

Personal Computer
FX-750P
OWNER'S MANUAL



CASIO®

Personal Computer
FX-750P
OWNER'S MANUAL

CASIO®

- 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 copyright law.
- CASIO Computer Co., Ltd. shall not be held responsible for any damages or losses resulting from the use of this manual.

Introduction

We thank you very much for purchasing the Casio FX-750P.

Regular handheld computers are generally provided with a memory unit in which data and programs can be entered, retrieved and erased. A major feature of the FX-750P, however, is its RAM card which serves as this memory unit. Since the RAM card can be freely inserted or removed from the main frame, replacing or saving data is considerably easier to perform compared with the conventional cassette tape. The RAM card is also provided with a backup battery to retain the programs and data stored in the card.

Since the FX-750P can be loaded with up to 2 RAM cards, address logs, customer lists and reports can be speedily prepared on the spot if data and programs are prepared in advance by card units.

Another feature of the FX-750P is its many functional characteristics. Abundant functions such as numerical functions, statistical functions and others enable complex scientific, technical and statistical calculations to be performed with a simple operation.

This manual is arranged for easy understanding of the functions and for full utilization of the FX-750P in studying business and science and engineering subjects.

Please use this manual as a guide to enable the FX-750P to serve you faithfully for many years.

- Chapter 1 Installing the batteries, inserting and removing the RAM card and precautions in handling.
- Chapter 2 Usage as a sophisticated scientific calculator
- Chapter 3 Fundamental of the BASIC program
- Chapter 4 Explanation of BASIC commands together with actual examples
- Chapter 5 Command reference
- Chapter 6 Practical library for immediate use
- Chapter 7 List of functions convenient as a guide

Prior to Operation

This FX-750P has undergone rigid inspection processes based on Casio's high level electronics technology and quality control. To realize long years of service from this computer, please take the following points into consideration when using this unit.

■ Precautions in Use

- Never attempt to disassemble the main frame or the RAM card as they are composed of high precision electronic components. Refrain from subjecting the computer to shocks by throwing or dropping or by strong static electricity, and also refrain from subjecting the computer to sharp changes in temperature. Furthermore, when the temperature is low, display response may be slow or the display may blank out but this will return to normal if the temperature rises to normal.
- The connector portion on the computer is exclusively for the FA-20 character printer with cassette interface so no other devices should be plugged into this connector. Always cap the connector when using the computer independently to prevent touching the contacts unnecessarily.
- Never touch the contacts of the RAM card. When the RAM card is not being used in the computer, move the pawl on both sides of the RAM card and cover the contacts. Then, slip into the card case supplied and store.
- Since the computer is not provided with a built-in RAM, always insert a RAM card into the RAM card slot before using the computer.
- When replacing the batteries, always remove the RAM cards first to avoid changes of the RAM card's contents.
- If the lock switch retaining the RAM card is not in locked position, power will not go on even if the power switch is set to ON. Always set the lock switch to LOCK position when using.
- The memory contents may change or key operation may not be possible if the computer or the RAM card is subjected to strong static electricity. If this occurs, remove the batteries temporarily and insert again.

- Always set the power switch of the computer to OFF when connecting to optional devices.
- Replace the batteries once every 2 years regardless of whether the computer is used or not. Never leave exhausted batteries in the computer as battery fluid may leak and cause damage to the computer.
- Avoid using volatile fluids such as thinner or benzine to clean the computer. Wipe off with a soft dry cloth or with cloth dipped in a neutral detergent.
- Do not turn off the power switch when executing a program or when performing a calculation.
- Since the computer is made up of precision electronic parts, avoid giving a strong shock while a program is being executed; otherwise the program execution may be stopped or the memory content may be changed.
- When a malfunction occurs, contact the store where it was purchased or a nearby dealer.
- Before seeking service, please read this manual again and check the power supply as well as the program for logic errors.

CONTENTS

CHAPTER 1	MAIN FRAME + RAM CARD = FX-750P	1
1-1	Power Supply and Battery Replacement	2
1-2	RAM Card	4
	1. Features of the RAM Card	4
	2. Method of Using the RAM Card and Precautions	5
	3. Contents of a RAM Card (Internal Memory Allocation)	12
	4. Types of RAM Cards and Number of Bytes	14
	5. Method of Application with RAM Card Combinations	15
	6. System Area and User Area When Using 2 RAM Cards	18
	7. Error Messages after Inserting or Removing of a RAM Card	21
1-3	Operating Your FX-750P	23
	Understanding the Display	24
	Key Functions	26
1-4	Option	31
	Printer Section	32
	Cassette Interface Section	32
	Data Storing and Loading (PUT, GET)	35
CHAPTER 2	CALCULATION FUNCTIONS	37
2-1	Prior to Calculations	38
2-2	Manual Calculations and Input Corrections	42
2-3	Numerical Functions	46
2-4	Character Functions	53
2-5	Statistics Calculations	55
2-6	Scientific Constants	61

CHAPTER 3	“BASIC” REFERENCE	65
3-1	Program	66
3-2	The Flow Chart	67
3-3	Programming	71
3-4	Variables	74
3-5	Program Input and Execution	79
3-6	Program Modification	83
3-7	Precautions for Practical Programs in Chapter 4	86
CHAPTER 4	APPLICATIONS	89
4-1	Foreign Currency Conversions	90
4-2	Circuit Calculations (Impedance calculation)	104
4-3	Analysis of Variance	118
4-4	Regression Analysis (Linear regression)	127
4-5	Management of the Household Budget	134
4-6	Weekly Schedule	150
4-7	\bar{X} -R Control	168
4-8	ABC Analysis	172
4-9	Chemical Calculations (Radiation attenuation calculation)	180
4-10	Number Guessing Game	194
CHAPTER 5	COMMAND REFERENCE	203
5-1	Manual Commands	205
	CONT	205
	DELETE	207
	EDIT	208

CONTENTS

	LIST, LLIST	209
	LOAD [ALL], LOAD ,A [,M]	210
	NEW [ALL]	213
	PASS	214
	PROG	215
	RUN	215
	SAVE [ALL], SAVE ,A	216
	STAT LIST, STAT LLIST	217
	SYSTEM	218
	VERIFY	219
5-2	Program Commands	220
	ANGLE	220
	BEEP	221
	CHAIN	222
	CLEAR	223
	CLS	224
	DIM	225
	END	227
	ERASE	227
	FOR ~ TO ~ STEP/NEXT	228
	GET	230
	GOSUB/RETURN	231
	GOTO	232
	IF ~ THEN ~ ELSE	233
	INPUT	234
	LET	235
	LOCATE	236
	ON ~ GOSUB	237
	ON ~ GOTO	238
	PRINT, LPRINT	239
	PRINT ON, PRINT OFF	241

	PUT	242
	READ, RESTORE, DATA	243
	REM	245
	STAT	246
	STAT CLEAR	247
	STOP	247
	TRON, TROFF	249
	WAIT	250
5-3	Character Functions	251
	ASC	251
	CHR\$	251
	VAL	252
	STR\$	253
	DMS\$	254
	HEX\$	255
	LEFT\$	256
	RIGHT\$	257
	MID\$	258
	LEN	259
	INKEY\$	259
5-4	Control Functions	260
	TAB	260
	USING	261
5-5	Numerical Functions	263
	SIN, COS, TAN	263
	ASN, ACS, ATN	264
	HYP SIN, HYP COS, HYP TAN	265
	HYP ASN, HYP ACS, HYP ATN	266
	SQR	266
	LGT, LOG	267
	EXP	267

