, also a type of program.

### BASIC is a Programming Language

Just like humans use various different "languages," programs use different languages with different syntaxes.

This product uses a language called "BASIC," which is known for being easy to understand because it uses wording that resembles human language (English, to be precise).

For example...

[Print characters on the screen! - PRINT instruction](#)

[Make a beep! - BEEP instruction](#)

[Memorize numbers! - Variable and assignment instructions](#)

[Check which number is bigger or smaller! - Comparison and branch instructions](#)

[Repeat the operation! - Loop instructions](#)

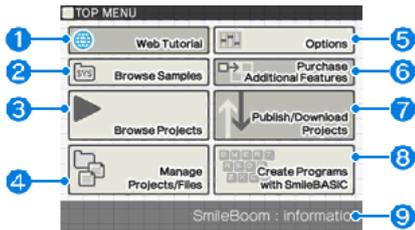
[Save the data! - SAVE instruction](#)

There are lots of different BASIC instructions. Once you master them, they can be used to do lots of different things, such as moving characters according to button presses on a controller, or generating sounds, in order to create fun games and tools.

Of course, if you make mistakes when writing out these instructions, you'll create errors known as **bugs**.

Many people now working in companies creating games or systems started off by learning how to program with BASIC. Our biggest hope is that Petit Computer will give people a taste of just how fun programming can be.

## 12 About the TOP MENU



The TOP MENU is the first screen that appears when you start SmileBASIC.

### ① Web Tutorial

This opens the official "Introduction to SmileBASIC" page in your browser. This page provides instructions for beginners who are new to SmileBASIC and BASIC programming. (An Internet connection is required.)



A dialog will appear to confirm that the Internet will be accessed.



When you want to return to SmileBASIC, press the HOME button to exit the browser.

The information on the page is updated regularly, so the content may vary from that shown here

### ② Browse Samples

This feature allows you to view the programs and games included as samples. Select the file you wish to view and press the Confirm button to start SmileBASIC and run the program.

- EX1-8** Programs of increasing complexity
- GAME1-7** Games in different genres
- SB????** Various tool programs

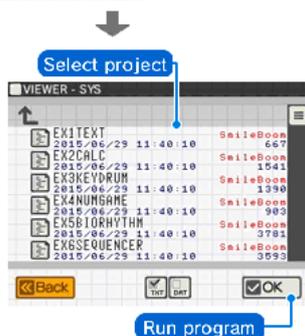
Please refer to the "About Sample Programs" page for details.

### ③ Browse Projects

This feature allows you to run saved projects. Select a project and press the Select File button. Next, select the project to run and press the Confirm button, which will start SmileBASIC and run the program.



The contents of the SYS folder are the same as those displayed when "Browse Samples" is selected.



If any of the following apply, the program currently running will be stopped, and you will be returned to the TOP MENU.

- If the program has run to the end
- If an error has occurred in the program
- If the user has pressed the START button

- If the END or STOP instructions have been executed

If the program stops with the message ===Press ENTER to Exit== displayed, please press the ENTER key on the Touch Screen. You will then be returned to the TOP MENU.

#### ④Manage Projects/Files

By default, files such as programs and other data are saved to a project folder called DEFAULT. Once you get to the point where you have lots of different projects, and too many files to manage, you can create new project folders to organize them into. Please refer to the "Manage Projects/Files" page for more details.

#### ⑤Options

From here you can configure various settings, such as assigning features to the SMILE button on the keyboard. Please refer to the "Options" page for more details.

#### ⑥Purchase Additional Features

New features and services to enhance SmileBASIC can be purchased from the Nintendo eShop. Press the button to display a product list. Then, select the product you wish to purchase and press the "Purchase" button, which will connect you to Nintendo eShop and start the purchase procedure. Please note that you must have a sufficient balance for the price of each product that you wish to purchase. (An Internet connection is required to access the Nintendo eShop.)

#### ⑦Publish/Download Projects

From here you can store programs you have created on the server, and publish them so they are available to other people. Please refer to the "Publish/Download Projects" page for more details.

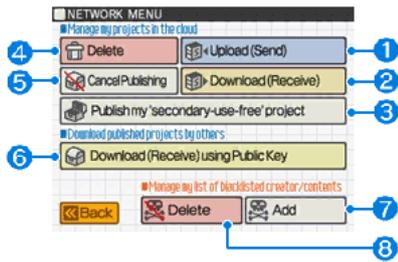
#### ⑧Create Programs with SmileBASIC

From here you can start SmileBASIC and begin creating your very own program. Please refer to "Create Programs" and the subsequent pages for more details.

#### ⑨Announcement Display Area

Important information regarding SmileBASIC, news of upcoming events, and other announcements from SmileBoom are displayed here. (If you do not wish to receive these announcements, please uncheck "Receive Announcements" in Options.)

## 13 Projects in the Cloud

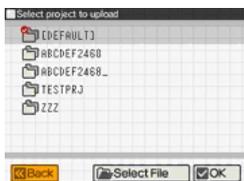


This menu is used when storing files on the SmileBoom server by connecting to the network, or when making completed projects open to the public.

An Internet connection is required in order to use this menu. In order to upload and download, you must also create and register a Nintendo Network ID. Please refer to the Operations Manual provided with Nintendo 3DS system for more details. The first time you try to use a feature that requires a Nintendo Network ID, an authentication dialog will appear.

### ① Upload (Send)

This allows you to download (receive) files which have been published on the SmileBoom server.



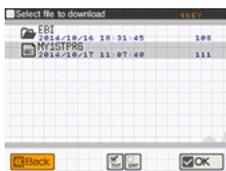
1. The SmileBoom server provides 10 storage locations for storing files, which are tied to your Nintendo Network ID. Each location is called a "Slot." One slot is used up whether you save all the files in a project or just a single file.
2. Use the +Control Pad to select a project to upload. To select a single file, press the Select File button and choose a file from the file list.
- ⚠ **You cannot upload the SYS folder or the DEFAULT folder.**
3. Press the Confirm button. A confirmation message will appear.
  - Yes - Upload will begin.
  - No - Upload will not begin.

Uploading large data will take a considerable amount of time. Please wait until the completion message appears.

Simply uploading files to the SmileBoom server does not mean that they have been published (made open to the public). They will be viewable to others once you have carried out the "Publish Project that Allows Secondary Use" process.

### ② Download (Receive)

This allows you to download the contents of slots containing projects (or files) that have been saved to the SmileBoom server.



1. Use the +Control Pad to select the slot to download files from, and then press the Confirm button. A confirmation message will appear.
  - Yes - Download will begin.
  - No - Download will not begin.

Downloading large data will take a considerable amount of time. Please wait until the completion message appears.

While the download is in progress, please do not turn off the power or remove the SD card or Game Card.

### ③ Publish Project that Allows Secondary Use

This feature allows you to publish projects you have uploaded to the SmileBoom server. Once a project is published, a "Public Key" will be generated, which you can use to give the project to others.

The publication process is as follows:

1. Accept the precautions concerning publication (Allow secondary use)
2. Select from your uploaded files
3. Confirm submission of the publication request
4. Review process (performed on the server side)

- If there are no issues, a Public Key will be issued



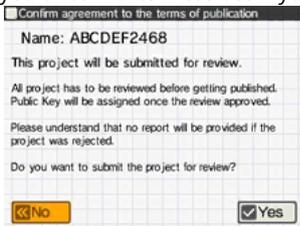
- A warning screen will appear showing the terms of publication. By pressing the Yes button, you are agreeing to abide by these terms.



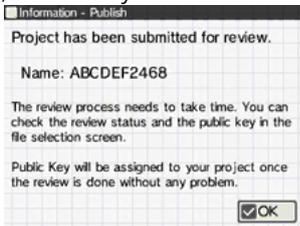
Once you publish a project, anyone in the world who owns this product and has been given the Public Key can view your work (program, data, etc.). Please be sure not to carelessly publish projects that contain libelous content or personally identifiable information such as addresses and telephone numbers. Such content, if distributed by a malicious user to the general public via the Internet, might ruin the life of the person involved.

Please also note that programs and resources (images and sounds) included in published projects are deemed to have been provided by their creators with permission for secondary use by third parties. Please be careful to avoid problems such as infringement of copyright. If a published project is the subject of a takedown notice, SmileBoom Co., Ltd. may delete the relevant project without prior notice.

- Use the +Control Pad to select the file you want to publish and press the A button to confirm. The precautions regarding publication will be displayed once more. Press Yes if you want to request publication.



- The review process will now begin. The file content will be analyzed on the SmileBoom server side. Once it is verified that there are no issues with it, a Public Key will be issued. The review process will require some time.



**⚠** In some cases, the result of the review process may be given as "Publication Declined." Publication may be declined because the project includes content that is contrary to public order, or because it includes files specified by a deletion request. However, we cannot answer any questions regarding the specific reasons for publication being declined. We apologize for any inconvenience, and thank you for your understanding. A project with the "Publication Declined" status cannot be given to others.

- The publication review process is now finished. Please wait for the result of the review from the server.

#### 4 Delete



- Use the +Control Pad to select the slot to delete files from, and then press the A button to confirm.
- Use the +Control Pad to select the files you want to delete, and press the A button to check them.
- Pressing the Confirm button will display a message confirming deletion.
  - Yes - Files will be deleted.
  - No - Files will not be deleted.

#### 5 Cancel Publication

This will cancel publication of a slot and make it inaccessible to others.



Use the +Control Pad to select a slot to depublish, and then press the A button to confirm your selection. A confirmation message will appear.

- Yes - Publication will be cancelled.
- No - Slot will remain open to the public.

## ⑥ Download Projects from the Network Using a Public Key

This allows you to upload saved projects (or files) to the SmileBoom server.



1. Enter the Public Key to access the server. If you enter an incorrect Public Key, you will not be able to download the file.
2. Select the project folder where the downloaded files will be stored.
3. Pressing the Confirm button will display a message asking if you wish to begin the download.
  - Yes - Download will begin.
  - No - Download will not begin.

Downloading large data will take a considerable amount of time. Please wait until the completion message appears.

While the download is in progress, please do not turn off the power or remove the SD card or Game Card.

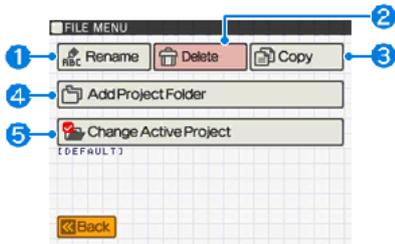
## ⑦ Add to the Blocked-User List

You can block communication from people who behave improperly or cause trouble by adding them to the blocked-user list.

A file list will appear. Please select a file from the creator you want to block to add them to the blocked-user list.

## ⑧ Clear the Blocked-User List

This will clear the blocked-user list. If you clear the list, you will be able to receive communication from the people you added to it once more. This option always clears all entries in the blocked-user list. It is not possible to select specific entries to delete.



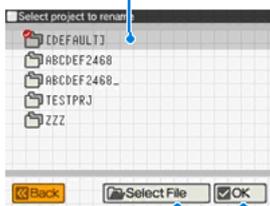
This menu allows you to create new "project folders" in which programs and data are saved, and to rename, copy, and delete files.

