PERSONAL COMPUTER

FX-730P

OWNER'S MANUAL

CASIO

PERSONAL COMPUTER
FX-730P
OWNER'S MANUAL

The contents of this manual may be subject to change without notice. Unlawful copying of all or any portion of this manual is strictly forbidden. Please be aware that the use of this manual for other than personal use without permission from CASIO is prohibited under the copyrighting law. CASIO Computer Co., Ltd. shall not be held responsible for any damages or losses resulting from the use of this manual.

CASIO®

Foreword

This manual contains easily comprehensible explanations on the numerous functions of this computer.

The first feature of this equipment is the fullness of its scientific functions including statistical calculations. Incorporating many numeric functions and basic statistics, this computer will enable you to carry out troublesome scientific and technological calculations as well as statistical calculations by simple operation. In addition, more advanced calculations can be made by using a BASIC program.

The second feature is the "FUNCTION MEMORY", which makes possible storage of numeric expressions to enable outputting calculation results by inputting the necessary numeric values. By using this function, you can easily make numeric calculations without any special BASIC program.

The third feature is the "DATA BANK function," which permits the use of this computer as an "electronic memo pad." Unlike a paper memo pad which requires laborious efforts later to put the individual entries in order or retrieve them, this computer performs such work quite simply.

As shown above, this computer can be used by a wide variety of people – persons who want to learn BASIC for the first time, those who already know BASIC and want to make full use of this computer.

We hope that this manual will enable you to make effective use of this computer for many years to come.

Prior to Operation

This computer was delivered to you through CASIO's strict testing process, high level electronics technology, and strict quality control.
To ensure a long life for your computer, please observe the following precautions.

■ Utilization Precautions

- Since this computer consists of precision electronic parts, do not disassemble it. Also do not apply an impact to it by throwing or dropping it, or do not expose it to rapid temperature changes. In addition, do not store it in a place with high temperatures or high humidity, or in a dusty place. When the computer is utilized in low temperatures, sometimes the display response is slow or does not operate. When normal temperature conditions are restored, however, the computer operation will become normal.
- Special care should be taken not to damage the computer by bending. For example, do not carry it in your hip pocket.
- Please do not connect units other than the FA-3, FA-5, SB-42 and FP-12S to the connector portion.
- Although the display sometimes becomes faint while the buzzer is sounding, it is not a malfunction. However, if the display becomes very faint, replace the batteries with new ones as soon as possible.
- Every two years, replace the batteries with new ones even if the computer is not used. Do not leave exhausted batteries inside it because trouble may occur due to battery leakage.
- Always keep the cap for the connector portion in place. Remove it only when peripherals are to be connected to the computer.
- If strong static electricity is applied to the computer, sometimes the memory content is changed, or key operation cannot be performed. To discharge static electricity accumulated in your body, touch a metallic substance like a door knob. If this occurs, remove the batteries, then replace them again.
- Always turn computer power off before connecting peripherals.
- To clean the computer, do not use volatile liquids such as benzine or thinner. Wipe it with a soft dry cloth, or a cloth dampened with a neutral detergent solution.

- Do not turn the power off during program execution or operation.
- Since the computer is made up of precision electronic parts, avoid dropping it while a program is being executed; otherwise the program execution may be stopped or the memory contents may be changed.
- When a malfunction occurs, contact the store where the computer was purchased or a nearby dealer.
- Before seeking service, please read this manual again, check the power supply, check the program for logic errors, etc.
- Note that the manufacturer assumes no responsibility for any loss or claims by third parties which may arise through use of this unit.
- Note that the manufacturer assumes no responsibility for any damages incurred as a result of data/program loss caused by malfunctions, repairs or battery replacements. Records of important data should be prepared to protect against such data/program losses.

Contents

CHAPTER 1 General Guide

1-1 Names of Components 2
1-2 Functions of Components 3
1-3 Power Supply 11
1-4 RAM Expansion Pack 13

CHAPTER 2 Manual Operations

2-1 Let's Operate the Computer 16
2-2 Begin with the Four Arithmetic Operations 21
2-3 Calculation Notes 23
2-4 Function Calculations 24
2-5 Statistical Calculations 36

CHAPTER 3 Using the "Function Memory"

3-1 Calculations with the Same Formula 44
3-2 Utilization for Preparing Tables 47

CHAPTER 4 Programming with BASIC Language

4-1 Writing Programs 52
4-2 Executing a Program 55
4-3 Variables 57
4-4 Method of Calculating the Program Length 64
4-5 Convenient Techniques 65
4-6 Error Messages and Debugging 67
4-7 Convenient Peripherals 72
4-8 Using a PB-100 Program 82

CHAPTER 5 Program Library

5-1 Rearrangement of Data (Sorting) 88
5-2 Horse Race Game 91

CHAPTER 6 Command Reference

6-1 Manual Commands 99
NEW [ALL] 99
RUN 100
LIST 101
PASS 103
SAVE [ALL] 105
LOAD [ALL] 106
VERIFY 107
CLEAR 107
6-2 Program Commands 108
END 108
STOP 108
[LET] 109
REM 110
INPUT 111
KEY$ 113
PRINT 114
CSR 115
GOTO 116
ON ~ GOTO 117
IF ~ THEN 118
FOR ~ TO ~ [STEP] NEXT 119
GOSUB 121
RETURN 122
ON ~ GOSUB 122

	DATA	123
	READ	124
	RESTORE	125
	PUT	126
	GET	128
	BEEP	130
	DEFM	131
	DIM	133
	ERASE	135
	MODE	136
	STAT CLEAR	137
	STAT	137
	STAT LIST	138
	SET	139
6-3	Character Functions	140
	LEN	140
	MID$	141
	VAL	142
	STR$	143
6-4	Numeric Functions	144
	SIN, COS, TAN	144
	ASN, ACS, ATN	145
	HYPSIN, HYPCOS, HYPTAN	146
	HYPASN, HYPACS, HYPATN	147
	LOG, LN	148
	EXP	148
	SQR	149
	CUR	149
	ABS	150
	SGN	150
	INT	151

	FRAC	152
	RND	152
	REC	153
	POL	154
	FACT	155
	NPR	156
	NCR	157
6-5	Statistic Functions	158
	EOX	158
	EOY	159
6-6	Others	160
	RAN#	160
	DEG	161
	DMS$	162
	HEX$	163
	&H	164
6-7	DATA BANK Commands	166
	NEW#	166
	LIST#	166
	SAVE#	168
	LOAD#	169
	READ#	170
	RESTORE#	172
	WRITE#	174

CHAPTER 7 Convenient DATA BANK Function

7-1	Specifying the MEMO IN Mode	178
7-2	Inputting Data	179
7-3	Displaying the Data Contents	182
7-4	Correcting Data	183

7-5 Retrieving (Searching) Data 185
7-6 Erasing Data 191
7-7 Adding and Inserting Data 192
7-8 Searching Using a Program 195
7-9 Application to Tabular Calculations 197
7-10 Combining with the Function Memory 201

Appendix

Character Code Table 204
Numeric Functions 205
Error Messages 208
Specifications 211
Index 214

1 General Guide

To those who have never touched a computer or are already familiar with computers, it is recommended that you first read this chapter carefully. The quicker you become familiar with the configuration of the computer the quicker you will be able to use it properly.

1-1 Names of Components

① Display window
② Alphabet keys
③ Shift key
④ Display contrast
⑤ Function memory keys
⑥ Function key
⑦ Numeral keys
⑧ Decimal point key
⑨ Exponent key
⑩ Execution key
⑪ Calculation command keys
⑫ Connector for peripherals
⑬ Power switch
⑭ Mode key
⑮ Memo/search key

1-2 Functions of Components

- **Power Switch**

When this switch is moved to the right, the power is turned on, and when it is moved to the left, the power is turned off.

- **Shift Key (Red [S] Key)**

If this key is pressed, the shift mode is selected ("[S]" is displayed) and the command or symbol printed above each key can be displayed. When it is pressed again, the shift mode is released and "[S]" disappears. (To distinguish this key from the alphabetical [S] key, it will be written as [SHIFT] from now on in this manual.)

- **Function Key (Blue [F] Key)**

If this key is pressed, the function mode is selected ("[F]" is displayed) and the function printed below each key can be displayed. When it is pressed again, the function mode is released and "[F]" disappears. (To distinguish this key from the alphabetical [F] key, it will be written as [FUNC] from now on in this manual.)

- **Numeral Keys, Decimal Point Key, Calculation Keys, and Execution Key**

Examine this key array carefully. It is the same as that of an ordinary calculator, isn't it? This part is used when the four arithmetic calculations (addition, subtraction, multiplication, division) are performed. However, the following differences exist. The [×] (multiplication) and [÷] (division) keys are different and there is no [=] key while there is an [EXE] (execution) key. This occurs because a computer uses an * (asterisk) for × and / (slash) for ÷, while the answer is obtained by the [EXE] key instead of the [=] key.

For example, an operation is performed by an ordinary calculator as 12 [X] 4 [÷] 3 [+] 7 [−] 5 [=] while this computer uses 12 [*] 4 [/] 3 [+] 7 [−] 5 [EXE] .

This computer can be used as an ordinary calculator as shown above. When followed by the [SHIFT] key, one of the numeral keys ([0] to [9]) can be used to specify a program area from P0 to P9 while the [·] key is used for power calculation ($x^y \rightarrow x\uparrow y$) and the [+] [−] [*] [/] keys are used to enter relational operators ($\geqq, \leqq, >, <$).

- **Alphabetical Keys, Space Key**

Using these keys, commands are entered, or programs are written. Each of the 26 alphabetical keys from [A] to [Z] functions as a memory (for storage locations).

Also, the [A] ~ [Z] keys have another function. When they are pressed after the [SHIFT] key, a symbol or BASIC command is displayed.

Press the space key ([SPC]) when a space is required.

Example: [SHIFT] [A] → **GOSUB**, [SHIFT] [U] → **?**

In addition, the alphabetical keys have another use in the extension mode (When [EXT] is pressed, "EXT" is displayed). When they are directly pressed, small alphabetical characters are displayed, and when they are pressed after the [SHIFT] key, special symbols are displayed.

Extension mode functions:

q	w	e	r	t	y	u	i	o	p
a	s	d	f	g	h	j	k	l	,
z	x	c	v	b	n	m	SPC	=	E

Functions provided when a key is pressed after the [SHIFT] key in the extension mode:

%	'	@	¥	[	]	&	←	↓	→
○	□	△	×	÷	♠	♥	♦	♣	∴
■	▪	°	_	Σ	Ω	μ	SPC	γ	σ

To release the extension mode, press [EXT] again.

This computer is provided with the [FUNC] key. When a key is pressed after [FUNC] key, one of the following functions is diaplayed.

Example: [FUNC] [Q] → SIN

SIN	COS	TAN	ASN	ACS	ATN	LOG	LN	EXP	SQR
HYP	ABS	INT	FRAC	SGN	RAN #	SET	RND(	DEG(	DMS$(
VAL(	STR$(	LEN(	MID$(	DATA	READ	RESTORE	SPC	&H	HEX$(

In the extension mode, capital alphabetical characters are displayed.

- **Equal key ([=])**

This key is not used to provide an answer for calculation, but is used for an assignment statement (see page 109) and for a condition in an IF statement (see page 118).

Also, when this key is pressed after the [SHIFT] key, a ≠ (not equal) symbol is displayed.

- **Exponent/Pi Key ([π/E])**

When this key is directly pressed, it is used to provide an exponent. For example, operate [1][·][2][3][E][4] for 1.23×10^4. When an exponent is a negative number, press the [−] key after this key. For example, operate [7][·][4][1][E][−][9] for 7.41×10^{-9}.

When this key is pressed after the [SHIFT] key, Pi (the ratio of the circumference of a circle to its diameter) is displayed.

- **Answer Key ([ANS])**

When this key is pressed after the [SHIFT] key, the result of manual or program calculation executed immediately before is displayed.

- **Function Memory Keys ([IN][OUT][CALC])**

These keys are for the Function Memory. They will be described in detail in Chapter 3.

- **Cursor Movement Keys ([LTOP LEND])**

These keys are used when correcting displayed characters. The cursor ("_" blinking in the display window) is moved right and left by these keys. Each time this key is pressed, the cursor moves by one character. When the key is held down, the cursor moves continuously through all of the characters present. When [SHIFT][LTOP] is pressed, the cursor moves to the left edge of the display (the beginning of the line) – LINE TOP function. When [SHIFT][LEND] is pressed, the cursor moves to the right of the last input character (the end of the line) – LINE END function.

- **Memo Key (MEMO)**

Pressed to use the Data Bank function. Also pressed for sequential recall or for recall after pressing a specified character in the RUN mode (press MODE) or in the input mode (press MODE).

- **All Clear Key (AC)**

This key erases any display. Also, it is pressed when an error occurs, or when the display blanks out by auto power off (see page 12). When a program is being executed, program execution is suspended by pressing this key.

- **Delete/Insert Key (INS DEL)**

This key is used to delete a character where the blinking cursor is positioned. After deletion, the character to the right of the cursor moves to the left. When it is pressed after the SHIFT key, the character where the blinking cursor is positioned is moved to the right to provide a space.

- **Engineering/Stop Key (STOP ← ENG)**

When this key is pressed, a calculation result or a numeric value displayed by a PRINT statement is converted into an exponent display. When this key is pressed repeatedly, the displayed exponent will decrease by 3 each time. The exponent can be increased by 3 each time by pressing SHIFT .

When pressed during program execution, this temporarily suspends program execution. When it is pressed while characters are being scrolled, the display is temporarily suspended. Execution is resumed when the EXE key is pressed.

- **Mode Key ([MODE])**

When specifying a computer mode or an angle unit, use this key in combination with [·], [0]~[9].

[MODE][·]····· This turns the key input buzzer sound on and off. When the buzzer is on, the "**BUZZER**" symbol lights up on the display.

[MODE][0]····· The "**RUN**" symbol is displayed for manual and program calculations. (RUN mode)

[MODE][1]····· The "**WRT**" symbol is displayed for program writing, checking and editing. (WRT mode)

[MODE][2]····· The "**TRACE ON**" symbol is displayed for execution of tracing. (For details, see page 71.)

[MODE][3]····· When the "**TRACE ON**" symbol is displayed, the execution trace mode is canceled and "TRACE ON" disappears.

[MODE][4]····· The "**DEG**" symbol is displayed specifying "degrees" as the angle unit.

[MODE][5]····· The "**RAD**" symbol is displayed specifying "radians" as the angle unit.

[MODE][6]····· The "**GRA**" symbol is displayed specifying "grads" as the angle unit.

[MODE][7]····· The "**PRT ON**" symbol is displayed and printer output is possible when a printer is connected to the computer.

[MODE][8]····· When "**PRT ON**" is displayed, the printer output mode is canceled and "PRT ON" disappears.

[MODE][9]····· The " [MEMO][IN] " symbols light up to indicate that the MEMO IN mode is specified. This mode permits memo data input to the DATA BANK. (For details, see Chapter 7.) To cancel this mode, press [MODE][0] .

- **Display Contrast Control**

When the display is dark or faint, depending on the battery condition or display view angle, adjust it by moving the control located on the left side of the computer.

To increase the display contrast, turn the control in the direction indicated by the arrow. To weaken the contrast, turn it in the opposite direction. If the contrast is still weak even after the control is at its highest-contrast position, the batteries have probably run down. If so, replace the batteries as soon as possible. (For battery replacement, see page 11.)

- **Connector for Peripherals**

Use this connector (I/O port) for connecting to optionally available peripherals. When a printer is to be used, connect on the FP-12S printer or SB-42 interface pack (for connecting the computer and FP-40). When a tape recorder is to be used, connect it with the cassette interface FA-3 or FA-5.

Do not insert anything except the FP-12S, SB-42, FA-3 or FA-5. When any options are not used, always cover the connector with the connector cover supplied.

- **ALL RESET Button**

This button is located on the back of the unit. If this button is pushed with a pointed object when the power is on, the computer will be reset to the state where no specification or no input is performed. This operation may be used when the computer is in locked state due to strong static electricity. Caution is required since all programs and data will be lost if pushed.

ALL RESET Button

1-3 Power Supply

The power supply system for the computer is divided into the main power supply (two CR2032 lithium batteries) and an auxiliary power supply for memory backup (one CR1220 lithium battery). If the display contrast remains weak even after adjustment (see page 9), replace the batteries as soon as possible because they are becoming exhausted.

Note:
Be sure to replace the batteries every two years regardless of their use in order to prevent the chance of malfunction due to battery leakage.

■ Battery Replacement

1) Switch off the power supply and remove the rear panel after removing the two screws.
2) Remove the batteries.
   Main batteries:
   Remove the battery cover after loosening the screw Ⓐ.
   Auxiliary battery:
   Remove the battery cover after loosening the screw Ⓑ.
3) Remove the exhausted batteries. (They can be removed easily by tapping the battery compartment with its opening facing downward.)
4) Wipe the surfaces of new batteries well with a dry cloth before inserting them with the ⊕ side up.
5) Press the batteries down with the battery cover and slide the cover to close the battery compartment.
6) Replace the screws on the rear panel and switch on the power supply.

- **Auxiliary Batteries**

The auxiliary battery is for memory backup. This battery remains in operation while the main battery is being replaced, thus preventing the program and data from vanishing.

Bear in mind that if both the main and auxiliary batteries are removed at the same time, the program and data will vanish. If the main and auxiliary batteries must be replaced at the same time, press the ALL RESET button with a pointed object after switching on main frame power supply.

Notes:

1. Frequent use of the buzzer shortens battery life.
2. When replacing the main batteries, be sure to replace both at the same time.
3. Never throw batteries into a fire. It will be dangerous as they may burst.
4. Care should be taken to ensure that battery polarity(⊕ , ⊖) is correct.

Keep batteries out of reach of children. If swallowed by accident, consult a doctor immediately.

■ Auto Power Off

This is an automatic power-saving function designed to prevent waste of power when a user forgets to switch off the power supply. The power supply is automatically cut off in 6 minutes upon completion of operation (except during program execution) or upon key-input waiting state following execution of an INPUT or PRINT statement.

In such a case, power supply can be resumed by turning the power switch off and then on again or by pressing the [AC] key.

Note:

Even when the power supply is cut off, variable content, program content and DATA BANK content will be retained but mode specifications ("WRT", "TRACE ON", "PRT ON", etc.) will be initialized (immediately after turning the power on).

1-4 RAM Expansion Pack

The standard free area (area where programs and data can be written) has 7520 bytes and 26 variables, but it can be expanded to a maximum of 15712 bytes by installing the optionally available RP-8 RAM pack.
The expanded area can be used in entirely the same way as the standard area.

■ How to Install the RAM Pack

< Preparations >

In view of the possibility that the internal circuitry of the RAM pack might be destroyed by static electricity, touch a metallic substance like a door knob to discharge the static electricity accumulated in your body before you handle the pack.

< Procedure >

1) Switch off the power supply. (Power switch → OFF)
2) Loosen the two screws provided on the rear panel of the main frame and remove the panel.
3) Insert the RAM pack into its compartment and tighten the three screws.
   *Never touch the connector portion of the RAM pack or the PCB pad portion of the computer body.
4) Tighten the screws on the rear panel.

5) Switch on the power supply and press the ALL RESET button with a pointed object.

- After installing or removing the RAM pack, be sure to press the ALL RESET button with a pointed object. If the ALL RESET button is not pressed, the memory contents may be changed or a meaningless display may be shown.
- Use care not to allow the connector portion of the pack or the PCB pad portion of the computer body to become dusty or dirty, and avoid getting fingerprints on them as this will cause poor contact.
- Be sure to place the removed pack in its case and store in a location where it is not subject to dust or dirt.

2 Manual Operations

One must at least operate the computer to become familiar with the unit. Even if you operate something wrong, the machine will not be broken. Since practice makes perfect, as the proverb says, begin practising simple operations.

2-1 Let's Operate the Computer

Try the computer and see how it works.
First slide the power switch to ON, and the following display will appear.

```
BUZZER        RUN DEG
Ready P0
```

First erase this display. To do so, press the [AC] key. "Ready P0" will vanish. Then "–" will begin blinking at the left end. This is called the "cursor" and indicates the starting point for character writing.

```
BUZZER        RUN DEG
```

The state in which this cursor is blinking is called "key-input waiting state", – namely, the blinking cursor indicates that the computer is waiting for a calculation or a command. The cursor is usually indicated by a blinking "–", but as characters are written continuously, it sometimes changes to a blinking " ▮ ". On this computer one line consists of up to 62 characters. The " ▮ " symbol appears as a warning signal when the number of written characters exceeds 55.
"BUZZER", "RUN" and "DEG" will probably appear at the top of the display. These are called mode displays and indicate the state of the computer. "RUN" indicates the RUN mode in which manual calculations and program execution can be performed. "BUZZER" shows that the buzzer is on. The buzzer beeps at each key input. "DEG" shows that the angle unit is the degree. In addition, other angle units are the radians ("RAD" lights up), which is specified by pressing [MODE][5] , and the grads ("GRA" lights up) specified by pressing [MODE][6] . Be careful about these angle units when handling a trigonometric function, inverse trigonometric function or coordinate transformation. Once an angle unit is specified, it remains in effect even when power is switched off.
The other modes displayed are the program writing mode ("WRT" lights up) specified by pressing [MODE][1] , the trace mode ("TRACE ON" lights up, see page

71) specified by pressing [MODE][2], the printer output mode ("PRT ON" lights up, see page 80) specified by pressing [MODE][7], the MEMO IN mode for the DATA BANK function ("[MEMO] [IN]" light up) specified by pressing [MODE][9] and the extension mode ("EXT" lights up) specified by pressing [EXT].

You will learn these as you become familiar with the computer.
Now actually press the keys to display the modes. If a confusion has arisen in mode display, switch the power supply off and then on again.

First try a simple calculation.

Example:

1 2 3 + 4 5 6 = 5 7 9

Press [AC].

```
_
```

Press keys according to the above equation.

[1][2][3][+][4][5][6]

```
123+456_
```

Then press [EXE] instead of [=] to find the answer.

[EXE]

```
579
```

The calculation is as simple as with an ordinary calculator, isn't it?
Now make a calculation including both multiplication and addition.

Example:

33 × 5 + 16 = 181

Here it is assumed that 34 has been input by mistake instead of 33.

[3][4][*][5][+][1][6]

```
34*5+16_
```

You notice the mistake, but don't worry. Press the cursor movement key (←) and bring the cursor to the wrong numeral.

← ← ← ← ← ←

```
34*5+16
```

The cursor and 4 blink by turns.

Then press the right key 3 .

3

```
33*5+16
```

Now the calculation formula has been corrected. Find the answer.

EXE

```
181
```

As shown above, when a mistake is noticed during the input process, it can be easily corrected by using the cursor movement keys. However, when a mistake is noticed after the EXE key has been pressed, start the calculation again from the beginning.

Now write characters using the alphabet keys.

These keys are arranged in the same manner as on typewriters (ASCII arrangement).

First write capitals.

Example: Input **A**, **B**, **C**, **X**, **Y** and **Z**.

First input A, B and C.

A B C

```
ABC_
```

Then input X, Y and Z.

X Y Z

```
ABCXYZ_
```

Next, insert a one-character space between ABC and XYZ. Bring the cursor to the position of X.

[←][←][←]

ABCXYZ

The cursor and X blink by turns.

Make a one-character space.

[SHIFT] [INS]

ABC_XYZ

To insert a space between characters in this manner, place the cursor where the space is to be inserted and press [SHIFT] [INS]. When desiring to insert some spaces, keep this key pressed after the [SHIFT] key.

This computer is capable of displaying small letters and special characters in addition to numerals and capitals. For displaying these characters, use the extension mode. See page 5.

Example:

Display the small letters **a**, **b** and **c**.

First specify the extension mode.

EXT lights up.

[EXT]

BUZZER EXT RUN DEG

Then input A, B and C.

[A][B][C]

BUZZER EXT RUN DEG

abc_

Example:

Display the marks ♠♥♦♣.

Since the computer is already in the extension mode, just press each of the relevant key after pressing the [SHIFT] key.

[SHIFT][←][SHIFT][←][SHIFT][←][SHIFT][←]

BUZZER EXT RUN DEG

abc♠♥♦♣_

Example:

Display the symbols Σ Ω μ .

Press keys as shown below in the extension mode:

Since the above marks and symbols are available, they may be used for various purposes. To cancel the extension mode, press EXT again and the "EXT" symbol disappears.

Now we believe you understand key operations. While you are practicing key operation, "**Error 2**" may be displayed and the pressed key locked. This is not a trouble but a message that the wrong operation has been performed. It is therefore called the "error message." In such a case, press the AC key. Then the error message will disappear and the computer will become operable again. There are several kinds of such error messages. For details, see page 68.

2-2 Begin with the Four Arithmetic Operations

Try simple calculations. Bear in mind that there is a priority sequence in operation, i.e., multiplication and division take precedence over addition and subtraction.

Example 1:

$23+4.5-53=-25.5$

Operation

[2][3][+][4][·][5][−][5][3][EXE] -25.5

*From here on, numerals will be no longer placed in boxes.

Example 2:

$56\times(-12)\div(-2.5)=268.8$

Operation

56[×][−]12[÷][−]2.5[EXE] 268.8

*In the case of a negative numeral, press the [−] key before pressing the numeral key.

Example 3:

$7\times8-4\times5=36$

Operation

7[×]8[−]4[×]5[EXE] 36

*Multiplications are executed first, followed by subtraction.

Example 4:

$(4.5\times10^{75})\times(-2.3\times10^{-78})=-0.01035$

Operation

4.5[E]75[×][−]2.3[E][−]78[EXE] -0.01035

*For exponent display, input an exponent after pressing the [E] key.

In addition to the calculations as shown above, algebraic calculations using variables are possible with this computer. These calculations are convenient when a certain value is used repeatedly.

For example:

$3x + 5 =$
$4x + 6 =$
$5x + 7 =$

If the value of x in the above calculations is 123.456, it is troublesome to press the same numeral keys repeatedly. A labor-saving method for such calculations is algebraic calculation using a variable. Use variable X.
First assign 123.456 to the variable X.

[X][=] 1 2 3 . 4 5 6 [EXE]

Where [=] does not mean "equal," but "assignment of the right side to the left side." Now start calculation.

3 [*][X][+] 5 [EXE]	375.368
4 [*][X][+] 6 [EXE]	499.824
5 [*][X][+] 7 [EXE]	624.28

Repetitive calculations can be made as simple as this when a variable is used.

This computer has 26 variables from A to Z, which makes possible storage of numerous values.
In the above example, the value of the variable X is constant while calculation formulas differ.
Please note, in a calculation where formulas are constant and the value of the variable differs the computer works in a different way. For example, in a calculation of a formula $3x + 5 =$ where x varies from 123 to 456 to 789 the computer uses a function to store numeric expressions (calculation formulas). This will be described in Chapter 3.

2-3 Calculation Notes

■ Priority Sequence in Calculation

As mentioned in the preceding section, calculations are subject to the rule of "priority sequence" (true algebraic logic) which requires that multiplication and division take precedence over addition and subtraction. This computer automatically judges the priority sequence. You simply input a numeric expression and the correct answer will be displayed.