CONTENTS

	ABS	268
	INT	268
	FRAC	269
	SGN	269
	ROUND	270
	DEG	271
	PI	272
	RND	272
5-6	Statistical Functions	273
	CNT	273
	SUMX, SUMY	273
	SUMXY	274
	SUMX2, SUMY2	274
	SDX, SDY	275
	SDXN, SDYN	275
	LRA	276
	LRB	276
	COR	277
	EOX	277
	EOY	278
5-7	Others	279
	&H	279
CHAPTER 6	PRACTICAL LIBRARY	281
6-1	Achievement Processing	282
6-2	Tabulation Program	291
6-3	Memo Card	299
6-4	Testing the Method of Appeal and Effectiveness of DM	304
6-5	Monster Road Game	308

CHAPTER 7	LIST OF FUNCTIONS AND STANDARDS	311
7-1	NUMERICAL FUNCTIONS	312
7-2	STATISTICS FUNCTIONS	314
7-3	CONTROL FUNCTIONS	314
7-4	CHARACTER FUNCTIONS	315
7-5	SCIENTIFIC CONSTANTS	315
7-6	MANUAL COMMANDS	316
7-7	PROGRAM COMMANDS	318
7-8	ERROR MESSAGES	320
7-9	CHARACTER CODE TABLE	325
7-10	LIST OF RESERVED WORDS	326
7-11	LIST OF FLOWCHART SYMBOLS	327
	SPECIFICATIONS	329
	INDEX	332

CHAPTER

1

MAIN FRAME + RAM CARD = FX-750P

This chapter includes battery replacement of the main frame and the RAM card, inserting and removing the RAM card as well as its use precautions, configuration of the main frame, etc.

It is important to understand these to utilize your FX-750P.

1-1 Power Supply and Battery Replacement

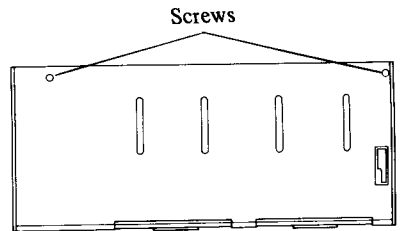
The FX-750P uses 2 lithium batteries (CR2032) as its power supply. Battery life is about 120 hours when using the main frame only but will be shorter if the buzzer is used frequently. If the display cannot be brightened with the contrast control (see page 25), it means that the batteries are exhausted so they should be replaced as soon as possible. Always replace both batteries at the same time.

* Always replace the batteries every 2 years even if the computer is not used as there will be danger of battery fluid leakage.

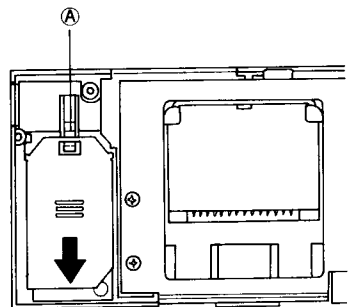
■ Replacing the Batteries

If RAM cards are set in the computer, remove the RAM cards before replacing the batteries. After replacing the batteries, return the RAM card to its same previous slot. (See page 7)

- 1) After setting the power switch to OFF, remove the 2 screws on the back and remove the back cover. (Use a good quality screwdriver with a fine tip.)

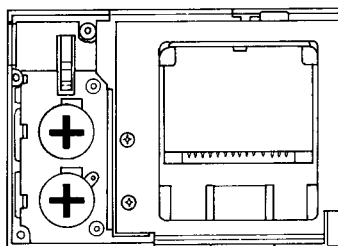


- 2) Slide off the battery retainer cover in the direction of the arrow while pushing down on clip (A) shown in the diagram.



3) Remove both of the exhausted batteries. (The batteries can be removed easily by facing the battery compartment downward and tapping lightly.)

4) Wipe off the new batteries with a dry cloth and insert with the positive (+) side facing up. Make sure that the (+) side of both batteries face up.



5) Slide in and close the battery retainer plate.

6) Fit the back cover and tighten with the 2 screws.

* **Do not throw exhausted batteries into a fire because they may burst.**

* **Keep the battery away from children. If it is swallowed, contact your doctor immediately.**

1-2 RAM Card

1. Features of the RAM Card

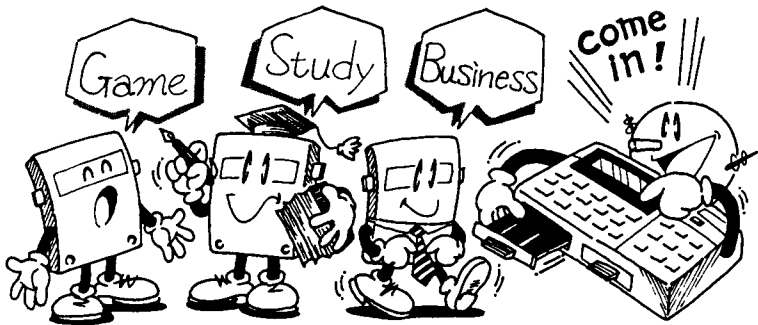
Although regular handheld computers are provided with a built-in memory to store data and programs, the FX-750P uses an easily removable “RAM card” in place of this memory. This is the first of this nature made on handheld computers and is highly convenient for saving and exchanging data and programs.

Cassette tapes were used to save and exchange data and programs in conventional handheld computers but the “RAM card” has eliminated working with cassette tapes and has greatly enhanced the usage of handheld computers. This system is highly convenient since data and programs can be easily exchanged and processed with a card unit. Since a battery built into the RAM card backs up the contents stored in the card, the contents will not be lost even if the RAM card is removed from the main frame.

RAM cards are available in 2 types which are the RC-4 (4K bytes) and the RC-2 (2K bytes) and up to 2 of any combination can be loaded in the computer.

The range of use may be expanded considerably if data and program are input in respective RAM cards since the cards can then be loaded and processed at any time and place necessary.

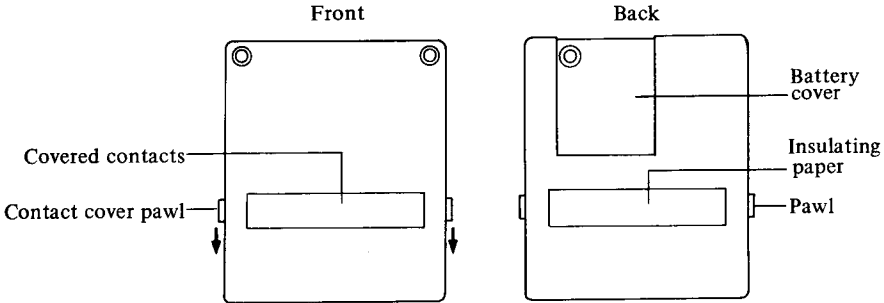
* This computer cannot be used unless a RAM card is loaded since it does not have a built-in RAM area.



2. Method of Using the RAM Card and Precautions

■ Precautions in Handling

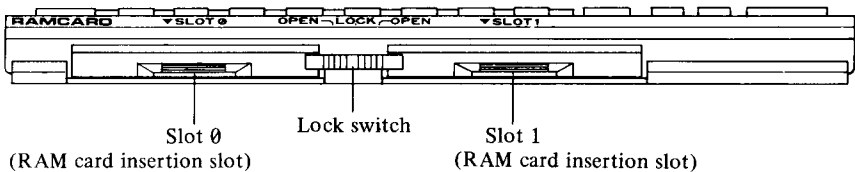
Although there are 2 types of RAM cards consisting of the RC-4 (4K bytes) and RC-2 (2K bytes) which are exclusive to the FX-750P, the method of handling is the same.



- Do not touch the contacts.
Gripping the pawl on both sides and pulling in the direction of the arrow will expose the contacts. Be careful not to touch these contacts with the finger or with any metallic object since doing so may render the RAM card ineffective. The contacts should always be covered when the RAM card is removed from the main frame.
- Do not subject the RAM card to strong electrostatic charges since this may change the stored data or make key inputs impossible. If this occurs, remove and reinsert the battery in the RAM card (in this case, all stored data are cleared).
- Never attempt to twist, bend or disassemble the RAM card.
- When removing the RAM card, always place in the case supplied to protect it from dust and keep in a place away from direct sunlight.