## 1 Rename

This is used to rename project folders and files.

1. Select the project folder or file you wish to rename.

Select file / folder to rename



※ Press this when you want to select a file within the folder. **Confirm selection**

2. Enter a new name using the simplified keyboard.
3. File/folder name will be changed.

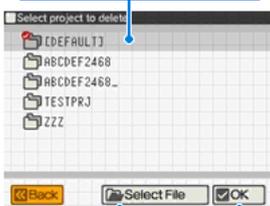
You can also change file names by using the RENAME instruction.

## 2 Delete

This is used to delete files that are no longer needed.

1. Select the Project folder containing the files you wish to delete.

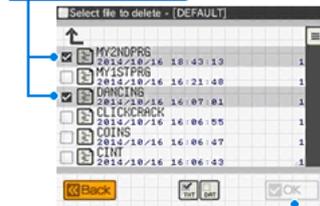
Select project folder



**Display file list** ※ Press this when you want to delete the project folder itself.

2. Select the files you want to delete.

Check files to delete



**Delete checked files**

3. A message confirming deletion of the files will appear.
  - Yes - File(s) will be deleted.
  - No - File(s) will not be deleted.

⚠ You cannot restore files once they have been deleted. Please be careful not to delete the wrong file by mistake.

You can also delete files by using the DELETE instruction.

## 3 Copy

This is used to copy files with new names.

#### ■ Copying a project

1. Select the project to copy
2. Input a new project name
3. Copy the project with the new name

#### ■ Copying one or more files

1. Select the project to copy from
2. Press the Select File button
3. Select the file(s) you wish to copy (You can select multiple files)
4. Select the project to copy to
5. Copy the selected file(s)

\*A confirmation screen will appear if any file names already exist.

### ④ Add Project Folder

This is used to create new project folders.

1. Enter the name of the new project folder using the simplified keyboard that appears.



2. The project folder will be created.  
Select whether or not you want this project to be the default project (whether or not you want to use it immediately).



1. - Yes - The new project folder will be set as default.
2. - No - Will not be set as default for the moment.

If you change the default project, the programs and files in the previous project folder will no longer be visible. You can change the default project by selecting "Change Active Project" below, or by using the PROJECT instruction.

### ⑤ Change Active Project

This is used to change the default project folder where files handled in BASIC are saved.

⚠ In the initial state, a project called DEFAULT is assigned.

- You will no longer be able to load the files in the previous project folder in BASIC, but they do still exist.  
If you reassign the previous project folder as the default project, your old files will be available again.

You can also change the default project by using the PROJECT instruction in BASIC.  
Entering PROJECT "" will restore the default project to its initial state.

### Concept of the Active Project

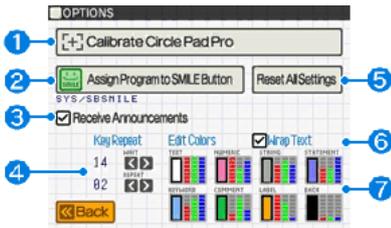
SmileBASIC assumes the following three conditions for the current project:

1. Current project at start-up time  
(specified using "Change Active Project" under "Manage Projects/Files")
2. Current project at non-execution time  
(set with the PROJECT instruction)
3. Current project at execution time  
(set during execution, e.g., of EXEC)

When one of the above is changed, the subordinate project settings will also be updated accordingly.

For example, if the current project at start-up time is changed, the current project at non-execution time and execution time will also be changed

When execution is started (by using RUN, executing a tool, or executing a program from the file viewer), the current project at execution time will be set as the initial value of the current project at non-execution time.



From here you can set the various options for the way SmileBASIC works.

### ① Calibrate Circle Pad Pro

From here you can calibrate the Circle Pad Pro. Attach the Circle Pad Pro, and then follow the on-screen instructions.

### ② Assign Program to SMILE Button

This lets you change the program that is assigned to the SMILE button on the keyboard.

#### ◆ Precaution

The OPTION TOOL instruction must be written at the top of a program in order to register it to the SMILE button.

### ③ Receive Announcements

Check this option if you want to receive announcements from SmileBoom, such as news of upcoming events. Announcements will appear at the bottom of the TOP MENU.

### ④ Key Repeat

When inputting characters in BASIC, if you hold down a key, it will be input repeatedly. From here you can set the repeat rate.

- WAIT - Time to wait after inputting the first character before inputting the second character
- REPEAT - Repeat interval after inputting the second character

### ⑤ Reset Settings

This restores the settings on the Options screen to their initial state.

### ⑥ Wrap Text

In the program EDIT mode, long lines of text are wrapped around. If you wish to turn this feature off and show them on single lines, uncheck this option. If turned off, horizontal scrolling will occur.

### ⑦ Edit Colors

In EDIT mode, different types of elements such as instructions and numerical values are displayed in different colors so that they can be recognized easily. From here you can change the display color for the different elements.

- TEXT (Characters)
- NUMERIC (Numbers)
- STRING (Character strings enclosed in "")
- STATEMENT (Control statements)
- KEYWORD (Instructions)
- COMMENT (Comments)
- LABEL (Labels beginning with @)
- BACK (Background)

When you select "Create Programs with SmileBASIC" from the TOP MENU, SmileBASIC will start. SmileBASIC has two operating modes, which are explained below.

## The Two Operating Modes

### ● DIRECT Mode



This mode is for inputting and executing instructions using the keyboard (Press the DIRECT button when in EDIT mode to switch to this mode)

This is the mode you will start in.



### ● EDIT Mode



This is a text editing mode for inputting programs (Press the EDIT button when in DIRECT mode to switch to this mode)



Programs are input in EDIT mode, and saved and run using instructions in DIRECT mode.

## Switching Modes

You can switch modes by using the system buttons under the keyboard.

### ◆ System Buttons



1	<b>EDIT Button</b> Switches from DIRECT mode to EDIT mode.
2	<b>DIRECT Button</b> Changes from EDIT mode to DIRECT mode.
3	<b>TOP MENU Button</b> Exits SmileBASIC and returns to the TOP MENU. The program will not be cleared.
4	<b>SMILE Button</b> Calls up the handy SMILE tool.

### ◆ About Program SLOTS



5	<b>Changing Program SLOT</b> This product provides four locations for storing programs, which are known as program SLOTS. Usually, SLOT0 is used. SLOT1-3 are supplementary, and do not necessarily need to be used.
---	---

- For information on other system buttons, see the "Editing Features Provided in EDIT Mode" page.

## DIRECT Mode

DIRECT mode is used for executing instructions on the fly. Even if the program is created in EDIT mode, it is run by executing the RUN instruction in DIRECT mode.

### ◆ Upper Screen - Console Screen

Instructions input via the keyboard and execution results are displayed here.



## 17 Using the Keyboard

Whether you're in DIRECT mode or EDIT mode, the keyboard displayed on the Touch Screen is always used to input characters.

### Switching Character Input Mode

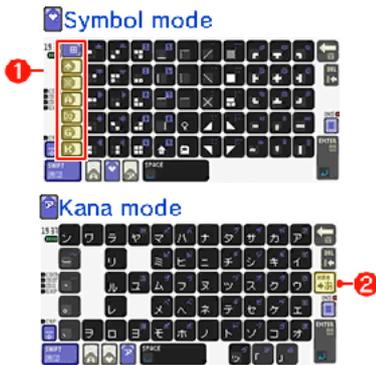
You can switch between alphanumeric characters, symbols, and Kana, as well as between upper and lower case.



1	<b>Character switching keys</b> Alphanumeric Symbols Kana
2	<b>SHIFT key</b> Switches between upper and lower case for the next single character that is input. Also switches some symbols.
3	<b>CAP key</b> Locks the SHIFT key after switching to upper or lower case so that the case is kept for subsequent inputted characters. Pressing this key again returns to normal input.

### About Inputting Symbols and Kana

If you use the character switching key to switch to symbol or Kana input, part of the keyboard will change, allowing you to input a greater variety of characters.



1	Switches between symbol categories. There are seven categories in total.
2	Switches between Hiragana and Katakana.

### Spaces and Line Breaks

If you want to insert a space between characters, press the SPACE key. If you want to begin a new line, press the ENTER key.



1	<b>SPACE key</b> Inputs a blank space equal to one character. In some BASIC instructions, you will have to input a space to delimit the elements. Be careful when inputting instructions. <pre>× BEEP5 ENTER ○ BEEP S ENTER   ↑   Press SPACE key to insert a space</pre>
2	<b>ENTER key</b> The ENTER key performs very important functions. <ul style="list-style-type: none"><li>● In DIRECT mode Sends instructions input on the screen to the computer for execution.</li></ul>

Input instructions will not be executed until you press the ENTER key.

● In EDIT mode

Inserts a line break and moves the cursor to the beginning of the next line.

In this electronic manual, the symbol  indicates that the ENTER key must be pressed at the end of an instruction.

Example:

```
PRINT "HELLO"
```

Press SPACE key Press ENTER key

## Commonly Confused Characters

### Zero and O

In BASIC, a strict distinction is made between the number zero and the letter o. If you input the wrong one, BASIC will not work as expected. Please be careful, as these keys are located close to each other on the keyboard.



### I, I (Uppercase i) and l (Lowercase L)

Although they look alike, these characters each have different meanings. Another example of confusing characters is a minus sign and a Kana macron (—). Please watch out that you don't get these confused.

### Quotation Marks and Separators

	<b>Double quotation mark</b> This is used frequently. Characters enclosed in double quotation marks are handled as a character string, not as a number. 135 -> numerical value 135 (one hundred and thirty-five) "135"-> character string (one three five)
	<b>Semicolon</b> This is used to, for example, separate an instruction from its parameters. PRINT "The amount is ";A ↑
	<b>Colon</b> This is used to, for example, list multiple instructions. BEEP 5:GOSUB @DM:PRINT "BYE" ↑ ↑  Please be careful, as in some cases a colon follows a semicolon. PRINT "Valid";:GOTO @TOP ↑ ↑
	<b>Comma</b> This is used to, for example, separate arguments or pieces of data. GLINE 0,0,399,239 ↑ ↑ ↑
	<b>Period</b> This is used to, for example, represent a decimal point. Please be careful, as in some cases a mix of periods and commas will occur. DATA 3.14, 1.08, 36.5 ↑ ↑ ↑

## Inserting and Overwriting Characters

There are two different modes used when inputting characters where there are existing characters: [insert mode](#) and [overwrite mode](#). Pressing the INS key switches between these two modes.



	<b>Insert mode</b> When you input a character, it will be inserted at the cursor position. The character strings to the right of the cursor will all move to the right.
	<b>Overwrite mode</b> When you input a character, the character at the cursor position will be replaced with the new character.

## Deleting Characters

Use the BS key  or DEL key to delete characters you have input.



### 1 BS key

Deletes the first character to the left of the cursor. The character strings to the right of the cursor will all move to the left.

### 2 DEL key

Deletes the first character to the right of the cursor (or in overwrite mode, the character the cursor is over). The character strings to the right of the cursor will all move to the left.

When in DIRECT mode, the UNDO key cannot be used to restore characters that have been overwritten or deleted using the BS or DEL keys. When in EDIT mode, the UNDO key can be used.