Here is the priority sequence in a calculation:

1) Functions (sin, cos, tan, etc.)
2) Power (↑)
3) Multiplication (✱), division (/)
4) Addition (+), subtraction (−)

Calculations are performed according to this priority sequence. When calculations happen to be equal in the priority sequence, priority is given to the calculation on the left. If there are parentheses, top priority should be given to the parenthesized calculation.

Example:

```
2 + 3 * SIN( 17+13 ) ↑ 2 = 2.75
          └──①──┘
        └────②────┘
        └──────③──────┘
    └──────④──────┘
└──────⑤──────┘
```

■ Number of Input/Output Digits and Calculation Digits

The range of input values (number of input digits) acceptable to this computer is 12 digits for a mantissa and 2 digits for an exponent. The same number of digits apply to internal calculations.

The displayed range of a value (number of output digits) is 10 digits for a mantissa and 2 digits for an exponent.

Example:

1 [.] 2345678912 [EXE]	1.234567891
12345678912 [✱] 100 [EXE]	1.234567891E12
12345678912 [✱] −100 [EXE]	−1.234567891E12

2-4 Function Calculations

This computer is capable of performing function calculations in addition to the four arithmetic operations.
The functions can be used in a program, but manual operation is described here.

This computer is provided with the following functions:

Name of function	Format	Function and input range	
Trigonometric function	SIN (Numeric expression) * hereafter X	sin	$\lvert X\rvert < 1440°$ (8π rad, 1600gra)
	COS (X)	cos	$\lvert X\rvert < 1440°$ (8π rad, 1600gra)
	TAN (X)	tan	$\lvert X\rvert < 1440°$ (8π rad, 1600gra) except when $\lvert X\rvert$ is odd multiple of $90°$ ($\pi/2$ rad, 100gra)
Inverse trigonometric function	ASN (X)	$\sin^{-1}$	$\lvert X\rvert \leqq 1, -90° \leqq ASN \leqq 90°$ (rad: $-\pi/2 \leqq ASN \leqq \pi/2$, gra: $-100 \leqq ASN \leqq 100$)
	ACS (X)	$\cos^{-1}$	$\lvert X\rvert \leqq 1,\ 0° \leqq ACS \leqq 180°$ (rad: $0 \leqq ACS \leqq \pi$, gra: $0 \leqq ACS \leqq 200$)
	ATN (X)	$\tan^{-1}$	$-90° \leqq ATN \leqq 90°$ (rad: $-\pi/2 \leqq ATN \leqq \pi/2$, gra: $-100 \leqq ATN \leqq 100$)
Hyperbolic function	HYP SIN (X)	sinh	$\lvert X\rvert \leqq 230.2585092$
	HYP COS (X)	cosh	$\lvert X\rvert \leqq 230.2585092$
	HYP TAN (X)	tanh	$\lvert X\rvert < 10^{100}$
Inverse hyperbolic function	HYP ASN (X)	$\sinh^{-1}$	$\lvert X\rvert < 5 \times 10^{99}$
	HYP ACS (X)	$\cosh^{-1}$	$1 \leqq X < 5 \times 10^{99}$
	HYP ATN (X)	$\tanh^{-1}$	$\lvert X\rvert < 1$
Square root	SQR (X)	$\sqrt{x}$	$X \geqq 0$
Cube root	CUR (X)	$\sqrt[3]{x}$	$\lvert X\rvert < 10^{100}$
Power	X ↑ X	x^y	$x < 0 \rightarrow y$: natural number
Exponential function	EXP (X)	e^x	$-10^{100} < X \leqq 230.2585092$

Name of function	Format	Function and input range	
Common logarithm	LOG (X)	$\log_{10} x$	X > 0
Natural logarithm	LN (X)	$\log_e x$	X > 0
Integer	INT (X)	$[x]$	Gives maximum integer not exceeding X (equal to Gaussian function $[x]$)
Fraction	FRAC (X)	FRAC	Gives decimal portion of X
Absolute value	ABS (X)	$\lvert x \rvert$	Gives absolute value of X
Sign	SGN (X)	sgn x	1 when X > 0 0 when X = 0 -1 when X < 0
Rounding off	RND (X, Number of digits)*	RND(	Gives the value of X which is rounded off at the specified digit. |Number of digits| < 100
Random numbers	RAN #	RAN #	Generates a 10-digit random number. 0 < RAN # < 1
π	π	π	Gives approximate value of ratio of circle circumference to diameter.
Decimal → sexagesimal conversion	DMS$ (X)*	DMS$ (	Converts decimal number given as X into sexagesimal character string in degrees, minutes and seconds. $\lvert X \rvert < 10^5$
Sexagesimal → decimal conversion	DEG (deg. [, min. [, sec.]])*	DEG (	DEG $(x, y, z) = x + y/60 + z/3600$. $\lvert \text{DEG}(x, y, z) \rvert < 10^{100}$
Decimal → hexadecimal conversion	HEX$ (X)*	HEX$ (	Converts value of X into 4-digit hexadecimal character string. -32769 < X < 65536

Name of function	Format	Function and input range	
Hexadecimal → decimal conversion	&H Hexadecimal character string	&Hx	Character string contains hexadecimal number within 4 characters.
Factorial	FACT (X)	$x!$	$0 \leqq X \leqq 69$ (0 and positive integer)
Permutation	NPR (n, r)*	nPr	$0 \leqq r \leqq n < 10^{10}$ (0 and positive integer)
Combination	NCR (n, r)*	nCr	$0 \leqq r \leqq n < 10^{10}$ (0 and positive integer)
Rectangular → polar coordinate transformation	POL (X, Y)* X, Y: numeric expressions	POL (	$\lvert X\rvert < 10^{100}$, $\lvert Y\rvert < 10^{100}$, $\lvert X\rvert + \lvert Y\rvert \neq 0$ r is given as a function value for assignment to variable X while value of θ is assigned to variable Y.
Polar → rectangular coordinate transformation	REC (r, θ)* r, θ: numeric expressions	REC (	$0 \leqq r < 10^{100}$, $\lvert\theta\rvert < 1440°$ (8π rad, 1600 gra) Gives x as a function value for assignment to variable X while value of y is assigned to variable Y.

Note:

In the case of asterisked functions, parameters must be parenthesized.

*Certain combinations or permutations may cause errors due to overflow during internal calculations.

Now perform calculations by using functions. Frequently used functions can be input at one touch of the respective function keys.

- **Trigonometric Functions (sin, cos, tan) and Inverse Trigonometric Functions ($\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$)**

When using these functions, be sure to specify the angle unit (degrees, radians, grads).

Example:

$\sin 12.3456° = 0.2138079201$

Operation:

[MODE] [4] (Angle unit: degrees (DEG))

[FUNC] [SIN] 1 2 . 3 4 5 6 [EXE] | 0.2138079201

Example:

$2 \cdot \sin 45° \times \cos 65.1° = 0.5954345575$

Operation:

2 [×] [FUNC] [SIN] 4 5 [×] [FUNC] [COS] 6 5 . 1 [EXE] | 0.5954345575

Example:

$\sin^{-1} 0.5 = 30°$

Operation:

[FUNC] [ASN] 0 . 5 [EXE] | 30

Example:

$\cos(\frac{\pi}{3}\text{rad}) = 0.5$

Operation:

[MODE] [5] (Angle unit: radians (RAD))

[FUNC] [COS] [SHIFT] [(] [SHIFT] [π] [/] 3 [SHIFT] [)] [EXE] | 0.5

Example:

$$\cos^{-1}\frac{\sqrt{2}}{2} = 0.7853981634\text{rad}$$

Operation:

[FUNC] [ACS] [SHIFT] [↓] [FUNC] [SQR] 2 [/] 2 [SHIFT] [↓] [EXE]

```
0.7853981634
```

Example:

$$\tan(-35\text{gra}) = -0.6128007881$$

[MODE] [6] (Angle unit: grads (GRA))

[FUNC] [TAN] [−] 3 5 [EXE]

```
-0.6128007881
```

- **Hyperbolic Functions (sinh, cosh, tanh) and Inverse Hyperbolic Functions ($\sinh^{-1}$, $\cosh^{-1}$, $\tanh^{-1}$)**

In the case of these functions, press [FUNC] [HYP] and then press the same keys as in the case of trigonometric and inverse trigonometric functions.

Example:

$$\sinh\left(-\frac{\pi}{2}\right) = -2.301298902$$

Operation:

[FUNC] [HYP] [FUNC] [SIN] [SHIFT] [↓] [−] [SHIFT] [π] [↓] [/] 2 [SHIFT] [↓] [EXE]

```
-2.301298902
```

Example:

$$\cosh^{-1} 1.5 = 0.9624236501$$

Operation:

[FUNC] [HYP] [FUNC] [ACS] 1 . 5 [EXE]

```
0.9624236501
```

- **Logarithmic Functions (log, ln), Exponential Function (e^x) and Power Function (x^y)**

Example:

$$\log 1.23 = \log_{10} 1.23 = 0.08990511144$$

Operation:

[FUNC] [LOG] 1 . 2 3 [EXE]

```
0.08990511144
```

Example:

$\ln 90 = \log_e 90 = 4.49980967$

Operation:

[FUNC] [LN] 9 0 [EXE]

```
4.49980967
```

Example:

$e^5 = 148.4131591$

Operation:

[FUNC] [EXP] 5 [EXE]

```
148.4131591
```

Example:

$123^2 = 15129$

Operation:

1 2 3 [SHIFT] [x^y] 2 [EXE]

```
15129
```

Example:

$123^3 = 1860867$

Operation:

1 2 3 [SHIFT] [x^y] 3 [EXE]

```
1860867
```

Example:

$10^{1.23} = 16.98243652$

Operation:

1 0 [SHIFT] [x^y] 1 . 2 3 [EXE]

```
16.98243652
```

Example:

$5.6^{2.3} = 52.58143837$

Operation:

5 . 6 [SHIFT] [x^y] 2 . 3 [EXE]

```
52.58143837
```

Example:

$123^{\frac{1}{7}} = \sqrt[7]{123} = 1.988647795$

Operation:

1 2 3 [SHIFT] [x^y] [SHIFT] [x^y] 1 [/] 7 [SHIFT] [x^y] [EXE]

```
1.988647795
```

- **Other Functions ($\sqrt{\ }$, $\sqrt[3]{\ }$, SGN, RAN#, RND, ABS, INT, FRAC)**

Example:

$\sqrt{2} + \sqrt{5} = 3.65028154$

Operation:

[FUNC] [SQR] 2 [+] [FUNC] [SQR] 5 [EXE]

3.65028154

Example:

$\sqrt[3]{27} = 3$

Operation:

[C] [U] [R] 2 7 [EXE]

3

Example:

Conversion into signs (positive number → 1, negative number → -1, 0 → 0)

Operation:

[FUNC] [SGN] 6 [EXE]

[FUNC] [SGN] 0 [EXE]

[FUNC] [SGN] [−] 2 [EXE]

1
0
-1

Example:

Generation of random numbers (0 < RAN# < 1 pseudorandom number)

Operation:

[FUNC] [RAN#] [EXE]

0.0756647782

(This value is not necessarily displayed.)

Example:

Round the result of 12.3 × 4.56 at the place of 10^{-2}.

$12.3 \times 4.56 = 56.088$

Operation:

[FUNC] [RND(] 1 2 . 3 [×] 4 . 5 6 [)] [−] 2 [SHIFT] [,] [EXE]

56.1

Example:

$|-78.9 \div 5.6| = 14.08928571$

Operation:

[FUNC] [ABS] [SHIFT] [,] [−] 7 8 . 9 [/] 5 . 6 [SHIFT] [,] [EXE]

14.08928571

Example:

The integer portion of 7800 ÷ 96 is 81.

Operation.

[FUNC] [INT] [SHIFT] [.] 7 8 0 0 [/] 9 6 [SHIFT] [.] [EXE]

```
81
```

*INT x gives an integer not exceeding x.

Example:

The decimal portion of 7800 ÷ 96 is 0.25.

Operation:

[FUNC] [FRAC] [SHIFT] [.] 7 8 0 0 [/] 9 6 [SHIFT] [.] [EXE]

```
0.25
```

- **Decimal-Sexagesimal Conversion (DEG, DMS$)**

Example:

14° 25′ 36″ = 14.42666667°

Operation:

[FUNC] [DEG] 1 4 [›] 2 5 [›] 3 6 [SHIFT] [.] [EXE]

```
14.42666667
```

Example:

12.3456° = 12° 20′ 44.16″

Operation:

[FUNC] [DMS$] 1 2 . 3 4 5 6 [SHIFT] [.] [EXE]

```
12°20'44.16
```

Example:

sin 63° 52′ 41″ = 0.897859012

Operation:

[MODE] [4] [FUNC] [SIN] [FUNC] [DEG] 6 3 [›] 5 2 [›] 4 1 [SHIFT] [.] [EXE]

```
0.897859012
```

- **Decimal-Hexadecimal Conversion (&H, HEX$)**

*A, B, C, D, E and F in hexadecimal numbers correspond to 10 ~ 15 in decimal numbers.

Example:

Convert a hexadecimal number into a decimal number.

Operation:

[FUNC] [&H] 1 0 [EXE]	16
[FUNC] [&H] 7 F F F [EXE]	32767
[FUNC] [&H] 8 0 0 0 [EXE]	-32768
[FUNC] [&H] F F F F [EXE]	-1

Example:

Convert a decimal number into a hexadecimal number.

Operation:

[FUNC] [HEX$(] 1 0 0 [SHIFT] [↵] [EXE]	0064
[FUNC] [HEX$(] 1 0 0 0 [SHIFT] [↵] [EXE]	03E8
[FUNC] [HEX$(] [−] 3 2 7 6 8 [SHIFT] [↵] [EXE]	8000
[FUNC] [HEX$(] 3 2 7 6 7 [SHIFT] [↵] [EXE]	7FFF
[FUNC] [HEX$(] 6 5 5 3 5 [SHIFT] [↵] [EXE]	FFFF

- **Factorial, Permutation and Combination (FACT, NPR and NCR)**

Example:

10 ! = 3628800

Operation:

[F] [A] [C] [T] 1 0 [EXE]	3628800

Example:

$_{10}P_4$ = 5040

Operation:

[FUNC] [nPr(] 1 0 [,] 4 [SHIFT] [↵] [EXE]	5040

Example:

$_{10}C_4 = 210$

Operation:

[FUNC] [nCr] 1 0 [,] 4 [SHIFT] [ans] [EXE]

210

- **Rectangular-Polar Coordinate Transformation (REC, POL)**

Here θ of polar coordinates (r, θ) is assumed to be obtained in radians.

Example:

The point $(5, \pi/6)$ in the polar coordinate is (4.330127019, 2.5) in the rectangular coordinate.

Operation:

[MODE] [5] (Angle unit: radians)

[FUNC] [REC] 5 [,] [SHIFT] [π] [/] 6 [SHIFT] [ans] [EXE]	(x)	4.330127019
[Y] [EXE]	(y)	2.5

Example:

The point (1, 1) in the rectangular coordinate is (1.414213562, 0.7853981634) in the polar coordinate.

Operation:

[FUNC] [POL] 1 [,] 1 [SHIFT] [ans] [EXE]	(r)	1.414213562
[Y] [EXE]	(θ)	0.7853981634

* In these functions, results are assigned to variables X and Y. Output values are the same as the content of variable X.

* In these functions, angle unit specification is as important as in the case of trigonometric or inverse trigonometric functions.

- **Specifying the Number of Significant Digits and the Number of Decimal Places**

"SET" is used for these specifications.

Specification of number of significant digits . . . SET E*n* ($n = 0 \sim 9$)

Specification of number of decimal places . . . . . SET F*n* ($n = 0 \sim 9$)

Release of specification . . . . . SET N

* "SET E0" used to specify the number of significant digits specifies 10 digits.

* When a specification is made, the result is displayed by the number of specified digits. (The digit next to the last specified digit is rounded off.) The original value remains in the computer.

Example:

100 ÷ 6 = 16.66666666···

Operation:

[FUNC/SET] [E] 4 [EXE] (Specified number of significant digits: 4)

1 0 0 [/] 6 [EXE] 1.667E01

Example:

123 ÷ 7 = 17.57142857···

Operation:

[FUNC/SET] [F] 2 [EXE] (Specified number of decimal places: 2)

1 2 3 [/] 7 [EXE] 17.57

Example:

1 ÷ 3 = 0.333333333···

Operation:

[FUNC/SET] [N] [EXE] (Specification released.)

1 [/] 3 [EXE] 0.3333333333

- **Engineering**

Example:

42.195km × 3 = 126.585km = 126585m

Operation:

42.195[×]3[EXE]	126.585
[ENG]	126.585E00
[ENG]	126585E03

Example:

78g × 96 = 7488g = 7.488kg

Operation:

78[×]96[EXE]	7488
[SHIFT][ENG]	7.488E03

2-5 Statistical Calculations

When collected data are analyzed, or when new data are obtained through forecasts or estimates, statistical calculation is indispensable, whether in office or technical work. This computer incorporating statistical calculation functions enables you to perform troublesome statistical calculations simply. Correlation coefficients or estimated values can be quickly obtained.

- **Statistical Data Input**

Prior to start of a statistical calculation, clear the exclusive statistical memories by pressing [SHIFT] [STAT] [SHIFT] [CLEAR] [EXE].
And input a statistical data as shown below in the RUN mode.

One-variable data

{ Individual data . . . . . [SHIFT] [STAT] data [EXE]
{ Plural identical data . . . . . [SHIFT] [STAT] data [SHIFT] [;] frequency [EXE]

Paired-variable data

{ Individual data . . . . . [SHIFT] [STAT] x data [,] y data [EXE]
{ Plural identical data . . . . . [SHIFT] [STAT] x data [,] y data [SHIFT] [;] frequency [EXE]

Here is an example of a stitistical data input.

Input data

x	1	3	5	5	3	3	3
y	2	4	4	6	3	3	3

Operation:

[MODE] [0] (RUN mode)
[SHIFT] [STAT] [SHIFT] [CLEAR] [EXE] (Statistical memories cleared.)
[SHIFT] [STAT] 1 [,] 2
[EXE]
[SHIFT] [STAT] 3 [,] 4 [EXE] [SHIFT] [STAT] 5 [EXE]※1 [SHIFT] [STAT] 5 [,] 6 [EXE]
[SHIFT] [STAT] 3 [,] 3 [EXE] [SHIFT] [STAT] 3 [,] 3 [SHIFT] [;] 2 [EXE]

```
Ready P0
STAT CLEAR _
_
STAT 1,2_
_
```

*1. y data can be omitted by using "[SHIFT] [STAT] x data ([SHIFT] [.] frequency) [EXE] ". In such a case y's values are the same as the previous ones. ([[SHIFT] [.] frequency] can be omitted.)

* Data x cannot be omitted.

* Input statistical data are stored even after power is off unless the STAT CLEAR command is executed by pressing [SHIFT] [STAT] [SHIFT] [CLEAR] [EXE]. Therefore, when inputting new statistical data, be sure to execute the STAT CLEAR command before input and after statistical calculations so as to clear statistical memories.

* When "frequency" is omitted, frequency is regarded as 1.

* Key input during statistical calculations is limited to 23 digits. Input of 24 or more digits is possible after performing [SHIFT] [STAT] 0 [SHIFT] [.] 0 [EXE] .

• Statistics Output

Statistics can be obtained by STAT LIST, STAT LIST 1, STAT LIST 2, EOX and EOY commands. This computer allows the calculation of the following statistics:

	Statistics	Formula
n	Number of statistical data processed	n
Σx	Sum of x data	Σx
Σy	Sum of y data	Σy
$\Sigma x \uparrow 2$	Sum of squares of x data	Σx^2
$\Sigma y \uparrow 2$	Sum of squares of y data	Σy^2
Σxy	Sum of products of x and y data	Σxy
$\Sigma x/n$	Mean of x data	$\frac{\Sigma x}{n}$
$\Sigma y/n$	Mean of y data	$\frac{\Sigma y}{n}$
$x\sigma_{n-1}$	Sample standard deviation of x data	$\sqrt{\frac{n\Sigma x^2-(\Sigma x)^2}{n(n-1)}}$
$y\sigma_{n-1}$	Sample standard deviation of y data	$\sqrt{\frac{n\Sigma y^2-(\Sigma y)^2}{n(n-1)}}$
$x\sigma_n$	Population standard deviation of x data	$\sqrt{\frac{n\Sigma x^2-(\Sigma x)^2}{n^2}}$
$y\sigma_n$	Population standard deviation of y data	$\sqrt{\frac{n\Sigma y^2-(\Sigma y)^2}{n^2}}$
a	Linear regression constant term	$\frac{\Sigma y-\mathrm{b}\cdot\Sigma x}{n}$
b	Linear regression coefficient	$\frac{n\Sigma xy-\Sigma x\cdot\Sigma y}{n\Sigma x^2-(\Sigma x)^2}$
r	Correlation coefficient	$\frac{n\Sigma xy-\Sigma x\cdot\Sigma y}{\sqrt{\lvert n\Sigma x^2-(\Sigma x)^2\rvert\ \lvert n\Sigma y^2-(\Sigma y)^2\rvert}}$
EOX	Estimated x value for a given y value	$\mathrm{EOX}(y)=\frac{y-\mathrm{a}}{\mathrm{b}}$
EOY	Estimated y value for a given x value	$\mathrm{EOY}(x)=a+x\cdot b$

STAT LIST (or STAT LIST 0) is for outputting all of the statistics.
STAT LIST 1 is for one-variable statistics alone and STAT LIST 2 for paired-variable statistics.
The details are explained in the following table:

	STAT LIST [0]	STAT LIST 1	STAT LIST 2
n	n	n	
Σx	Σx	Σx	
Σy	Σy		
Σx^2	$\Sigma x \uparrow 2$	$\Sigma x \uparrow 2$	
Σy^2	$\Sigma y \uparrow 2$		
Σxy	Σxy		
$\bar{x}$	$\Sigma x/n$	$\Sigma x/n$	$\Sigma x/n$
$\bar{y}$	$\Sigma y/n$		$\Sigma y/n$
$x\sigma_n$	$x\sigma_n$	$x\sigma_n$	$x\sigma_n$
$y\sigma_n$	$y\sigma_n$		$y\sigma_n$
$x\sigma_{n-1}$	$x\sigma_{n-1}$	$x\sigma_{n-1}$	$x\sigma_{n-1}$
$y\sigma_{n-1}$	$y\sigma_{n-1}$		$y\sigma_{n-1}$
a	a		a
b	b		b
r	r		r
	Output of all the statistics	Output of one-variable statistics	Output of paired-variable secondarily-processed statistics

Regression formula: $y = a + b \cdot x$

EOX and EOY are treated as functions. These can be calculated in the same way as ordinary numerical functions; "EOX y value" or "EOY x value". When x and y values are variables or numeric values, parentheses can be omitted in the same way as in the case of SIN, etc.
Now master statistical calculations by doing the following exercises:

Exercise:
The table at right shows the state of shipments of product x and product y. Determine the variance of shipments by finding the standard deviation.

Product \ Date	4	5	6	7	8
x	2	2	5	8	8
y	1	5	5	5	9

Operation:

First input the statistical data shown in the table. (Use the RUN mode.)

[SHIFT] [STAT] [SHIFT] [CLEAR] [EXE]

[SHIFT] [STAT] 2 [,] 1 [EXE]

[SHIFT] [STAT] 2 [,] 5 [EXE]

⋮

Input all x and y data.

Upon completion of data input, output all of the statistics one by one by using STAT LIST.

(Number of data)	[SHIFT] [STAT] [SHIFT] [LIST] [EXE]	n =5
(Sum of x data)	[EXE]	Σx =25
(Sum of y data)	[EXE]	Σy =25
(Sum of squares of x data)	[EXE]	Σx↑2=161
(Sum of squares of y data)	[EXE]	Σy↑2=157
(Sum of products of x and y data)	[EXE]	Σxy =149
(Mean of x data)	[EXE]	Σx/n=5
(Mean of y data)	[EXE]	Σy/n=5
(Standard deviation of x)	[EXE]	xσn =2.683281573
(Standard deviation of y)	[EXE]	yσn =2.529822128

Others omitted

Comparison of products x and y on the basis of the above calculation results shows that the sums total and the mean values are the same, but the standard deviation is larger in the case of product x. This suggests that there is a larger variance in shipment of this product.

Now find correlation coefficients and estimated values through a regression calculation with paired data.

Exercise:

The table below shows the ratios of advertising expenditure (advertising expenditure/operating expenses) and the ratios of operating profit (operating profit/sales amount) in seven chain supermarkets last year. Was advertising effective?

Store	1	2	3	4	5	6	7
Ratio of advertising expenditure (%)	0.8	2.1	2.5	1.8	3.1	4.0	1.0
Ratio of operating profit (%)	2.5	3.4	3.7	3.2	4.3	6.3	2.3

Approach:

Draw up a scatter diagram based on the table.

The scatter diagram suggests that profit increased with advertising expenditure. The line connecting the plotted points in the diagram is called the regression curve. In this case, it is almost linear and is therefore called linear regression. This linear regression is expressed by $y = a + b \cdot x$ where a is called the linear regression constant term and b the linear regression coefficient.

The correlation coefficient (r) is known to be within the range of $-1 \leqq r \leqq 1$. Correlationship is positive when $0 < r \leqq 1$, negative when $-1 \leqq r < 0$, and no correlationship when $r = 0$.

Now input data on the seven stores and obtain statistics.

Operation:

[SHIFT] [STAT] [SHIFT] [CLEAR] [EXE] (Statistical memories cleared)
[SHIFT] [STAT] 0 . 8 [,] 2 . 5 [EXE]
[SHIFT] [STAT] 2 . 1 [,] 3 . 4 [EXE]
⋮

(Input all the paired data.)

Next, output paired-variable statistics by using STAT LIST 2.

(Mean of x)	[SHIFT] [STAT] [SHIFT] [LIST] 2 [EXE]	Σx/n=2.185714286
(Mean of y)	[EXE]	Σy/n=3.671428571
(Standard deviation of x)	[EXE]	xσn =1.049392438
(Standard deviation of y)	[EXE]	yσn =1.245235819
(Sample standard deviation of x)	[EXE]	xσn-1=1.133473381
(Sample standard deviation of y)	[EXE]	yσn-1=1.345008409
(Linear regression constant term)	[EXE]	a =1.174221646
(Linear regression coefficient)	[EXE]	b =1.142512973
(Correlation coefficient)	[EXE]	r =0.9628252383

It is evident from the value of r that x and y have a positive correlation. Then, what advertising expenditure ratio should be adopted to bring the operating profit ratio to 5.7%? What operating profit ratio will result when the advertising expenditure ratio is 4.5%? Now estimate such values.

(Estimated value of x)	EOX5 . 7 [EXE]	3.961248986
(Estimated value of y)	EOY4 . 5 [EXE]	6.315530022

It is estimated from the above answers that the advertising expenditure ratio needed to bring the operating profit ratio to 5.7% is 3.96%, and that when the advertising expenditure ratio is 4.5%, an operating profit ratio of 6.32% will result.

3 Using the "Function Memory"

In this chapter we shall study the use of the "Function Memory" which is one of the features of the computer. This function greatly simplifies calculation of formula in which only the numeric values assigned to a variable differ.

3-1 Calculations with the Same Formula

This unit is provided with a very convenient function called "Function Memory". This function permits easy calculations by simply assigning numeric values to the variable as long as the formula is stored in advance.