- The RAM card is provided with a lithium battery for memory backup. Since the stored data will be lost if the battery is removed, first transfer the stored contents onto a cassette tape if it becomes necessary to replace the battery. The contents can then be loaded in the RAM card again after replacing the battery. (See page 33)
- Never remove the insulating paper as it serves to protect the contacts in the event that the RAM card is inserted backwards.
- Always ensure that a battery is in the RAM card when using.
- Do not use the RAM card in computers other than Casio's exclusively-used RAM card computers.

■ Precautions in Removing and Inserting the RAM Card

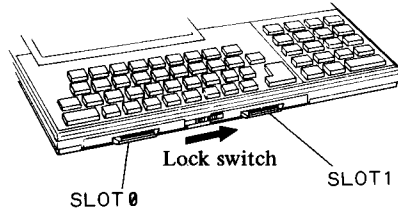


- Although 2 RAM cards can be loaded in this computer, the RAM card must always be inserted in slot 0 when using the main frame with only 1 RAM card. Key input will otherwise not be possible.

■ Setting the RAM Card in the Computer

Any combination of up to 2 RAM cards can be used in this computer. When using only 1 RAM card, however, it must always be inserted in slot 0. The method of insertion is the same for both slots 0 and 1.

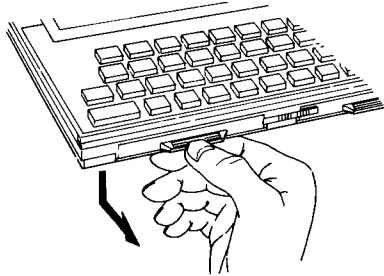
- 1) Set the computer power switch to OFF.
- 2) Slide the lock switch away from the slot in which the RAM card is to be set. (The computer power supply will be in OFF state at this time.)



Caution:

Pulling the card holder forcibly without sliding the lock switch will damage the lock switch.

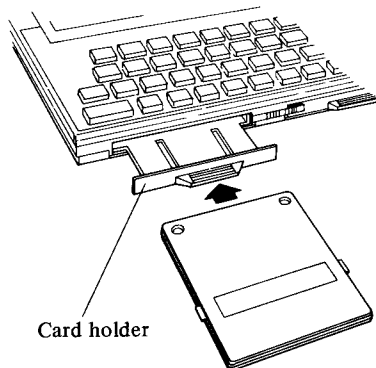
- 3) Pull out the card holder while pushing down on the lip of the holder.



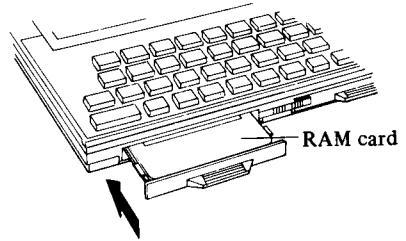
Caution:

The holder can only be pulled out part way. Attempting to pull the holder out forcibly will damage the holder.

- 4) Face the RAM card up (contacts facing up) and insert in the card holder with the contact cover in place.

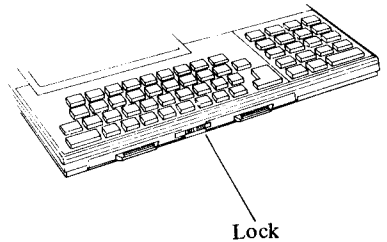


5) Push down lightly on the lip of the card holder so the RAM card will be level and put in completely.



6) Lift the card holder slightly and push in the direction of the arrow until a click is heard indicating that the holder is locked in place.

7) Slide the lock switch back to the center locking position.



Caution:

Power will not be supplied even if the computer power switch is turned on if the lock switch is not in locked condition. Therefore, do not forget to lock this switch after replacing a RAM card.

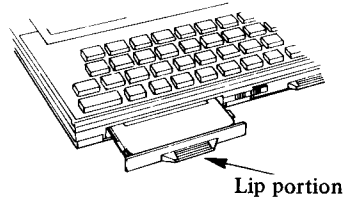
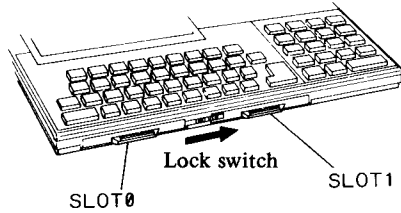
8) The computer can now be used at any time by setting the power switch to ON.

* If a message other than "READY P0" is displayed, it will mean that the RAM card is not correctly set. Refer to page 21 and reset the card correctly.

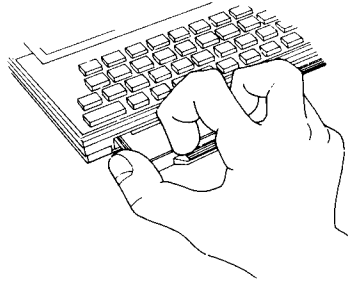
■ Removing the RAM Card

(The method is the same for both slots 0 and 1.)

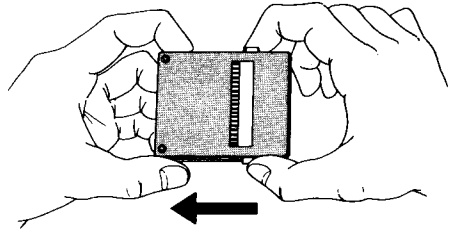
- 1) Set the computer power switch to OFF.
- 2) Slide the lock switch away from the slot from which the RAM card is to be removed to release the lock.
- 3) Pull the card holder out while pressing down on its lip portion.



- 4) Grip the both sides of the RAM card and pull out while pushing down the lip of the card holder lightly. Care should be taken not to touch the contacts at this time.



5) Since the contacts of the RAM card will be exposed at this time, cover the contacts again immediately by sliding the pawl on both sides of the cover.



Caution:

When not using a RAM card, always keep it in the RAM card case supplied.

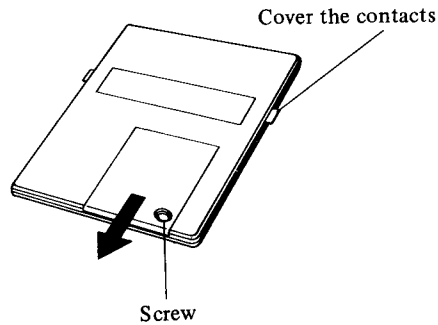
■ **Replacing the RAM Card Battery (for Memory Backup)**

A RAM card uses 1 lithium battery (CR2016) as a power supply for memory backup. Although the battery life is about 1 year for the RC-4, 2 years for the RC-2, this will be extended to about 2 years for the RC-4 when used in the FX-750P since it will be backed up by the computer batteries. However, as battery fluid may leak after 2 or more years of use, always replace the battery within 2 years.

- **Method of Replacing the RAM Card Battery**

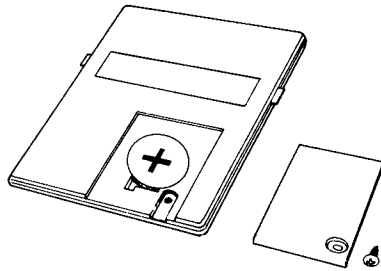
Leave the contacts covered when replacing a RAM card since touching the contacts will lead to trouble.

1. Remove the battery cover by removing the screw and pushing the cover in the direction of the arrow. Then pull out the exhausted battery.



2. Wipe off the new battery with a dry cloth and insert with the plus (+) side up, and remount the cover. Always ensure that the polarity is correct when inserting.

* Never throw the exhausted battery into a fire. It will be highly dangerous since the battery may explode.



* **Keep the battery away from children.**

If it is swallowed, contact your doctor immediately.