## Function Keys

At the top of the keyboard are five function keys which can be used to input frequently used instructions with a single touch.



You can change the functions of each of the keys by using the [KEY instruction](#).

### Example: Change function key 3 to FILES

In DIRECT mode, execute

```
KEY 3,"FILES"
```

## Program Read / Write Support Features

If you press the L button while the keyboard is being displayed, the support buttons for reading and writing program files will appear in the function key section. Press the LOAD button to open the file list dialog, from which you can select and LOAD files. Press the SAVE button to select and SAVE files. You can also check the name of the file that is currently loaded.

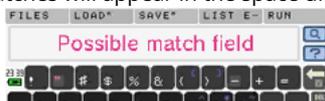


## Instruction Prediction Feature

This feature helps out when you're inputting instructions. When you input the first few letters of an instruction, a list of matches will be displayed.

### 1. Possible Match Field

Possible matches will appear in the space under the function keys.



### 2. Input the first letter

For example, if you press the G key, a list of instructions beginning with G will appear in the possible match field.



### 3. Narrow down the selection

Next, press the C key. The list of possible matches will be changed, narrowing it down to instructions beginning with GC.



If you touch a possible match, it will be input.

## 18 What is DIRECT Mode?

In DIRECT mode, instructions are passed to the computer directly.

### OK Mark - Waiting for an Instruction to be Input

"OK" is displayed on the console screen, with the cursor blinking on the next line.

```
OK
|
```

This is called the "instruction waiting" state. When you are about to enter an instruction, make sure that OK is being displayed.

### Inputting BASIC Instructions

Let's try out some simple instructions.

#### ◆Generate a sound (BEEP instruction)

Input the following from the cursor position. A funny sound will be generated.

```
BEEP 5 ENTER
```

- Insert a space after "BEEP" using the SPACE key.
- At the end, press the ENTER key.
- If you do not hear a sound, please check the volume control.

Next, trying inputting the instruction with a different number specified after "BEEP."

```
BEEP 7 ENTER
```

A different sound from before was generated.

Information specified together with an instruction like this is called an "argument." Depending on the instruction, the arguments will have different meanings, or there may be no arguments.

#### ◆Clear the screen (CLS instruction)

Input the following:

```
CLS ENTER
```

All the characters on the screen have disappeared, and now only "OK" (instruction waiting) is displayed. The CLS instruction is used to clear the screen.

#### ◆Change the color of characters (COLOR instruction)

The console screen uses white as the standard color, but this can be changed to a different color.

Input the following instruction:

```
COLOR 3 ENTER
```

- Please be careful not to input the number zero instead of the letter O in COLOR.

The execution result will be as follows:

```
COLOR 3
OK ← Execution result
|
```

The previously white "OK" is now displayed in red. The argument 3 is a numerical value that means "red."

If you input "COLOR 15" and press the ENTER key, the color will return to white.

```
COLOR 15
OK
|
```

The range of available color numbers is from 0 to 15. However, 0 is transparent, and 1 is black. Be careful when specifying these, because the characters will become invisible.

### About Errors

Computers interpret instructions exactly as they are entered. If you mistype just a single letter in an instruction, the computer will not execute it. Try inputting a fake instruction ("ABC") and press the ENTER key.

```
ABC
Undefined function ← An error
occurred
OK
|
```

A warning beep sounds, and the message "Undefined function" is displayed.

This is an error message meaning "This feature does not exist."

Please refer to the "Error Message Table" page for more information on error messages.

### What is a Program?

In DIRECT mode, instructions are executed one by one as they are input, which means you have to input instructions each time you want to

execute them.

However, to generate complex actions, such as those seen in games, it's necessary to input all the instructions together, in advance.

This is achieved by using a "program." A program is a set of multiple instructions arranged sequentially. On the next page, we'll try writing a program.

Let's try combining some simple instructions to write a program.  
Switch to EDIT mode so that you can begin inputting the program.



## Input a Simple Program

Now, try inputting a program that executes the following procedure. The program is made up of three lines in total.

1. Clear the screen - CLS instruction
2. Print "HELLO" - PRINT instruction
3. Generate a sound - BEEP instruction

### ◆ Input the first line and press the ENTER key

On the first line, input the CLS instruction, an instruction that clears the screen.



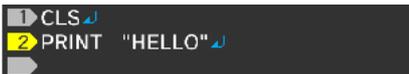
The yellow mark at the beginning means that this is the line currently being input. A line number is assigned automatically.

The  mark on the right side of the line is called a line feed mark, and indicates the end of a line.

Once you have finished inputting one line, press the ENTER key to begin a new one.

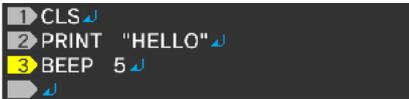
### ◆ Input the next line

On the second line, input an instruction to print "HELLO" on the screen. The instruction for printing characters is "PRINT."



The PRINT instruction should be accompanied by an argument to specify the character string to print. One of the rules in BASIC is that character strings must be enclosed in double quotation marks ("). Please make sure to input these.

Input the third line in the same way to complete the program.



## Running a Program (RUN Instruction)

Next, let's switch to DIRECT mode and run the program you have input.



The instruction for running programs is "RUN."



If the program is input correctly, the screen will be cleared, "HELLO" will be printed, and a sound will be generated.



If the expected result is not attained, go back to EDIT mode and check for errors.

### ◆ Press START to force the running program to stop

Programs begin running from the first line, and stop at the last line. However, some programs may run endlessly. To stop such a program, press START on the 3DS system. The program will be aborted.

## Erasing Programs (NEW Instruction)

To erase all the programs you have input in one go, execute the NEW instruction in DIRECT mode.

All programs will be erased.



**⚠ If NEW is executed, all program SLOTS will be erased. Please be very careful!**

### ◆ Erase only a specific program SLOT

The NEW instruction has an argument that can specify the program SLOT to erase. You can specify a numerical value between 0 to 3, which corresponds to SLOT0 to SLOT3.

- NEW 0 -> Erases only SLOT0
- NEW 1 -> Erases only SLOT1

NEW 2 -> Erases only SLOT2  
NEW 3 -> Erases only SLOT3  
NEW -> Erases all SLOTS

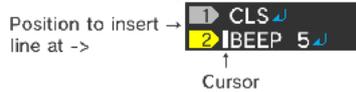
EDIT mode provides various editing features that will help you to write programs easily.

### Insert an Empty Line, Divide a Line and Combine Lines

The following methods allow you to insert empty lines in the middle of the program, divide lines, and combine multiple lines.

#### ◆ Insert an empty line

1. Move the cursor to the beginning of the line before which you wish to insert a line.



2. Press the ENTER key.



#### ◆ Divide one long line into two lines

1. Move the cursor to the position at which you wish to divide the line.

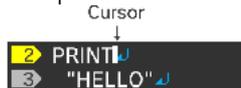


2. Press the ENTER key.



#### ◆ Combine two lines into one

1. Move the cursor so that it is positioned before the line feed mark on the first of the two lines.



2. Press the DEL key.



### Edit Multiple Lines/Undo

The system buttons in EDIT mode provide useful features for editing programs, such as the ability to copy and paste a specified range.



<b>1</b>	<p><b>SEL button - Select Range</b> Allows you to select a range in the program that you want to copy or delete.</p> <ol style="list-style-type: none"> <li>1. Move the cursor to the first character in the target range.</li> <li>2. Press the SEL button.</li> <li>3. Move the cursor to the last character in the target range. The selected range will now be shown with a white background.</li> </ol>
<b>2</b>	<p><b>COPY button - Copy Selection</b> Captures the selected range of lines internally. * The content on the screen will not change.</p>
<b>3</b>	<p><b>PASTE button - Paste Copied Contents</b> Pastes the lines captured internally using the COPY button at the cursor position.</p>
<b>4</b>	<p><b>L.INS button - Add a Line</b> Inserts a line after the line at which the cursor is positioned, and moves the cursor to the new line.</p>
<b>5</b>	<p><b>UNDO button - Undo</b> When you have accidentally deleted or pasted, or otherwise made a mistake, you can use this button to return to the state before the action.</p>

### Moving Within a Long Program

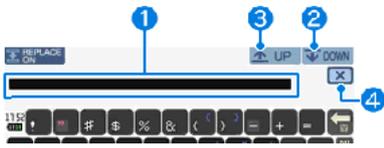
- Push the +Control Pad up or down while pressing the L button to move to the next or previous screen.
- Push the Circle Pad up or down while pressing the L button to jump to the first or last line in the program.

### Search/Replace Words Within the Program

#### ◆ Search feature

In EDIT mode, you can search for specified words within the program.

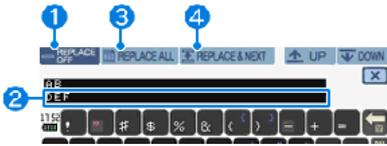
Press the Find button at the top right of the keyboard to switch the keyboard to search mode.



1	<b>Search keyword input field</b> Input the word you want to search for.
2	<b>Down button (Find Next)</b> Searches downwards from the cursor position and displays search results with a white background.
3	<b>Up button (Find Previous)</b> Searches upwards from the cursor position and displays search results with a white background.
4	<b>Exit Search button</b> Exits search mode.

◆ **Replace feature**

Press the Replace Mode button at the top left of the Search feature to switch to replace mode. This mode allows you to replace the search keyword with a different word.



1	<b>Search Mode button</b> Press the button to return to search mode.
2	<b>Replace keyword input field</b> Input the word you wish to replace the search keyword with.
3	<b>Replace All button</b> Replaces all search results.
4	<b>Replace &amp; Next button</b> Searches downwards from the cursor position and replaces the first search result. Press this button again to replace the next search result.

Programs you create can be saved to an SD card.

### SAVE Instruction - Save a Program

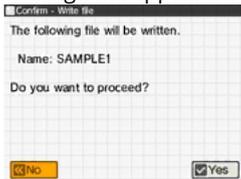
Use the "SAVE" instruction in DIRECT mode to save programs.

**Format** SAVE "File\_name"  
 - The only characters that can be used in file names are alphanumeric characters and \_ (underscore).

1. As an example, let's try saving a program as "TEST1".

```
SAVE "TEST1" 
```

2. A confirmation message will appear on the Touch Screen.



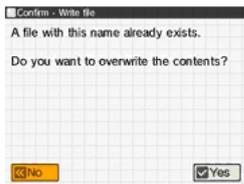
- Yes - Begins saving the file.
- No - Does not save the file.

Do not turn off the power or remove the SD card while files are being saved. Please wait until the operation finishes and a completion message appears.

3. Once saving has finished, a completion message will appear. Please press OK.

#### ◆ If a file with the same name already exists

If a file with the same name already exists in the SD card, a confirmation message will appear.



- Yes - Saves the file, overwriting the existing file.  
⚠ Overwriting will cause the contents of the previous file to be lost. Please be careful, as this operation cannot be undone.
- No - Does not save the file.