The following keys are used for the "Function Memory".

[IN] ..... Stores the contents currently written.

[OUT] ..... Displays the stored content.

[CALC] ..... If the stored content is a formula, the desired numeric value will be assigned to the used variables and the calculated result will be displayed.

A simple example is given below to learn the use of these three keys.

Example:

Obtain the value of y for each of the values assigned to x when $y = 3.43 \text{ Cos } x$. (Calculate in three decimal places.)

x	8°	15°	22°	27°	31°
y					

Operation:

First specify the angle unit and number of decimal places.

[MODE][4] (Angle unit: "DEG")

[FUNC][SET][F]3[EXE] (Obtain in three decimal places by rounding off the 4th decimal place.)

Next, input a formula, and press the [IN] key to store it.

[Y][=]3.43[*][FUNC][COS][X][IN]

Press the [OUT] key to confirm that the formula has been stored.

[OUT] Y=3.43* COSX_

Then, start calculating by pressing the [CALC] key.

[CALC]	X ?
8 [EXE]	Y= 3.397
[CALC]	X ?
15 [EXE]	Y= 3.313
[CALC]	X ?
22 [EXE]	Y= 3.180
[CALC]	X ?
27 [EXE]	Y= 3.056
[CALC]	X ?
31 [EXE]	Y= 2.940

As shown in this example, the "Function Memory" is ideal for calculating a formula in which only the numeric values assigned to a variable differ.
If we add a semicolon (;) at the end of the formula when storing, the formula can be executed repeatedly by pressing the [EXE] key instead of the [CALC] key.

Example:
Obtain the value of V for the respective values of r when $V = 4/3\pi r^3$. (Calculate in three decimal places by rounding off.)

r	4.579	7.381	9.244	6.133	1.416
V					

Operation:
First specify the number of decimal places.

[FUNC] [SET] [F] 3 [EXE]

Then input the formula.

[V] = 4 [/] 3 [*] [SHIFT] [π] [*] [R] [SHIFT] [$^{-1}$] 3 [SHIFT] [$^{-1}$] [IN]

Start the calculations.

[CALC]	R ?
4.579[EXE]	V= 402.162
[EXE]	R ?
7.381[EXE]	V= 1684.357
[EXE]	R ?
9.244[EXE]	V= 3308.784
[EXE]	R ?
6.133[EXE]	V= 966.290
[EXE]	R ?
1.416[EXE]	V= 11.893
[MODE][⊘]	Ready P0

Repeated operations can be terminated by pressing [MODE][⊘].

3-2 Utilization for Preparing Tables

Multiple formulas can be written by separating with colons (:). Tables such as that shown below can be easily prepared by using this method.

Example:

Complete the following table. (Calculate in three decimal places by rounding off.)

X	Y	P = X · Y	Q = X / Y
4.27	1.17		
8.17	6.48		
6.07	9.47		
2.71	4.36		
1.98	3.62		

Operation:

[FUNC/SET] [F] 3 [EXE] (Specification of number of decimal places)

[P] [=] [X] [*] [Y] [SHIFT] [:] [Q] [=] [X] [/] [Y] [SHIFT] [:] [IN] (Storing the formula)

[CALC] (Calculation starts)

4.27 [EXE]

1.17 [EXE]

[EXE]

[EXE]

⋮

```
X ?
Y ?
P= 4.996
Q= 3.650
X ?
```

⋮

Continue to input the values of X and Y in this manner, and the values of P and Q will be calculated in successive order and the table will be completed as shown below.

X	Y	P = X · Y	Q = X / Y
4.27	1.17	4.996	3.650
8.17	6.48	52.942	1.261
6.07	9.47	57.483	0.641
2.71	4.36	11.816	0.622
1.98	3.62	7.168	0.547

Messages can also be added by enclosing them in quotation marks (" ") immediately after the variables. This will be convenient since the message will be displayed at time of input and it will be possible to tell at a glance what value is being input.

Example:

Complete the following table. (Calculate in two decimal places by rounding off.)

Radius (r)	Height (h)	Volume of a cylinder ($U = \pi r^2 h$)	Volume of a cone ($V = 1/3U$)
1.205	2.227		
2.174	3.451		
3.357	7.463		

Operation:

[FUNC] [SET] [F] [2] [EXE]

[U] [SHIFT] ["] [C] [Y] [L] [I] [N] [D] [E] [R] [SHIFT] ["] [=] [SHIFT] [π] [*] [R] [SHIFT] ["] [R] [A] [D] [I] [U] [S] [SHIFT] ["] [SHIFT] [!] 2

[*] [H] [SHIFT] ["] [H] [E] [I] [G] [H] [T] [SHIFT] ["] [SHIFT] [:] [V] [SHIFT] ["] [C] [O] [N] [E] [SHIFT] ["] [=] [U] [/] [3] [SHIFT] [:] [IN]

[CALC] (Calculation starts.)

1.205[EXE]

2.227[EXE]

[EXE]

[EXE]

⋮

```
RADIUS ?
HEIGHT ?
CYLINDER= 10.16
CONE= 3.39
RADIUS ?
```

⋮

If the values of radius (r) and height (h) are input in this manner, volume (U) of the cylinder and volume (V) of the cone will be calculated successively and the table will be completed as shown below.

Radius (r)	Height (h)	Volume of a cylinder ($U = \pi r^2 h$)	Volume of a cone ($V = 1/3U$)
1.205	2.227	10.16	3.39
2.174	3.451	51.24	17.08
3.357	7.463	264.22	88.07

By using this "Function Memory", simple repetitive calculations can be performed easily without the need to use "Program Calculations" which will be explained in the next chapter.

- Some of the points requiring care in using the "Function Memory" are listed below.

1) Character string of up to 62 characters can be stored with the [IN] key. The 63rd character and after will be discarded. Since spaces included in the commands and functions input with the one-key command will be counted as characters, delete these spaces with [DEL] if there are too many characters.
2) The stored contents will be retained even if power is turned off or if Auto Power off function is activated.
3) Error will occur when [CALC] is pressed if the stored content is other than a formula.
4) The functions in the fomula must be numeric functions.
5) The variables in the formula must be numeric variables A ~ Z (see page 57).
6) If an exclusive character variable $ (see page 57) is included in the formula, the content of the variable $ will be used. (e.g. VAL($)*A , etc.)

*It will be convenient to use the "Function Memory" in combination with the DATA BANK function. See Chapter 7, Section 7-10 "Combining with the Function Memory" on how to use this combination.

4 Programming with BASIC Language

In this chapter we will explain programs using the BASIC language. BASIC is one of the programming languages and is currently used in practically all personal computers. Publications on BASIC language are also readily available on the market. In this chapter, we shall focus our attention on special precautionary points relative to programming with BASIC language. Details in relation to commands and grammar of BASIC will be explained in Chapter 6 "Command Reference".

4-1 Writing Programs

The example shown below is a program using BASIC in which the value of y is calculated with the formula $y = 2x^2 + 91x + 125$ by inputting the value of x.

```
10 INPUT"x=",X
20 Y=2*X↑2+91*X+125
30 PRINT"y=";Y
40 GOTO 10
50 END
```

A BASIC program is a collection of "lines" with each line being composed of numerals (integers from 1 to 9999 can be used) called line numbers and statements (commands such as INPUT or formulas such as Y=2*X↑2+91*X+125). The program will also be executed in successive order starting from the smallest line number.

We will start now by inputting the above program.

Press [MODE][1]. The symbol "**WRT**" should appear in the upper part of the display.

The program must be written only when in the WRT mode as shown above. The large characters "P 0123456789" indicate the usage state of the program areas.

*The computer is provided with 10 program areas (P0 ~ P9) in which multiple programs can be stored separately. The program area can be changed by pressing any number key from [P0] to [P9] after pressing [SHIFT].

The blinking number indicates the program area currently specified. The numbers displayed indicate open areas. If a program has been written, that area number changes to a cursor (_). In the display shown below for example, the area specified is P5 and programs are already written in areas P1, P8 and P9.

The number at the upper right part of the display indicates the remaining capacity (number of bytes) of the area in which programs or data can be written (free area). The free area will be 7520 bytes when nothing is written and these numbers will decrease each time a program or data is written.

[N][E][W][A][L][L][EXE]

This operation erases the programs in all program areas and variable contents, and prepares program area P0.
Let us now write the previously mentioned program.

[1][0][SHIFT][INPUT][SHIFT]["][EXT][X][EXT][=][SHIFT]["][,][X][EXE]
[2][0][Y][=][2][*][X][SHIFT][:][2][+][9][1][*][X][+][1][2][5][EXE]
[3][0][SHIFT][PRINT][SHIFT]["][EXT][Y][EXT][=][SHIFT]["][SHIFT][;][Y][EXE]
[4][0][SHIFT][GOTO][1][0][EXE]
[5][0][E][N][D][EXE]

The program will be stored with the above key operation. **Do not forget to press the [EXE] key at the end of each line** since the line will be stored only by pressing [EXE]. Since one-key input is also possible for frequently used commands such as INPUT and PRINT, speedy inputs will be possible. (For example, the five characters "INPUT" can be input by pressing [SHIFT][X]. [I][N][P][U][T] can also be applied.)

- **Method of Correcting Program Errors**

1) When an error is found during input before [EXE] is pressed:
   Move the cursor to the error position by a cursor movement key ([←] or [→]) and correct the error. (Same as in Chapter 2)
2) When an error is found after storing the line by pressing [EXE]:
   Enter LIST line No. [EXE] in the WRT mode to display the contents of the line and correct as in 1) above.

In either case, always press the [EXE] key after correcting since corrections will not be made unless this key is pressed.

*Press [SHIFT] [↑ EXE] to display and correct the line immediately before a line being displayed.

*If there are additional programs following the corrected line in the case of 2), the next line will be displayed when [EXE] is pressed. If no other correction is required, press the [AC] key to clear the display.

- **Addition and Deletion of Lines**

1) To add a line, simply input it after clearing the display by pressing the [AC] key in the WRT mode. If a line is input with an already used line number, this line will have priority and the previously stored line will be erased.
2) To delete a line, enter the line number to be deleted after clearing the display with the [AC] key and press [EXE].

- **Erasing Programs**

1) To erase a program in the currently specified program area, enter NEW [EXE] in the WRT mode.
2) To erase the programs in all program areas (P0 ~ P9) and all data at one time, enter NEW ALL [EXE] in the WRT mode.

*The stored program will be retained even when the power is turned off.

4-2 Executing a Program

To execute a stored program, specify the RUN mode by pressing [MODE] [0]. The display will then appear as shown below.

Symbol indicating the RUN mode.

```
BUZZER      RUN DEG
Ready P0
```

Program area number currently specified.

The program in the currently specified program area will start if the RUN command is executed here. If the previously stored program is executed, it will be as shown below.

[MODE] [0]		Ready P0
[R] [U] [N]		RUN_
[EXE]	(Program starts.)	x=?
[2]	(2 is entered for x.)	2_
[EXE]	(The value of y is displayed.)	y= 315
[EXE]	(Jumps to line 10.)	x=?

To pause execution of a program, press the [STOP] key. "STOP" will then appear at the upper right part of the display and the execution will stop. If the [STOP] key is pressed again, the currently specified program area number and the stopped line number will be displayed. Execution of the program will resume when [EXE] is pressed.

	BUZZER RUN DEG x=?
[STOP]	BUZZER RUN DEG STOP x=?
[STOP] (Stops at line 10.)	BUZZER RUN DEG STOP P0-10
[EXE] (Resumes execution.)	BUZZER RUN DEG ?

Press [MODE] [⊘] to terminate execution of a program. The display will then return to the initial state of the RUN mode.

[MODE] [⊘]

```
Ready P0
```

- **Other Methods of Executing a Program**

1) RUN Line Number [EXE] :

   This operation will cause the program to start from the specified line.

2) [SHIFT] [P0] ~ [P9] :

   If a number key is pressed after the [SHIFT] key in the RUN mode, the program in the area specified by the number key will be executed from the first line.

*Variables are not cleared when executing a program.

4-3 Variables

Twenty-six variables named A to Z are provided as a standard feature of the computer and calculation results and numeric values can be stored in these variables. These variables have already been used for manual calculations, and they will serve as "memories" when the computer is used as a calculator.
Capital alphabetical letters A to Z are used for variable names and these will be called simple variables. Subscripts in parentheses such as A(3) and X(4, 5) can also be added to the variable name. Variables in this form are called array variables and are used when handling large volumes of data.
All of the variables mentioned up to this point are for numeric values and are called numeric variables. Variables to which character strings are assigned are called character variables. Character variables are indicated by adding a dollar sign ($) after the variable name such as A$. Character variables can also be used as array variables such as C$(12). In addition to these variables, the computer is provided with the exclusive character variable $.
In other words, the computer is provided with the following variables.

	Simple variable	Array variable
Numeric variable	A、B、C…、Z	A(0)、X(2,2)…… etc.
Character variable	A$、B$、…、Z$、$	A$(0)、X$(2,2)…… etc.

The numeric variable will store numeric values of up to 12 digits (12 digits for a mantissa and 2 digits for an exponent) and the character variable will store character string of up to 7 characters. The exclusive character variable $ will store character string of up to 62 characters.

< Two Array Declarations >

This computer is capable of array variable declarations by DIM statements and array declarations by DEFM statements. By using a DIM statement, it is possible to use array variables of up to three dimensions. The DEFM statement is used when desiring to expand the number of variables but it can also be used to declare array variables. A maximum of up to eight array variables can be used simultaneously with a DIM statement.

< DIM mode and DEFM mode >

It will not be possible to specify an array by a DIM statement and a DEFM statement at the same time. That is, the computer will select either the DIM mode or the DEFM mode and will display the mode such as "DEFM" if in the DEFM mode. When power is switched ON, the computer will revert to the state it was in before power was switched OFF. If a DIM statement was executed, it will revert to the DIM mode and, if a DEFM statement was executed, it will revert to the DEFM mode.

< Precautions when Switching Over >

Precautions will be required when switching modes since the array declared with DIM will be cleared when switching from the DIM mode to the DEFM mode, and the array declared with DEFM will be cleared when switching from the DEFM mode to the DIM mode.

< Array Declaration with a DIM Statement >

This computer is capable of declaring array variables of up to three dimensions. These array variables consist of two types which are the character array variables and the numeric array variables.

1) **Declaration of one-dimensional array variables**

A DIM statement is used to declare array variables and a one-dimensional array variable will be specified as follows.

```
DIM A(10)
```

*The numeral 10 in parenthesis is called a subscript.

There are 11 elements in this array consisting of A(0), A(1), A(2), ... A(10). These array elements exist independent of variables such as A, B, C, etc. which are numeric variables. A CLEAR statement or ERASE statement is therefore used when clearing these arrays. When declaring two array variables, punctuate the array variables with a comma.

```
DIM A(3),B(3)
```

We can then use array variables such as A(0), A(1), A(2), A(3), B(0), B(1), B(2), B(3).

For a character array variable, add the symbol $ after the variable name.

```
DIM A$(4)
```

This declares the array elements A$(0), A$(1), . . . A$(4) as the character array variables.

2) **Declaration of two-dimensional array variables**

When declaring two-dimensional array variables with a DIM statement, punctuate the subscripts with a comma.

```
DIM A(2,3)
```

The array variable is now declared in two-dimensions. There are 12 array elements as shown in the table below.

	y			
x	A(0, 0)	A(0, 1)	A(0, 2)	A(0, 3)
	A(1, 0)	A(1, 1)	A(1, 2)	A(1, 3)
	A(2, 0)	A(2, 1)	A(2, 2)	A(2, 3)

For a character array variable, add the symbol $ after the variable name.

As in the case of one-dimensional array variables, multiple array variables can be declared at the same time by punctuating with commas.

3) **Declaration of three-dimensional array variables**

Punctuate the subscripts with commas similar to when declaring two-dimensional array variables.

```
DIM A$(1,2,3)
```

Array declaration is in three dimensions with character array variable A$. There are 24 array elements (2 × 3 × 4) and since they exist independent of the character variable A$, the contents of the array variable remains unchanged regardless of what is substituted for A$. Similar to the one-dimensional array variable, it is possible to declare multiple array variables at the same time by punctuating with commas.

4) **Effective range of array variables**

One array can be used in multiple programs since array variables are common for all program areas from 0 to 9. Since eight bytes will be required for one array element, care will be required to ensure that there will be sufficient memory capacity when declaring different array variables in each program.

For method of using array variables with the DIM statement, refer to Chapter 5 "Program Library" and Chapter 6 "Command Reference."

< Variable Expansion >

Although 26 variables (A to Z) are provided as a standard feature of the computer, additional variables can be used by variable expansion. The variables can be expanded by using the DEFM command and specifying with the format "DEFM Number expanded (numeric expression)".

Example:

To add 20 variables for a total of 46, operate as shown below in the RUN or WRT mode.

[D][E][F][M][2][0][EXE]

```
BUZZER      RUN DEG   DEFM
A··Z:26   DEFM:20
```

In manual operation, the number of variables will be displayed as shown in the above.

Number of variables can be expanded up to a maximum of 966.
The remaining capacity of free area will decrease 8 bytes for each expanded variable so care will be required since there may be insufficient space to create a program if expanded excessively. The table below shows the relations between the number of variables and the maximum capacity of the free area. This shows that a large number of variables can be used with the RP-8 RAM expansion pack.

Number expanded	Number of variables	Standard free area	Free area with the RAM expansion pack
0	26	7520	15712
1	27	7512	15704
2	28	7504	15696
≀	≀	≀	≀
171	197	6152	14344
172	198	6144	14336
173	199	6136	14328
≀	≀	≀	≀
939	965	8	8200
940	966	0	8192
941	967	–	8184
≀	≀		≀
1195	1221		6152
1196	1222		6144
1197	1223		6136
≀	≀		≀
1963	1989		8
1964	1990		0

Capacity with nothing stored

The newly expanded variables are used as array variables following variable Z. The relation of each variable will be as shown below.

A(26) = B(25) = ····· = Y(2) = Z(1)
A(27) = B(26) = ····· = Y(3) = Z(2)
⋮ ⋮ ⋮ ⋮ ⋮
A(965) = B(964) = ····· = Y(941) = Z(940)

The DEFM command cannot only be used in manual operation but can also be used in a program. For example, if we wish to use array variables Z (0) to Z (20) in a program and assign a numeric value J to Z (J), the program can be created as follows.

Example:

```
10 DEFM 20          ···20 variables are expanded.
20 FOR J=0 TO 20 }
30 Z(J)=J        } Numeric values are assigned to the array
40 NEXT J        } variables.
     ⋮
```

*When the DEFM command is used in a program, the number of variables will not be displayed when executed.

The DEFM command is also used to display the current number of variables. DEFM only will be executed in this instance.

[D][E][F][M][EXE]

```
BUZZER  RUN DEG  DEFM
A··Z:26   DEFM:20
```

*When "DEFM" only is used in a program, the number of variables will be displayed when executing the program.

Notes:

1. If the variables are expanded, that specification will be saved even when the power is turned off. Execute DEEM 0 to return to the standard 26 variables.

   *The variables will return to the standard 26 if NEW ALL [EXE] is operated in the WRT mode or if changed over to the DIM mode.

[D][E][F][M][0][EXE]

```
BUZZER  RUN DEG  DEFM
A··Z:26 DEFM:0
```

2. If the number of bytes used to expand the variables is specified in excess of the remaining free area, a MEMORY OVER error (Error 1) will occur to protect the stored programs and data.

< Precautions in Using Variables >

1) If the names of the numeric variable and character variable are the same, the same memory space will be used. For this reason, it will therefore not be possible to use a numeric variable A and a character variable A$ at the same time. If the following program is executed, an error (Error 6) will occur on line 20. (See page 69 for details.)

```
10 A$="CASIO"
20 PRINT A
30 END
```

…… Assigns character string "CASIO" to character variable A$.

…… Displays the content of numeric variable A.

* Excluding array variables in the DIM mode.

2) Care should be taken when using an array variable in the DEFM mode since it uses the same memory space as some of the simple variables. In the figure shown below, the variables combined with equal signs (=) use the same memory space. (Although numeric variables are shown, these relations are the same as for character variables.)

```
A = A(0)
B = A(1) = B(0)
C = A(2) = B(1) = C(0)
D = A(3) = B(2) = C(1) = D(0)
⋮   ⋮   ⋮   ⋮   ⋮
Y = A(24) = B(23) = C(22) = ……… = Y(0)
Z = A(25) = B(24) = C(23) = …… = Y(1) = Z(0)
↑   └──────────────────────────────────┘
Simple variables      Array variables
```

For example, if we execute the following program, the content of variable C will be 10.

```
10 DEFM
20 C=0
30 A(2)=10
40 PRINT C
50 END
```

3) Since variables are common in all program areas, care should be taken in assigning variables when creating programs using multiple program areas.

4-4 Method of Calculating the Program Length

The maximum capacity of the free area used for programs and DATA BANK is 7520 bytes. When the RAM expansion pack is used, the free area will be 15712 bytes. This free area decreases as programs and data are written. (The figures shown at the upper part of the display in the WRT mode indicates the remaining number of bytes.) The number of bytes required to write programs and data are calculated as shown below.

- **Line number** . . . . . Two bytes for one regardless of the number between 1 to 9999.
- **Command** . . . . . One byte for one command.
- **Function** . . . . . One byte for one function.
- **Character** . . . . . One byte for one character (a space will also be considered a character.)
- **EXE key** . . . . . One byte will be required when the EXE key is pressed at the end of a line.
- If an array variable is declared with a DIM statement, eight bytes will be required for one array element.
- If the number of variables is expanded with the DEFM statement, eight bytes will be required for each variable expanded.

When using many array variables or when setting up a long program, it will be necessary to consider the length of the program according to the above calculations and to trim any unnecessary portions.

4-5 Convenient Techniques

Following are two techniques that are convenient to know when programming.

- **Using the Program Areas**

This unit is provided with ten program areas (P0 to P9) into which individual programs can be stored. One method of using these areas is to locate the main routine in one program area and the subroutine in the other area. A simple example is shown below.

Example:

Program area P0 (Main routine)

```
10 X=12
20 PRINT
30 PRINT CSRX;"@";
40 GOSUB #1
50 GOTO 20
60 END
```

Program area P1 (Subroutine)

```
10 K$=KEY$
20 IF K$="4" THEN X=X-1: IF X<0 THEN
   X=0
30 IF K$="6" THEN X=X+1: IF X>23 THEN
   X=23
40 RETURN
```

When the main routine in P0 is executed, @(unit price mark) appears at the center of the display. Press the [4] key to move @ to the left and the [6] key to move @ to the right. These are the only keys that can move @.

In this program, the main routine in P0 displays @ and the subroutine in P1 calculates the moving position of @ by key input. The subroutine in P1 is accessed by GOSUB #1 on line 40 of the main routine.

This method of allocating routines with coherent functions in one program area facilitates the usage of long programs.

- **Using Arrays with the DIM Mode**

In this example, we will process two types of data by using two-dimensional array variables.

Example:

Store the names and heights of 15 people.

```
P0 10 DIM A$(1,14)
   20 FOR C=0 TO 14
   30 INPUT "NAME",A$ (0,C)
   40 INPUT "HEIGHT",A$(1,C)
   50 NEXT C
   60 END

P1 10 INPUT "NAME",N$
   20 FOR C=0 TO 14
   30 IF N$=A$(0,C) THEN PRINT
      A$(1,C);"cm":GOTO 10
   40 NEXT C
   50 PRINT "NO NAME"
   60 GOTO 10
```

Input the names and heights to P0 since this is the input program. P1 is the program to display the height of the pertinent person when a name is entered. Variable C is used for controlling the FOR ~ NEXT loop and variable N$ is used for temporary storage when searching a name.

Array A$ (0, 0) ~ A$ (0, 14) is an array variable to store the names of 15 people and array A$ (1, 0) ~ A$ (1, 14) is an array variable to store the heights of 15 people.

*On line 30 in P1, when the name of the input person is concerned, his height is displayed. Then execution is jumped to line 10. To input a different name, press the EXE key when the height of previous person is displayed.

4-6 Error Messages and Debugging

An error in a program is called a "bug", and tracing and correcting this error is called "debugging".

Debugging can be carried out easily in the computer since it is provided with an automatic check mechanism that displays an error message if there is an error in program execution or in the grammar of the BASIC language. It is important to trace bugs persistently since there may be certain cases when a bug does not become an actual error but the desired results cannot be obtained.

1. Debugging with the Error Message

With the display as shown below, the error message reveals the type of the error, the program area and the line number where the error occurred.

The Error 2 display is called an "error code" and indicates the type of error. The various error codes are shown in the following list together with causes and countermeasures (debugging methods).

< Error Message List >

Error code/ Meaning	Cause	Countermeasure
Error 1 Memory over or system stack over	• Unable to write programs or expand variables due to insufficient capacity of free area. • Calculating area (stack) unable to hold formula since the formula is excessively complex. • Unable to write data in the data bank since capacity is insufficient. • Nine or more arrays were declared.	• Erase unnecessary programs with the NEW command or reduce the number of variables. • Separate and simplify the formula. • Clear the array
Error 2 Syntax error	• Format error in the program or formula. • The formats of left side and right side in the assigned statement differ. (Such as character type and numeric type) • Attempted to read character in a numeric variable with READ/ READ#. • Character string operation exceeded 62 characters.	• Correct the error in the input program. • Change numeric variable to character variable or check for character (including space) in the DATA statement. • Shorten the character string.
Error 3 Mathematical error	• When the calculation result of a formula exceeds 10^{100}. (Overflow) • When arguments are outside the input range of numeric functions. • When the results are uncertain or impossible. (Attempted to divide with a 0)	• Correct the formula or the data. • Check the data.

Error code/ Meaning	Cause	Countermeasure
Error 4 Undefined error	• No jump destination for the GOTO or GOSUB statements. • There is no data to be read with READ/READ# or RESTORE#. • The line number specified with RESTORE does not exist.	• Specify the correct jump destination. • Write data • Correct the line number.
Error 5 Argument error	• When the argument is outside the input range of commands and functions requiring arguments. • The subscript in the array is outside the input range. • Attempted to specify two arrays with the same name but different subscripts.	• Correct the argument error. • Change the subscript. • Change the array name.
Error 6 Variable error	• Attempted to use a variable that was not added. • Attempted to use the same variable name for a numeric variable and a character variable. • Attempted to use an array name subscript that was not declared.	• Expand the variables with the DEFM statement. • Change the variable name for the numeric variable and character variable. • Use after declaring the array or correct the array name subscript.
Error 7 Nesting error	• When the RETURN statement is used other than when executing a subroutine. • When the FOR statement and NEXT statement do not correspond or when the variable of the NEXT statement does not match that of the FOR statement. • When the subroutine nesting (calling a subroutine from a subroutine) exceeds eight levels.	• Correspond GOSUB ~ RETURN or FOR ~ NEXT correctly. • Correct the subroutine or FOR loop nesting level within the range.