3. Contents of a RAM Card (Internal Memory Allocation)

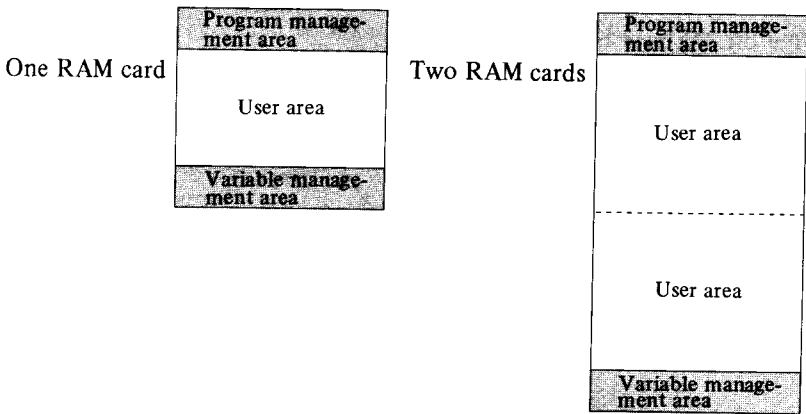
Since the memory configuration of a card will greatly affect storage capacity, it will be important to initially ascertain the general size of the program and the amount of data when creating a program or a data format.

We shall now take a peek inside a RAM card.

The interior of the RAM card is divided into 3 broad blocks.

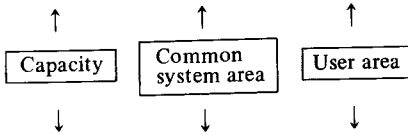
The upper block is the program management area for BASIC, the lower block is allocated for use with variables and the center block is the user area (where the program and data are stored).

An interesting point here is that 1 upper and lower area (system area) will suffice regardless of whether 1 or 2 RAM cards are used. This therefore means that a greater user area can be realized from the standpoint of the number of bytes by using 2 cards.



The number of bytes in the user area will differ depending on the combination of the RAM cards. A 4K byte RAM card is used in the following example to explain the number of bytes.

One 4K byte card: $4096 \text{ bytes} - 1296 \text{ bytes} = 2800 \text{ bytes}$



Two 4K byte cards: $8192 \text{ bytes} - 1296 \text{ bytes} = 6896 \text{ bytes}$

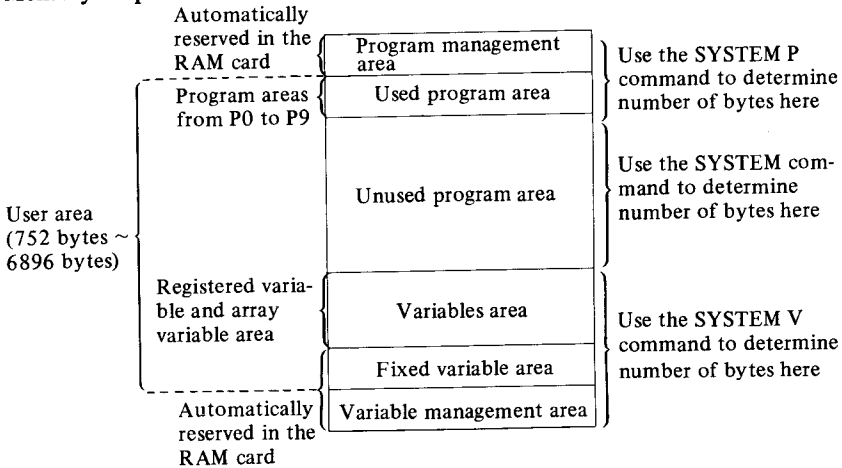
Common system area:

- { Fixed variable area = 208 bytes
- { System area = 1088 bytes
- { Program management area = 632 bytes
- { Variable management area = 456 bytes

Since the RAM card itself has an independent program management area and a variable management area, a different program can be executed immediately by simply changing the RAM card.

If the RAM card in slot 1 is used for data and this RAM card is replaced with several cards, care will be required on program area and data area combinations. (See page 19)

★ Memory Map



Convenient SYSTEM commands are available to determine the number of bytes currently being used in each area of the RAM card. (See page 218)

4. Types of RAM Cards and Number of Bytes

Type	Storage capacity	Bytes
RC-2	2K bytes	2048 bytes
RC-4	4K bytes	4096 bytes

* One RC-4 RAM card (4K bytes) will be supplied with FX-750P when you purchase.

The FX-750P is provided with 2 slots for the RAM cards so either one RC-2 (2K bytes) or one RC-4 (4K bytes), or any combination of the two can be used.

Number of Bytes Depending on RAM Card Combinations

- Although the combinations are optional when using 2 RAM cards in the FX-750P, once data and program have been entered in a certain RAM card combination, that same combination and the same slots must always be used. An error will be displayed if combined with a different RAM card or if only 1 RAM card is used, or if inserted in the wrong slot. (See page 21)

Combination	Slot 0	Slot 1	Capacity (byte)	User area (byte)	Common system area
1	RC-2	—	2048	752	Program management area 632 bytes
2	RC-4	—	4096	2800	
3	RC-2	RC-2	4096	2800	Fixed variable area 208 bytes Variable management area 456 bytes
4	RC-2	RC-4	6144	4848	
5	RC-4	RC-2	6144	4848	Total 1296 bytes
6	RC-4	RC-4	8192	6896	

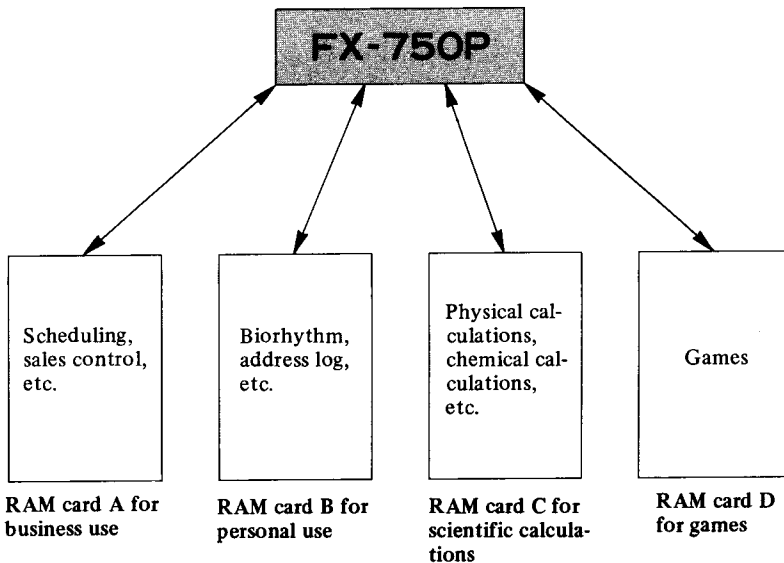
5. Method of Application with RAM Card Combinations

The exclusive use of each RAM card for storing a special program or a group of data enables easy retrieval of desired program or data just like finding a page through the index of a book.

We shall now describe how the RAM cards can be used in different combinations.

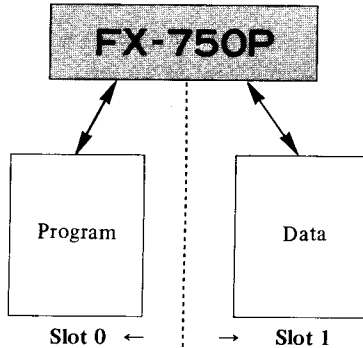
■ Classifying Data and Programs with RAM Cards

If the data and programs created with the FX-750P are stored in RAM cards and classified by cards with indices, the computer can respond quickly to your needs since it will then be simply necessary to insert the card.



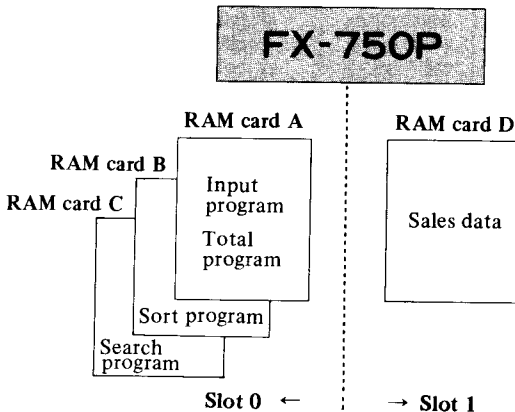
■ When Using 2 RAM Cards in Combination

Since the FX-750P is provided with 2 card insertion slots, it can naturally be used as an extended memory if 2 cards are inserted. The computer will have high general purpose usage as it is also possible to use the cards separately for program use and for data use.