#### ◆ Application: Save another program SLOT

The SAVE instruction saves the contents of SLOT0. If you want to save the program contained in SLOT1, please input "PRG1:" before the file name to specify the SLOT number.

```
SAVE "PRG1:TEST1" 
```

In the same way, you can save the contents of SLOT2 or of SLOT3 by specifying "PRG2:" or "PRG3:" respectively.

#### ◆ Supplementary: Support feature for saving programs

When you want to SAVE the program currently being edited, you can also do so by pressing the SAVE button, which appears when you press the L button while the keyboard is being displayed on the Touch Screen, and then selecting the file to save.



### LOAD instruction - Load a Program

You can use the LOAD instruction to load and run programs you have saved.

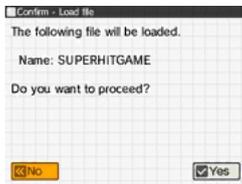
⚠ When you load a program, the program you are currently inputting in SLOT0 will be overwritten and lost.

**Format** LOAD "File\_name"  
 File name specified when the file was saved

1. As an example, let's try loading a file called "TEST1".

```
LOAD "TEST1" 
```

2. A confirmation message will appear on the Touch Screen.



- Yes - Begins loading the file.
- No - Does not load the file.

Do not turn off the power or remove the SD card while files are being loaded. Please wait until the operation finishes and a completion message appears.

3. Once loading has finished, a completion message will appear. Please press OK.

### ◆Application: Load into another program SLOT

The LOAD instruction loads a program into SLOT0. However, by inputting "PRG1:" before the file name, you can load the program into SLOT1.

```
LOAD "PRG1:TEST1" 
```

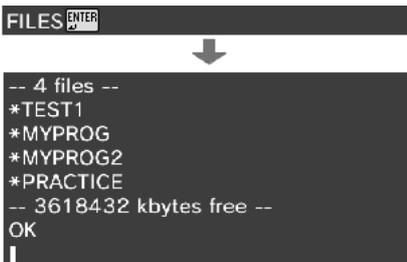
In the same way, inputting "PRG2:" or "PRG3:" will load the program into SLOT2 or SLOT3 respectively.

## FILES Instruction - Display a File List

You can use the FILES instruction to display a list of the files saved in the SD card.

**Format** FILES

- Display a list of files saved in the SD memory card on the console screen



## DELETE Instruction - Delete a File

You can use the DELETE instruction to delete files saved in the SD card.

⚠ You cannot restore files once they have been deleted. Please be very careful not to input wrong file name by mistake.

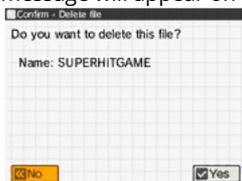
**Format** DELETE "File\_name"

- File name specified when the file was saved

1. As an example, let's delete a file called "TEST1".

```
DELETE "TEST1" 
```

2. A confirmation message will appear on the Touch Screen.



- Yes - Deletes the file.
- No - Does not delete the file.

Do not turn off the power or remove the SD card while files are being deleted. Please wait until the operation finishes and a completion message appears.

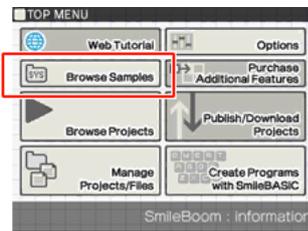
3. Once deletion has finished, a completion message will appear. Please press OK.

## 22 About Sample Programs

This product includes sample programs. Please refer to each sample program's description for details.

### How to Play the Sample Programs

To run a sample program, press the "Browse Samples" button from the TOP MENU, and then select and run the program from the file list.



### How to Use the Sample Programs

You can use the sample programs in the same way as programs you have created yourself. The sample programs are all stored within the SYS project.

1. Switch to DIRECT mode
2. LOAD "SYS/File\_name"
3. RUN
4. Press START to stop the running program
5. View the contents of the program in EDIT mode

Any of the sample programs can be saved under a different name using the SAVE instruction. They can be freely modified as you wish, without asking SmileBoom for approval.

### Basic Samples

These are sample programs using simple BASIC instructions that will help you learn to program. Use Edit mode to see what each of the programs do.

File Name and Short Description
EX1TEXT Printing characters to the console
EX2CALC Simple calculator using character input
EX3KEYDRUM Simple piano and drums using the keyboard
EX4NUMGAME Number guessing game
EX5BIORHYTHM Graphical biorhythm display
EX6SEQUENCER Simple touch-operated sequencer
EX7ALIEN Demo showing movement of multiple enemies and bullet shooting
EX8TECDEMO Demo explaining and evaluating each instruction

### Advanced Samples

These are samples that utilize various different BASIC instructions. The different tool samples can be used when preparing data needed for a game you are creating.

You can view the contents of these sample programs in EDIT mode, just like the basic samples, so once you understand how to program with BASIC, you can add your own features as required.

File Name and Short Description
GAME1DOTRC Racing game made up of pictographs, where the player clears all the on-screen dots while dodging enemy attacks
GAME2RPG Role-playing game where the player moves through a 3D-style dungeon while defeating enemies
GAME3JUMP

Side-scrolling platform game where the player heads for the goal while dodging enemies and obstacles

**GAME4SHOOTER**

Bullet hell ("danmaku" or "Barrage") shooting game where the player defeats enemies, which grow stronger with each stage, to earn score

**GAME5VS**

Simple versus fighting game where large characters fight with swords

**GAME6TALK**

Nonsensical fortune-telling game where the player answers gibberish questions to receive mumbo jumbo answers

※ This sample is not localized because TALK instruction only speaks Japanese

**GAME7EXPAD**

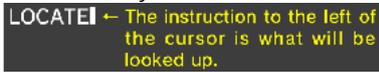
Can only be played with the Circle Pad Pro. Juggling game using the sticks

This product provides a HELP feature that allows you to instantly check the format and usage of each instruction. This feature can be used both in DIRECT mode and in EDIT mode.

### How to Use the HELP Feature

As an example, let's try displaying the help for the LOCATE instruction.

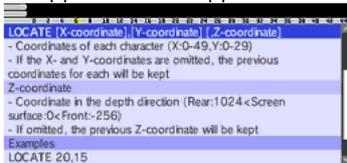
1. Input the instruction you want to check. DO NOT press the ENTER key.



2. Press the HELP button on the keyboard.

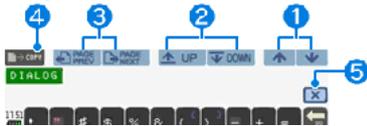


3. The help text appears on the upper screen.



When the help text is too long, only part of it will be displayed on the screen. Please press the buttons shown below to scroll up and down in order to view the whole of the text. Note that an instruction may have multiple help pages that show different uses for it, such as different arguments used with the instruction. If there is a notation such as (1/2 ->) in the header section at the top of the page, you can switch to the other page(s) by moving the stick to the left or right.

#### ◆ How to navigate the help text



<b>1</b>	Scrolls up or down one line in the help text. You can also scroll by pushing the Circle Pad up or down.
<b>2</b>	Scrolls up or down one section at a time. Comment -> Arguments -> Return Values -> Examples
<b>3</b>	Changes the current help page. You can also change page by pushing the Circle Pad left or right.
<b>4</b>	Copies the usage example(s). You can then press the PASTE button to paste and use the copied example(s) at the cursor location.
<b>5</b>	Exits the HELP feature.

#### ◆ Understanding the help text

##### Example: LOCATE instruction

1. Title: Format



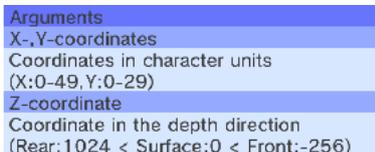
- Shows how to arrange the instruction and its argument(s).
- Arguments enclosed in [] are optional.

2. Comment



- Explains what the instruction does.

3. Arguments



- Explains how the argument(s) work.
- Some instructions may have multiple arguments.

4. Return Values

Return Values

None

- Shows the return values, if the instruction (function) has any.

#### 5. Examples

Examples

LOCATE 20,15

- Shows specific usage example(s).

- These can be copied using the COPY button.

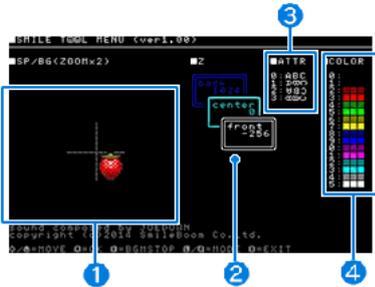
### Supplementary HELP Feature

When you press the HELP button without inputting a word to look up, an explanation on how to use the HELP feature and the instruction list will be displayed.

Pressing the SMILE button on the keyboard will start the SMILE tool. You can use it to check information that's required by your programs. You can also run advanced tools, such as a map editor.

### SMILE Tool Screen

#### ◆ Upper Screen



- |          |   |
|----------|---|
| <b>1</b> | Allows you to check the previously registered definition number of a sprite character and its image. You can push the Circle Pad up or down to check information on other characters. |
| <b>2</b> | Shows the Z-coordinate of the display element. The farthest forward is -256, while the farthest back is 1024.   |
| <b>3</b> | Shows the values and rotation directions that can be specified when rotating characters using the ATTR instruction.   |
| <b>4</b> | Shows the numbers for each of the colors/background colors that can be specified with the COLOR instruction.  |

#### ◆ Touch Screen



- |          |  |
|----------|--|
| <b>1</b> | <b>List display field</b><br>Displays the list for the entry selected out of ②-⑥.<br>Use the +Control Pad to select an entry from the list.  |
| <b>2</b> | <b>Switch to the sound effect list</b><br>Allows you to check the preset sounds used for BEEP. Press the A button to play back each entry.   |
| <b>3</b> | <b>Switch to the BGM list</b><br>Allows you to check the preset musical pieces for BGMPLAY.<br>Press the A button to play back each entry. Press the Y button to stop.   |
| <b>4</b> | <b>Switch to the MML instrument tone list</b><br>Allows you to check the instrument numbers (equivalent to GM tone generators) that can be used with the BGMPLAY instruction.<br>Press the A button to play back each entry. Press the Y button to stop. |
| <b>5</b> | <b>Switch to the sprite definition image list</b><br>Allows you to check the currently defined SPDEF definition numbers and contents.  |
| <b>6</b> | <b>Switch to the BG image list</b><br>Allows you to check the currently defined BG images.   |
| <b>7</b> | <b>Advanced editing tools</b><br>- PAINT (Character creation)<br>- MAP (BG screen creation)<br>- ANIM. (Animation creation)<br>- WAVE (Sampling and waveform editing)<br><br>Please refer to the next section for more details.                          |
| <b>8</b> | <b>Exit the SMILE tool</b><br>Exits the SMILE tool.<br>* You can also exit the tool by pressing the X button.  |
| <b>9</b> | <b>Calculator input</b><br>Allows you to move to a specified entry within the list. Input a number and press ENTER to change the currently selected entry to that list   |

number.

## Advanced Editing Tools

These are the advanced editing tools that can be run from the SMILE tool. Each tool goes into file mode when the Y button is pressed, and closes when the X button is pressed.