Error code/ Meaning	Cause	Countermeasure
Error 7 Nesting error	• When the FOR loop nesting (inserting a loop within a loop with nesting form) exceeds four levels. • The CLEAR statement was used in the FOR ~ NEXT loop.	• Move the CLEAR statement outside the FOR ~ NEXT statement.
Error 8 Protect error	• When the following occurs with the password specified. 1) Input of a different password 2) Execution of a prohibited command 3) Editing of a program 4) Loading programs with different passwords. 5) Inputting data in the data bank 6) Calling data from the data bank	• Clear the password.
Error 9 Option error	• SAVE, SAVE# or PUT command was executed without an interface. • When the signal input with the LOAD, LOAD# or GET command is erratic and cannot be loaded. • A printer is not connected. • When the printer is not sufficiently charged. • Paper jammed in the printer.	• Connect a tape recorder. • Reduce the playback volume of the tape recorder. • Set the tone control of the tape recorder to middle position. • Change the cassette tape. • Clean the head of the tape recorder. • Charge the printer. • Remove the paper jammed in the printer.

If an error occurs, specify the WRT mode by pressing [MODE] [1] after releasing the error by [AC] key and correct by calling the error line with the LIST command.

2. **Debugging When Error Is Not Displayed**

If the desired result cannot be obtained without any error message displayed, there is a "bug" that does not become an error somewhere in the program. In this case, debugging is carried out while executing the program.

- **Debugging with the STOP Command**

Since the program execution will pause if the STOP command is written immediately after the line containing the variable to be checked, display and check the contents of the variable by entering variable to be checked followed by [EXE]. To continue executing the program, press [EXE].

- **Debugging with the TRACE Mode**

Press [MODE] [2] and "TRACE ON" will be displayed. This mode is called the TRACE mode. If a program is executed in the TRACE mode, there will be a pause after each line (after each statement when using multistatements). Press [EXE] to advance to the next execution.

The TRACE mode is used to find the bug by tracing the flow of the program.

Pressing [MODE] [3] causes "TRACE ON" to disappear and cancel the TRACE mode.

Other causes of bugs that can be considered are "errors in the variable" and "errors in the subscript of an array variable". The program should therefore be closely checked.

4-7 Convenient Peripherals

Although the computer can be conveniently used as an independent unit, optional peripherals are also available.

Character printer with cassette interface (FP-40)
Cassette interface (FA-3) (FA-5)
Character printer (FP-12S)
Interface pack (SB-42)

The cassette interfaces enable programs in the computer to be quickly stored on a cassette tape or loaded from the tape. It will also be possible to store data in variables and the DATA BANK. The character printers print out program contents, data and calculation results.

- **Connections**

*Be sure to switch off the power when connecting the computer with each peripheral.

We will now give a brief description of each function. If further details are desired, please refer to the command descriptions in Chapter 6.

- **Storing Programs and Data on a Cassette Tape**

To store programs and data on a cassette tape, connect a tape recorder to the computer by using the FP-40, FA-3 or FA-5.

<Connecting the terminals>

Each terminal is connected as follows:

MIC: MIC terminal. In the case of a stereo-recorder, it is desirable that the RIGHT terminal is used.

EAR: EAR, MONITOR or EX SP terminal. In the case of a stereo-recorder, connect to the terminal that was used for recording.

REM: Connect the REM terminal to the recorder REMOTE terminal.

*Although the three cables may be connected at the same time, some recorders produce noise when both the MIC and EAR terminals are connected. In this case, connect only the MIC terminal during recording (from the computer to tape), and connect only the EAR terminal during playback (from tape to the computer).

- **Operation of the Tape Recorder**

<With the REMOTE terminal connected>

- The SAVE or PUT command is executed with the tape recorder set to "RECORD".
- The LOAD or GET command is executed with the tape recorder set to "PLAYBACK".

<With the REMOTE terminal disconnected>

- The SAVE or PUT command is executed with the tape recorder set to "RECORD".
- The tape recorder is set to "PLAYBACK" and started after first executing the LOAD or GET command.

■ Program Storing and Loading

Sometimes the program cannot be stored because of the capacity of free area. If the previous program is erased, it cannot be used again. In cases of this nature, the cassette interface is very helpful.

Commands for storing programs on a cassette tape are "SAVE" or "SAVE ALL". "SAVE" can only store a program located in a single program area, while "SAVE ALL" can simultaneously store programs located in all program areas.

SAVE Command

[MODE] [0]

```
Ready Pn
```

→ The program located in this program area can be stored.

SAVE ALL Command

Programs located in all program areas can be stored.

The SAVE and SAVE ALL commands are manually executed.

Example:

```
SAVE [EXE]
SAVE "CASIO" [EXE]
SAVE ALL [EXE]
SAVE ALL "FX" [EXE]
```

Characters enclosed with the quotation marks (") after SAVE and SAVE ALL are file names which are placed with stored programs. These programs can be loaded later by specifying these names. Up to 8 characters can be used for a file name.

LOAD and LOAD ALL commands are used to load programs from a cassette tape to the computer. The proper use of these commands depends on whether programs were stored by SAVE or SAVE ALL.

Storing \ Loading	LOAD	LOAD "file name"	LOAD ALL	LOAD ALL "file name"
SAVE	O	X	X	X
SAVE "file name"	O	O	X	X
SAVE ALL	X	X	O	X
SAVE ALL "file name"	X	X	O	O

Items marked with "O" can be loaded; those marked with "X" cannot be loaded.
File names must be identical.

Example:

```
LOAD [EXE]
LOAD "file name" [EXE]
LOAD ALL [EXE]
LOAD ALL "file name" [EXE]
```

When programs are loaded by LOAD or LOAD ALL, a display depending on the storing format appears.

Storing format	Display
SAVE	PF:
SAVE "file name"	PF: file name
SAVE ALL	AF:
SAVE ALL "file name"	AF: file name

A program stored by a SAVE command can be loaded to any of the program areas by a LOAD command.

Example:
Stores the program of P0.

↓

Loads it to P9.

Precautions:

Sometimes a program cannot be stored or loaded smoothly. If this happens, check the following items.

- "Error 9" is displayed during storing.
  [Check point]
  Check if the computer is properly connected to the cassette interface.
- "Error 9" is displayed during loading.
  [Check points]
  If the tape is stretched, replace it with a new one.
  If the head of the tape recorder is dirty, clean it.
  Set the tone control of the tape recorder to medium.
- No error is displayed but loading is attempted without success.
  [Check points]
  If the tape recorder output volume is low, increase the volume near MAX.
  Check if the output standard of the tape recorder is in accordance with that of the cassette interface. See the owner's manual of the cassette interface for the standard.

■ Storing and Loading of Data in the DATA BANK

All the data in the DATA BANK can be stored on a cassette tape at once by using "SAVE #".

SAVE # "File name"
Up to 8 characters.

Up to 8 characters can be placed inside " " for a file name, the same as when a program is stored.

Example:

SAVE # "MEMO" [EXE]

"LOAD #" is used to load data in the DATA BANK from a tape to the computer.
The previous data are erased when new data are loaded.
However, if ",M" is specified at the end of the LOAD # command, the data from the tape are loaded following the previously stored data left intact.

LOAD# **"File name"**
Up to 8 characters.

Example:

LOAD# "MEMO" [EXE]

When data in the DATA BANK are being loaded, a display depending on the storing format appears.

Storing format	Display
SAVE #	MF:
SAVE # "file name"	MF: file name

■ Data Storing and Loading

A program always has data; it is troublesome to enter these data from the keyboard each time.
Try a method by which data in the computer are stored on tape and loaded again.
To store data on a tape, "PUT" is used.
Variables are specified in a PUT command. A file name can also be specified.

PUT **"File name"** **Variable 1, Variable 2**
Up to 8 characters.

For a file name, up to 8 characters can be placed inside " " as for program storing.
If the exclusive character variable ($) is used, specify it first. Then next two variable names are specified to determine the beginning and end of the variables to be stored.

Example:
Store the contents of the exclusive character variable ($) and 13 variables from A to M.

```
PUT $,A,M
```

Store the contents of array variable A(10) with the file name "DATA" in the DIM mode.

```
PUT "DATA"A(*)
```

Store the contents of 36 variables from A to Z (10) with the file name "CASIO" in the DEFM mode.

```
PUT "CASIO" A,Z(10)
```

*Assuming that the variables are expanded.

Since the variable names specify the beginning and end of the variables to be stored, place them in alphabetical order (e.g., "A, Z"). A specification such as "Z, A" cannot be performed.
When the variables are character variables, "A, Z" can be specified instead of "A$, Z$".
"GET" is used to load data from a tape to the computer. Variables are specified in a GET command. A file name can also be specified.

GET **"File name"** **Variable 1, Variable 2**
Up to 8 characters.

Example:
Load data to the exclusive character variable ($) and 3 variables from X to Z.

```
GET $,X,Z
```

Load the data of file name "DATA" into array variable A (15) in the DIM mode.

```
GET "DATA"A(*)
```

Load the data of file name "FX" into array variables G (0) to G (59) in the DEFM mode.

```
GET "FX" G(0),G(59)
```

*Assuming that the variables are expanded.

For details, refer to GET and PUT in Chapter 6 "Command Reference". When data is being loaded by a GET command, a display depending on the storing format appears.

Storing format	Display
PUT $, A, Z	VF:
PUT "file name" G, P	VF: file name

- **Printing of Programs, Data and Calculation Results**

An exclusive character printer (FP-40 or FP-12S) is used for printing out programs, data and calculation results on paper. See page 72 for the connection method.

Printing is carried out by key operation in the PRINT mode ("PRT ON" displayed). The PRINT mode can be specified by pressing [MODE] [7] and can be canceled by pressing [MODE] [8] .

1. Printing Program Contents

Execute the LIST command after pressing [MODE][7] in the RUN mode. If it is desired to print out the contents of all program areas from P0 to P9, enter LIST ALL [EXE] .

After completing printing, do not forget to press [MODE][8] to cancel the PRINT mode.

2. Printing Out Calculation Results

When desiring to print calculation results, specify the PRINT mode by pressing [MODE][7] or by writing "MODE 7" in the program. It will be more convenient to write "MODE 7" in the program if only certain parts are to be printed.

*When writing in the program, enter [M][O][D][E] instead of pressing the [MODE] key.

3. Printing Data in the DATA BANK

To print the memo data stored in the DATA BANK (See Chapter 7), execute the LIST# command after pressing [MODE][7] in the RUN mode.

*After printing is finished, cancel the PRINT mode by pressing [MODE][8] .

4-8 Using a PB-100 Program

Programs prepared for the PB-100, PB-300, FX-700P and FX-802P can be utilized with this computer.
This computer is provided with more commands than them; its utilization is more convenient.
The BASIC language used by this computer is almost the same as that used by the above computers.

■ Different Points

• Additional Commands

PASS (Program protection)
BEEP (Buzzer sound)
READ (Reads data from a DATA statement)
DATA (Writes data)
RESTORE (Specifies data to be read)
ON ~ GOTO (Indirect specification of a GOTO statement)
ON ~ GOSUB (Indirect specification of a GOSUB statement)
REM (Comment statement)
DIM (Array declaration)
ERASE (Clears array)
LIST V (Confirmation of array variable)
LIST# (Lists memo data in the DATA BANK)
LOAD# (Loads memo data in the DATA BANK)
SAVE# (Stores memo data in the DATA BANK)
NEW# (Erases memo data in the DATA BANK)
RESTORE# (Searches memo data in the DATA BANK)
READ# (Reads memo data in the DATA BANK)
WRITE# (Writes memo data in the DATA BANK)

• Additional Functions

DEG (Sexagesimal → decimal conversion)
DMS$ (Decimal → sexagesimal conversion)
STR$ (Converts a numeric value to a character string)
&H (Hexadecimal → decimal conversion)
HEX$ (Decimal → hexadecimal conversion)

- **Modified Commands**

This computer	PB-100/PB-300/FX-700P/FX-802P
NEW (NEW ALL)	CLEAR (CLEAR A)
CLEAR	VAC
IF ~ THEN	IF ~;
SAVE ALL	SAVE A
LOAD ALL	LOAD A
VERIFY	VER
DEFM (Can be written in a program.)	DEFM (Can only be performed manually.)

- **Modified Functions**

This computer	PB-100/PB-300/FX-700P/FX-802P
KEY$	KEY
MID$	MID

In spite of these different points, a program prepared by the PB-100/PB-300/FX-700P/FX-802P can be fundamentally utilized with this computer.
However, it is better that programs be rewritten for this computer so that it can be easily used or can be easily reconsidered later.

Example:
PB-100 program

```
10 VAC
20 FOR A=1 TO 20
30 INPUT Z(A)
40 IF Z(A)>80:B=B+1:GOTO 90
50 IF Z(A)<60:C=C+1:GOTO 90
60 IF Z(A)>40:D=D+1:GOTO 90
70 IF Z(A)>20:E=E+1:GOTO 90
80 F=F+1
90 NEXT A
    :
    :
```

This example is part of a program to enter data and distribute them according to their size. Although the program could be used as it is, correct the following items.

Change "VAC" on line 10 to "CLEAR".

```
10 CLEAR
```

Change ";" on lines 40 to 70 to "THEN".

```
40 IF Z(A)>80 THEN B=B+1:GOTO 90
      ⋮
```

Since variable expansion is necessary in this program, write the DEFM command, manually executed in the PB-100/PB-300/FX-700P/FX-802P, at the beginning.

```
5 DEFM 20
```

Example:

PB-100 program

```
10 INPUT "I=1/O=2/P=3",N
20 IF N<1 THEN 10
30 IF N>3 THEN 10
40 GOTO N*100
      ⋮
```

This program is used to determine branch destination according to the work.

To adapt it for this computer, modify it as follows by using an ON ~ GOTO statement.

```
10 INPUT "I=1/O=2/P=3",N
20 ON N GOTO 100,200,300
30 GOTO 10
      ⋮
```

The program is simplified by utilizing an ON ~ GOTO statement as mentioned above; testing the data N is deleted.

Programs and data stored on tape by CASIO's handheld computers can be loaded as they are to this computer. However, the reverse operation is not always possible. Therefore precautions shall be taken. The relationships are as follows.

This computer → PB-110, PB-220, PB-410, FX-720P, FX-820P, FX-770P, FX-785P, FX-790P

SAVE / LOAD	PF	AF	MF	With password		
				PF	AF	MF
LOAD	○			○		
LOAD ALL		○			○	
LOAD#			○			○

This computer → PB-100F, FX-710P

SAVE / LOAD	PF	AF	MF	With password		
				PF	AF	MF
LOAD	○			○		
LOAD ALL		○			○	

This computer → PB-100, PB-300, FX-700P, FX-802P

SAVE / LOAD	PF	AF	MF	With password		
				PF	AF	MF
LOAD	○					
LOAD ALL		○				

○ : Can be loaded.

/ : Cannot be loaded.

PRECAUTIONS

- When transferring a program prepared with this computer to other CASIO's computers (excluding PB-110, PB-220, PB-410, FX-720P, FX-820P, FX-770P, FX-785P and FX-790P), a READ#, WRITE# or RESTORE# command must not exist in the program. In the case of PB-100 series, use KEY and MID in place of KEY$ and MID$.
- A program created with other CASIO's computers may not be correctly executed in this computer.

  If a numeric expression is used at the IF ~ THEN branch destination, an error occurs. In this case, correct it to IF ~ THEN GOTO branch destination.
- When transferring a program prepared with this computer to other CASIO's computers (excluding FX-785P and FX-790P), a DIM or ERASE command must not exist in the program.

  Use the array variable in the DIM statement after first changing to a DEFM statement.

  Be careful of the method of determining the array variable when changing. (See page 57.)

5 Program Library

In this chapter we will deal with slightly longer comprehensive programs. The first is a sorting program in which data are rearranged. Array variables are used in an interesting manner in this program. The second is an exciting game program. Please note the usage of the various commands from the examples of these programs.

5-1 Rearrangement of Data (Sorting)

Sorting of disorderly data often becomes important in the creation of practical programs. Here we will introduce a program using a method called "bubble sort".

Consider a program to display the ranking of eight examinees from No. 101 to 108 by sorting their points supplied.

- **Program List**

```
 10 REM bubble sort
 20 INPUT "Number of data",C
 30 REM DIMENSION
 40 DIM D(3,C)
 50 REM data set
 60 FOR A=1 TO C
 70 READ D(1,A),D(2,A)
 80 NEXT A
 90 PRINT "sorting";
100 REM sort
110 FOR A=1 TO C-1
120 FOR B=A+1 TO C
130 IF D(2,A)≥D(2,B) THEN 160
140 D=D(1,A):D(1,A)=D(1,B):D(1,B)=D
150 D=D(2,A):D(2,A)=D(2,B):D(2,B)=D
160 NEXT B
170 NEXT A
180 FOR A=1 TO C:D(3,A)=A: NEXT A
190 FOR A=1 TO C-1
200 IF D(2,A)>D(2,A+1) THEN 220
210 D(3,A+1)=D(3,A)
220 NEXT A
300 REM output
310 FOR A=1 TO C
320 BEEP : PRINT
330 PRINT D(3,A); CSR5;"No";D(1,A); CS
    R15;"→";D(2,A)
340 NEXT A
350 PRINT "OVER"
```

```
 360 END
1000 DATA 101,40
1010 DATA 102,60
1020 DATA 103,80
1030 DATA 104,60
1040 DATA 105,60
1050 DATA 106,80
1060 DATA 107,20
1070 DATA 108,95
```

- **Variables List**

Variable	Role	Variable	Role
A	Loop control variables	D (2, 1)	Scores
B		⋮	
C	Number of data		
D	Used for conversion	D (2, C)	
D (1, 1)	Examinees' numbers	D (3, 1)	Ranking
⋮		⋮	
D (1, C)		D (3, C)	

- **Program Description**

Line Nos. 20 – 40 . . . . Inputs number of data and declares array.

Line Nos. 50 – 80 . . . . Loads data in line No. 1000 and after into array.

Line Nos. 100 – 170 . . . Main routine for sorting. Compares scores successively and swaps the score and the examinee's number with line number 140 and 150 if the large and small scores are reversed.

Line Nos. 180 – 220 . . . Assigns ranking to data sorted according to sequence of scores.

Correspondence of array variables will be as shown below in relation to a person of ranking A

Examinees' Nos.		Scores		Ranking
D (1 , A)	⟷	D (2 , A)	⟷	D (3 , A)

Line Nos. 300 – 360 . . . Displays sorted data in the order of ranking, examinee's No. and score.

Execution example:

Operation	Display
[R][U][N][EXE]	Number of data?
[8]	8_
[EXE]	sorting
	1 No 108 → 95
[EXE]	2 No 106 → 80
[EXE]	2 No 103 → 80
[EXE]	4 No 105 → 60
[EXE]	4 No 102 → 60
[EXE]	4 No 104 → 60
[EXE]	7 No 101 → 40
[EXE]	8 No 107 → 20
[EXE]	OVER

- Since the sorted data remain in array D (1, 1) ~ D (3, C), they can be used in a different program.

5-2 Horse Race Game

Place your bet by considering the odds on 4 horses in this horse race game. Start with $20.00 and make a fortune by picking the dark horse.

- **Program List**

```
 10 REM Horse Race
 20 CLEAR : DIM A(3,4)
 30 R=1:$="♠♥♦♣"
 40 PRINT "< Horse Race >";
 50 FOR J=1 TO 5: BEEP : BEEP 1: NEXT
    J
 60 PRINT
 70 PRINT "HORSE";
 80 FOR J=1 TO 4: PRINT J; MID$(J,1);:
     NEXT J
 90 GOSUB 1000: GOSUB 1000
100 PRINT : BEEP
110 INPUT "How many players ",P
120 IF P>5 THEN 110
130 IF P<1 THEN 110
140 PRINT "ALL PLAYERS HAVE $20";
150 GOSUB 1000
160 DIM X(2,P),Y$(P)
170 FOR J=1 TO P:X(2,J)=20: NEXT J
180 REM initialize
190 G=0
200 FOR J=1 TO 4
210 A(1,J)=0:A(2,J)= RAN#:A(3,J)=1+ IN
    T(10↑(1.2-A(2,J)))
220 NEXT J
230 PRINT : PRINT "<RACE";R;">";: GOSU
    B 1000
240 REM bet memory
250 FOR J=1 TO P
260 PRINT :X(1,J)=0:Y$(J)=""
270 IF X(2,J)=0 THEN 450
280 PRINT "PLAYER";J;" HAS $";X(2,J);
290 GOSUB 1000
300 PRINT : PRINT "RATE ";
```

```
310 FOR K=1 TO 4
320 PRINT CSRK*5; MID$(K,1);A(3,K);
330 NEXT K
340 GOSUB 1000: BEEP
350 PRINT CSR0;"P";J;"→";
360 A$= KEY$: IF A$="" THEN 360
370 IF A$<"0" THEN 360
380 IF A$>"4" THEN 360
390 N= VAL(A$): IF N=0 THEN 450
400 A$= MID$(N,1):Y$(J)=A$
410 BEEP : PRINT : PRINT "PLAYER";J;"
    ";A$;
420 INPUT " MONEY ",X(1,J)
430 IF X(2,J)<X(1,J) THEN 410
440 X(2,J)=X(2,J)-X(1,J)
450 NEXT J
460 PRINT
470 PRINT " < START! >";
480 FOR K=1 TO 10: BEEP : NEXT K
490 PRINT
500 REM main loop
510 IF G=2 THEN 600
520 FOR J=1 TO 4
530 IF G≥1 THEN 560
540 PRINT CSRA(1,J);" ";
550 IF RAN#*(0.9+A(2,J)/10)>0.7 THEN A
    (1,J)=A(1,J)+1
560 IF A(1,J)=23 THEN G=G+1
570 PRINT CSRA(1,J); MID$(J,1);
580 NEXT J
590 GOTO 500
600 REM goal
610 PRINT CSR0;"GOAL!";
620 FOR J=1 TO 7: BEEP : BEEP 1: NEXT
    J
630 GOSUB 1000
640 FOR J=1 TO 4
650 IF A(1,J)=23 THEN H=A(3,J):A$= MID
    $(J,1)
```

```
 660 NEXT J
 670 F=0
 680 FOR J=1 TO P
 690 M=0: IF X(1,J)=0 THEN 730
 700 IF Y$(J)=A$ THEN M=X(1,J)*H
 710 PRINT : BEEP
 720 PRINT "PLAYER";J;" →PRIZE $";M;
 730 X(2,J)=X(2,J)+M: GOSUB 1000
 740 PRINT : BEEP : IF X(2,J)=0 THEN F=
     F+1
 750 PRINT "PLAYER";J;" HAS $";X(2,J);:
      GOSUB 1000
 760 NEXT J
 770 PRINT : BEEP : IF F=P THEN 830
 780 PRINT "REPLAY [Y/N] ?";
 790 A$= KEY$: IF A$="" THEN 790
 800 IF A$="Y" THEN R=R+1: GOTO 180
 810 IF A$="N" THEN 830
 820 GOTO 790
 830 PRINT : PRINT "GAME OVER"
 840 END
1000 REM timer sub
1010 FOR K=1 TO 150: NEXT K
1020 RETURN
```

• Variables List

Variable	Role	Variable	Role
A$	For keys and characters	K	Loop control variable
A (1, 1)	Position of the spade (♠)	M	For calculating prize
A (1, 2)	Position of the heart (♥)	N	For horse number
A (1, 3)	Position of the diamond (♦)	P	Number of players
A (1, 4)	Position of the club (♣)	R	Race number
A (2, 1)	Random number of (♠)	X (1, 1)	Player 1's bet
A (2, 2)	Random number of (♥)	X (1, 2)	Player 2's bet
A (2, 3)	Random number of (♦)	⋮	⋮
A (2, 4)	Random number of (♣)		
A (3, 1)	Odds on (♠)	X (1, P)	Player P's bet
A (3, 2)	Odds on (♥)	X (2, 1)	Player 1's holdings
A (3, 3)	Odds on (♦)	X (2, 2)	Player 2's holdings
A (3, 4)	Odds on (♣)	⋮	⋮
F	For determining game over		
G	For determining goal	X (2, P)	Player P's holdings
H	Odds on winning horse	$	For selecting a horse's character (♠ , ♥ , ♦ or ♣)
J	Loop control variable		

- **Game Description**

One to five persons can play this game with each player starting with $20.00. There are 4 horses numbered from 1 to 4 with 1 being the spade ♠, 2 the heart ♥, 3 the diamond ♦, and 4 the club ♣. Select a horse from 1 to 4. A player selecting 0 passes a race since there is no horse numbered 0.

The odds on each horse are displayed for each race. If the horse selected by a player wins, the player receives an amount equal to the odds times his bet. If the selected horse does not win, the player loses his bet. If a player's holdings drop to 0, he must drop out of the game. Game is over when all player's lose their holdings.

Following is a description of the sequential displays and key operations in a sample game.

1) Start game.

[R][U][N][EXE] (Displays title.)	<Horse Race>
(Describes horse.)	HORSE 1♠ 2♥ 3♦ 4♣

2) Input number of players.

	How many players ?
[2] (Enters 2 for two players.)	2_
[EXE] (Initial holding.)	ALL PLAYERS HAVE $20
(First race.)	<RACE 1>

3) Input horse and bet.

(Holdings of player 1)	PLAYER 1 HAS $ 20
(Odds displayed.)	RATE ♠ 12 ♥ 3 ♦ 7 ♣ 12
	[1] [2] [3] [4] ↓ ↓ ↓ ↓
(Inputs horse of player 1.)	P 1→ ♠ 12 ♥ 3 ♦ 7 ♣ 12
[2] (Selects ♥ horse.)	PLAYER 1 ♥ MONEY ?
[1][0] (Bets $10.00)	10_
[EXE] (Holdings of player 2)	PLAYER 2 HAS $ 20
(Odds displayed.)	RATE ♠ 12 ♥ 3 ♦ 7 ♣ 12
(Inputs horse of player 2.)	P 2→ ♠ 12 ♥ 3 ♦ 7 ♣ 12
[3] (Selects ♦ horse.)	PLAYER 2 ♦ MONEY ?
[5] (Bets $5.00)	5_

4) Race starts.

[EXE]

	< START! >
(Exciting race is being taken.)	♣♠♦♥

5) Race ends.

(Placing decided. ♦ wins.)	GOAL! ♥ ♣ ♠♦
(Player 1 loses.)	PLAYER 1 →PRIZE $ 0
(Holdings of player 1)	PLAYER 1 HAS $ 10
(Player 2 wins $35.00.)	PLAYER 2 →PRIZE $ 35
(Holdings of player 2)	PLAYER 2 HAS $ 50
(Do you wish to play again?)	REPLAY [Y/N] ?

6) Press [Y] to advance to the next race and press [N] for "**GAME OVER**".

- This game is programmed so the lower the odds the easier to win and horses with high odds are difficult to win. Good luck!

6 Command Reference

The following descriptions apply symbols and terms frequently used in the syntax.

- $\left\{\begin{matrix}\times\times\times\times\\ \bigcirc\bigcirc\bigcirc\bigcirc\end{matrix}\right\}$ ..... One of the elements inside { } must be selected. The { } itself must not be written.
- [OOOO] ..... The element inside [ ] can be omitted. The [ ] itself must not be written.
- OOOO* ..... The element with * on the top right can be repeatedly used. The * itself must not be written.
- Numeric expression .... Numeric value, calculation expression, and numeric variable such as 10, 2 + 3, A, S * Q.
- Character expression ... Character constant, character variable, and character expression such as "ABC", X$, N$ + M$.
- Expression ..... General name of numeric expressions and character expressions.
- Parameter ..... An element that accompanies a command.
- Ⓟ ..... Can only be executed in a program.
- Ⓜ ..... Can only be executed manually.
- Ⓐ ..... Can be executed both manually and in a program.
- Ⓕ ..... Function instruction that can be executed both manually and in a program.