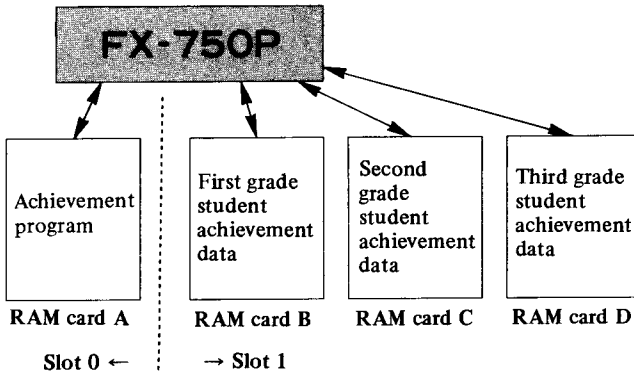
1. Processing of One Data Item by Various Programs

When conducting sales analysis for example, various analysis will be possible on 1 data if the input program, total program, sort program and search program are stored in the RAM card in slot 0 and the sales data in the RAM card in slot 1.



2. Processing of Multiple Data with a Single Program

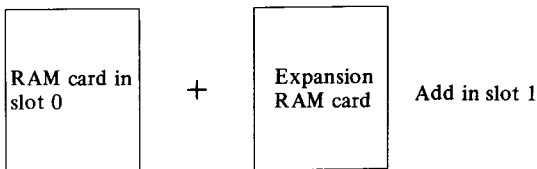
When processing multiple data, much faster processing will be possible with RAM cards when compared with processing by cassette tapes.



When separating programs and data into 2 RAM cards in this manner, always be sure to insert the program RAM card in slot 0 and the data RAM card in slot 1.

3. Memory Expansion Possible While Operating

Memory will frequently run short while creating a program. Since the FX-750P uses a 2-slot system, memory area can be easily expanded by simply adding a RAM card in slot 1.



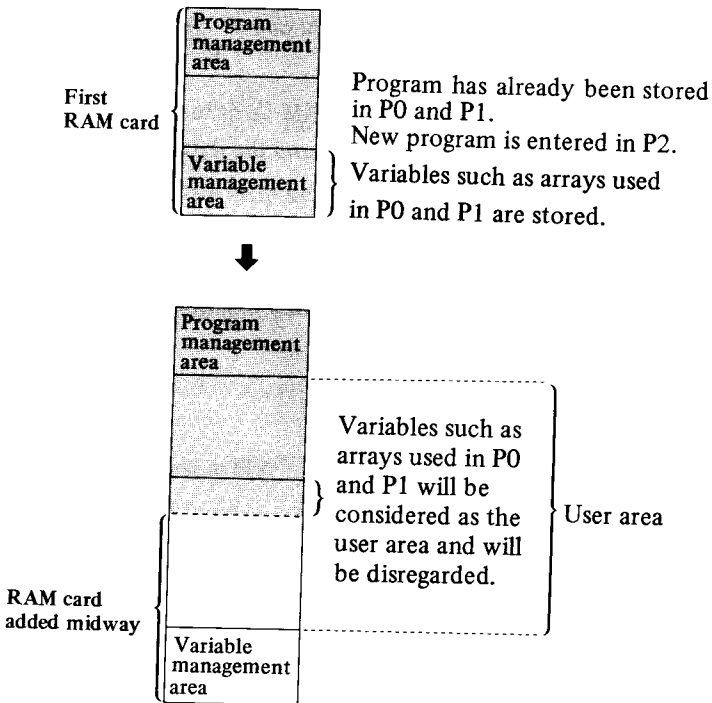
When creating a slightly larger program, it is recommended that a new RAM card be used, or the NEW ALL command be executed at the beginning to provide reserve memory area; if the memory is expanded midway with a certain number of programs already stored in the initial RAM card, the arrays already executed and variable contents are considered as the user area and are disregarded.

6. System Area and User Area When Using 2 RAM Cards

It was already mentioned that the system area would be split into an upper and lower section and that more user area would be available if 2 RAM cards are used. However, care will be required in relation to the following points.

1. When Expanding the Memory During Operation

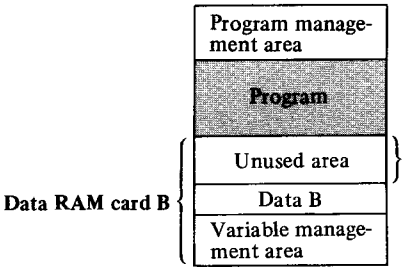
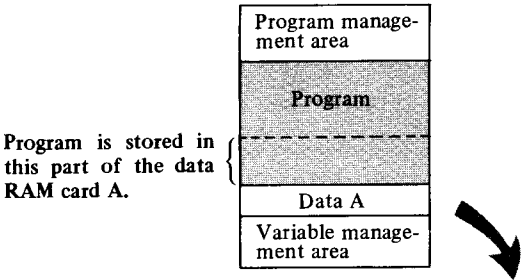
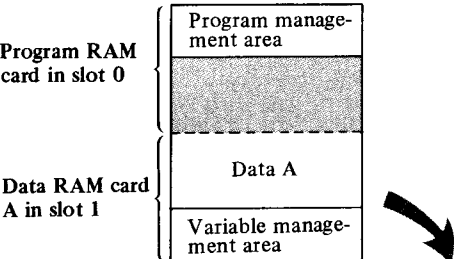
There will be no problem in expanding the memory if the NEW ALL command is executed before using the first card. However, if the memory is expanded midway with a program having been entered, the variable area in which arrays have been executed will be considered as the user area and will be disregarded. So it will not be possible to call the data.



2. When Using 2 RAM Cards Separately

There will be no problems when 2 RAM cards are used as a set with the RAM card in slot 1 used for memory expansion. When using separately as a program card and a data card however, the storage capacity of each RAM card must not be exceeded.

Initialized RAM Card Contents



Due to the separation of the whole program requiring the use of two RAM cards, insertion of another RAM card instead of the card A causes wrong program execution.

The program in this part will be lost.

Also, be careful when arranging the arrays for data since correct processing will not be possible if data is stored in a program area.

When using 2 RAM cards separately in this manner, programs and data exceeding 2048 or 4096 bytes (characters) must not be stored in 1 RAM card. Therefore, there is a SYSTEM command to check whether the program and data are overflowing the RAM card.

The amount of program used can be checked by the SYSTEM P command.

SYSTEM P 

```
PROG 1003 BYTES USED
```

The amount of variable area used can be checked by the SYSTEM V command.

SYSTEM V 

```
VAR 664 BYTES USED
```

When executing the program, be careful to ensure that this value does not exceed 2048 (RC-2: 2K byte RAM card) or 4096 (RC-4: 4K byte RAM card). A list of values with various card combinations is shown below for your reference.

Slot 0	Slot 1	SYSTEM P	SYSTEM V
2K	2K	2048	2048
2K	4K	2048	4096
4K	2K	4096	2048
4K	4K	4096	4096

The programs and data can be stored in 1 RAM card within the range shown.

7. Error Messages after Inserting or Removing of a RAM Card

In the FX-750P, it is possible to perform automatic checks of whether a RAM card has been correctly set (in relation to program and variable areas) after inserting a RAM card and setting the power switch to ON.

“READY P0” will be displayed if the card is correctly set. However, if the following message is displayed, reset the RAM card correctly. Also, be careful when answering the error message Y/N? since all programs and data will be erased by pressing Y .