### PAINT (Character creation)



This tool is used for creating character data for BG images and sprites. Choose colors and select tools as you draw characters in the EDIT area. Press the Y button to go into file mode, then input L and press ENTER to load the character data. Input S and press ENTER to save the data. The saved data can be used from your programs.  
(Example) `LOAD"GRP5:MYBG"`

### MAP (BG screen creation)

This tool is used for arranging BG characters to create BG screen data, for example a cityscape. Select characters from the character chart to paste them into the EDIT area. The saved data can be used from your programs by loading it into an array.

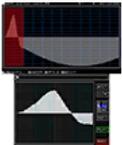


### ANIM. (Animation creation)



This tool is used for making adjustments to SPDEF definition contents and registering animation data for SPANIM. If you start an animation playback test, the program data in SLOT1 will be overwritten.

### WAVE (Sampling and waveform editing)



This tool is used for sampling from the microphone and using waveforms you have created as instrument sounds. A saved file can be used as an instrument sound for BEEP or BGMPPLAY in the WAVSET instruction by loading it as an array.

## 25 "PRINT" and Variables

The following will introduce the basic instructions required for writing programs. Please make sure to read it if you're new to programming.

### Print Characters - PRINT Instruction

"PRINT" is an important instruction that displays characters on the screen. Please try inputting the following program.

```
1 PRINT "HELLO" ↵
2 PRINT "A" ↵
```

Once you have input the program, run it in DIRECT mode.

```
RUN ↵
HELLO
A
OK
|
```

The PRINT instruction prints character strings enclosed in double quotations as-is.

#### ◆How semicolons (;) and commas (,) work

Input ; (semicolon) to the right of "HELLO" on the first line, and then run the program.

```
1 PRINT "HELLO"; ↵
2 PRINT "A" ↵
HELLOA
OK
|
```

"HELLO" and "A" have now been joined together.

Normally, the PRINT instruction causes a line break to occur automatically after printing the specified character string. However, if you add a semicolon (;), subsequent characters will follow directly after the printed string.

Next, change the semicolon (;) to a comma (,).

```
1 PRINT "HELLO", ↵
2 PRINT "A" ↵
HELLO  A
OK
|
```

If you use a comma (,), subsequent characters will be printed after a set space.

### The Difference between "A" and A - Understanding Variables

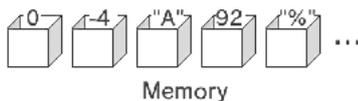
What will happen if you forget to insert double quotations when you were supposed to input PRINT "A"?

```
1 PRINT A ↵ ← Not enclosed in ""
RUN
0 ← Zero is displayed
OK
|
```

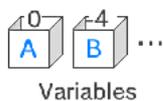
0 is displayed. This is because A without double quotations means "variable A."

#### ◆What is a variable?

Computers contain lots of memory, which can store numbers and characters in individual pieces.



In BASIC, pieces of memory that store values are managed by giving them names. A piece of memory that has been given a name is called a "variable."



"PRINT A" without double quotations means "print the contents of the variable A" instead of "print the character A".

In this case, the contents of the variable A happened to be 0, so 0 was printed.

```
PRINT "A" — Character A
PRINT A — Variable A
```

More details about variables are explained on the next page.

## Print Characters from a Chosen Location - LOCATE Instruction

You can use the LOCATE instruction to specify the location (coordinates) at which characters should be printed with the PRINT instruction.

**Format** LOCATE X-coordinate, Y-coordinate

- The X-coordinate specifies the number of characters to the right (0-49)
- The Y-coordinate specifies the number of characters down (0-29)

The following program prints HELLO at the position X=10, Y=3.

```
1 CLS ↵
2 LOCATE 10,3 ↵
3 PRINT "HELLO" ↵
```



### ◆ The depth of characters can also be specified

You can also specify the depth location (Z-coordinate) to display at. This can be used to achieve 3D effects when in 3D mode.

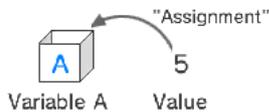
**Format** LOCATE X-coordinate, Y-coordinate, Z-coordinate

- The Z coordinate specifies the depth (positive values = into the screen, zero = on the 3D screen surface, negative values = in front of the screen)
- For depth into the screen, a value in the range 0 to 1024 should be specified, and for depth in front of the screen, a value in the range 0 to -256 should be specified

The following will explain how to use variables to store and calculate numerical values.

### Assigning a Value to a Variable with the = Sign

Putting something (a value) inside a variable is called "assignment."



In this figure, the value 5 is assigned to variable A.

In BASIC, this is written as follows:

**A=5**

This doesn't mean "A is equal to 5," but is an instruction meaning "assign 5 to A." If variable A does not exist yet, an empty piece of memory will be created and given the name A.

Let's try using the PRINT instruction to check if the 5 really was assigned to variable A.

```

1 A=5 ← Assigns 5 to variable A
2 PRINT A ← Prints the contents of variable A
↓ Execution result
5
OK
|
```

#### ◆ Character strings cannot be assigned to numerical variables

To be precise, the variable described here is called a "numerical variable." You cannot assign character strings to this type of variable. For example, inputting A="HELLO" will cause an error.

To assign a character string to a variable, you need to use a "string variable," which will be described later.

#### ◆ Formulas are also allowed

As well as single values, you can also write formulas to the right of the = sign. However, you need to use an asterisk (\*) as the multiplication sign, and a slash (/) as the division sign.

Let's prepare two variables, A and B, and assign to them the calculation results of "2+3" and "3÷2" respectively.

```

1 A=2+3
2 B=3/2
3 PRINT A, B
↓ Execution result
5 1.5
OK
|
```

#### ◆ Calculations between variables

You can also perform calculations between variables, as well as between numerical values.

```

1 A=2
2 B=3
3 C=A*B+1 ← Assigns the value of A x B + 1 to C
4 PRINT A, B, C ← Prints the values of A, B and C
```

In the third line, the content of A is multiplied by the content of B, 1 is added, and the result is assigned to C.



The execution result will be as follows:

```

2 3 7
OK
|
```

#### ◆ Find the area of a circle

Let's try writing a program to calculate the area of a circle with a radius of 2.

The area of a circle is radius x radius x pi. Here, let's use 3.14 as pi.

```

1 R=2
2 PI=3.14
3 S=R*R*PI
4 PRINT "The area is ";S
```

In the second line, the decimal number 3.14 is assigned to the variable PI. Variable names do not have to be a single character. You can use names with any length you wish, as long as they begin with a letter character and consist only of alphanumeric characters and underscores (\_).

In the third line, the area is calculated, and assigned to the variable S.

Pay attention to the PRINT instruction in the fourth line. This instruction prints "The area is " first, and then directly after that, as specified by the semicolon (;), prints the calculation result S.

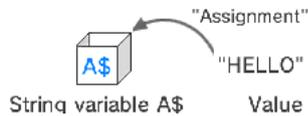
The execution result will be as follows:

```
The area is 12.56
OK
|
```

## Character Strings and String Variables

To assign a character string to a variable, you need to use what's called a "string variable."

String variable names should have a \$ sign at the end.



In BASIC, this is written as follows:

```
A$="HELLO"
```

You can use the PRINT instruction to print the value of a string variable in the same way as that of a numerical variable. The following program assigns "ALICE" to the string variable NAME\$, and then uses the PRINT instruction to print the string.

```
1 NAME$="ALICE" ↵
2 PRINT "HELLO ";NAME$ ↵
```

↓ Execution result

```
HELLO ALICE
OK
|
```

### ◆ Adding string variables

You can use addition to join string variables.

```
1 A$="HELLO " ↵
2 B$="WORLD" ↵
3 C$=A$+B$ ↵ ← Add A$ and B$
4 PRINT C$ ↵
```

↓ Execution result

```
HELLO WORLD
OK
|
```

Now let's have a look at programs that use characters input from the keyboard, and ones that automatically change how processing is handled depending on certain conditions.

### Input Characters - INPUT

In the program shown on the previous page for finding the area of a circle, the value for the radius variable is fixed internally. Unless you modify the program, it cannot be used to calculate the area of a circle with a different radius value.

```
1 R=2 ← The radius value is fixed at 2
2 PI=3.14
3 S=R*R*PI
4 PRINT "The area is ";S
```

So, it'll be more convenient if you make it so that any value can be input externally (from the keyboard) for R in the first line, instead of it being fixed internally. The instruction used to achieve this is "INPUT."

```
1 INPUT R ← Input R from the keyboard
2 PI=3.14
3 S=R*R*PI
4 PRINT "The area is ";S
```

When you run the program, you will see the following:

```
RUN ← The INPUT instruction is waiting for
?
|
```

The INPUT instruction displays a "?" mark on the screen, and waits for a value to be input from the keyboard. Once a value is input, it is assigned to a variable (in this case, R).

To find the area of a circle with a radius of 3, input 3 and press the ENTER key.

```
RUN
? 3 ← 3 has been input
The area is 28.26
OK
|
```

#### ◆ Display a guidance message for input

As soon as it's run, the program immediately displays "?", which will be confusing to people who aren't familiar with the program. To solve this problem, the INPUT instruction has a feature for displaying a guidance message.

The following program displays "What is the radius?" on the screen before waiting for the variable R to be input.

```
1 INPUT "What is the radius ";R
2 PI=3.14
3 S=R*R*PI
4 PRINT "The area is ";S
```

↓ Execution result

```
RUN
What is the radius? 3
The area is 28.26
OK
|
```

#### ◆ INPUT instruction summary

<b>Format</b>	INPUT "Guidance message"; Variable - Guidance message is optional. (If omitted, the semicolon can also be omitted.) - String variables are also allowed for the variable.
<b>Usage example</b>	INPUT "How old are you"; AGE This instruction waits for a value to be input from the keyboard, and then assigns the input value to the variable AGE.

### Jump to a Specified Location - GOTO and Label

The program shown above exits after printing the area of one circle.

In order to make multiple calculations, it will be more convenient if the program returns to the first line once it has reached the last line. This can be achieved by using the "GOTO" instruction.

<b>Format</b>	GOTO Jump target label name - The label name should begin with @ and consist of arbitrary alphanumeric characters (Example: @TOP).
<b>Usage example</b>	GOTO @TOP This instruction forces a jump to the line with the label "@TOP".

Programs run lines in order, starting with the first. However, by using GOTO, you can force a program to jump to a specified line. You must assign a name called a "label" in advance to the line to be jumped to. If you put @ (at mark) at the beginning of a line, the line will be

handled as a GOTO label.

```

1 @TOP ← Prepare a label
2 INPUT "What is the radius";R
3 PI=3.14
4 S=R*R*PI
5 PRINT "The area is ";S
6 GOTO @TOP ← Jump to @TOP
  
```

In this program, a jump target label (@TOP) is prepared in the first line.

This line only works as a sign, and performs no action itself. In the second to fifth lines, the radius is input, and the result is printed. Then, in the sixth line, the GOTO instruction forces a jump to the "@TOP" label, enabling the program to repeat the process from the top line.

↓ Execution result  
(program still running)

```

RUN
What is the radius? 3 ← First input
The area is 28.26
What is the radius? 2 ← Second input
The area is 12.56
What is the radius? | ← Waiting for third
input
  
```

◆ Press START to force the running program to stop

The GOTO instruction will cause this program to always return to the top, so in order to stop it, please press START.