Example:

DATA [data] [, [data]]*

Since all data are provided with a bracket [ ], it will also be possible to write "DATA" only. Since ,[data] is provided with [ ]*, this element can be written repeatedly. This can therefore be written "DATA data, data, ..." If we omit the first [data], this can also be written "DATA, data, data, ..."

GOTO { **Line No.** / **# program area No.** }

There are two different ways to write this statement as shown below.

1) GOTO line No.

2) GOTO # program area No.

NEW [ALL] Ⓜ

Function:

Program erase. Erases programs and variables.

Parameter:

When ALL is specified, all P0 ~ P9 programs and variables are erased.

Explanation:

1) If ALL is not specified, the program in the presently specified program area is erased. Clearing the contents of variables and canceling of the expanded variables are not performed.
2) If ALL is specified, the programs in all program areas and variables are erased. The number of variables specified by DEFM will be initialized to 26 and the computer will be in the DIM mode.
3) Cannot be executed while a password is specified.
4) Cannot be used in a program.
5) Can only be executed in the WRT mode.

*NEW ALL can be abbreviated as NEW A.

Example:

[MODE] [1] NEW [EXE]

RUN [Execution Start Line]

line No.

Ⓜ

Function:

Program execution.

Parameter:

Starting line number: $1 \leq$ line number < 10000

Explanation:

1) Executes a program from a specified line (when the line number is omitted, execution starts from the beginning of the program).
2) When a specified line number does not exist, execution starts from the line with the closest larger number.
3) Variables are not cleared.

Example:

```
10 PRINT "LINE 10"
20 PRINT "LINE 20"
30 END
```

RUN EXE	LINE 10
RUN 20 EXE	LINE 20

LIST [{ line No. / ALL / V }]

Ⓜ

Function:

Displays program contents or the variable names and subscripts of the declared arrays.

Parameter:

Line No.:	No. of the first line to be displayed.
ALL:	Displays the contents of all P0 ~ P9 programs sequentially.
V:	Displays variable names and subscripts of the declared arrays.

Explanation:

1. RUN Mode

1) Sequentially displays the content of a program from a line number if it is specified, or from the beginning if it is omitted.
2) Since the content of a program is automatically displayed sequentially, press the [STOP] key to stop this. Press the [EXE] key to display the next line and after.
3) Array variable names are displayed in sequence one at a time. Press [EXE] to display the next array variable name.
4) In the PRINT mode (when "PRT ON" is displayed), the display is not stopped but is made sequentially at high speed.

2. WRT Mode

1) Displays the content of a program from a line number if it is specified, and from the beginning if it is omitted.
2) Since each line is displayed for edit in the WRT mode, if edit is not required, press the [EXE] key to advance to the next line. Also, if the [SHIFT] key is pressed before the [EXE] key, the previous line is displayed.
3) The array variable name is displayed one at a time with LIST V. Press [EXE] and the next array variable name will be displayed.

- When ALL is specified, the contents of all P0 ~ P9 programs are sequentially displayed. In this case they are sequentially advanced even in the WRT mode, so edit cannot be performed.
- This command cannot be used while a password is specified.

*LIST ALL can be abbreviated as LIST A.

Example:

```
LIST [EXE]
LIST 30 [EXE]
LIST V [EXE]
```

PASS "Password" Character string

Ⓜ

Function:

Specifies or cancels a password.

Parameter:

Password: $1 \leqq$ character string $\leqq 8$.

Explanation:

1) If this command is executed when a password is not specified, a password is specified for all program areas (P0 ~ P9).
2) If this command is executed while a password is specified, this password is canceled only when entering the corresponding password. When passwords do not correspond, a protect error (Error 8) occurs.
3) A password consists of a 1 ~ 8 character string in which spaces, alphabetical characters, numerals, special symbols, etc. can be used. However, (") cannot be used.
4) While a password is specified, commands such as LIST, LIST ALL, LIST #, LIST V, NEW, NEW ALL, and NEW# cannot be used. Also no writing (WRT mode) can be made; it is attempted, an error (Error 8) occurs.
5) Cannot be used in a program.
6) A password can be maintained while the power switch is off.
7) If a program is stored on a cassette tape by a SAVE or SAVE ALL command while a password is specified, this password is also stored. When a program with a password attached is loaded from a cassette tape by a LOAD or LOAD ALL command, the password is also loaded. Also, when a currently specified password in the mainframe and the password of a program loaded from a cassette tape are different, the program cannot be loaded from a cassette tape (Error 8).

Precaution:

If the password is forgotten after specifying, press the ALL RESET button on the back panel and cancel the password. Note that this operation causes all programs and variable contents to be cleared.

Example:

PASS "CASIO" [EXE]

* The same procedure is used for specifying and also cancelling the password.

SAVE [ALL] ["File name"] Character string Ⓜ

Function:

Stores a program on a cassette tape.

Parameter:

ALL:	Stores the programs in all program areas.
File name:	1 ≦ character string ≦ 8. Can be omitted.

Explanation:

1) When ALL is omitted, the content in the presently specified program area is stored.
2) When ALL is used, the contents of all P0 ~ P9 program areas are stored.
3) When a password is specified, the storing is performed with that password. Therefore, the password is the same as that stored when the program is loaded by the LOAD command.

*SAVE ALL can be abbreviated as SAVE A.

Example:

```
SAVE [EXE]
SAVE "CASIO"[EXE]
SAVE ALL "FX"[EXE]
```

LOAD [ALL] ["File name"] Character string Ⓜ

Function:

Loads a program from a cassette tape.

Parameter:

ALL:	Loads the programs in all program areas.
File name:	1 ≦ character string ≦ 8. Can be omitted.

Explanation:

1) When ALL is omitted, a program stored by "SAVE" is read into the presently specified program area.
2) When ALL is used, programs stored by "SAVE ALL" are read into the P0 ~ P9 program areas.
3) With "LOAD ALL", the stored programs in the computer will be erased and a new program will be loaded from the cassette tape.
4) When a file name is specified, a program with the same file name will be searched and loaded from the cassette tape. If the file name is omitted, the first program found on the cassette tape will be loaded.
5) If a program with a password is loaded, that password will also be loaded.

*LOAD ALL can be abbreviated as LOAD A.

SAVE and LOAD Relationship

	LOAD	LOAD "File name"	LOAD ALL	LOAD ALL "File name"
SAVE	○	×	×	×
SAVE "File name"	○	○	×	×
SAVE ALL	×	×	○	×
SAVE ALL "File name"	×	×	○	○

* File names are assumed identical. ○ . . . Can be loaded.
× . . . Cannot be loaded.

VERIFY

["File name"]
Character string

Ⓜ

Function:

Checks the status of a program and data stored on a cassette tape.

Parameter:

File name: 1 ≦ character string ≦ 8. Can be omitted.

Explanation:

1) When a file name is specified, the file with this name is checked.
2) When the file name is omitted, checks the first file that appears on the cassette tape.
3) The parity check system is used to check a storing format.

Example:

```
VERIFY EXE
VERIFY "PROG1" EXE
```

CLEAR

Ⓐ

Function:

Clears all variables including array variables.

Explanation:

1) Clears all variables; all numeric variables are cleared to 0 and all character variables to a null.
2) This command can be used both in a program and manually.
3) Since control variables are also cleared in a FOR ~ NEXT loop (see page 119), an error (Error 7) occurs during NEXT statement execution.

*The CLEAR command functions the same as VAC.

END

Ⓟ

Function:
Terminates program execution.

Explanation:
Since program execution is terminated, the next program is not executed even if it exists.

STOP

Ⓟ

Function:
Temporarily suspends program execution.

Explanation:
1) Temporarily suspends program execution and displays "STOP" after which input waiting occurs.
2) After suspension, execution is resumed by pressing the [EXE] key.
3) If the [STOP] key is pressed while execution is stopped by a STOP statement, the program area number and line number are displayed.
4) Calculations can be performed by manual operation when execution is suspended by STOP.

[LET] { Numeric variable = numeric expression / Character variable = character expression } Ⓟ

Function:

Assigns the value on the right side of the equal (=) sign to the variable on the left.

Explanation:

1) A numeric expression corresponds to a numeric variable, and a character expression corresponds to a character variable.
2) LET can be omitted.

Example:

```
10 LET X=12
20 Y=X↑2+2*X-1 ............ LET can be omitted.
30 PRINT Y
40 A$="CASIO" ............... LET can be omitted.
50 LET B$=A$+"FX"
60 PRINT B$
70 END
```

REM

Comment
Character string

Ⓟ

Function:

Statement that expresses a comment.

Explanation:

1) Written in a program. Content after REM in one line is treated as comment statement and is therefore not executed.

```
10 REM TEST:A=50
```
Not executed.

2) When a command to be executed is written on the same line, write a multi-statement sign (:) before the REM statement.

Example:

```
10 REM AREA
20 INPUT "R=",R
30 S=π*R↑2:REM CALCULATION
40 PRINT S
50 END
```

Determines program name.

Adds comment to the line.

INPUT

[“Message statement”,] variable name (P)
Character string

[, [“Message statement”,] Variable name]*
Character string

Function:

Assigns input from the keyboard to a variable.

Parameter:

Message:	Character string.
Variable name:	Numeric variable name or character variable name.

Explanation:

1) Input from the keyboard is assigned to a variable when the [EXE] key is pressed.
2) If there is a message, “?” will be displayed after the message and the computer will be in key input wait status.
3) If the message is omitted, “?” only will be displayed.
4) In the key input wait status with the INPUT statement, the message and “?” will be cleared with the first key input.
5) Error (Error 2) will occur if character string data is input to a numeric variable. Since “?” will be displayed again and a numeric value input will be requested when the [AC] key is pressed, make a new input. If [MODE] [⊘] is pressed, program execution will be terminated.
6) A numeric expression can be used as the input to a numeric variable and the value of the numeric expression will be assigned to the numeric variable. In this case, even one alphabetical letter will be regarded as a numeric expression (variable).
7) If the [EXE] key only is pressed without inputting the numeric value or numeric expression, the value in the numeric variable will not change.
8) If the [EXE] key only is pressed without inputting a character expression when inputting to a character variable, null will be assigned to the character variable.

9) If a character string data exceeding seven characters is input to a character variable, the first seven characters will be significant and the eighth character and after will be disregarded.

Example:

```
10 INPUT A
20 INPUT "NAME=",B$
30 INPUT "C$=",C$,"D$=",D$
40 INPUT "FORM=",$
     ⋮
```

KEY$

Ⓟ

Function:

Enters one character from the keyboard.

Explanation:

1) Accepts the input of only one character from the keyboard.
2) Numerals, alphabetical letters and symbols can be input.
3) The data entered will be in single character form.
4) Null will occur when there is no key input.
5) Since "?" is not displayed and input waiting also does not occur, KEY$ is usually used in combination with an IF statement.

*KEY$ can be abbreviated as KEY.

Example:

```
10 PRINT"BEEP";
20 A$=KEY$
30 IF A$="0"THEN BEEP 0
40 IF A$="1"THEN BEEP 1
50 IF A$="E"THEN 70
60 GOTO 20
70 PRINT:PRINT"END"
80 END
```

Repeated until the [E] key is pressed.

*A low sound will be generated when the [0] key is pressed and a high sound will be generated when the [1] key is pressed. "END" will be displayed and the program will be terminated when the [E] key is pressed.

PRINT [Output element] [{ , ; } [Output element]]* Ⓟ

Function:
Displays an output element.

Parameter:

Output element:	Output control function (CSR), numeric expression, character expression.

Explanation:

1) Displays an output element. When an output control function is added, the element is displayed at the location determined by this function.
2) Values are displayed for numeric expressions and character expressions.
3) When an output element is a numeric expression, a position for sign (+, −) is placed before the value. However, the + sign is displayed as a blank.
   - **Character display** . . . . . ␣␣␣·····␣ (Output element)
   - **Numeral display** . . . . . ␣␣␣␣·····␣ (Sign, Output element)
4) When an output element is a numeric expression and the mantissa is more than 10 digits, the 11th digit is rounded off. When an exponent exists besides the mantissa, an exponent sign (E) and a two digit exponent are displayed.
5) "," and ";" can be used as punctuation between output elements. When "," is used, the execution stops (STOP is displayed) after the first output element is displayed, then the next output element is displayed by pressing the [EXE] key. When ";" is used, the next output element is displayed continuously after the first one.
6) When no output element is specified (only PRINT is written), the display is cleared and is not stopped.
7) The display is not stopped during printing in the print mode ([MODE] [7]).
8) The format for displaying numeric values can be specified by a SET statement.

Example:

```
10 PRINT 1/3
20 PRINT "A=";A
30 PRINT "SIN 30",SIN 30
40 PRINT "END";
50 PRINT
60 END
```

CSR — Output location specification

Numeric expression

Ⓕ

Function:

Displays an output element from a specified location.

Parameter:

Output location specification: Numeric expression. Values below decimal point are discarded.

$0 \leqq$ specification < 24

Explanation:

1) Used in a PRINT statement to specify the location of an output element.
2) The output location of the left end is 0.

Example:

```
10 FOR L=0 TO 23
20 PRINT CSRL;"A";CSR23-L;"B"
30 NEXT L
40 END
```

*Characters A and B are shifted from the left and right respectively each time the EXE key is pressed.

GOTO

{ **Branched line No.** / line No. / # **program area No.** / Number 0 to 9 }

Ⓟ

Function:

Unconditionally branches to a specified location.

Parameter:

Line No.:	1 to 9999.
Program area No.:	0 to 9.

Explanation:

1) Branches to a specified location.
2) When a branched location is a line number, branches to the specified line in the current program area and executes the program. When the branched line number does not exist, an error (Error 4) occurs.
3) When the branched location is a program area number, branches to the specified program area and executes the program from the beginning.

*A numeric expression can be used for the branched line number and the program area number.

Example:

```
 10 PRINT "START";
 20 GOTO 100
 30 PRINT "LINE 30"
 40 END
100 BEEP 0:BEEP 1
110 PRINT
120 PRINT "LINE 120"
130 GOTO 30
```

ON Branch condition / Numeric expression **GOTO** [Branched location] [, [Branched location]]* Ⓟ

★ Branched location { Line No. / # program area No.

Function:

Branches to a specified location according to the branch condition.

Parameter:

Branch condition:	Numeric expression. Values below the decimal point are discarded.
Line No.:	1 to 9999.
Program area No.:	0 to 9.

Explanation:

1) Branches according to the integer part of the value in a branch condition expression. Branched locations are allocated sequentially according to

   ON A GOTO $\underset{A=1}{\underline{100}}$, $\underset{A=2}{\underline{200}}$, $\underset{A=3}{\underline{300}}$, ·····

2) When the value of the expression is smaller than 1, or when an appropriate branched location does not exist, the next statement is executed without branching.
3) As many branched locations that can fit on one line can be written.

Example:

```
 10 INPUT "A=",A
 20 ON A GOTO 100,200,300
 30 PRINT "OTHER"
 40 GOTO 10
100 PRINT "LINE 100":GOTO 10
200 PRINT "LINE 200":GOTO 10
300 PRINT "LINE 300":GOTO 10
```

*When 1 ~ 3 is entered, branchings to 100 ~ 300 are performed respectively, otherwise "OTHER" is displayed.

IF Conditional expression / Comparison expression **THEN** { Statement [: statement]* / Branched location } Ⓟ

★ Branched location { Line No. / # program area No.

Function:

When a conditional expression is true, the statements after THEN are executed. Also, when a statement after THEN is a branched location, branching is performed.

Parameter:

Conditional expression:	Used in the form of a comparison expression.
Line No.:	1 to 9999.
Program area No.:	0 to 9.

Explanation:

1) When the conditional expression is true, the statements after THEN are executed or branching is performed.
2) When the conditional expression is false, the next line is executed.
3) The related operators used in a conditional expression:
   - = The item on the left is equal to the item on the right.
   - ≠ The item on the left is not equal to the item on the right.
   - < The item on the right is larger than that on the left.
   - > The item on the right is smaller than that on the left.
   - ≦ The item on the right is larger than or equal to that on the left.
   - ≧ The item on the right is smaller than or equal to that on the left.
4) When two or more conditional expressions exist, several IF ~ THEN statements can be written sequentially.

   IF ~ THEN IF ~ THEN .....

5) Since the statements after THEN in one line will be executed only after all conditional expressions are true, statements to be executed regardless of the conditional expression must be written as a multistatement before IF or on a separate line.

*When a statement exists after THEN, ";" can be used instead of THEN.

Example:

```
10 N=12
20 PRINT CSR N;"@";
30 K$=KEY$
40 IF K$="4"THEN N=N-1:IF N<0 THEN N=0
50 IF K$="6"THEN N=N+1:IF N>23 THEN N=23
60 PRINT
70 GOTO 20
```

*"@" is shifted to the left when the [4] key is pressed and is shifted to the right when the [6] is pressed.

FOR Control variable name = Initial value (Numeric expression) **TO** Final value (Numeric expression) Ⓟ
[**STEP** Increment (Numeric expression)] **NEXT** Control variable name

Function:

Repeats process contained between FOR and NEXT statements a number of times specified by the control variable. The value of this variable is changed, from the initial to the final one, by the increment for each repetition of the process.

Parameter:

Control variable name:	Simple numeric type variable name. An array variable cannot be used.
Initial value:	Numeric expression
Final value:	Numeric expression
Increment:	Numeric expression The value 1 is taken in default of this.

Explanation:

1) Repeats process contained between FOR and NEXT statements a number of times specified by the control variable. The value of this variable is changed, from the initial to the final one, by the increment for each repetition of the process. When the value of the control variable exceeds the final value, repetition is terminated.
2) When the initial value is larger than the final value, the execution between FOR ~ NEXT is performed only once.
3) When the increment is positive, the value of the control variable increases by each increment. When the increment is 0, execution will be repeated endlessly. When the increment is negative, the value of the control variable decreases by each increment. If increment is omitted, one will be specified.
4) A NEXT statement must always correspond to a FOR statement and must be written after it.
5) FOR ~ NEXT loops can have the following nested structure.

```
10 FOR I=1 TO 10
20 FOR J=11 TO 20
30 PRINT I;":";J
40 NEXT J
50 NEXT I
60 END
```

6) Nesting can be performed with up to 4 levels.
7) When a FOR ~ NEXT loop is terminated, the value of the control variable exceeds the final value by the value of the increment.
8) A branching out of a FOR ~ NEXT loop can be performed. If branching inside a FOR ~ NEXT loop by an IF statement or GOTO statement is attempted, an error occurs.

GOSUB

Branched line No. / Line No. / # program area No. / Number 0 to 9

Ⓟ

Function:

Performs a branching to a specified subroutine.

Parameter:

Line No.:	1 to 9999.
Program area No.:	0 to 9.

Explanation:

1) Performs a branching to a subroutine. A return from this subroutine is performed by executing RETURN.
2) To make a subroutine inside a subroutine is called nesting which can be performed with up to 8 levels.
3) Return to the statement next to the GOSUB statement is performed by RETURN.
4) Return to the main routine cannot be performed by an IF statement or GOTO statement. Therefore, be sure to perform return by a RETURN statement.
5) When the branched line No. does not exist, an error (Error 4) occurs.

*A numeric expression can also be used for a branched line number and a program area number.

Example:

```
 10 PRINT "MAIN 10"
 20 GOSUB 100
 30 PRINT "MAIN 30"
 40 END
100 PRINT "SUB 100"
110 GOSUB 200
120 RETURN
200 PRINT "SUB 200"
210 RETURN
```

RETURN

Ⓟ

Function:

Provides a return from the subroutine to the main program.

Explanation:

Returns to a statement located just after the statement which called the subroutine.

ON Branch condition / Numeric expression **GOSUB** [Branched location] [, [Branched location]]* Ⓟ

★ Branched location { Line No. / # program area No.

Function:

Branches to a subroutine according to a branch condition.

Parameter:

Branching condition:	Numeric expression. Values below the decimal point are discarded.
Line No.:	1 to 9999.
Program area No.:	0 to 9.

Explanation:

1) Performs a subroutine branching by the integer part of the value in a branch condition expression. Branched locations are allocated sequentially according to the value of the expression.

   ON B GOSUB $\underset{B=1}{\underline{1000}}$, $\underset{B=2}{\underline{2000}}$, $\underset{B=3}{\underline{3000}}$……

2) When the value of the expression is smaller than 1 or an appropriate branching location does not exist, the next statement is executed without branching.
3) As many branching locations as can fit in one line can be written.

Example:

```
 10 INPUT A
 20 ON A GOSUB 100,200,300
 30 GOTO 10
100 PRINT "SUB 100":RETURN
200 PRINT "SUB 200":RETURN
300 PRINT "SUB 300":RETURN
```

*When 1 ~ 3 is entered, a branching to the corresponding subroutine occurs.

DATA

[data] [, [data]]*
Constant Constant

Ⓟ

Function:

Stores data.

Parameter:

Data: Character constant or numeric constant.

Explanation:

1) Used to write data that is read by a READ statement.
2) Plural data can be written by punctuation with ",".
3) If only a DATA statement is executed without a READ statement, no function is performed.
4) When a character constant includes ",", place it inside " ".

```
DATA ABC, DEF, "GHI,JKL", ……
     1st  2nd   3rd
```

5) When data is omitted, a character string with a length of 0 is taken by default.

```
DATA A, ,B  → DATA A,"",B
DATA ,      → DATA "","" ──┐ Null string
DATA        → DATA ""    ──┘
```

6) Even if a command is added to a DATA statement with a colon (:), it will be regarded as data and will not be executed.

```
DATA 12,34,56:INPUT A
           └──────────┘
           (regarded as data)
```

A space following data is not disregarded. Therefore, an error will be generated if a space is inserted after numerical data.

Example: DATA 1, 2␣, 3, 4␣ These spaces are not disregarded.

READ Variable name [, [variable name]]* Ⓟ

Function:

Reads the content of a DATA statement.

Parameter:

Variable name:	Numeric variable or character variable. An array variable can be used.

Explanation:

1) Allocates data in the currently specified DATA statement sequentially to a specified variable.
2) Only numeric type data can be read for a numeric variable.
3) Data in DATA statements are read sequentially with the smallest line number first, and sequentially from the beginning in a statement.
4) After the necessary data are read by a READ statement, the following data are read by the next READ statement.
5) The first data in the program area where a READ statement exists is read by the first execution of this statement after which data in the program area at that time are read sequentially.
6) The specification of data to be read can be changed by a RESTORE statement.
7) When the number of data in a DATA statement is smaller than the number of variables in a READ statement, an error (Error 4) occurs.
8) When a space exists at the beginning of data, it is skipped.

Example:

```
10 DATA 1,2,3
20 READ A,B
30 PRINT A;B
40 DATA 4,5
50 READ C,D,E
60 PRINT C;D;E
70 END
```

*Reads data sequentially from a DATA statement and displays them.

RESTORE [Line No.] Numeric expression Ⓟ

Function:

Specifies the location of data to be read by a READ statement.

Parameter:

Line No.: Numeric expression. Values below the decimal point are discarded.
$1 \leq$ line No. ≤ 9999

Explanation:

1) Specifies a DATA statement where data to be read by a READ statement exist.
2) When a line number is omitted, the data specification is cancelled. After this, the first data in the program area where a READ statement exists are specified and read by the first READ statement that is executed.
3) When a line number of the program area is specified by a RESTORE statement, data of the DATA statement with this line number are read sequentially by the READ statement.
4) When a specified line number does not exist or a DATA statement does not exist on a specified line number and after, an error (Error 4) occurs.

Example:

```
10 DATA 1,2,3
20 DATA 4,5
30 READ A,B,C,D,E
40 RESTORE 10
50 READ F,G
60 RESTORE 20
70 READ H,I
80 PRINT A;B;C;D;E;F;G;H;I
90 END
```

PUT

["File name"] variable 1 [, Variable 2]*
Character string

Ⓐ

Function:

Stores data on a cassette tape.

Parameter:

File name:	1 ≦ Number of characters of character string ≦ 8. Can be omitted.
Variable 1, variable 2:	Specification of the variable to be stored. Array variables also possible.

Explanation:

1) Stores the contents of variables on a cassette tape.
2) Variable specifications are written as follows.

PUT A . . . . . Content of variable A.
PUT A, Z . . . . . Contents of the 26 variables from A to Z.
PUT A(0), A(100) . . . . . Contents of the 101 variables from A(0) to A(100).
PUT $, D, W . . . . . Contents of the exclusive character variable $ and 20 variables from D to W.

When the content of the exclusive character variable $ must be stored, write $ first.

3) Write as follows in case of array variables defined in the DIM mode.

PUT A(5), A(9) . . . . . Contents of A(5), A(6), A(7), A(8), A(9)

PUT A(0,0,1), A(1,0,0) . . . Contents of A(0,0,1), A(0,0,2), A(0,1,0), A(0,1,1), A(0,1,2), A(0,2,0), A(0,2,1), A(0,2,2) and A(1,0,0) when DIM A(2,2,2) is defined.

PUT A$(*) . . . . . Stores all array contents of A$. Will be PUT A$(*) regardless of whether the array is two-dimensional or three-dimensional. Two or more array variables cannot be stored with one PUT statement.

*Write as follows in case of array variables defined in the DEFM mode.

PUT A, A(5) . . . . . Contents of the six variables A ~ A(5). This is the same as PUT A, F.

PUT Z, Z(20) . . . . . Contents of the 21 variables Z ~ Z(20).

4) Can be executed both manually and in a program.

5) $ need not be attached even when using a character variable.

Example:

Enter PUT A,D when A and C are numeric variables and B and D are character variables.

```
PUT  A,D
```

GET

["File name"] variable 1 [, Variable 2]*
Character string

Ⓐ

Function:

Loads data stored on a cassette tape into a variable.

Parameter:

File name:	1 ≦ Number of characters of character string ≦ 8. Can be omitted.
Variable 1, variable 2:	Specification of the variable to be loaded. Array variables also possible.

Explanation:

1) Loads data stored on a cassette tape into a specified variable.
2) Variable specifications are written as follows.

GET A . . . . . Loads in variable A.

GET A, Z . . . . . Loads in variables from A to Z.

GET A(0), A(100) . . . . . Loads in variables from A(0) to A(100).

GET $, D, W . . . . . Loads in the exclusive character variable $, and in variables from D to W.

3) Write as follows in case of array variables defined in the DIM mode.

GET A(5), A(9) . . . . . Loads into A(5), A(6), A(7), A(8), A(9).

GET A(0,0,1), A(1,0,0) . . . Loads into A(0,0,1), A(0,0,2), A(0,1,0), A(0,1,1), A(0,1,2), A(0,2,0), A(0,2,1), A(0,2,2), A(1,0,0) when DIM A(2,2,2) is defined.

GET A$(*) . . . . . Loads all array A$ data stored with PUT A$(*).

When the number of elements in GET is greater than in PUT, only the PUT data will be loaded. Conversely, when GET elements are fewer all the GET elements will be loaded.

*Descriptions such as A$(*) cannot be used in the DEFM mode.

*Write as follows in case of array variables defined in the DEFM mode.