Furthermore, when two cards which can be used independently are set, READY P0 may be displayed even if operation is not correct.

* Check all items thoroughly in regard to inserting or removing a RAM card.

```
???
```

Cause: No RAM card in slot 0.

Remedy: Set computer power switch to OFF and insert a RAM card in slot 0.

```
CLEAR P:Y/N?
```

Cause: Program area contents is faulty.

Examples:

1. When a pair of cards are used for storing a program, the card for slot 1 is not set or the other card is set into slot 1 by mistake.
2. Only the data card is set in slot 0 when using separate program and data cards.
3. When the stored programs and data were destroyed due to exhaustion of the battery in the RAM card.

Remedy:

1. Reset the 2 cards in their correct slots.
2. Press the Y key. (In this case, all programs are erased, and “READY P0” is displayed.)
3. Replace the RAM card battery.

```
CLEAR D:Y/N?
```

Cause: Variable area contents are faulty.

- Examples:**
1. When a pair of cards are used in the same format for storing data, the card for slot 1 is not set or the other card is set into slot 0 by mistake.
 2. When using separate program and data cards, the program card is set in slot 0 but the data card is not set in slot 1.
 3. When the format is destroyed due to exhaustion of the RAM card battery.

- Remedy:**
1. Reset the 2 RAM cards in their respective slots.
 2. Press the Y key. (In this case, all variable contents are erased, and "READY P0" is displayed.)
 3. Replace the RAM card battery.

```
NEWALL Y/N?
```


Cause: Contents in the program area and data area are faulty.

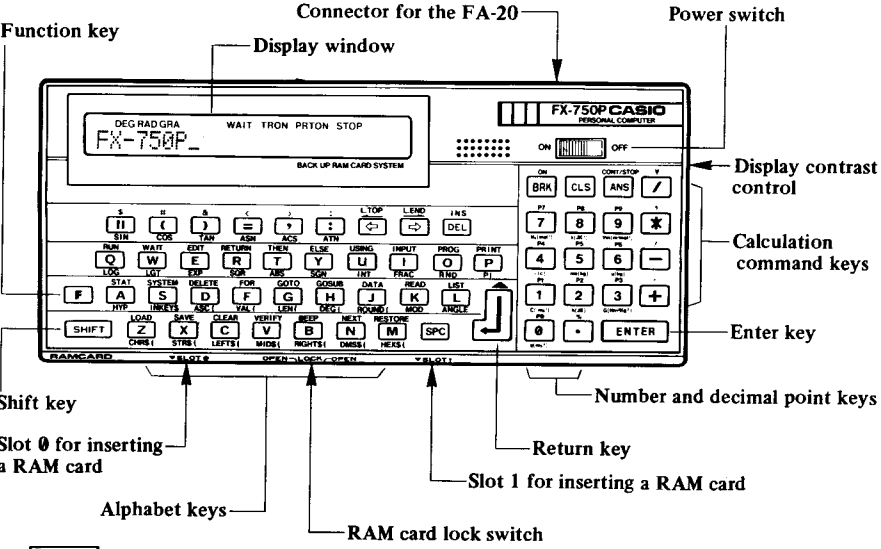
Example: The program and data card are set in reverse.

- Remedy:**
1. Set the program card in slot 0 and the data card in slot 1.
 2. Press the Y key. (In this case, all programs and variable contents will be erased.)

* RAM cards created for the FX-750P cannot be used in any other computer.

1-3 Operating Your FX-750P


The keyboard of your FX-750P is arranged with numeric keys and calculation command keys, etc. located to the right of the return key  key, and the alphabet keys (regular typewriter arrangement) and symbol keys to the left. To have a clear understanding of the functions of the FX-750P, it is desirable that you actually perform each function when explained.



ON  OFF **Power Switch:**

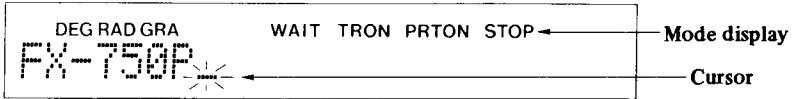
Slide to the left to turn power on. READY P0 will then be displayed to indicate that the FX-750P is ready for inputs. This switch can also be used to restore power from an "Auto Power Off" state.

★ Auto Power Off Function (Automatic power cut-off)

This is a power saving function that automatically turns off the power about 6 minutes after the last key operation in the event that the power switch is inadvertently left on. When desiring to turn the power on again, simply press the  key or reset the power switch to ON.

Although the memory contents and programs will not be lost even if the power goes off, all angle and mode specifications ("WAIT", "TRON", etc.) will be canceled. This function is inoperative when a program is being executed.

■ Understanding the Display



24 characters fit horizontally in the display window with each character composed of 35 dots (5 wide x 7 high). However, a maximum of up to 79 characters can be entered in a calculation formula or in a line of BASIC program as the screen will roll to the left if more than 24 characters are entered. A total of 128 character types, including alphanumeric and special symbols, can also be used with the FX-750P. (0 will be shown as 0.)

* Graphic characters cannot be entered directly from the keyboard but they can be displayed by using the CHR\$ function. (See page 143)

Angle units “DEG”, “RAD”, “GRA” or mode displays “TRON”, “STOP”, etc. are displayed at the upper part of the screen when that particular mode is specified.

- **DEG** (Degree mode): This mode is specified by and is an angle unit showing a right angle as 90°. This is initialized when the power is on. (See page 47)
- **RAD** (Radian mode): This mode is specified by and is an angle unit showing a right angle as $\frac{\pi}{2}$ radian.
- **GRA** (Grade mode): This mode is specified by and is an angle unit showing a right angle as 100 grades.
- **TRON** (Trace mode): This mode shows the proceeding of the program execution and is specified by . Entering cancels the trace mode. (See page 249)
- **PRTON** (Printer mode): This is the printer output mode and is specified by . Entering cancels the print mode. (See page 241)
- **STOP**: Appears when a program execution is stopped. (See page 247)
- **WAIT**: Appears when a program execution is temporarily stopped for displaying the contents of a PRINT statement. (See page 250)

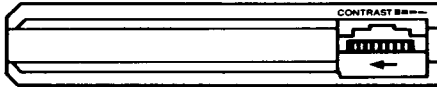
★ Cursor

When the power is turned on and characters or numerical data are entered, "READY P0" will go off and a " _ " symbol blinks at the right end of the characters displayed. This is called the cursor. The cursor indicates the position of the next character to be entered and moves one character at a time to the right. This is also used to correct input errors.

Although a maximum of 79 characters can be entered per line, the cursor blinks with a " _ " symbol up to the 71st character and then changes to a " ■ " symbol to caution you that the end of the line is near.

— Contrast Adjustment —


A brightness control is located on the right side of the FX-750P. Turn this in the direction of the arrow if you wish a brighter display.











* If the display does not become bright even at the maximum position, replace the batteries as soon as possible since, in all probability, they will be exhausted. (See page 2)

■ Key Functions

In the FX-750P, multiple functions can be performed with one key by simply changing the mode. Press one key only for the Direct Mode, press one key while pressing the **SHIFT** key for the Shift Mode and press one key while pressing the **F** key for the Function Mode.

* In this manual,  requires the simultaneous pressing of these two keys.




























































For example, the functions of the  key in each mode are as follows.

	Key operation	Display
		Q..... Direct mode
 RUN	 	RUN..... Shift mode
 LOG	 	LOG..... Function mode

This means that with one key you can enter Q in the direct mode, RUN (program execution) in the shift mode and LOG (natural logarithm) in the function mode. The functions of the principal keys in each mode are listed below.

1. Direct Mode

In this mode, the character or symbol shown on the key will be entered or the function will be performed when that key is pressed.

~ , **Number and decimal point keys**

- Use these keys when you wish to enter a desired number or decimal point.

Calculation command keys

- Use these keys when you wish to make an arithmetic calculation. The and keys are used here for \times and \div , respectively.