Next, let's modify the program so that it will stop automatically depending on the given input. In order to achieve this, conditional judgment is used.

**Conditional Judgment - IF...THEN**

In BASIC, it is possible to check the value of a variable, and execute instructions only if the value meets a certain condition. The IF...THEN instruction is used for this purpose.

<b>Format</b>	IF Conditional expression THEN Instruction to execute - - The conditional expression should be a comparison, such as A==0 or A>4  == Equal to != Not equal to > Greater than < Smaller than >= Equal to or greater than <= Equal to or smaller than - After "THEN," specify the instruction to execute once the condition is met
<b>Usage example</b>	IF A\$=="YES" THEN PRINT "Bingo" If the value of the variable A\$ is "YES," this instruction displays "Bingo" and then proceeds to the next line. Otherwise, it does nothing, and simply proceeds to the next line.

Let's try modifying the previous program so that it will close when zero is input for "What is the radius?"

Insert the conditional judgment instruction after the radius is input.

```

1 @TOP
2 INPUT "What is the radius";R
3 IF R==0 THEN END ← Conditional judgment
4 PI=3.14
5 S=R*R*PI
6 PRINT "The area is ";S
7 GOTO @TOP
  
```

IF R==0 THEN END" means "if R is equal to 0, execute the END instruction" (close the program).

Please note that for comparisons, you must use "R==0" instead of "R=0." This is different from conventional BASIC usage.

↓ Execution result

```

RUN
What is the radius? 3 ← Input 3 for the first time
The area is 28.26
What is the radius? 0 ← Input 0 for the second time
OK ← Program closes
|
  
```

**Terminate the Program - END Instruction**

The "END instruction," which is used in the above program, is an instruction used to close a running program.

You may not need to use this instruction for programs without conditional branching, because they will terminate when they reach the last line.

<b>Format</b>	END
---------------	-----

Terminate the program.

## Supplement: IF...THEN...ELSE...ENDIF

- For users with programming knowledge

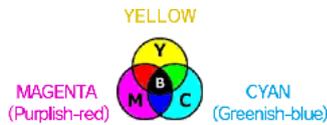
You can also use IF...THEN...ELSE in this product.

Also, by using ENDIF, you can write instructions spanning multiple lines that address both cases where the condition is satisfied and where it is not satisfied.

<b>Format</b>	IF "Conditional expression" THEN Instructions to be executed when the condition is met [ELSE Instructions to be executed when the condition is NOT met...] ENDIF
---------------	--

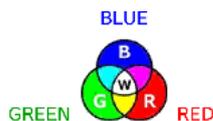
The colors used in programs (RGB) are different from those used for printed materials such as books (CMY). The following is a brief explanation of the colors used in computers.

### Subtractive Primary Colors (for printing)



Just like paint colors, the colors used in printing become murky when blended. In the above figure, the point where all three colors meet is B (black). Blending these colors completely will produce jet black. Ink cartridges used in printers also use these colors. (However, since printers cannot produce completely jet black by simply blending the colors, black ink is often supplied separately.)

### Additive Primary Colors (for TVs)



Video game screens use liquid crystal displays, which create color by emitting different colored lights. Unlike the colors used for printing, blending the three additive primary colors (RED (red component), GREEN (green component) and BLUE (blue component)) will make the light stronger, producing a color that is closer to WHITE. These colors are called RGB, an acronym for Red, Green and Blue. In programming, RGB is used when specifying colors.

### RGB Values of Commonly Used Colors

Color Chart	R	G	B	Color Name
■ ■ ■	0	0	0	Black
■ ■ ■	255	255	255	White
■ ■ ■	224	224	224	Light Gray
■ ■ ■	128	128	128	Gray
■ ■ ■	64	64	64	Dark Gray
■ ■ ■	255	0	0	Red
■ ■ ■	255	96	208	Pink
■ ■ ■	160	32	255	Purple
■ ■ ■	80	208	255	Light Blue
■ ■ ■	0	32	255	Blue
■ ■ ■	96	255	128	Yellow-Green
■ ■ ■	0	192	0	Green
■ ■ ■	255	224	32	Yellow
■ ■ ■	255	160	16	Orange
■ ■ ■	160	128	96	Brown
■ ■ ■	255	208	160	Pale Pink

For example, if you want to draw a red line on the screen, write the following:

```
GLINE 0,0,399,239,RGB( 255,0,0 )
```

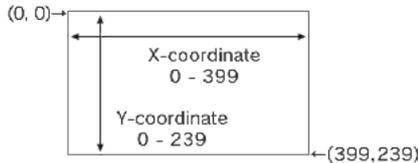
At the end of this line is the instruction RGB(), which takes the value of the red component R (0-255), the value of the green component G and the value of the blue component B, in this order. If you want to draw a green line, write RGB(0,255,0). For a blue line, write RGB(0,0,255).

The following will introduce the basic instructions used to display graphics.

### Graphic Screen and Coordinates

In this product, graphics are drawn on the graphic screen, which is a screen existing behind the console screen.

The graphic screen has a resolution of 400 horizontal dots x 240 vertical dots, which are handled as X-coordinates and Y-coordinates respectively.



### Try Using Graphic Instructions

First of all, let's input a simple graphic instruction in DIRECT mode to draw a straight line on the screen. Of course, you can also write this instruction as a program in EDIT mode.

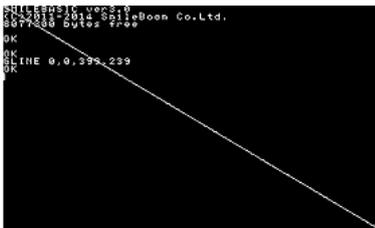
#### ◆ Draw a straight line (GLINE instruction)

Please input the following:

```
GLINE 0, 0, 399, 239 ENTER
```

The GLINE instruction is used to draw a straight line on the graphic screen.

The execution result will be as follows:



A line crossing the screen diagonally is displayed.

<b>Format</b>	<p>GLINE Start point X, Start point Y, End point X, End point Y, Color</p> <ul style="list-style-type: none"> <li>- The start point specifies the coordinates at which the straight line starts, and the end point specifies the coordinates at which it ends.</li> <li>- Color (color code) is optional.</li> </ul>
<b>Usage example</b>	<p>GLINE 0, 0, 399, 239</p> <p>Draws a straight line from the start point (0, 0) to the end point (399,239).</p>

#### ◆ Clear the graphic screen (GCLS instruction)

The CLS instruction cannot be used to clear contents drawn on the graphic screen. In order to do so, you should instead use the "GCLS" instruction.

<b>Format</b>	<p>GCLS</p> <p>Clear the graphic screen</p>
---------------	---

\* In this product, instructions beginning with G generally relate to the graphic screen.

```
GCLS ENTER
```

↓ Execution result



Only the line has disappeared; the characters have not.

The console screen and the graphic screen are managed separately. The CLS instruction only clears the console screen, while the GCLS instruction only clears the graphic screen.

If you want to read about the structure of the screen in more detail, please refer to the "Screen Structure" page.

### About Colors

Some graphic instructions, such as the GLINE instruction, allow you to specify the drawing color. Colors can be specified in multiple ways. If you have experience with graphics software, it should be easy to specify colors using RGB.

#### ◆How to specify colors using RGB

In this color specification method, each of the three additive primary colors (Red (R), Green (G) and Blue (B)) are specified by a value in the range 0-255.

For example, bright red is expressed as follows:

RGB(255, 0, 0)  
 ↑    ↑    ↑  
 Red Green Blue  
 value value value

The following color specifications are close to the eight basic colors used in early computers. If you use them as a base and modify each of the values, you can create more natural-looking colors.

Black	■	RGB(0,0,0)
Red	■	RGB(255,0,0)
Green	■	RGB(0,255,0)
Blue	■	RGB(0,0,255)
Yellow	■	RGB(255,255,0)
Light Blue	■	RGB(0,255,255)
Purple	■	RGB(255,0,255)
White	■	RGB(255,255,255)

#### ◆GCOLOR instruction

The GCOLOR instruction specifies the color of subsequent graphics that are drawn.

<b>Format</b>	GCOLOR Color code - Color code should be specified in RGB format.
<b>Usage example</b>	GCOLOR RGB(255,0,0) This specifies that subsequent graphics will drawn in a red color.

### Other Graphic Instructions

The following will introduce some of the instructions used to draw basic figures, such as quadrangles and circles. For more instructions, please refer to the pages on the sample programs and the instruction list.

#### ◆GPSET instruction - Plot a dot

<b>Format</b>	GPSET X-coordinate, Y-coordinate, Color code - X-coordinate, Y-coordinate - Coordinates to plot a dot at - Color code is optional.
<b>Usage example</b>	GPSET 199,119

#### ◆GBOX instruction - Draw a quadrangle

<b>Format</b>	GBOX Start point X, Start point Y, End point X, End point Y, Color - Start point X,Y - Coordinates for the top left of the quadrangle - End point X,Y - Coordinates for the bottom right of the quadrangle - Color code is optional.
<b>Usage example</b>	GBOX 0,0,100,80

#### ◆GFILL instruction - Draw a quadrangle and fill it with a color

<b>Format</b>	GFILL Start point X, Start point Y, End point X, End point Y, Color - Start point X,Y - Coordinates for the top left of the quadrangle - End point X,Y - Coordinates for the bottom right of the quadrangle - Color code is optional.
<b>Usage example</b>	GFILL 110,40,50,20

#### ◆GCIRCLE instruction - Draw a circle

<b>Format</b>	GCIRCLE Center X, Center Y, Radius, Color - Center X,Y - Coordinates for the center of the circle - Radius - Radius of the circle - Color code is optional.
<b>Usage example</b>	GCIRCLE 110,40,50,20

#### ◆GPAINT instruction - Fill the inside of a figure with a color

<b>Format</b>	GPAINT Start X, Start Y, Fill color, Border color - Start X,Y - Coordinates from which to start filling (The inside of the figure to be filled should be specified.) - Fill color code - Color with which to fill the area (Optional) - Border color code - Color to use for the border of the fill area (Optional)
---------------	--

**Usage  
example**

GPAINT 110, 40, RGB(0,255,255)

This fills an area with light blue, starting from the specified coordinates (110,40) and stopping when a color other than the base color is reached

## 30 Sound Instructions

The following will introduce the basic instructions used to play sound effects and BGM. There are also lots of other applied instructions. Please refer to the sample programs for information on how to use them.

### Play a Sound Effect - BEEP Instruction

The BEEP instruction we talked about on the page describing DIRECT mode is used to play short sound effects. Using this instruction, you can play a sound effect chosen from the list of various preset effects by specifying the effect number.

<b>Format</b>	BEEP Sound effect number - Sound effect number should be in the range 0-133. - Sound effect number can be omitted (If omitted = 0).
<b>Usage example</b>	BEEP 8

In applied usage, you can tune the frequency, volume, and pan-pot. Please input the instruction and check the description given in the HELP feature.

You can check the available sound effect numbers and their contents from the "BEEP" list, which is displayed when the SMILE button is pressed.

### Play BGM - BGMPLOY Instruction ①

The BGMPLOY instruction allows you to easily play BGM in your programs, for example in games. This product provides 43 ready-to-use preset BGM pieces.