GET A, A(5) . . . . . Loads into the six variables A ~ A(5) same as in the case of GET A,F.

GET Z, Z(20) . . . . . Loads into the 21 variables Z ~ Z(20).

4) A variable name stored by PUT can be different from the name read by GET.
5) When the number of stored data is smaller than the number of variables to be loaded, only the data are loaded sequentially in the variables from the specified first variable.
6) When a file name is specified, data with the same file name are loaded from the cassette tape. When the file name has been omitted, data will be loaded from the first data found on a cassette tape.
7) This can be executed both manually and in a program.
8) GET will be executed in the DIM mode without distinguishing between character and numeric variables. Error will therefore occur when executing an array if data stored (PUT) as a character array is loaded (GET) into a numeric array. If data stored (PUT) as a numeric array is loaded (GET) into a character array, the contents of the character array will become null.

< Example 1 >

```
PUT  A$(*)
GET  A(*)
       ↓
PRINT  A(0)←Error 6 will occur in this line (when executing an array).
```

< Example 2 >

```
PUT  A(*)
GET  A$(*)
     ↓
PRINT  A$(0)←Null will be displayed.
```

BEEP [{0 / 1}] Ⓐ

Function:

Generates a beep sound.

Parameter:

0: Low sound

1: High sound

0 is taken by default.

Explanation:

1) Generates a high or low beep sound.

2) Can be executed both manually and in a program.

Example:

```
10 BEEP:INPUT"N=",N
20 FOR B=1 TO N
30 BEEP 0:BEEP 1:PRINT:PRINT B;
40 NEXT B
50 PRINT:GOTO 10
```

* Beep sounds are generated number of times specified.

DEFM

[Number of variables to be added]
Numeric expression

Ⓐ

Function:

Changes from DIM mode to DEFM mode.

Provides variable expansion.

Parameter:

Number of added variables: Numeric expression. Values below the decimal point are discarded. Can be omitted.

$0 \leqq$ Number of added variables $\leqq 940$

$0 \leqq$ Number of added variables $\leqq 1964$

(When RP-8 RAM expansion pack is loaded.)

Explanation:

1) Expands the number of variables.
   The arrays defined in the DIM mode will be cleared at this time.
2) Number of added variables can be specified according to the number of remaining bytes.
3) The free area for programs and DATA BANK decreases eight bytes for each variable expansion.
4) Added variables are used as array variables.
5) When the number of added variables is omitted after DEFM, the number of currently specified variables is displayed.
6) Can be executed both manually and in a program. When executed manually, the newly specified status (number of added variables + 26 basic variables) is displayed. When executed in a program, the newly specified status is not displayed.
7) An error (Error 1) will occur if an attempt is made to expand the variables beyond the number of remaining bytes in the free area.
8) Specification of expansion will be retained even if the power is turned off. Execute DEFM 0 to cancel the variable expansion and return to the 26 basic variables.

Example:

```
DEFM 10 EXE        A··Z:26 DEFM:10
DEFM EXE           A··Z:26 DEFM:10
```

```
 10 DEFM 10:CLEAR
 20 FOR J=1 TO 10
 30 PRINT"Z(";J;")=";
 40 INPUT Z(J)
 50 NEXT J
 60 FOR J=1 TO 10
 70 S=S+Z(J)
 80 NEXT J
 90 BEEP:PRINT"Sum=";S
100 DEFM 0
110 END
```

*Obtains total of the 10 numeric values input.

DIM [Array name] [, Array name]*

Ⓐ

Function:

Declares an array.

Parameter:

One-dimensional to three-dimensional arrays can be specified.

1) Array name (i) when one-dimensional
2) Array name (i, j) when two-dimensional
3) Array name (i, j, k) when three-dimensional
4) Changes from DEFM mode to DIM mode if the array name is defaulted.
   Array names i, j, k may be specified by the numeric expression $0 \leqq i, j, k \leqq 255$. Discards decimal values.
   Upper case alphabetical letters may be used for the array name.

Explanation:

1) Declares array variable names.
2) Character arrays can be declared by attaching a $ sign immediately after the array variable name. The storable character length of a character array is a maximum of seven characters. If an attempt is made to assign a character string with more than seven characters, all beyond seven characters will be disregarded.
3) The same array name can be used for a numeric array name and a character array name.
4) Up to eight arrays can be declared. Error 1 will occur if nine or more are declared.
5) Error 5 will occur if arrays are declared with the same array variable name and the different size (i, j, k) of the subscript. When array variables of the same size are declared, contents of all array variables will become 0 or null.

Example

```
10 DIM A(10)
20 A(3)=7
30 DIM A(10)
40 PRINT A(3)  ← 0 will be displayed.
```

6) Error 6 will occur if an attempt is made to use an undeclared array variable in the DIM mode.
7) Error 1 will occur when memory is insufficient.
8) Multiple array variables can be declared at one time by punctuating with commas.

Example

```
DIM A(3),A$(5),B(20,3)
```

9) Contents of all array variables will be initialized to 0 or null string by executing the DIM statement.
10) Array declarations can be cancelled with CLEAR, ERASE, NEW ALL or DEFM.

*If a DIM statement is executed when in the DEFM mode, the DEFM mode will be cancelled and all expanded variables will be cleared.

11) Eight bytes of memory will be required for each array declared with the DIM statement.

ERASE Variable name [, variable name]* Ⓐ

Function:
Deletes array variables.

Parameter:
Specifies array variable name.

Explanation:
1) Deletes array variables confirmable with LIST V.
2) Specification of array variable names are variable names in one upper case alphabetical character.

Example
ERASE A, A$, B will be specified when the variable name displayed by LIST V is **A(1), A$(1, 1, 1), B(1).**
3) Nothing will occur if a nonregistered variable name is specified.
4) If memory capacity becomes low due to array declarations, it may be increased by executing the ERASE command.

MODE Numeric expression Ⓟ

Function:

Sets the state of the computer.

Parameter:

Numeric expression: Values below the decimal point are discarded.
$4 \leq$ numeric expression < 9

Explanation:

1) Sets the angle unit and PRINT mode or releases these modes depending on the numeric expression used.
2) Settings are as follows.
   MODE 4 . . . . . Sets the angle unit to degrees.
   MODE 5 . . . . . Sets the angle unit to radians.
   MODE 6 . . . . . Sets the angle unit to grads.
   MODE 7 . . . . . Displays "PRT ON" and sets the PRINT mode.
   MODE 8 . . . . . Releases the PRINT mode.
3) Same setting as by the [MODE] key. However, the RUN mode and WRT mode cannot be set using this command. Also, input cannot be performed with the [MODE] key, but by pressing the [M][O][D][E] keys.

Example:

```
10 MODE 4
20 A=SIN 30:PRINT"A=";A
30 MODE 5
40 B=COS(π/6):PRINT"B=";B
50 END
```

*You can find the "DEG" symbol on the display changes to the "RAD" symbol during program execution.

STAT CLEAR

Ⓐ

Function:
Initializes basic statistics.

Explanation:
1) Clears the contents of memories used for statistical calculations: n (number of data), Σx (sum of data x), Σy (sum of data y), Σx^2 (sum of the square of data x), Σy^2 (sum of the square of data y) and Σxy (sum of the product of data x and y).
2) Initializes basic statistics to start a new calculation. Always execute this command when performing new statistical calculations.

Example:

STAT CLEAR [EXE]

STAT

value of data X [[, value of data Y] [; frequency]]

Ⓐ

Function:
Inputs statistical data and the frequency of the data.

Parameter:

Value of data x:	numeric expression
Value of data y:	numeric expression
Frequency:	numeric expression

Explanation:

1) Inputs statistical data and frequency to the statistics memory.
2) If the value of data x is omitted, the previous value will be used as x.
3) If the value of data y is omitted, the previous value will be used as y.
4) If frequency is omitted, one will be considered the frequency.

STAT LIST [{0, 1, 2}] Ⓐ

Function:

Outputs amount of statistics.

Parameter:

0 . . . . Outputs all statistics.
1 . . . . Outputs statistics for one variable.
2 . . . . Outputs statistics for paired variables.
When omitted . . . Outputs all statistics. (Same as STAT LIST 0.)

Explanation:

1) Outputs statistics calculated based on the input data STAT LIST, STAT LIST 0, STAT LIST 1 or STAT LIST 2.
2) Refer to pages 38 to 39 relative to the statistics to be output and the calculating formulas.

Example:

```
   10 STAT CLEAR:CLEAR
   20 FOR N=1 TO 5
   30 READ X,Y
   40 STAT X,Y
   50 NEXT N
   60 STAT LIST
   70 END
 1000 DATA 3,5,4,9,2,1,6,4,6,9
```

SET $\begin{Bmatrix} Fn \\ En \\ N \end{Bmatrix}$ Ⓐ

★ *n* is an integer from 0 to 9.

Function:

Specifies the output format for numeric data.

Parameter:

F*n*: Specifies the number of decimal places.

E*n*: Specifies the number of significant digits.

N: Releases a specification.

Explanation:

1) Specifies the number of decimal places or significant digits.
2) For specifying the number of decimal places (F*n*), a value from 0 to 9 is used.
3) For specifying the number of significant digits (E*n*), a value from 0 to 9 is used. Also "SET E0" indicates a 10-digit specification.
4) Both specifications are released by "SET N".
5) After executing this command, the specified digits are given by rounding off.
6) The number of digits specified by this command is for the display only and a 12-digit mantissa can remain in the computer.
7) This can be executed both manually and in a program.

Example:

```
10 X=10*SQR2
20 SET F3:PRINT X
30 SET E3:PRINT X
40 SET N:PRINT X
50 END
```

LEN (Simple character variable)

Ⓕ

Function:

Obtains the length of the character string in a simple character variable.

Parameter:

Simple character variable: An array character variable cannot be used.

Explanation:

1) Counts the number of characters in a simple variable.
2) Usable character variables are simple character variables (A$, Y$, $, etc.). Array character variables such as B$(3) cannot be used.

Example:

```
10 INPUT"String=",$
20 PRINT"Length=";LEN($)
30 GOTO 10
```

MID$ (Location [, Number of characters]) Ⓕ

Numeric expression Numeric expression

Function:

Fetches the specified number of characters from a specified location of the exclusive character variable ($).

Parameter:

Location:	Numeric expression. Values below the decimal point are discarded. $1 \leqq$ location < 101
Number of characters:	Numeric expression. Values below the decimal point are discarded. $1 \leqq$ number of characters < 101.

When omitted, all characters after the specified location are fetched.

Explanation:

1) Fetches a specified number of characters from a specified location of the exclusive character variable ($).
2) When the specified location is out of the character string, a null is obtained.
3) When the length of the character string after the specified location is smaller than the specified number of characters, all the characters after the specified location are fetched.

*MID$ can be abbreviated as MID.

Example:

```
10 $="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 INPUT"location=",M,"length=",N
30 PRINT MID$(M,N)
40 GOTO 20
```

VAL (Simple character variable) Ⓕ

Function:

Converts characters in a simple character variable into a numeric value.

Parameter:

Simple character variable: An array character variable cannot be used.

Explanation:

1) Converts characters in a simple character variable into a numeric value.
2) When the content of a character variable includes +, −, •, **E** or **E**⁻, it is converted into a numeric value as it is.
   When **A$ = "-12.3", VAL(A$) → -12.3**
3) When the content of a character variable starts with a character other than a numeral, +, −, or •, an error occurs.
   When **A$ = "A45", VAL(A$) → error (Error 2)**
4) When a character other than a numeral is inserted in the middle, only the part before this character is converted to a numeric value.
   When **A$ = "78A9", VAL(A$) → 78**

Example:

```
10 Z$="123"
20 PRINT VAL(Z$)+45
30 END
```

*If this program is executed, the numeric values 123 and 45 will be added and 168 will be displayed.

STR$ (Numeric expression) Ⓕ

Function:

Converts the value of a numeric expression into a character string.

Parameter:

Numeric expression: Numeric value, calculation expression, numeric variable, numeric array variable.

Explanation:

1) Converts the value of a numeric expression into a character string.
2) When the numeric expression is a calculation expression, the calculation result is converted into a character string.
3) When a numeric expression is positive, the sign digit is deleted and only the numerals are converted.

Example:

```
10 A=123
20 PRINT STR$(A)+"45"
30 END
```

*If this program is executed, character strings "123" and "45" will be connected and "12345" will be displayed. (Same as "123" + "45")

6-4 Numeric Functions

SIN Argument (Numeric expression) **COS** Argument (Numeric expression) Ⓕ

TAN Argument (Numeric expression)

Function:

Obtains the value of a trigonometric function for a given argument.

Parameter:

Argument: Numeric expression

- $1440° < |\text{argument}| < 1440°$ (degrees)
- $8\pi < |\text{argument}| < 8\pi$ (radians)
- $1600 < |\text{argument}| < 1600$ (grads)

However, for TAN, "|Argument| = $(2n - 1) * 1$ right angle" is excluded.

1 right angle = $90° = \frac{\pi}{2}$ rad = 100 grad.

Explanation:

1) Obtains the value of a trigonometric function for a given argument.
2) The value depends on the angle unit setting; DEG, RAD or GRA (by the MODE key or MODE command).
3) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

ASN Argument: Numeric expression **ACS** Argument: Numeric expression Ⓕ

ATN Argument: Numeric expression

Function:

Inverse trigonometric function that obtains an angle for a given argument.

Parameter:

Argument: Numeric expression.
For ASN, ACS, $-1 \leq$ argument ≤ 1.

Explanation:

1) Inverse trigonometric function that obtains an angle for a given argument.
2) The value depends on the angle unit setting (by the MODE key or MODE command).
3) The values of the functions are given within the following range.

Degrees (DEG)	Radians (RAD)	Grads (GRA)
$-90° \leq \mathrm{ASN}\,x \leq 90°$	$-\frac{\pi}{2} \leq \mathrm{ASN}\,x \leq \frac{\pi}{2}$	$-100 \leq \mathrm{ASN}\,x \leq 100$
$0° \leq \mathrm{ACS}\,x \leq 180°$	$0 \leq \mathrm{ACS}\,x \leq \pi$	$0 \leq \mathrm{ACS}\,x \leq 200$
$-90° \leq \mathrm{ATN}\,x \leq 90°$	$-\frac{\pi}{2} \leq \mathrm{ATN}\,x \leq \frac{\pi}{2}$	$-100 \leq \mathrm{ATN}\,x \leq 100$

4) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

HYPSIN

Argument
Numeric expression

HYPCOS

Argument
Numeric expression

HYPTAN

Argument
Numeric expression

Ⓕ

Function:

Obtains the value of a hyperbolic function for a given argument.

Parameter:

Argument: Numeric expression
$|\text{argument}| \leqq 230.2585092$ (HYPSIN, HYPCOS)
$|\text{argument}| < 10^{100}$ (HYPTAN)

Explanation:

1) Obtains the value of a hyperbolic function for a given argument.

HYPSIN : $\sinh x = (e^x - e^{-x})/2$
HYPCON : $\cosh x = (e^x + e^{-x})/2$
HYPTAN : $\tanh x = (e^x - e^{-x})/(e^x + e^{-x})$

2) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

HYPASN Argument
Numeric expression

HYPACS Argument
Numeric expression

HYPATN Argument
Numeric expression

Ⓕ

Function:

Obtains the value of an inverse hyperbolic function for a given argument.

Parameter:

Argument: Numeric expression

HYPASN: $|\text{argument}| < 5 \times 10^{99}$

HYPACS: $1 \leqq \text{argument} < 5 \times 10^{99}$

HYPATN: $|\text{argument}| < 1$

Explanation:

1) Obtains the value of an inverse hyperbolic function for a given argument.

$$\text{HYPASN} : \sinh^{-1} x = \log_e (x + \sqrt{x^2 + 1})$$

$$\text{HYPACS} : \cosh^{-1} x = \log_e (x + \sqrt{x^2 - 1})$$

$$\text{HYPATN} : \tanh^{-1} x = \frac{1}{2} \log e \frac{1 + x}{1 - x}$$

2) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

LOG Argument (Numeric expression) LN Argument (Numeric expression) Ⓕ

Function:

Obtains the value of a logarithmic function for a given argument.

Parameter:

Argument: Numeric expression.
$0 <$ argument

Explanation:

1) Obtains the value of a logarithmic function for a given argument.
   - **LOG Common logarithmic function** $\log_{10}x$, $\log x$
   - **LN Natural logarithmic function** $\log_e x$, $\ln x$
2) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

EXP Argument (Numeric expression) Ⓕ

Function:

Obtains the value of an exponential function for a given argument.

Parameter:

Argument: Numeric expression.
$-10^{100} <$ argument ≤ 230.2585092

Explanation:

1) Obtains the value of an exponential function (e^x) for a given argument.
2) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

SQR

Argument
Numeric expression

Ⓕ

Function:

Obtains the square root of a given argument.

Parameter:

Argument: Numeric expression. $0 \leq$ argument

Explanation:

1) Obtains the square root ($\sqrt{x}$) of a given argument.
2) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

CUR

Argument
Numeric expression

Ⓕ

Function:

Obtains the value of a cube root for a given argument.

Parameter:

Argument: Numeric expression. $|\text{argument}| < 10^{100}$

Explanation:

1) Obtains the cube root ($\sqrt[3]{x}$ or $x^{\frac{1}{3}}$) of a given argument.
2) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

ABS

Argument
Numeric expression

Ⓕ

Function:
Obtains the absolute value for a given argument.

Parameter:
Argument: Numeric expression.

Explanation:
1) Obtains the absolute value of a given argument.
2) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

SGN

Argument
Numeric expression

Ⓕ

Function:
Obtains a value (1, 0 or -1) corresponding to the sign of a given argument.

Parameter:
Argument: Numeric expression.

Explanation:
1) Gives a value that corresponds to the sign of an argument.
   When an argument is positive, 1
   When an argument is 0, 0
   When an argument is negative, -1
2) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

INT

Argument
Numeric expression

Ⓕ

Function:

Obtains the maximum integer that does not exceed a given argument.

Parameter:

Argument: Numeric expression.

Explanation:

1) Obtains the maximum integer that does not exceed a given argument.

```
INT 12.56→12
INT -78.1→-79
INT 12→12
```

2) This is the same function as the Gaussian function ($[x]$) used in mathematics.
3) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

FRAC

Argument
Numeric expression

Ⓕ

Function:
Obtains the decimal part of a given argument.

Parameter:

Argument:	Numeric expression.

Explanation:

1) Obtains the decimal part of a given argument. Its sign agrees with the sign of the argument.
2) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

RND

(Argument , **digit location)**
Numeric expression , Numeric expression

Ⓕ

Function:
Obtains the value of a given argument rounded off at the specified location.

Parameter:

Argument:	Numeric expression.
Location:	Numeric expression. Values below the decimal point are discarded. Displays the exponent n when the location to be rounded off is 10^n. $\lvert \text{location} \rvert < 100$

Explanation:

1) Obtains the value of a given argument which is rounded off at the specified location.
2) If RND (x, y) is executed, x will be rounded off at 10^y.
   Rounds off at the third decimal place $(10^{-3}) \rightarrow$ RND $(x, -3)$
   Rounds off at the 100s position $(10^2) \rightarrow$ RND $(x, 2)$
3) The parenthesis cannot be omitted.

REC (r coordinate, θ coordinate) / Numeric expression

Ⓕ

Function:

Transforms polar coordinates (r, θ) to rectangular coordinates (x, y).

Parameter:

r coordinate:	Numeric expression. $0 \leq r < 10^{100}$
θ coordinate:	Numeric expression
	DEG: $\lvert\theta\rvert < 1440°$
	RAD: $\lvert\theta\rvert < 8\pi$ rad
	GRA: $\lvert\theta\rvert < 1600$ gra

Explanation:

1) Transforms polar coordinates (r, θ) to rectangular coordinates (x, y) using the following relational expressions. $x = r\cos\theta$
$y = r\sin\theta$
2) The x coordinate of (x, y) will be given as the output of the function, and the value of the x coordinate will be assigned to variable X and the value of the y coordinate will be assigned to variable Y at the same time.
3) An error will occur if $r < 0$.
4) The output values will correspond to the setting of the angle unit (DEG, RAD or GRA).

Example:

```
10 CLEAR
20 MODE 4
30 INPUT "r =",R
40 INPUT "t (°)=",T
50 X=REC(R,T)
60 PRINT"x=";X
70 PRINT"y=";Y
80 GOTO 30
```

*(x, y) will be displayed if (r, θ) is input.

POL (x coordinate, y coordinate) Ⓕ

Numeric expression Numeric expression

Function:

Transforms rectangular coordinates (x, y) to polar coordinates (r, θ).

Parameter:

x coordinate:	Numeric expression	$\lvert x\rvert + \lvert y\rvert > 0$
y coordinate:	Numeric expression	

Explanation:

1) Transforms rectangular coordinates (x, y) to polar coordinates (r, θ) using the following relational expressions.

$$\begin{cases} r = \sqrt{x^2 + y^2} \\ \cos\theta = \dfrac{x}{\sqrt{x^2+y^2}}, \quad \sin\theta = \dfrac{y}{\sqrt{x^2+y^2}} \end{cases}$$

2) The r coordinate of (r, θ) is given as the output of the function, and the value of the r coordinate will be assigned to variable X and the value of the θ coordinate will be assigned to variable Y at the same time.
3) Calculation of the θ coordinate will correspond to the angle unit (DEG, RAD or GRA). The value of θ will be given within the following range.
   DEG: $-180° < \theta \leqq 180°$
   RAD: $-\pi$ rad $< \theta \leqq \pi$ rad
   GRA: -200 gra $< \theta \leqq 200$ gra
4) An error will occur when $\lvert x\rvert + \lvert y\rvert = 0$.

Example:

```
10 CLEAR
20 MODE 4
30 INPUT"X=",A
40 INPUT"Y=",B
50 PRINT"r =";POL(A,B)
60 PRINT"t =";DMS$(Y)
70 GOTO 30
```

*(r, θ) will be displayed if (x, y) is input.

FACT

Argument
Numeric expression

Ⓕ

Function:

Obtains the factorial value for a given argument.

Parameter:

Argument: Numeric expression
The argument must take an integer value with the range of ($0 \leqq$ argument $\leqq 69$).

Explanation:

1) Obtains the factorial value ($x!$) of the argument x.
2) An error will occur if argument x contains fractions.
3) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

Example:

FACT 10 EXE (10!)	3628800
FACT 69 EXE (69!)	1.711224524E98

NPR (Value of n, value of r) Ⓕ

Numeric expression, Numeric expression

Function:

Obtains the permutations (nPr) for given values of n and r.

Parameter:

Value of n: Numeric expression
Value of r: Numeric expression

n and r are integer values with the range of $0 \leqq r \leqq n < 10^{10}$.

Explanation:

1) This function obtains permutations $_n\mathrm{P}_r \left(= \frac{n!}{(n-r)!}\right)$
2) An error occurs if n or r contains fractions.

Example:

NPR(5,3)EXE ($_5$P$_3$)	60
NPR(5,0)EXE ($_5$P$_0$)	1

NCR (Value of n, value of r)

Numeric expression, Numeric expression

Ⓕ

Function:

Obtains the combinations (nCr) for given values of n and r.

Parameter:

Value of n: Numeric expression
Value of r: Numeric expression

n and r are integer values with the range of $0 \leqq r \leqq n < 10^{10}$.

Explanation:

1) Obtains the combinations $_nC_r \left(= \frac{n!}{r!(n-r)!}\right)$.
2) An error occurs if n or r contains decimals.

Example:

NCR(5,3)[EXE] ($_5C_3$)	10
NCR(5,0)[EXE] ($_5C_0$)	1

EOX

Argument
Numeric expression

Ⓕ

Function:

Obtains the estimated value of x for a value of y given as the argument in the paired variable statistics of (x, y).

Parameter:

Argument: Numeric expression giving the value of y.

Explanation:

1) Obtains the estimated value of x for the value of y according to the linear regression expression $y = a + bx$ in the paired variable statistics of (x, y).

$$\mathrm{EOX}(y) = \frac{y - a}{b}$$

2) The values of linear regression constant term a and linear regression coefficient b are determined by the statistical data.
3) The value of EOX(y) will be uncertain when $b = 0$. The input value y is with the range of $|y| < 10^{100}$ and the estimated value of x is with the range of $|\mathrm{EOX}(y)| < 10^{100}$ when $b \neq 0$.
4) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

EOY

Argument
Numeric expression

Ⓕ

Function:

Obtains the estimated value of y for x a value of given as the argument in the paired variable statistics of (x, y).

Parameter:

Argument: Numeric expression giving the value of x.

Explanation:

1) Obtains the estimated value of y for the value of x according to the linear regression expression $y = a + bx$ in the paired variable statistics of (x, y).

$$\mathrm{EOY}(x) = a + bx$$

2) The values of linear regression constant term a and linear regression coefficient b are determined by the statistical data.
3) Input value x is with the range of $|x| < 10^{100}$ and the estimated value of y is with the range of $|\mathrm{EOY}(x)| < 10^{100}$.
4) As a rule, the argument is enclosed in parenthesis but the parenthesis can be omitted if the argument is a variable or a numeric value.

RAN

Ⓕ

Function:

Obtains a random number from 0 to 1.

Explanation:

1) Obtains a pseudo-random number from 0 to 1 within 10 digit mantissa.

$$0 < \text{random number} < 1.$$

Example:

Provides a random number with 1 digit from 0 to 9.

```
INT(RAN# * 10)
```

Provides a random number with 1 digit from 1 to 6.

```
INT(RAN# * 6)+ 1
```

Provides a random number with 2 digits from 10 to 99.

```
INT(RAN# * 90)+ 10
```

DEG (Degree [, Minute [, Second]]) Ⓕ

Numeric expression Numeric expression Numeric expression

Function:

Converts a sexagesimal expressed by given degrees, minutes and seconds to a decimal.

Parameter:

Degree:	Numeric expression.
Minute:	Numeric expression.
Second:	Numeric expression.

$|\text{DEG}(\text{degree, minute, second})| < 10^{100}$

Explanation:

1) Converts a sexagesimal expressed by given degrees, minutes and seconds to a decimal.

$$\text{DEG}(a, b, c) = a + \frac{b}{60} + \frac{c}{3600}$$

2) The minutes and seconds can be omitted, and they will be considered 0.
3) The parenthesis cannot be omitted.

Example:

```
DEG(12,34,56) EXE        12.58222222

10 INPUT A,B,C
20 PRINT DEG(A,B,C)
30 END
```

DMS $

(Argument)
Numeric expression

Ⓕ

Function:

Converts a given decimal argument to a character string in the sexagesimal notation.

Parameter:

Argument: Numeric expression.
|Numeric expression| $< 10^{100}$

Explanation:

1) Converts a decimal given as a numeric expression to a character string in the sexagesimal notation.
2) Although degrees, minutes and seconds will be displayed with the range of |numeric expression| $< 10^5$, the value of the numeric expression itself will be displayed if outside this range.
3) The result will be given as a character string.

Example:

DMS$(180/π)EXE	57°17'44.81
DMS$(45.678)EXE	45°40'40.8
DMS$(99999.999)EXE	99999°59'56.4
DMS$(100000.1)EXE	100000.1

```
10 INPUT A
20 $=DMS$(A)
30 PRINT $
40 END
```

HEX $

(Argument)
Numeric expression

Ⓕ

Function:

Converts a given decimal argument to a 4-digit hexadecimal character string.