Parentheses keys

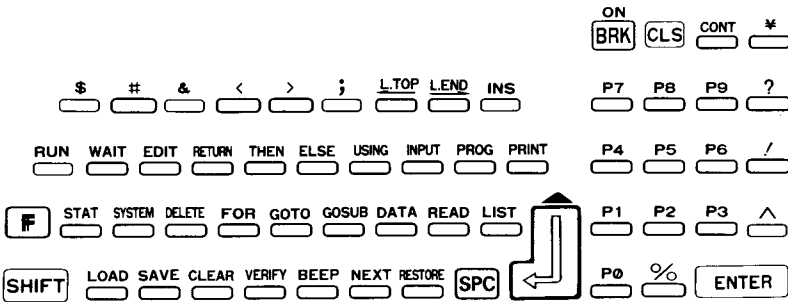
- Press these keys when you wish to enter a parenthesis.

Equal key

- This is used to indicate equal conditions in the assignment statement or judgement in the IF statement.

2. Shift Mode (

A BASIC command (GOTO, PRINT, BEEP, etc.) or a symbol shown above a key will be displayed if that particular key is pressed while pressing the key.

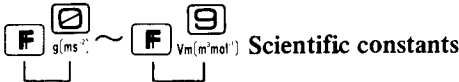
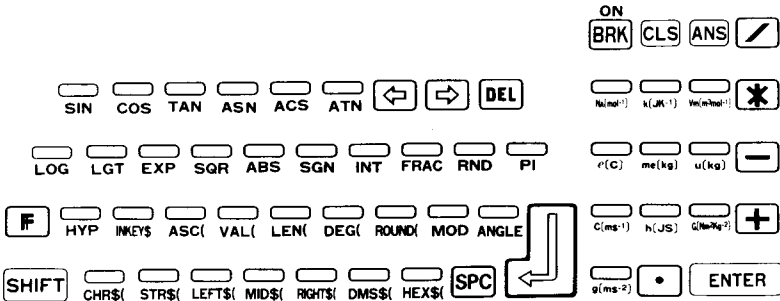


~ **Program area keys**

- Use these keys to specify a program area. If a program has already been written in that area, that program will be executed.

3. Function Mode (F)

The numerical function (SIN, PI, etc.), character function (ASC, CHR\$, etc.) or scientific constants shown under a key will be displayed if that particular key is pressed while pressing the **F** key. This will be explained further in Chapter 2.



- Use these keys to call scientific constants such as Plank's constant or Avogadro's constant.

4. Editing Keys and Special Keys


INS

DEL Delete/insert key

- Deletes the character above the cursor in the direct mode and moves all characters on the right one space to the left to close the gap.
- If this key is pressed while pressing the **SHIFT** key, all characters above and to the right of the cursor will move to the right to open a space.


L.TOP

**Cursor left key/line top key**

- The cursor moves 1 character to the left each time this key is pressed. If this key is held down, the cursor will move continuously to the left end of the line.
- If this key is pressed together with the  key, the cursor will move to the beginning of the line.

L.END

**Cursor right key/line end key**


- The cursor moves 1 character to the right each time this key is pressed. If this key is held down, the cursor will move continuously to the right end of the line.
- If this key is pressed together with the  key, the cursor will move to the end of the line.

**Screen clear key**


- Clears the display and moves the cursor to the left end of the display.

CONT/STOP


**Answer key**

- Use this key to display the answer of the manual calculation previously made, or the value output through the performance of the PRINT or LPRINT statement, or the value of the scientific constant called.
- This key stops program execution temporarily. To restart the program execution, press this key together with the  key.

**Return key**

- Press this key at the end of each line when writing a program. Also, use this key to execute a command in a program.
- If this key is pressed together with the  key in the EDIT mode, the previous line can be displayed.

ENTER Enter key

- Functions as (=) in a manual calculation to obtain an answer. This also functions the same as the  key during execution of a program when waiting for data input.

ON
BRK Break key

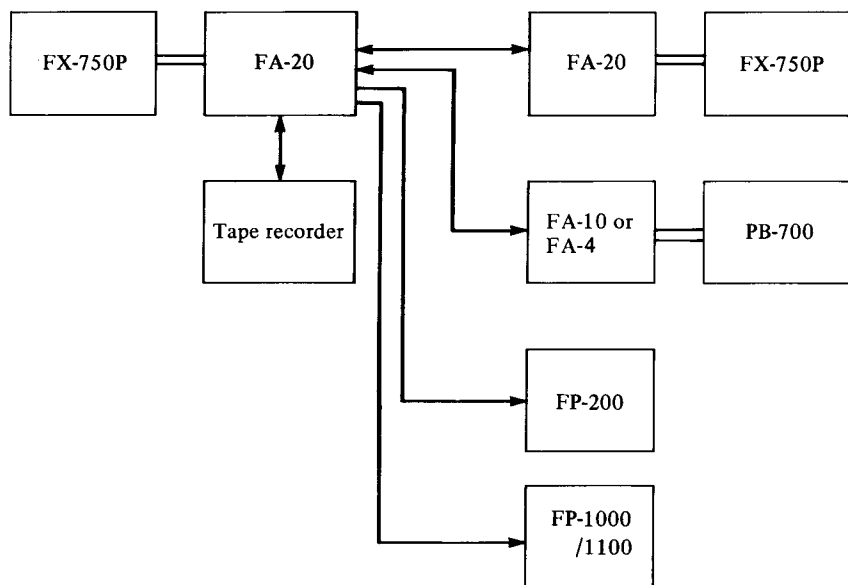
- Use this key when you wish to stop execution of a program.
- Use this key to restore power when it is turned off by the auto power off function.

1-4 Option

The FX-750P can be used conveniently as stand-alone equipment, but a character printer with a cassette interface < FA-20 > is optionally available. (Do not use anything other than the FA-20 with this computer.)

The printer section of the FA-20 can print out program lists and calculation results, etc.

The cassette interface section of the FA-20 is for connection to a cassette tape recorder so that the program and data prepared by the FX-750P can be stored on a tape and loaded from a tape. It also makes possible mutual communication between the FX-750P connected to another FA-20 and the PB-700 connected to the FA-10 or FA-4, and transfer to the FP-1000/1100 and the FP-200.



For FX-750P, FA-20 and cassette tape recorder connections and the points requiring particular attention, refer to the instruction manual supplied with the FA-20. Only the basic operating procedure is described here.

■ Printer Section

It often happens that program content modifications or debugging becomes necessary during programming. In such a case, the capability of printing out the program list comes in very handy. The capability of storing the printout is also convenient.

For printing out a program or various data, use an LLIST command.

Print command	Content
LLIST	Prints out the program in the currently specified program area.
LLIST ALL	Prints out all programs in all program areas (P0 to P9).
LLIST V	Prints out registered variable names and declared array variable names.

■ Cassette Interface Section

One of the features of the RAM card is immediate usability. Operation can be started immediately just by inserting the card on the main frame. The card is available in a variety of kinds for particular purposes.

However, it is recommended to store programs and data on a cassette tape under the following conditions.

- When replacing battery of a RAM card.
- When program or data are loaded to another RAM card.
- When storing programs or data as a spare one.

Here is how to store a program or data on a cassette tape and, conversely, to load the program or data stored on a tape to the FX-750P.

● Program Storing (SAVE)

When storing or loading a program, it is necessary that the CMT cable (cable for connecting with cassette tape recorder) be properly connected to the FA-20 and the cassette tape recorder. (For the connection method, see the FA-20 instruction manual.)

For program storing, there are three commands: "SAVE", "SAVE ALL" and "SAVE, A". "SAVE" is for storing the program in the currently specified program area. "SAVE ALL" is for storing all programs in the program areas P0 to P9.

"SAVE, A" is for storing the program in the currently specified program area on a tape with the ASCII code format. Storing with the ASCII code permit loading by the FP-1000/1100 or the FP-200.