<b>Format</b>	BGMPLOY Track number, BGM number - Track number: Specifies the target track when multiple tunes are played at the same time (0-7) Optional - BGM number: Specifies the number of the preset tune to play (0-42) * 128-255 are user defined tunes.
<b>Usage example</b>	BGMPLOY 12

Some preset tunes will end after being played once, while others will loop until the BGMSTOP instruction, described below, is executed.

You can check the available BGM numbers and their contents from the "BGM" list, which is displayed when the SMILE button is pressed.

### Stop BGM - BGMSTOP Instruction

This instruction is used to stop BGM that is currently playing.

<b>Format</b>	BGMSTOP - Immediately stops the music on all tracks
---------------	--

If you specify the target track, you can also make the music fade out before stopping.

<b>Format</b>	BGMSTOP Track number, Fade-out time - Track number: Track on which to stop the music (0-7) - Fade-out time: Number of seconds for which to gradually decrease the volume before stopping the music If 0 or no value is specified, the music will stop immediately.
<b>Usage example</b>	BGMSTOP 0,2

Input and run the following program:

```
1 BGMPLOY 12 ↵  
2 WAIT 60*5 ↵  
3 BGMSTOP 0, 3 ↵
```

1The first line starts playing BGM No. 12 (As the track number is omitted, the BGM is played on track 0).

The second line is an instruction called WAIT, which waits for a specified amount of time (in units of 1/60th of a second).

"WAIT 60" waits for one second. The WAIT instruction used here waits for 5 seconds.

The third line fades out the BGM playing on track 0 for 3 seconds before stopping it.

As a result, the BGM is played, then after 5 seconds it starts to fade out, and then after 3 seconds it stops completely.

### Play a musical scale - BGMPLOY Instruction ②

You can also play music using MML (Music Macro Language), which was used in old versions of BASIC.

Execute the following instruction:

```
BGMPLOY "CDEFG2AB<C" ENTER
```

This is a method for playing music where a character string enclosed in double quotations (") is interpreted as a score. Each alpha character represents a note.

#### ◆BGMPLOY instruction (Play MML)

<b>Format</b>	BGMPLAY "MML" - MML: Character string to play, which is a simplified representation of a score (See below)
<b>Usage example</b>	BGMPLAY ":0CCC :1REE :2RRG" Uses three channels to play a chord

### Main MML Elements

A thru G - Note (Play a sound)

C(Do) D(Re) E(Mi)

F(Fa) G(Sol) A(La) B(Si)

\* Examples of musical scale notes

# or + - Halftone up

C# D# E# F# G# A# G#

C+ D+ E+ F+ G+ A+ G+

- (minus) - Halftone down

C- D- E- F- G- A- B-

R - Rest

Number after a note - Tone length

C1 C2 C4 C8 C16 C32

From left, Whole note, Half note, Quarter note, Eighth note, Sixteenth note, and Thirty-second note respectively

. (period) after tone length - Dotted note

C1. C2. C4. C8. C16. C32.

Increases each tone length by half

Lx - Default tone length

(where x is a number of a tone length)

Ox - Octave of a tone

(where x is a range from 0 thru 8)

< One octave up

> One octave down

:x - Channel

(where x is a range from 0 thru 15)

Tx - Tempo

(where x is a range from 1 thru 240)

Vx - Change in volume

(where x is a range from 0 thru 127)

@x - Change in tone

(where x is a range from 1 thru 127)

By inputting MML and characters and then pressing the HELP button, you can view detailed explanations for MML.

Petit Computer allows you to set depth values for graphics, characters, sprites, and so forth, making them appear to "pop out" (or "sink in") through the effect of parallax between the left and right eyes. While 3D effects can help heighten the expressiveness of your works, they can also cause eye strain if overused. Please follow the precautions given on this page and be careful to avoid using extreme parallax in your programs. If you ever find the parallax setting too intense or experience symptoms including eye strain when viewing someone else's work, please take a break to rest your eyes.

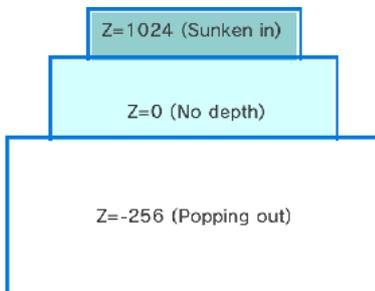
### **⚠ WARNING - 3D FEATURE ONLY FOR CHILDREN 7 AND OVER**

Viewing of 3D images by children 6 and under may cause vision damage.

Use the Parental Control feature to restrict the display of 3D images for children 6 and under. See the Parental Controls section in the operations manual of your Nintendo 3DS system for more information.

### **About 3D Effects**

3D effects are only supported on the upper screen. Various different display objects can be given perspective by setting the depth coordinate (Z-coordinate) for them. (Use LOCATE for the console screen, GPRIO for the graphic screen, SPOFS for sprites, and BGOFS for BG to specify the Z-coordinate)



### **Precautions Regarding 3D Effects**

When implementing 3D effects in your work, please pay attention to the following points and adjust your work accordingly in order to prevent eye strain.

- 1) Do not use white too much for areas which will serve as the background
- 2) Do not construct depth with only two levels: the background and the foreground.

White is a color that makes it difficult to perceive depth. When only using a two-level depth setting (rear and front) it is difficult for the eyes to perceive depth because there are few objects to use for comparison.

- 3) Decrease the parallax of images that "pop out" from the screen

Since images that pop out from the screen surface (those with a Z setting value of less than 0) tend to cause eye strain, please adjust the setting so that they are within the screen. If you set 3D effects near the edge of the screen, the parallax may not appear on the screen, which makes it difficult for the eyes to perceive depth.

If you find that any work created by others uses very intensive 3D effects, please lower the 3D setting when using it. Furthermore, if you experience eye strain during the creation process, please take a break to rest your eyes.

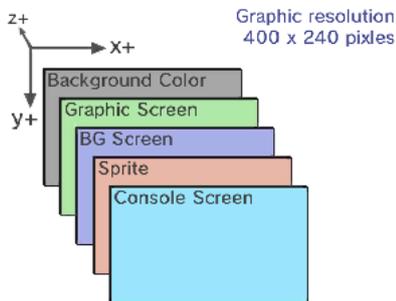
Please also refer to "Health and Safety Information" from the HOME Menu.

## 33 Screen Layout

In Petit Computer, multiple screens with different roles are displayed superimposed over each other, such as the console screen, which displays characters (text), and the graphic screen, which displays graphics.

### Upper Screen (3D Screen)

This upper screen consists of five screens. In order from innermost to foremost, these are: the background screen, graphic screen, BG screen, sprite screen, and console screen.

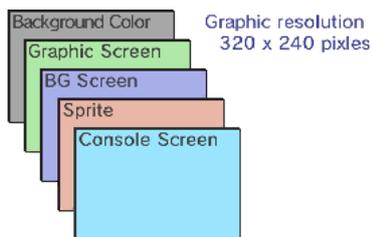


<b>Background Color</b>	A single-color screen that is displayed behind all the other screens.
<b>Graphic Screen</b>	The screen that displays graphics drawn with graphic instructions.
<b>BG Screen</b>	The screen used to create game maps and so forth by filling different areas with different tiles. <ul style="list-style-type: none"><li>- Up to 128x127 tiles of 16x16 dots each can be arranged and displayed on the screen</li><li>- There are four layers, allowing for multi-scrolling and other effects</li></ul>
<b>Sprite</b>	Element used for foreground characters, such as the hero of a game, that are displayed in front of the BG screen. <ul style="list-style-type: none"><li>- Up to 512 sprites can be used in total for the upper and touch screens</li><li>- The basic size is 16x16 dots, which can be changed individually to any required size</li><li>- An instruction to display multiple sprites continuously, producing animation, is available.</li></ul>
<b>Console Screen</b>	Text screen where characters can be written using instructions such as PRINT.

The 3D screen has depth, the amount of which is expressed by the Z-coordinate. The reference surface is  $Z=0$ , with Z going negative in front of the screen. The range of depth is 1024 to -256. However, the screen has fewer visible graduations than this, and the results will also change according to adjustment of the 3D depth slider.

### Touch Screen

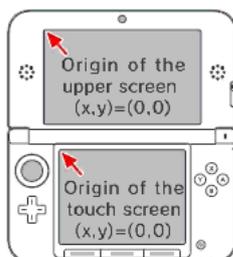
The structure is the same as the upper screen, except the screen size is different. The keyboard is only displayed on the touch screen.



As with the upper screen, the order of superimposed display elements is managed with the Z-coordinate.

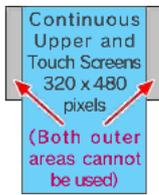
### Positional Relationship between the Upper and Touch Screens

The upper and touch screens are arranged vertically so that the horizontal center positions of the two screens are aligned. The coordinate origin (0,0) for both screens is at the top left.



### Continuous Display of the Upper and Touch Screens

You can use the upper and touch screens as one continuous screen by combining them using the `XSCREEN 4` instruction.

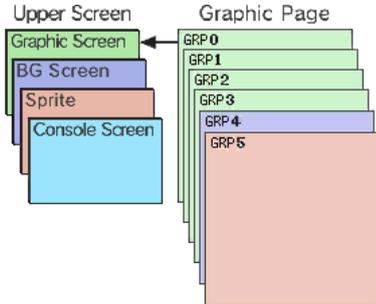


## Graphic Page

SmileBASIC provides a total of six locations for storing source images to be displayed on the screen. These are called [graphic pages](#).

Each page has a name from GRP0 to GRP5 and corresponds to a certain display screen.

For example, the graphic page called "GRP0" corresponds to the graphic screen on the upper screen.



If you use a graphic instruction to draw a graphic or place graphic data in GRP0, it will be displayed on the upper screen's graphic screen.

The following indicates which graphic page corresponds to which screen.

### ●Upper Screen (DISPLAY 0)

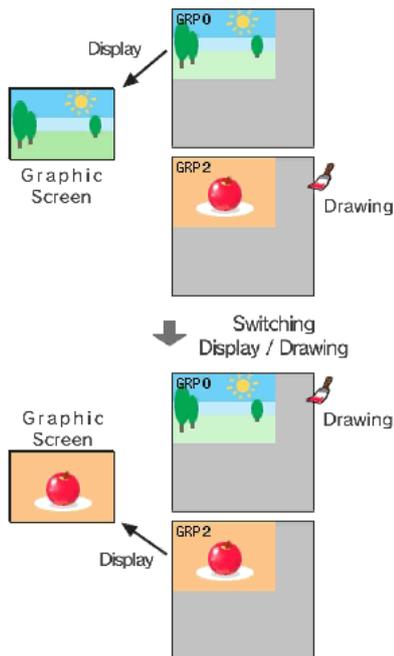
Graphic Screen	GRP0
BG Screen	GRP5
Sprite	GRP4

### ●Touch Screen (DISPLAY 1)

Graphic Drawing	GRP1
BG Screen	GRP5
Sprite	GRP4

## Display Page and Drawing Page

Drawing complicated graphics may take a long time, and the graphic may appear on-screen in an unfinished state. In order to prevent this, you can use a second graphic page to display while drawing is being processed on the first page.