Parameter:

Argument: Numeric expression
-32769 < numeric expression < 65536

Explanation:

1) Converts a decimal given as a numeric expression to a 4-digit hexadecimal character string.
2) The value of a numeric expression given as the argument is handled as an integer with decimals discarded.
3) When the value of an argument exceeds 32768, it will be handled as the value after subtracting 65536.

<**Example**>
40000 will be handled as follows.
40000 − 65536 = -25536

Example:

HEX$(10000)EXE	2710
HEX$(65535)EXE	FFFF
HEX$(−1) EXE	FFFF

```
10 INPUT"X=",X
20 PRINT "X=&H";HEX$(X)
30 GOTO 10
```

&H Character string

Ⓐ

Function:

Converts a hexadecimal character string to a decimal value by placing this function at the beginning of a given hexadecimal character string.

Parameter:

Character string: Hexadecimal numeric string. (Up to 4 digits)
$-32768 \leq$ &H character string ≤ 32767

Explanation:

1) If placed at the beginning of a hexadecimal, it will be converted to a decimal integer value. Although &H is shown as a function since it functions opposite to HEX$, strictly speaking, it is not a function but is actually a hexadecimal identifier.
2) Since the conversion result will be a decimal integer with the range of -32768 to +32767, &HFFFF, for example, will not indicate 65535 but -1.
3) The 0 placed at the top of a hexadecimal and spaces in a hexadecimal character string are disregarded.

<**Example**>

&H0010 indicates 16 of a decimal number.
&HA ␣ B indicates 171 of a decimal number.
&H ␣ A indicates 10 of a decimal number.
(␣ means a space.)

4) An error (Error 2) will occur if a hexadecimal character string exceeds four digits or if there is a character other than a hexadecimal in the character string.

<**Example**>

&H 10000 → Error 2 (Five digits)
&H AG → Error 2 (G is not a hexadecimal)
&H A# → Error 2 (# is not a hexadecimal)

Example:

```
 10 CLEAR
 20 READ H$
 30 IF H$="end"THEN 80
 40 M$="&H"+H$
 50 A=VAL(M$)
 60 PRINT M$;"=" ";A;"␣:␣";
 70 GOTO 10
 80 PRINT H$;:BEEP:END
100 DATA 10,100,1000,7FFF
110 DATA 8000,ABCD,FFFF
120 DATA end
```

Space

*Converts hexadecimals to decimals.

NEW#

Ⓜ

Function:

Erases memo data in the DATA BANK.

Explanation:

1) Erases all stored data.
2) Cannot be executed when a password is specified.
3) Can only be executed in the WRT mode.

Example:

[MODE] [1]
NEW # [EXE]
[MODE] [0]

LIST#

Ⓜ

Function:

Displays all memo data in the DATA BANK.

Parameter:

Outputs all memo data in the DATA BANK with record numbers attached.

Explanations:

1) Displays all memo data stored in the DATA BANK from the smallest record number in the order stored.
2) The contents displayed are the record number and memo data.
3) Since the memo data will be displayed automatically in sequential order, press the [STOP] key to stop the display. Press the [EXE] key to resume the display.
4) In the PRT ON mode ([MODE] [7]), the memo data will be displayed successively at high speed and will be printed at the same time.

5) Execution is not possible when a password is being specified. (An error will occur.)
6) Cannot be used in a program.
7) Cannot be executed in the MEMO IN mode ([MODE][9]).

Example:

LIST#[EXE]

```
L I S T #
1 510-01,TELEVISION,.$330
2 510-02,RADIO,.$80
3 510-03,TAPE RECORDER,.$100
4 510-04,STEREO,.$850
5 510-05,VIDEO RECORDER,.$750
6 510-06,DESKTOP CALCULATOR,.$50
7 510-07,PERSONAL COMPUTER,.$650
8 END
```

SAVE#

["File name"]
Character string

Ⓜ

Function:

Stores memo data in the DATA BANK on a cassette tape.

Parameter:

File name: 1 ≦ character numbers of character string ≦ 8.
Can be omitted.

Explanation:

1) Stores all memo data in the DATA BANK on a cassette tape.
2) Since memo data cannot be stored with SAVE or SAVE ALL, always load memo data with SAVE#.
3) If a password has been specified, storing is performed with this password. Therefore, the same password must be specified when the loading is performed by the LOAD# command.
4) Cannot be executed in the MEMO IN mode.

Example:

```
SAVE#[EXE]
SAVE#"CASIO"[EXE]
```

LOAD#

["File name"] [, M]
Character string

Ⓜ

Function:

Loads memo data in the DATA BANK from a cassette tape.

Parameter:

File name:	1 ≦ character numbers of character string ≦ 8. Can be omitted.
M:	(If M is specified, additional memo data can be loaded.)

Explanation:

1) In the case of LOAD# ["File name"]
   a) Loads memo data in the DATA BANK from a cassette tape after erasing all memo data currently stored in the DATA BANK.
   b) Loads first memo data found on a cassette tape being played back if the file name is omitted.
   c) This cannot be executed in the MEMO IN mode.
   d) This cannot be executed in a program.

2) In the case of LOAD# ["File name"] ,M
   a) Loads additional memo data in the DATA BANK from a cassette tape following the memo data currently stored in the DATA BANK.

   For b) to d), same as for LOAD#.

Example:

```
LOAD# EXE
LOAD#"CASIO" EXE
```

READ#

Variable name [, variable name]*

Ⓟ

Function:

Reads memo data from the DATA BANK.

Parameter:

Variable name:	Numeric variable or character variable. An array variable can also be used.

Explanation:

1) Sequentially reads stored data to a variable.
2) Only numeric type data can be read for a numeric variable. If character type data are used, an error (Error 2) occurs.
3) After the necessary data are read by a READ# statement, the following data are read by the next READ# statement.
4) When memo data in the DATA BANK are punctuated by ",", they are read in the order in which they are written.

<Example>

DATA

No. 1 A, X, Y
No. 2 B, Z
No. 3 C

⬇

Reading sequence

A→X→Y→B→Z→C

5) When data to be read does not exist, an error (Error 4) occurs.
6) The data sequence to be read can be modified by RESTORE# (see page 172).

7) When a space exists at the beginning of memo data in the DATA BANK, it is skipped.

<Example>

```
X,␣Y,Z
  └→ This space skipped.
```

8) When data is inside " ", the character string inside " " is read.

Example:

< Data >	< Program >
No. 1 1,2,3	10 A=0
No. 2 4,5,6	20 READ#$
No. 3 7,8,9	30 IF $="" THEN 60
No. 4 10,	40 A=A+VAL($)
	50 GOTO 20
	60 PRINT "Σx=";A
	70 END

* Reads numeric data from the DATA BANK to obtain a sum.

RESTORE# ["Searched character string" / Character expression [, [{0 / 1}]] Ⓟ [, { Line number / # program area number }]]]

Function:

Searches memo data in the DATA BANK and specifies the sequence of the data to be read by READ#.

Parameter:

Searched character string:	Character expression. When a character string is used, place it inside " ".
Line number:	Numeric expression. 0 < line number < 10000
Program area No.:	Numeric expression. 0 ≦ program area No. < 10

Explanation:

1) Searches memo data in the DATA BANK and specifies the sequence of data to be read by the following READ# statement.
2) The relationship between a parameter and data searching is as follows.
   a) RESTORE#
      When the searched character string and after are omitted, data are read from the beginning by the following READ#.
   b) RESTORE# "searched character string"
      Memo data having the searched character string at the beginning is read by the following READ#.
   c) RESTORE# "searched character string", {0 / 1}
      When 0 is specified, it is the same as b).
      When 1 is specified, the first data of the line that includes searched data is read by the following READ# statement.
   d) RESTORE# "searched character string", [{0 / 1}], { **line number** / **#program area No.** }

      When executing searching, it jumps to the specified line or a program area if appropriate data does not exist.

* In b) and c), when appropriate data does not exist, an error (Error 4) occurs.

* In d), when a branching line number does not exist, or when a program does not exist in the program area, an error (Error 4) occurs.

*If a parameter is assigned, the appropriate data will be searched from the data and on to be read by the next READ# statement. Enter RESTORE# : RESTORE# "searched character string" when desiring to search from the first data.

Example:

< Memo Data >

Record 1 Smith, 03-347-4811, San Diego
Record 2 Jones, 075-351-1161, Princeton
Record 3 Williams, 06-314-2681, Cleveland
Record 4 Edwards, 045-211-0821, Cambridge

< Program >

```
  10 RESTORE#
  20 GOSUB 1000
  30 RESTORE#"J"
  40 GOSUB 1000
  50 RESTORE#"Cl",1
  60 GOSUB 1000
  70 RESTORE#"Aa",1,200
  80 GOSUB 1000
  90 END
 200 BEEP:PRINT"Memo End"
 210 END
1000 READ#$:PRINT$
1010 RETURN
```

Displays memo data stored at the beginning of the DATA BANK.

Displays data having the first character J.

Searches data having the first two characters Cl and displays the first data on that line.

Jumps to line 200 if data with the first two characters Aa does not exist.

Subroutine that reads and displays memo data.

< Execution Example >

```
RUN[EXE]
[EXE]
[EXE]
[EXE]
```

Smith
Jones
Williams
Memo End

WRITE#

[Data [, Data]*]
expression expression

Ⓟ

Function:

Rewrites or deletes memo data in the DATA BANK.

Parameter:

Data: Numeric expression or character expression. When a character string is used, place it inside " ".

Explanation:

1) Writes data in the record area currently specified by RESTORE#.
2) Data are newly written without any relationship to data existence in the appropriate record area.
3) When no data is specified, stored data in the record area are deleted.
4) When plural data exist, these data can be written on the same record area by using "," for punctuation.
5) After the necessary data are written by the first WRITE# statement, the following data are written by the next WRITE# statement.
6) When writing memo data, one step will be required in addition to the number of characters.

   Memo data ABC ~ XYZ
   ↑
   (26 characters) + 1 step = 27 steps

Example:

```
 10 REM data write
 20 RESTORE#                 } Writes new memo data.
 30 WRITE#"X,Y,Z"
 40 GOSUB 1000
 50 PRINT"␣"
110 REM data change
120 RESTORE#                 } Rewrites memo data.
130 FOR J=1 TO 3
140 WRITE# STR$(J)
150 NEXT J
```

```
 160 GOSUB 1000
 170 PRINT"␣"
 210 REM data clear
 220 RESTORE#
 230 WRITE#
 240 RESTORE#
 250 READ#$
 260 END
1000 REM display sub
1010 RESTORE#
1020 FOR J=1 TO 3
1030 READ#$:PRINT$;
1040 NEXT J
1050 RETURN
```

Deletes memo data.

Subroutine to display memo data.

< Execution Example >

[MODE] [1]
NEW# [EXE]
[MODE] [⊘]
RUN [EXE]
[EXE]
[EXE]

Erases all memo data in the DATA BANK.

```
XYZ
123
Error4 P0-250
```

↓

This error occurs when data is deleted and no memo data remains in the DATA BANK.

7 Convenient DATA BANK Function

The DATA BANK can hold all your private memoranda such as names, telephone numbers, addresses, dates, etc. The object data can be speedily retrieved from the large amounts of stored data without the need for a special program. Furthermore, combined with a BASIC program, you can use this function for an expansive range of duties such as schedules, totalizing, etc. The use of the DATA BANK function is explained together with actual examples. For details on the exclusive commands (LIST#, RESTORE#, READ#, WRITE#) for the DATA BANK, refer to Chapter 6 "Command Reference".

7-1 Specifying the MEMO IN Mode

With its DATA BANK function, this computer can be used as an "electronic memo pad" in which data (memo data) can be conveniently written and from which the necessary data can be retrieved by simple key operations.
In order to utilize the DATA BANK function, it will first be necessary to input and store data. Specify the MEMO IN mode to input data to the DATA BANK. Pressing [MODE][9] will specify the MEMO IN mode and the display will appear as shown below.

The symbol " [MEMO][IN] " at the upper center of the display shows that the MEMO IN mode is currently specified. The number at the upper right is the record number which indicates the memo data line.
The above display shows that the cursor is blinking at the left end with nothing stored in the DATA BANK as yet and the computer is in the key-input waiting state for Record 1. If the record number happens to be 5, it will mean that four lines of memo data have already been stored in the DATA BANK.

7-2 Inputting Data

First input the names and telephone numbers of 10 people by assuming that the DATA BANK function will be used as a private "electronic telephone directory".

BROWN	03-021-1234	SMITH	0899-02-1007
ELLIS	011-041-7386	SULLIVAN	078-039-7132
FOX	06-021-6602	WATTS	0467-01-3569
JONES	052-031-6221	YOUNG	0425-01-0038
MILLS	03-063-2935	HOYT	03-054-4321

Start with entering the data for BROWN in the MEMO IN mode as follows. Separate the name and telephone number by inserting a "," (comma) between them.

Operation	Display
[B][R][O][W][N]	BROWN_
[,]	BROWN,_
[0][3][-]	BROWN,03-_
[0][2][1][-][1][2][3][4]	BROWN,03-021-1234_
[EXE]	BROWN,03-021-1234

If the [EXE] key is pressed after completing input of data, the cursor will disappear and the BROWN's data will be stored in the DATA BANK as the memo data of record 1.

Press the [EXE] key once more. The display will be cleared and the record number will change to 2. The blinking cursor indicates that the computer is in the key-input waiting state.

[EXE]	_

Next, enter the ELLIS' data in Record 2.

Operation:

[E][L][L][I][S]	BUZZER DEG MEMO IN 2 ELLIS_
[,]	BUZZER DEG MEMO IN 2 ELLIS,_
[0][1][1][-]	BUZZER DEG MEMO IN 2 ELLIS,011-_
[0][4][1][-][7][3][8][6]	BUZZER DEG MEMO IN 2 ELLIS,011-041-7386_
[EXE]	BUZZER DEG MEMO IN 2 ELLIS,011-041-7386

In the same manner, enter the names and telephone numbers from FOX to HOYT in successive order. Do not forget to press the [EXE] key at the end of each telephone number.

Notes:

1. There is a reason for separating the name and telephone number with "," (comma). Since the comma is a special symbol indicating separation of data in one record in the DATA BANK and, since it will serve an important role when retrieving the memo data later, always be sure to enter the comma.

2. In the above example, pressing the [EXE] key again after storing one record with the [EXE] key will clear the display and cause the record number to advance to the next. When entering data continuously, however, it will not be necessary to press the [EXE] key twice each time in this manner to clear the display. Data will be stored by pressing the [EXE] key once at the end of each record. When the first character of next data is entered, the previous data will be automatically erased from the display. In this case, the previous record number remains on the display until the [EXE] key. For example, the FOX's data can be entered after the ELLIS' data by the following procedure.

Operation

[EXE] (Completes input of ELLIS' data.)

```
BUZZER   DEG   MEMO IN   2
ELLIS,011-041-7386
```

[F][O][X][,]

```
BUZZER   DEG   MEMO IN   2
FOX,_
```

[0][6][-]

```
BUZZER   DEG   MEMO IN   2
FOX,06-_
```

[0][2][1][-][6][6][0][2]

```
BUZZER   DEG   MEMO IN   2
FOX,06-021-6602_
```

[EXE] (Record number changes to 3.)

```
BUZZER   DEG   MEMO IN   3
FOX,06-021-6602
```

When data input of all 10 persons is over, press the [EXE] key once again to clear the display.

[EXE] (Completes input of HOYT's data.)

```
BUZZER   DEG   MEMO IN   10
HOYT,03-054-4321
```

[EXE] (Clears the display.)

```
BUZZER   DEG   MEMO IN   11
```

7-3 Displaying the Data Contents

To confirm the input data, display the data contents for the ten people now stored. Specify the RUN mode by pressing [MODE][0].

[MODE][0]

```
BUZZER    RUN DEG
Ready P0
```

Use the LIST# command to display the entire contents of the DATA BANK. The data in each record will be displayed successively at approximately 1.5 sec. intervals together with the record number.

Operation

[SHIFT][LIST][SHIFT][#]

[EXE]

```
BUZZER    RUN DEG
LIST #_
```

```
BUZZER    RUN DEG
1 BROWN,03-021-1234
```

```
BUZZER    RUN DEG
2 ELLIS,011-041-7386
```

```
BUZZER    RUN DEG
3 FOX,06-021-6602
```

⋮

```
BUZZER    RUN DEG
9 YOUNG,0425-01-0038
```

```
BUZZER    RUN DEG
10 HOYT,03-054-4321
```

```
BUZZER    RUN DEG
Ready P0
```

Press the [STOP] key to stop the display temporarily for checking the contents. Press the [EXE] key to resume the following display.
This enables checking of any input data errors.

*The LIST# command can also be executed in the WRT mode ([MODE][1]).

7-4 Correcting Data

Assume that the contents of the DATA BANK were displayed with the LIST# command and, upon checking, it was found that the telephone number of JONES' in Record 4 was wrong.

Wrong 052-031-6211 → Correct 052-031-6221

In this cases, specify the MEMO IN mode and correct the data with the following procedure.

Operation

MODE 9	BUZZER DEG MEMO IN 11	Specifies the MEMO IN mode.
MEMO	BUZZER DEG MEMO IN 1 BROWN,03-021-1234	Displays the memo data in Record 1.
EXE	BUZZER DEG MEMO IN 2 ELLIS,011-041-7386	Advances to the next record when EXE key is pressed.
EXE	BUZZER DEG MEMO IN 3 FOX,06-021-6602	
EXE	BUZZER DEG MEMO IN 4 JONES,052-031-6211	Displays the memo data in Record 4.
←	BUZZER DEG MEMO IN EDIT 4 JONES,052-031-6211_	Press the cursor movement key to specify the EDIT mode.
← ←	BUZZER DEG MEMO IN EDIT 4 JONES,052-031-6211	Move the cursor to the position to be corrected.
2	BUZZER DEG MEMO IN EDIT 4 JONES,052-031-6221	Correct the data.
EXE	BUZZER DEG MEMO IN 4 JONES,052-031-6221	Press the EXE key after the correction.

If the MEMO IN mode is specified with the MODE 9 keys, the record number next to the last record stored will be displayed (record number 11 in this case). Press the MEMO key to display the data in Record 1 in this state. The record number will then advance each time the EXE key is pressed.

Although the JONES' data in Record 4 is displayed in this manner, the cursor is not displayed. If a cursor movement key (← or →) is pressed, the " EDIT " symbol will appear and the cursor will be displayed. This state is the EDIT mode and memo data can be corrected in this mode. Move the cursor with a cursor movement key to the position to be corrected and, after making the necessary corrections, press the EXE key. The corrected data will then be stored. (The " EDIT " symbol will disappear.)

7-5 Retrieving (Searching) Data

With the DATA BANK function, data retrieval can be performed directly by pressing the [MEMO] key.

1) Searching with the [MEMO] Key

If the [MEMO] key is pressed in the RUN mode ([MODE][⊘]), the name, BROWN and his telephone number in Record 1 will be displayed. (The " MEMO " symbol will appear.)

Operation

[MODE][⊘]

```
BUZZER    RUN DEG
Ready P0
```

[MEMO]

```
BUZZER    RUN DEG    MEMO    1
BROWN,03-021-1234
```

If the [MEMO] key is pressed again, only his telephone number will be displayed.

[MEMO]

```
BUZZER    RUN DEG    MEMO    1
03-021-1234
```

If the [MEMO] key is pressed once again, the ELLIS' data in Record 2 will be displayed.

[MEMO]

```
BUZZER    RUN DEG    MEMO    2
ELLIS,011-041-7386
```

Each time the [MEMO] key is pressed in this manner, data in the same record separated with "," (comma) will be displayed (the first 24 characters when a long data). When display of all data in one record is over, it will advance to the next record.

After all memo data stored in the DATA BANK have been displayed, the display will be cleared, and the " MEMO " symbol will disappear. If the MEMO key is pressed here again, data in Record 1 will be displayed again.

(The data in Record 10 is displayed by pressing the MEMO key repeatedly.)

MEMO

MEMO (The "MEMO" symbol disappears and the cursor blinks.)

MEMO (Displays the data in Record 1 again.)

```
BUZZER   RUN DEG   MEMO   10
HOYT,03-054-4321

BUZZER   RUN DEG   MEMO   10
03-054-4321

BUZZER   RUN DEG
_

BUZZER   RUN DEG   MEMO   1
BROWN,03-021-1234
```

2) Quick Search with the EXE Key

Memo data searched and displayed with the MEMO key can be advanced in record units by using the EXE key.

MODE ⊘

MEMO

EXE

EXE

```
BUZZER   RUN DEG
Ready P0

BUZZER   RUN DEG   MEMO   1
BROWN,03-021-1234

BUZZER   RUN DEG   MEMO   2
ELLIS,011-041-7386

BUZZER   RUN DEG   MEMO   3
FOX,06-021-6602
```

*The data in this example is displayed at a time because it is within 24 characters. If the data is long, however, the first 24 characters of the data will be displayed.

Even if the [EXE] key is pressed when the last record stored is displayed, the last data will then remain displayed.

(The data in Record 10 is displayed by pressing the [EXE] key repeatedly.)

[EXE] (Display remains unchanged.)

3) Backward Search with the [SHIFT] [▲EXE] Keys

Press [SHIFT] [▲EXE] to display a previous record when searching for a memo data with the [MEMO] key and [EXE] key.

If backward search is repeated in record units with [SHIFT] [▲EXE] similar to quick search with the [EXE] key, the command will return to the data in Record 1. Record 1 will then continue to be displayed even if the [SHIFT] [▲EXE] keys are pressed again. This operation can be repeated until the data in Record 1 is displayed.

(Data in Record 3 currently being displayed.)

[SHIFT] [▲EXE]

[SHIFT] [▲EXE] (Returns to Record 1.)

[SHIFT] [▲EXE] (Display remains unchanged.)

4) Conditional Search

Memo data are displayed in successive order from Record 1 when searching with the [MEMO] key or [EXE] key but time will be required to display the necessary data if the amount of stored data becomes great.

In this case it will be convenient to use "conditional search".

For example, we will search for a name starting with "S". Press [S] [MEMO].

[S] [MEMO]

BUZZER RUN DEG MEMO 6
SMITH,0899-02-1007

[MEMO]

BUZZER RUN DEG MEMO 7
SULLIVAN,078-039-7132

[MEMO]

BUZZER RUN DEG
_

The SMITH's data in Record 6 will be displayed. If [MEMO] is pressed again, the SULLIVAN's data in Record 7 will be displayed. If [MEMO] is pressed once again, the display will be cleared and the cursor will blink. This means that there are no more names starting with "S".

If there are multiple pertinent data, they will be displayed in the ascending order of record numbers.

The specified condition need not be a single character as in the above but can be longer character string. For example, if the 8 characters SULLIVAN are used as the condition, data beginning with these 8 characters will be retrieved.

[S] [U] [L] [L] [I] [V] [A] [N] [MEMO]

BUZZER RUN DEG MEMO 7
SULLIVAN,078-039-7132

[MEMO]

BUZZER RUN DEG
_

5) Additional Conditional Search

Assume that the JONES' data in Record 4 is currently being displayed as a result of searching with the condition "J" specified. It will also be possible at this time to make a new search of data after Record 4 by specifying an additional condition.

For example, search for a telephone number starting with an area number of 03 after Record 4. Press [0] [3] [SHIFT] [MEMO].

(Assuming that JONES' data is being displayed by [J] [MEMO])

[0][3][SHIFT][MEMO] (Specifies an additional condition.)

[MEMO]

[MEMO]

```
BUZZER   RUN DEG   MEMO   4
JONES,052-031-6221
BUZZER   RUN DEG   MEMO   5
03-063-2935
BUZZER   RUN DEG   MEMO   10
03-054-4321
BUZZER   RUN DEG
....
```

If [0][3][SHIFT][MEMO] are pressed as in the above, data starting with condition "03" will be displayed after Record 4. If there are multiple data with condition "03", the second corresponding data and after will be displayed each time the [MEMO] key is pressed.

If conditional search with condition "03" is performed in the above example instead of the additional conditional search, the first data displayed will become BROWN's telephone number in Record 1.

[0][3][MEMO] (Simple conditional search)

```
BUZZER   RUN DEG   MEMO   4
JONES,052-031-6221
BUZZER   RUN DEG   MEMO   1
03-021-1234
```

Thus, the additional conditional search allows you to find a record satisfying the initial condition and, after altering the condition, to retrieve data satisfying that condition from the subsequent data. This function enables you to narrow down the search by altering the condition.

6) Confirming with the Cursor Movement Key

Telephone numbers only will be displayed if search is made by specifying the condition "03". Use the cursor movement key [←] to confirm the name of the person with this telephone number.

[0][3][MEMO] (Conditional search)

[←][←][←][←][←][←]

```
BUZZER   RUN DEG   MEMO   1
03-021-1234
BUZZER   RUN DEG   MEMO   1
BROWN,03-021-1234
```

The telephone number moves one space to the right each time the ← key is pressed and the name will appear from the left.
Operating SHIFT L.TOP instead of pressing the ← key 6 times displays the data from the beginning.

0 3 MEMO

SHIFT L.TOP

```
BUZZER    RUN DEG    MEMO    :
03-021-1234
```

```
BUZZER    RUN DEG    MEMO    :
BROWN,03-021-1234
```

If the contents in one record exceed 24 characters, it will not be possible to display these at one time. The → key can then be used to shift the display to the left to bring the characters hidden on the right into view. Pressing SHIFT L.END will cause the last 24 characters in the record to be displayed.

7-6 Erasing Data

Erasing memo data can be performed in the EDIT mode as in data correction (see page 183).

1) To Erase Part of a Record

Display the desired data in the MEMO IN mode and specify the EDIT mode by pressing a cursor movement key. Then move the cursor to the character that you wish to erase. Press [DEL] to erase the character. After erasing unnecessary character. After erasing unnecessary characters, be sure to press the [EXE] key.

2) To Erase All Data in a Record

Display the desired record in the MEMO IN mode and specify the EDIT mode by pressing a cursor movement key. Then clear the display by pressing the [AC] key. And press the [EXE] key.

All data in that record will be erased and the current record number will be automatically assigned to the next record.

To erase all data stored in the DATA BANK, execute the NEW# command in the WRT mode ([MODE] [1]).

[N] [E] [W] [SHIFT] [#] [EXE]

The NEW# command should be executed with caution to avoid erasing any important data in the DATA BANK.

7-7 Adding and Inserting Data

It is possible to add a new data next to the last record or to insert a new record between stored records.

1) Adding Data

[MODE] [9]

```
BUZZER        DEG        MEMO IN        11
```

If the MEMO IN mode is specified, the computer will go into the key-input waiting state and the record number will be displayed. Record number 11 is displayed and the cursor is blinking in the above example. The computer is waiting for input of data for Record 11. This also means that data for 10 records have already been stored.

A new data will be stored in Record 11 by pressing [EXE] .

2) Inserting Data

To insert new data between stored records, first specify the MEMO IN mode and display the record where the new data will be stored. Next, input the data to be inserted and press [SHIFT] [EXE] . (Be careful here since pressing [EXE] only will cause the data to be stored in a record next to the last record.)

For example, assume we wish to insert the HOYT's data between FOX's data in Record 3 and JONES' data in Record 4. First, display the JONES' data in the MEMO IN mode.