In the initial state, the same graphic page is used for the drawing page and the display page: GRP0 (or GRP1 for the touch screen). However, it is possible to specify a different page using the [GPAGE instruction](#).

## Color Specification

SmileBASIC allows you to use 65536 colors for the whole screen. The way colors are specified differs between the graphic and the console screens.

Display Element	# of Colors / Color Specifications
Graphic Page	- 32768 colors per pixel - Use the RGB function to specify colors
Console Screen (Text)	Select from 16 colors (including transparent)

### ● Specifying colors on a graphic page

Colors are represented internally as 5 bits for each RGB color + 1 bit for transparency (RGBA=5551). However, when specifying colors, the RGB function should be used, and an 8-bit value for each RGB component specified.

### ◆ Examples of drawing color specification using the RGB function

GCOLOR RGB( R, G, B )

- Specifies color tone values in the range 0-255 for R (Red), G (Green) and B (Blue)

GCOLOR RGB( A, R, G, B )

- For A (transparency), the value 255 specifies "opaque," and any other value specifies "transparent"

### ● Specifying colors on the console screen (text)

You can set the character and background color for each character. Select a color for each one, out of 16 colors. The color and number mapping is displayed on the upper screen of the SMILE tool.

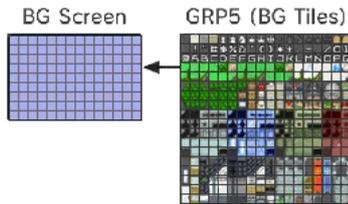
## 34 BG (Backgrounds)

Petit Computer 3's SmileBASIC provides structures called BG and sprites that make it easy to create game screens with movement.

### About BG

BG is an abbreviation for background, and refers to a common structure used for creating background scenery in console games. Tiles of 16x16 dots in size are arranged together in order to create one larger image (the background). By arranging identical tiles together, you can easily create areas of uniform graphics, such as oceans or fields.

In this product, various BG tiles commonly used in games are predefined. These can be used immediately by simply placing them on the BG screen using the BGPOT instruction.



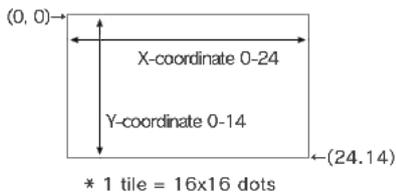
#### ◆BG Tiles

The BG tiles are preloaded in the GRP5 graphic page. Each tile is 16x16 dots in size and has a tile number in the range 0-1023. You can check which number corresponds to which tile from the touch screen by using the SMILE tool.

#### ◆BG Screen/Layers

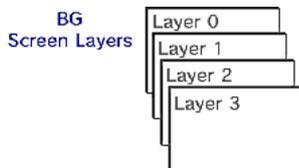
BG tiles are displayed on-screen only after they are placed on the BG screen.

The same BG tile can be placed repeatedly on the screen. The BG screen has coordinates in tile units. One screen can contain 25 tiles horizontally and 15 tiles vertically. Please note that by using the BGSCREEN instruction, you can also use a BG screen that is larger than the display screen.



The BG screen consists of 4 superimposed layers, and BG tiles can be placed on each one. These layers can be used to, for example, distinguish between the background and the foreground, or to enable multi-scrolling.

When placing BG tiles, a layer number (0-3) must be specified.



You can use the BGOFS instruction to change the display location of each layer in dot units. This makes scrolling possible.

You can also change the order in which the layers are superimposed by specifying the Z-coordinate in the BGOFS instruction. This also affects depth in 3D display.

### Place a BG Tile - BGPOT Instruction

This instruction is used to place a BG tile on the specified BG screen layer.

<b>Format</b>	BGPOT Layer, X, Y, Tile number - Layer: Number of the layer on which to place the tile (0-3) - X, Y: Coordinates at which to place the tile - Tile number: Number of the tile to place (0-1023)
<b>Usage example</b>	BGPOT 0,12,7,1 This places BG tile 1 near the center of the screen on layer 0

### Fill an Area with Repeated BG Tiles - BGFILL Instruction

This instruction is used to fill a specified rectangular area with repeated instances of one BG tile.

<b>Format</b>	BGFILL Layer, Start point X, Start point Y, End point X, End point Y, Tile number - Layer: Number of the layer that includes the area to fill with the tile (0-3) - Start point X, Y: Coordinates for the top left of the target area - End point X, Y: Coordinates for the bottom right of the target area - Tile number: Number of the tile to fill the area with (0-1023)
<b>Usage example</b>	BGFILL 0, 1, 1, 23,13, 2

This will fill an area of one tile around the inside of the circumference of layer 0 with BG tile 2

## Change the Display Location of the BG Screen - BGOFS Instruction

This instruction is used to change the display location and depth of a specified layer.

<b>Format</b>	BGOFS Layer, X, Y, Z - Layer: Number of the layer to move (0-3) - X, Y: Amount (in dots) by which to move the display location Positive values move the layer left or upwards - Z: Coordinates in the depth direction (Rear: 1024 - Screen surface: 0 - Front: -256) Optional
<b>Usage example</b>	BGOFS 0, -3, -4 This moves the display location of layer 0 by 3 dots to the right and 4 dots downwards

## Clear the BG Screen - BGCLR Instruction

This instruction is used to clear a specified BG screen layer.

<b>Format</b>	BGCLR Layer - Layer: Number of the layer to clear (0-3)
<b>Usage example</b>	BGCLR 0 This clears the display of layer 0

## Other BG Instructions

There are lots of other BG instructions. Please refer to HELP and the sample programs for information on how to use them. The following is a list of the main instructions.

### ◆BGSCREEN instruction

Changes the maximum size of the BG screen

### ◆BGSCALE instruction

Scales the BG screen up/down

### ◆BGROT instruction

Rotates the BG screen

### ◆BGCOPY instruction

Copies a specified range from the BG screen to another location

### ◆BGANIM instruction

Performs animation using the BG

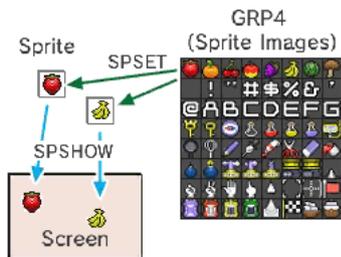
While the BG provides the still backdrop for games, sprites allow the creation of moving objects such as player and enemy characters, bullets, and so forth.

**About Sprites**

Like the BG, sprites are also a commonly used structure in console games.

Sprite images are displayed on-screen as single moving objects. The standard size for an image is 16x16 dots, but this can be changed to any size. Up to 512 sprites can be displayed on-screen. However, displaying too many at one time will cause slowdown.

In this product, various different sprite images that are commonly used in games are predefined. Using the SPSET instruction, these can be assigned to a sprite created with an arbitrary management number. They can then be displayed on screen by executing the SPSHOW instruction.

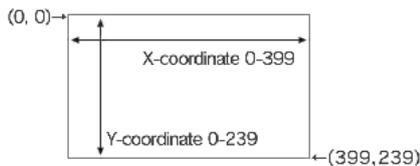


◆ **Sprite characters**

The sprite images are preloaded in the GRP4 graphic page. Although the basic size of an image is 16x16 dots, you can also use other sizes. To check image definition numbers, please press "SPDEF" from the SMILE tool.

◆ **Displaying sprites on-screen**

Using the SPSET instruction, you can prepare up to 512 sprites to display. Specify the image definition number to assign to each sprite number. Use the SPOFS instruction to specify the display coordinates of a sprite in dot units. This is different from BG coordinates, which are specified in tile units.



You can also change the order in which sprites are superimposed by specifying the Z-coordinate. This coordinate is treated in the same way as the Z-coordinate in the BG, so it is also possible to display sprites in front of, or hidden behind, the BG. The Z-coordinate also affects depth in 3D views.

**Prepare a Sprite - SPSET Instruction**

Use this instruction to prepare a sprite to display on the screen. Assign a management number in order to identify this sprite from the others, and select an image to use for the sprite.

<b>Format</b>	SPSET Management number, Image definition number - Management number: Arbitrary number to assign to the new sprite (0-511) - Image definition number: Number of the image to use for this sprite (0-1023)
<b>Usage example</b>	SPSET 0,1 This prepares sprite No. 0 and sets image 1 (an orange) as its appearance

The SPSET instruction prepares the target sprite in memory and displays it on the screen. (If you do not wish to display it, use the SPHIDE instruction.)

**Change the Display Location of a Sprite - SPOFS Instruction**

This instruction is used to change the display location and depth of a specified sprite. This is often used to move game characters.

<b>Format</b>	SPOFS Management number, X, Y, Z - Management number: Management number of the target sprite (0-511) - X, Y: Display coordinates on the screen - Z: Coordinates in the depth direction (Rear: 1024 - Screen surface: 0 - Front: -256) Optional
<b>Usage example</b>	SPOFS 0, 192, 112 This displays sprite No. 0 near the center of the screen.

**Scale a Sprite Up/Down - SPSCALE Instruction**

This instruction is used to scale up or scale down a specified sprite.

<b>Format</b>	SPSCALE Management number, Magnification X, Magnification Y - Management number: Management number of the target sprite (0-511) - Magnification X, Magnification Y: Horizontal and vertical magnification of the original size 0.5=50%, 1.0=100%, 2.0=200%
<b>Usage example</b>	SPSCALE 0, 1.5, 1.5 This increases the size of sprite No. 0 by 1.5 times vertically and horizontally.

## Rotate a Sprite - SPROT Instruction

This instruction is used to rotate a specified sprite.

<b>Format</b>	SPROT Management number, Angle - Management number: Management number of the target sprite (0-511) - Angle: Clockwise rotation angle (0-360) Angles beyond the upper limit or negative angles (counterclockwise rotation) can also be set.
<b>Usage example</b>	SPROT 0, 30 This rotates sprite No. 0 by 30 degrees clockwise

## Change the base point of a sprite - SPHOME instruction

This instruction is used to change the base point of a sprite when moving it, scaling it up or down, or rotating it. In the initial state, the base point of a sprite is its top left corner.

<b>Format</b>	SPHOME Management number, Base point X, Base point Y - Management number: Management number of the target sprite (0-511) - Base point X, Y: Coordinates relative to the top left corner of the sprite If the size of the sprite is 16x16 dots, specifying X=8, Y=8 will set the base point at around the center of the sprite.
<b>Usage example</b>	SPHOME 0, 8, 16 This moves the base point of sprite No. 0 by 8 dots to the right and 16 dots downwards from its top left corner.

## Other Sprite Instructions

There are many other sprite instructions. Please refer to HELP and the sample programs for information on how they are used. The following are the other main instructions.

### ◆ SPHIDE instruction

Hides a sprite

### ◆ SPSHOW instruction

Shows a hidden sprite

### ◆ SPANIM instruction

Performs sprite animation

### ◆ SPLINK instruction

Links multiple sprites to produce a multi-jointed character

### ◆ SPUNLINK instruction

Releases the link made with the SPLINK instruction

### ◆ SPCLR instruction

Stops using a specified sprite and releases the memory