[MODE] [9] [MEMO] [EXE] [EXE] [EXE]

```
BUZZER        DEG        MEMO IN        4
JONES,052-031-6221
```

Next, enter the initial character [H] of the HOYT. The previous display will be cleared and "H" will be displayed. Continue entering the HOYT's data.

[H]

[O] [Y] [T] [,] [0] [3] [−] [0] [5] [4] [−] [4] [3] [2] [1]

```
BUZZER        DEG        MEMO IN        4
H_
```

```
BUZZER        DEG        MEMO IN        4
HOYT,03-054-4321_
```

Finally, press [SHIFT] [▲ EXE] .

[SHIFT] [▲ EXE]

```
BUZZER      DEG      MEMO IN        4
HOYT,03-054-4321
```

The HOYT's data is now stored in Record 4 and the record numbers of the JONES' data and after have shifted down by one. Press the [EXE] key and confirm that the JONES' data is stored in Record 5.

[EXE]

```
BUZZER      DEG      MEMO IN        5
JONES,052-031-6221
```

The HOYT's data however, is also stored in Record 11.

[EXE] [EXE] [EXE] [EXE] [EXE] [EXE]
(Displays the HOYT's data in Record 11.)

```
BUZZER      DEG      MEMO IN        11
HOYT,03-054-4321
```

Since this data is no longer needed, erase it by the following procedure.

[←] (The EDIT mode is specified.)

```
BUZZER      DEG      MEMO IN EDIT   11
HOYT,03-054-4321_
```

[AC] (Clears the display.)

```
BUZZER      DEG      MEMO IN EDIT   11
```

[EXE] (Completes erasing. "EDIT" disappears.)

```
BUZZER      DEG      MEMO IN        11
```

The HOYT's data is now inserted between FOX and JONES and the "electronic telephone directory" is now arranged in alphabetical order.

Examples of data output before and after rearranging are as follows:

< Before rearranging >

```
 1 BROWN,03-021-1234
 2 ELLIS,011-041-7386
 3 FOX,06-021-6602
 4 JONES,052-031-6221
 5 MILLS,03-063-2935
 6 SMITH,0899-02-1007
 7 SULLIVAN,078-039-7132
 8 WATTS,0467-01-3569
 9 YOUNG,0425-01-0038
10 HOYT,03-054-4321
```

< After rearranging >

```
 1 BROWN,03-021-1234
 2 ELLIS,011-041-7386
 3 FOX,06-021-6602
 4 HOYT,03-054-4321
 5 JONES,052-031-6221
 6 MILLS,03-063-2935
 7 SMITH,0899-02-1007
 8 SULLIVAN,078-039-7132
 9 WATTS,0467-01-3569
10 YOUNG,0425-01-0038
```

7-8 Searching Using a Program

As long as the input data consists of only two items such as names and telephone numbers, it will not be particularly inconvenient to search by manual operation using the [MEMO] key as described in the previous sections. However, if the amount of data increases and the input items also increase to three, four or more, the length of a record will exceed 24 characters. It will then become necessary to move the display with the [MEMO] key or the cursor movement key in order to call the desired data.

This type of mass data can be handled easily if data is searched by using a BASIC program.

We will introduce a search method with a program using the data mentioned in the previous section (see page 179). This method can also be applied easily to data of three items or more.

- **Outline of the Program**

If this program is executed, the request "Name?" is displayed. If we enter the name of the person whose telephone number is to be searched, the telephone number will be displayed if it is stored in the DATA BANK. If it is not stored, "No Data!!" will be displayed.

Since this program is for displaying only, specify the MEMO IN mode ([MODE][9]) to enter the names or telephone numbers.

- **Program List**

```
10 DIM Z$(5)
20 RESTORE#
30 BEEP : INPUT "Name",$
40 IF $="" THEN 30
50 FOR J=1 TO 5
60 Z$(J)= MID$(J*7-6,7)
70 NEXT J
80 F=0
90 RESTORE# Z$(1)+Z$(2)+Z$(3)+Z$(4)
   +Z$(5),1,140
```

```
  100 READ# $,$
  110 F=1
  120 GOSUB 1000: PRINT $
  130 GOTO 90
  140 IF F=0 THEN PRINT "No Data !!"
  150 GOTO 20
 1000 FOR J=1 TO 3
 1010 BEEP 1: BEEP 0
 1020 NEXT J
 1030 RETURN
```

197 bytes

- **Variables Table**

Variable name	Contents
J	Control variable for the FOR ~ NEXT loop
F	Flag variable (F = 1 when the pertinent person exists. F = 0 if the pertinent person does not exist.)
Z$(1) ~ Z$(5)	Stores the name entered.
$	For reading data

- **Execution Example**

Operation

[MODE] [⊘] [R] [U] [N] [EXE]	Name?
[M] [I] [L] [L] [S]	MILLS_
[EXE]	03-063-2935
[EXE]	Name?
[N] [E] [W] [T] [O] [N]	NEWTON_
[EXE]	No Data !!
[EXE]	Name?

7-9 Application to Tabular Calculations

Vertical and horizontal tabular calculations as shown below are frequently required in practical calculations. Although this type of calculation was carried out using the array (see page 66), tabular calculations can be simplified further by considering the DATA BANK as being one large table.

n \ m	Product A	Product B	Product C	Product D	Horizontal total
Branch X	5329	4280	3602	2310	
Branch Y	2682	6313	4203	1128	
Branch Z	5113	3229	5176	965	
Vertical total					

- **Outline of the Program**

Since the size of the table will be requested when this program is executed, enter the number of horizontal items (m) and then the number of vertical items (n). Enter data; one column at a time vertically from the top. "Calculation" is displayed and calculation starts after all data are input. When calculation is over, the results are displayed in the order of vertical total and horizontal total.

- **Program List**

```
 10 BEEP : INPUT "(m x n)",A,B: ERAS
    E F: DIM F(A+1,B+1)
 20 FOR C=1 TO A
 30 FOR D=1 TO B
 40 PRINT "(";C;"x";D;")";
 50 BEEP : INPUT F(C,D)
 60 $="N"+ STR$(D)+","+ STR$(F(C,D))
 70 IF A=C THEN $=$+",0"
 80 WRITE# $
 90 NEXT D
100 WRITE# "M"+ STR$(C)+",0"
110 NEXT C
120 PRINT "Calculation";
140 FOR C=1 TO B+1
150 F(A+1,C)=0
```

```
160 NEXT C
170 FOR D=1 TO A+1
180 F(D,B+1)=0
190 NEXT D
200 RESTORE#
210 FOR C=1 TO A
220 FOR D=1 TO B
230 READ# $,F(C,D)
240 F(A+1,D)=F(A+1,D)+F(C,D)
250 IF A=C THEN WRITE# F(A+1,D)
260 F(C,B+1)=F(C,B+1)+F(C,D)
270 NEXT D
280 READ# $: WRITE# F(C,B+1)
290 NEXT C
300 PRINT
310 RESTORE#
320 FOR C=1 TO A
330 RESTORE# "M"
340 READ# $,E
350 PRINT $;"=";E
360 NEXT C
370 RESTORE#
380 IF A>1 THEN RESTORE# "M"+ STR$(A
    -1)
390 READ# $,$
400 FOR C=1 TO B
410 READ# $,E,E
420 PRINT $;"=";E
430 NEXT C
440 BEEP : PRINT "OVER"
450 PRINT : END
```

500 bytes

- **Variables Table**

Variable name	Contents
A	Number of horizontal items in the table
B	Number of vertical items in the table
C	Control variable for a FOR ~ NEXT loop
D	Control variable for a FOR ~ NEXT Loop
E	For reading data
F	For calculating vertical total
F(1) ~	The array for calculating horizontal total (Prepares number of vertical items in table.)
$	For preparing writing data to the DATA BANK and for reading space by the READ #$.

- **Program Execution Examples**

Calculate the vertical and horizontal totals using the table on page 197.

Operation

[MODE] [⊘] [R] [U] [N] [EXE]

4 [EXE] (Inputs number of horizontal items.)

3 [EXE] (Inputs number of vertical items.)

5 3 2 9 [EXE] (Inputs the value in the first vertical column.)

2 6 8 2 [EXE]

⋮

```
(m × n)?
?
( 1× 1)?
( 1× 2)?
( 1× 3)?
```

Input data in the 2nd, 3rd and 4th columns in sequential order. After all data are input, "Calculation" will be displayed. When calculation is over, the following results will be displayed.

⋮

965 [EXE]	Calculation
(Displays vertical total of the 1st column.)	M1= 13124
[EXE] (Displays vertical total of the 2nd column.)	M2= 13822
[EXE] (Displays vertical total of the 3rd column.)	M3= 12981
[EXE] (Displays vertical total of the 4th column.)	M4= 4403
[EXE] (Displays horizontal total of the 1st line.)	N1= 15521
[EXE] (Displays horizontal total of the 2nd line.)	N2= 14326
[EXE] (Displays horizontal total of the 3rd line.)	N3= 14483
[EXE] (Calculation is over.)	OVER
[EXE]	_

*Since all data are stored as memo data in the DATA BANK with this program, it will be necessary to first execute the NEW# command in the WRT mode when executing this program using new data.

7-10 Combining with the Function Memory

A practical method of using the DATA BANK function is to use it in combination with the Function Memory. Store formulas and equations in the DATA BANK and press the [MEMO] key to call the desired formula or equation. And calculate by storing the retrieved formula in the Function Memory by pressing the [IN] key.

Assume that the following formula and equation are stored in the DATA BANK.

1	QUADRATIC EQUATION
2	X=(−B+SQR(B↑2−4∗A∗C))/(2∗A): X=(−B−SQR(B↑2−4∗A∗C))/(2∗A)
3	HERON
4	S"s"=(A+B+C)/2:S=SQR(S∗(S−A)∗(S−B)∗(S−C))
	⋮

A formula to calculate the root of a quadratic equation and Heron's formula are stored in the DATA BANK.

Now, calculate the root of the quadratic equation. Press [Q][MEMO] to search the quadratic equation.

[Q][MEMO]

```
QUADRATIC EQUATION
```

After confirming the name, press the [EXE] key and the calculation formula is displayed.

[EXE]

```
X=(-B+SQR(B↑2-4*A*C))/(2
```

Then store the calculation formula in the Function Memory.

[IN]

```
_
```

Now we are ready to calculate.
Input 2 for a, -3 for b and -10 for c as an example.

[CALC]	B ?
−3[EXE]	A ?
2[EXE]	C ?
−10[EXE]	X= 3.108495283
[EXE]	X=-1.608495283

This function is highly convenient since calculating formulas stored in the DATA BANK can be applied to the Function Memory as they are.

Note:
When storing a formula name and calculating formula in the DATA BANK, do not store them in one record. Store the calculating formula in the record following the record in which the formula name is stored.

Appendix

Character Code Table

	(Space)	+	−	*	/	↑	!	"	#	$	>	≥	=	≤	<	≠
Numerals	0	1	2	3	4	5	6	7	8	9	.	π	)	(	$\bar{E}$	E
Capital letters	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	Q	R	S	T	U	V	W	X	Y	Z	r	σ				
Small letters	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
	q	r	s	t	u	v	w	x	y	z						
Symbols	?	,	;	:												
Graphic symbols	○	Σ	°	△	@	×	÷	♠	←	♥	♦	♣	μ	Ω	↓	→
	%	¥	□	[	&	_	'	.	]	■	\	▒				

*The characters and symbols in the above table are lined in sequence, with the space being the smallest and ▒ being the largest. (" ▒ " can be displayed by pressing [SHIFT][=] in the extension mode.)

Numeric Functions

Name of function	Format	Function and input range	
Trigonometric function	SIN (Numeric expression) * hereafter X	sin	$\lvert X\rvert < 1440°$ (8π rad, 1600gra)
	COS (X)	cos	$\lvert X\rvert < 1440°$ (8π rad, 1600gra)
	TAN (X)	tan	$\lvert X\rvert < 1440°$ (8π rad, 1600gra) except when $\lvert X\rvert$ is odd multiple of $90°$ ($\pi/2$ rad, 100gra)
Inverse trigonometric function	ASN (X)	$\sin^{-1}$	$\lvert X\rvert \leqq 1, -90° \leqq ASN \leqq 90°$ (rad: $-\pi/2 \leqq ASN \leqq \pi/2$, gra: $-100 \leqq ASN \leqq 100$)
	ACS (X)	$\cos^{-1}$	$\lvert X\rvert \leqq 1,\ 0° \leqq ACS \leqq 180°$ (rad: $0 \leqq ACS \leqq \pi$, gra: $0 \leqq ACS \leqq 200$)
	ATN (X)	$\tan^{-1}$	$-90° \leqq ATN \leqq 90°$ (rad: $-\pi/2 \leqq ATN \leqq \pi/2$, gra: $-100 \leqq ATN \leqq 100$)
Hyperbolic function	HYP SIN (X)	sinh	$\lvert X\rvert \leqq 230.2585092$
	HYP COS (X)	cosh	$\lvert X\rvert \leqq 230.2585092$
	HYP TAN (X)	tanh	$\lvert X\rvert < 10^{100}$
Inverse hyperbolic function	HYP ASN (X)	$\sinh^{-1}$	$\lvert X\rvert < 5 \times 10^{99}$
	HYP ACS (X)	$\cosh^{-1}$	$1 \leqq X < 5 \times 10^{99}$
	HYP ATN (X)	$\tanh^{-1}$	$\lvert X\rvert < 1$
Square root	SQR (X)	$\sqrt{x}$	$X \geqq 0$
Cube root	CUR (X)	$\sqrt[3]{x}$	$\lvert X\rvert < 10^{100}$
Power	X ↑ X	x^y	$x < 0 \rightarrow y$: natural number
Exponential function	EXP (X)	e^x	$-10^{100} < X \leqq 230.2585092$

Name of function	Format	Function and input range	
Common logarithm	LOG (X)	$\log_{10} x$	X > 0
Natural logarithm	LN (X)	$\log_e x$	X > 0
Integer	INT (X)	$[x]$	Gives maximum integer not exceeding X (equal to Gaussian function $[x]$)
Fraction	FRAC (X)	FRAC	Gives decimal portion of X
Absolute value	ABS (X)	$\lvert x \rvert$	Gives absolute value of X
Sign	SGN (X)	sgn x	1 when X > 0 0 when X = 0 - 1 when X < 0
Rounding off	RND (X, Number of digits)*	RND(	Gives the value of X which is rounded off at the specified digit. |Number of digits| < 100
Random numbers	RAN #	RAN #	Generates a 10-digit random number. 0 < RAN # < 1
π	π	π	Gives approximate value of ratio of circle circumference to diameter.
Decimal → sexagesimal conversion	DMS$ (X)*	DMS$ (	Converts decimal number given as X into sexagesimal character string in degrees, minutes and seconds. $\lvert X \rvert < 10^5$
Sexagesimal → decimal conversion	DEG (deg. [, min. [, sec.]])*	DEG (	DEG $(x, y, z) = x + y/60 + z/3600$. $\lvert \text{DEG}(x, y, z) \rvert < 10^{100}$
Decimal → hexadecimal conversion	HEX$ (X)*	HEX$ (	Converts value of X into 4-digit hexadecimal character string. -32769 < X < 65536

Name of function	Format	Function and input range	
Hexadecimal → decimal conversion	&H Hexadecimal character string	&Hx	Character string contains hexadecimal number within 4 characters.
Factorial	FACT (X)	$x!$	$0 \leqq X \leqq 69$ (0 and positive integer)
Permutation	NPR (n, r)*	nPr	$0 \leqq r \leqq n < 10^{10}$ (0 and positive integer)
Combination	NCR (n, r)*	nCr	$0 \leqq r \leqq n < 10^{10}$ (0 and positive integer)
Rectangular → polar coordinate transformation	POL (X, Y)* X, Y: numeric expressions	POL (	$\lvert X\rvert < 10^{100}$, $\lvert Y\rvert < 10^{100}$, $\lvert X\rvert + \lvert Y\rvert \neq 0$ r is given as a function value for assignment to variable X while value of θ is assigned to variable Y.
Polar → rectangular coordinate transformation	REC (r, θ)* r, θ: numeric expressions	REC (	$0 \leqq r < 10^{100}$, $\lvert\theta\rvert < 1440°$ (8π rad, 1600 gra) Gives x as a function value for assignment to variable X while value of y is assigned to variable Y.

Note:

In the case of asterisked functions, parameters must be parenthesized.

*Certain combinations or permutations may cause errors due to overflow during internal calculations.

Error Messages

Error code/ Meaning	Cause	Countermeasure
Error 1 Memory over or system stack over	• Unable to write programs or expand variables due to insufficient capacity of free area. • Calculating area (stack) unable to hold formula since the formula is excessively complex. • Unable to write data in the data bank since capacity is insufficient. • Nine or more arrays were declared.	• Erase unnecessary programs with the NEW command or reduce the number of variables. • Separate and simplify the formula. • Clear the array
Error 2 Syntax error	• Format error in the program or formula. • The formats of left side and right side in the assigned statement differ. (Such as character type and numeric type) • Attempted to read character in a numeric variable with READ/ READ#. • Character string operation exceeded 62 characters.	• Correct the error in the input program. • Change numeric variable to character variable or check for character (including space) in the DATA statement. • Shorten the character string.
Error 3 Mathematical error	• When the calculation result of a formula exceeds 10^{100}. (Overflow) • When arguments are outside the input range of numeric functions. • When the results are uncertain or impossible. (Attempted to divide with a 0)	• Correct the formula or the data. • Check the data.

Error code/ Meaning	Cause	Countermeasure
Error 4 Undefined error	• No jump destination for the GOTO or GOSUB statements. • There is no data to be read with READ/READ# or RESTORE#. • The line number specified with RESTORE does not exist.	• Specify the correct jump destination. • Write data • Correct the line number.
Error 5 Argument error	• When the argument is outside the input range of commands and functions requiring arguments. • The subscript in the array is outside the input range. • Attempted to specify two arrays with the same name but different subscripts.	• Correct the argument error. • Change the subscript. • Change the array name.
Error 6 Variable error	• Attempted to use a variable that was not added. • Attempted to use the same variable name for a numeric variable and a character variable. • Attempted to use an array name subscript that was not declared.	• Expand the variables with the DEFM statement. • Change the variable name for the numeric variable and character variable. • Use after declaring the array or correct the array name subscript.
Error 7 Nesting error	• When the RETURN statement is used other than when executing a subroutine. • When the FOR statement and NEXT statement do not correspond or when the variable of the NEXT statement does not match that of the FOR statement. • When the subroutine nesting (calling a subroutine from a subroutine) exceeds eight levels.	• Correspond GOSUB ~ RETURN or FOR ~ NEXT correctly. • Correct the subroutine or FOR loop nesting level within the range.

Error code/ Meaning	Cause	Countermeasure
Error 7 Nesting error	• When the FOR loop nesting (inserting a loop within a loop with nesting form) exceeds four levels. • The CLEAR statement was used in the FOR ~ NEXT loop.	• Move the CLEAR statement outside the FOR ~ NEXT statement.
Error 8 Protect error	• When the following occurs with the password specified. 1) Input of a different password 2) Execution of a prohibited command 3) Editing of a program 4) Loading programs with different passwords. 5) Inputting data in the data bank 6) Calling data from the data bank	• Clear the password.
Error 9 Option error	• SAVE, SAVE# or PUT command was executed without an interface. • When the signal input with the LOAD, LOAD# or GET command is erratic and cannot be loaded. • A printer is not connected. • When the printer is not sufficiently charged. • Paper jammed in the printer.	• Connect a tape recorder. • Reduce the playback volume of the tape recorder. • Set the tone control of the tape recorder to middle position. • Change the cassette tape. • Clean the head of the tape recorder. • Charge the printer. • Remove the paper jammed in the printer.

Specifications

Type:

FX-730P

Fundamental Calculation Functions:

Negative numbers, exponents, parenthetical addition, subtraction, multiplication and division (with priority sequence judgement function – true algebraic logic).

Built-in Functions:

Trigonometric/inverse trigonometric functions (angle units of degrees/radians/grads), hyperbolic/inverse hyperbolic functions, logarithmic/exponential functions, square roots, cube roots, powers, conversion to integer, deletion of integer portion, absolute values, signs, designation of number of significant digits, designation of number of decimal places, decimal ↔ sexagesimal conversions, decimal ↔ hexadecimal conversions, rectangular ↔ polar coordinate transformations, factorials, permutations, combinations, rounding, random number generations, π.

Statistical Calculation Functions:

Number of data, sum of x, sum of y, sum of x^2, sum of y^2, sum of $x \cdot y$, mean of x, mean of y, standard deviation of x (2 types), standard deviation of y (2 types), linear regression constant term, linear regression coefficient, correlation coefficient, estimated value of x, estimated value of y.

Commands:

INPUT, PRINT, GOTO, ON ~ GOTO, FOR ~ TO ~ STEP ~ NEXT, IF ~ THEN, GOSUB, ON ~ GOSUB, RETURN, READ, DATA, RESTORE, STOP, END, RUN, LIST, LIST V, LIST ALL, MODE, SET, CLEAR, NEW, NEW ALL, ERASE, DIM, DEFM, PASS, REM, BEEP, LET, SAVE, SAVE ALL, LOAD, LOAD ALL, PUT, GET, VERIFY, NEW#, LIST#, LOAD#, SAVE#, READ#, WRITE#, RESTORE#, STAT, STAT CLEAR, STAT LIST.

Program Functions:

DMS$, KEY$, CSR, LEN, MID$, STR$, VAL, HEX$.

Calculation Range:

$\pm 1 \times 10^{-99} \sim \pm 9.999999999 \times 10^{99}$ and 0. Internal operation uses 12-digit mantissa.

Program System:

Stored system

Program Language:

BASIC

RAM Capacity:
Standard 7,520 bytes (system area: 672 bytes) (15,712 bytes with the RP-8 RAM expansion pack)

Number of Program Areas:
Maximum 10 (P0 through P9)

Number of Variables:
26 fixed variables, exclusive character variable and array variables.

Number of Stacks:
Subroutine: 8 levels
FOR ~ NEXT loop: 4 levels
Numeric values: 6 levels
Operators: 12 levels

Number of Display Digits:
10-digit mantissa (including minus sign) and 2-digit exponent
Display contents: **BUZZER, EXT, [S], [F], RUN, WRT, DEG, RAD, GRA, DEFM, [MEMO], [IN], [EDIT], TRACE ON, PRT ON, STOP**

Display Elements:
24-digit dot matrix liquid crystal display

Main Components:
C-MOS VLSI and others

Power Supply:
2 lithium batteries (CR2032) for the mainframe
1 lithium battery (CR1220) for memory backup

Power Consumption:
Maximum 0.07W

Battery Life:
1) Continuous program execution: Approx. 86 hours
2) Continuous display of 5555555555 at 20°C (68°F): Approx. 205 hours
5.5 months when unit is used 1 hour per day.
*Note: 1 hours includes 10 minutes of condition 1) and 50 minutes of condition 2).
Memory protection battery: Approx. 2 years (with main batteries installed)

Auto Power-off:
Approximately 6 minutes

Ambient Temperature Range:
0°C to 40°C (32°F to 104°F)

Dimensions:

12.5H × 165W × 77Dmm
(½"H × 6½"W × 3"D)

Weight:

139 g (4.9 oz) including batteries.

Index

A

&H 164
ABS 150
ACS 145
ALL RESET button 10
Angle unit 8
Argument 144
Array variables 57, 133
ASCII 18
ASN 145
ATN 145

B

BASIC 51
BEEP 130
Beep Sound 130
Bug 67
BUZZER symbol 16

C

Character expression 98
Character Functions 140
Character variables 57, 111
CLEAR 107
Comparison expression 118
Control variable 119
Correlation coefficients 40
COS 144
CSR 115
CUR 149
Cursor 6, 16
Cursor movement keys 6

D

DATA 123
DATA BANK Commands 166
Debugging 67
Decimal portion 31
DEFM 131
DEG 161
Degrees 8
Delete key 7
DIM 133
DMS$ 162

E

Edit 101
EDIT mode 191
END 108
Engineering 7, 35
EOX 158
EOY 159
Equal key (=) 6
ERASE 135
Error 67
Error code 67
Error message 20, 67
Estimated value 42, 158
Exclusive character variables 57, 126
EXP 148
Exponent 6, 21
Exponent display 21
Expression 98
Extension mode 5

F

FACT 155
Factorial 26, 155
File name 75, 105
FOR ~ TO ~ [STEP] NEXT 119
FRAC 152
Free area 53, 62
Function key 3, 27

G

Gaussian function 25, 151
GET 128

GOSUB 121
GOTO 116
GRA (grades) 8, 144

H

Hexadecimal number 32
HEX$ 163
HYPACS 147
HYPASN 147
HYPATN 147
HYPCOS 146
HYPSIN 146
HYPTAN 146

I

IF ~ THEN 118
INPUT 111
Insert key 7, 19
INT 151
Integer portion 31

K

KEY$ 113

L

LEN 140
LET 109
Line number 52, 64
Linear regression coefficient 41, 158
Linear regression constant term 41, 158
Linear regression expression 41, 158
LIST 101
LIST # 166
LIST V 101
LN 148
LOAD [ALL] 106
LOAD # 169
LOG 148
Loop 120

M

Main routine 65
Mantissa 23
Manual operation 24
MEMO IN mode 8, 178
Message 48, 111
MID$ 141
MODE 8, 136
Multistatement 110

N

NCR 157
Nesting 69, 120
NEW [ALL] 99
NEW # 166
NPR 156
Null 107, 113
Number of bytes 53, 64
Number of decimal places 34, 139
Number of significant digits 34, 139
Numeral keys 3
Numeric expression 98
Numeric variables 57, 111

O

ON ~ GOSUB 122
ON ~ GOTO 117
One-variable statistics 39
Output element 114

P

Paired variable statistics 39
Parameter 98

Parity check 107
PASS 103
Password 103
Peripherals 9, 72
Permutation 26, 156
POL 154
Polar coordinate 33, 153
PRINT 114
PRINT mode 8, 80
Priority Sequence 23
PUT 126

R

RAD (radians) 8, 144
RAN# 160
Random numbers 25, 160
READ 124
READ# 170
REC 153
Rectangular coordinate 33, 153
Regression curve 41
REM 110
RESTORE 125
RESTORE# 172
RETURN 122
RND 152
RUN 100
RUN mode 8, 16, 55

S

SAVE [ALL] 105
SAVE# 168
SET 139
Sexagesimal 25, 31, 161
SGN 150
Simple variables 57
SIN 144
SQR 149
Standard deviation 38
STAT 137
STAT CLEAR 137
STAT LIST 138
Statements 52
Statistical data 36
STOP 108
STR$ 143
Subroutine 65

T

TAN 144
TRACE Mode 71

V

VAL 142
Variable Expansion 60, 131
Variables 22, 57
VERIFY 107

W

WRITE# 174
WRT mode 8, 54

GUIDELINES LAID DOWN BY FCC RULES FOR USE OF THE UNIT IN THE U.S.A. (not applicable to other areas).

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- reorient the receiving antenna
- relocate the computer with respect to the receiver
- move the computer away from the receiver
- plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful: "How to Identify and Resolve Radio-TV Interference Problems". This booklet is available from the US Government Printing Office, Washington, D.C., 20402, Stock No. 004-000-00345-4.