Examples:

```
SAVE ↵
SAVE "CASIO" ↵
SAVEALL ↵
SAVEALL "AA" ↵
SAVE , A ↵
SAVE "CASIO" , A ↵
```

Characters in double quotation marks when storing programs are called the file name. The file name can be omitted, but we recommend the use of such names because they will make it possible to specify a particular program when loading it later. Up to 8 characters or symbols are used for a file name.

● Program Loading (LOAD)

For program loading, there are four commands: "LOAD", "LOAD ALL", "LOAD, A" and "LOAD, M". When using a LOAD command, it should correspond to the format of the SAVE command which was used when storing the program. In the case of the "LOAD, M" command, the program stored with the

ASCII code format can be merged with the program in the main frame. When the "LOAD" or "LOAD, A" command is used, the previous program is erased. The "LOAD, M" and the "LOAD "file name", M" commands permit program loading additional to the previous one. If the same line number exists, the one loaded later is given priority and retained.

Examples:

- LOAD ↵
- LOAD "CASIO" ↵
- LOAD ALL ↵
- LOAD ALL "AA" ↵
- LOAD, A ↵
- LOAD "CASIO", A ↵
- LOAD, M ↵
- LOAD "CASIO", M ↵

● Relationship Between SAVE and LOAD

Storing	LOAD	LOAD "file name"	LOAD ALL	LOAD ALL "file name"	Display during loading
SAVE	○	×	×	×	PFB
SAVE "file name"	○	○	×	×	file name PFB
SAVE ALL	×	×	○	×	AFB
SAVE ALL "file name"	×	×	○	○	file name AFB
SAVE ,A	×	×	×	×	PFA
SAVE "file name" ,A	×	×	×	×	file name PFA

Storing	LOAD ,A	LOAD "file name" ,A	LOAD ,M	LOAD "file name" ,M	Display during loading
SAVE	×	×	×	×	PFB
SAVE "file name"	×	×	×	×	file name PFB
SAVE ALL	×	×	×	×	AFB
SAVE ALL "file name"	×	×	×	×	file name AFB
SAVE ,A	○	×	○	×	PFA
SAVE, "file name" ,A	○	○	○	○	file name PFA

○ – Can be loaded. × – Cannot be loaded.

■ Data Storing and Loading (PUT, GET)

Storing or loading of the contents of variables cannot be performed by a SAVE or LOAD command. To store the contents of variables, use the PUT command. To load the contents of variables, use the GET command.

Example of data storing:

```
PUT"CASIO" A,B,C
          (file name) (Variable name)
```

Example of data loading

```
GET"CASIO" A,B,C
          (file name) (variable name)
```

The file name can be omitted, but we recommend that it be used for easy discrimination when loading.

Attention

Tape recorder characteristics or cassette tape quality sometimes prevents accurate storing of program or data. Therefore, cultivate the habit of ascertaining storing accuracy after each storing operation by loading the program or data stored on the cassette tape to the main frame.

There are many causes for inaccurate storing. Here are the principal ones:

- Wrong connection of cable, etc.
- Wrong operations for storing or loading.
- Inappropriate volume of the cassette tape recorder at the time of loading.
- Problems with the cassette tape itself: the tape is stained, damaged or the like.
- The tape recorder head is unclean.

CHAPTER

2

CALCULATION FUNCTIONS

This chapter is to enable you to learn the basic operations from simple manual calculations to using the many numerical functions, character functions, statistical functions and scientific constants that are the features of the FX-750P.

Numerical functions and character functions can be performed easily through one-key functions.

In this chapter, the following method of notation will be used for key operations.

- Alphabets, numbers and commands will not be enclosed.

Example:

$\boxed{C} \boxed{O} \boxed{S} (\boxed{P} \boxed{I} \boxed{/} \boxed{3} \boxed{)} \boxed{ENTER}$

$\rightarrow \boxed{COS} (\boxed{P} \boxed{I} \boxed{/} \boxed{3} \boxed{)} \boxed{ENTER}$

- Although one-key commands and one-key functions with the \boxed{SHIFT} key and \boxed{F} key can also be performed with the alphabet keys, we shall indicate with alphabet keys only from now on.

Example:

$\boxed{SHIFT} \boxed{RUN} \rightarrow \text{RUN} \leftarrow \boxed{R} \boxed{U} \boxed{N}$

$\boxed{F} \boxed{LOG} \rightarrow \text{LOG} \leftarrow \boxed{L} \boxed{O} \boxed{G}$

2-1 Prior to Calculations

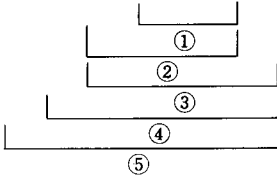
■ Calculation Priority Sequence

Although all calculations have a sequence of priority, this sequence is memorized in the FX-750P and calculations will be performed automatically in priority sequence and the correct answer displayed by simply pressing the keys according to the numerical expression.

Sequence of Priority

1. Parentheses calculations
 2. Functions (sin, cos, tan, etc.)
 3. Powers (^)
 4. Discrimination of signs (displays “ - ” only)
 5. Multiplication and division (*, /)
 6. MOD (Remainder calculation): Discards fractions of numerical values.
 7. Addition and subtraction (+, -)
 8. Relational operators (>, <, >=, <=, etc.)
- * If the sequence of priority is the same, calculations will start from the left (beginning).

Example: $3+5*INT(86/7)^2=723$



■ Using the Relational Operator

Relational operators are used as conditional expressions in IF statements. (See page 233)

- = Left and right sides are equal.
- <, >, > < Left and right sides are not equal.
- < The left side is smaller than the right.
- > The left side is larger than the right.
- = >, > = The left side is equal to or larger than the right.
- = <, < = The left side is equal to or smaller than the right.

Examples:

IF A>30 THEN 500

Jumps to line 500 if A is larger than 30.

IF B\$=" 6 " THEN BEEP 0

A low pitched beep will sound if B\$ is equal to "6".

IF CHR\$(65)>CHR\$(N)THEN W=3

If CHR\$(65), that is, A is larger than CHR\$(N), the value of 3 is assigned to W.

■ Calculation Methods**Formats**

$\frac{x+y}{2}$	$\rightarrow (x+y)/2$
x^3	$\rightarrow x^{\wedge}3$
$x^2+2xy+y^2$	$\rightarrow x^{\wedge}2+2*x*y+y^{\wedge}2$
$(-y)^2$	$\rightarrow (-y)^{\wedge}2$
$(x^y)^2$	$\rightarrow x^{\wedge}y^{\wedge}2$
x^{y^2}	$\rightarrow x^{\wedge}(y^{\wedge}2)$
Remainder of $\frac{x}{y}$	$\rightarrow x \text{ MOD } y$

Examples

$0.5^{\wedge}0$	$\rightarrow 1 \quad (0.5^0 = 1)$
$-0.5^{\wedge}0$	$\rightarrow -1 \quad (-0.5^0 = -1)$
$(-0.5)^{\wedge}0$	$\rightarrow 1$
$0.5^{\wedge}0.5$	$\rightarrow 0.7071067812$
$(-0.5)^{\wedge}0.5$	$\rightarrow \text{MA ERROR}$
$123 \text{ MOD } 5$	$\rightarrow 3$
$10 \text{ MOD } -6$	$\rightarrow 4$
$-10 \text{ MOD } -6$	$\rightarrow -4$

■ Character String Operators <+>

Character strings can be linked with the + sign. Numerical value enclosed with double quotation marks (") will be handled as character string. Operators other than + cannot be used as character string operators.

Example:

```
" AB " + " 123 " [ENTER] → AB123
A$ = " FX- " [↵]
B$ = " 750P " [↵]
C$ = A$ + B$ [↵]
C$ [ENTER] → FX-750P
```

■ Error Displays

1. Attempting to divide by "0" will cause an MA ERROR.
2. Exceeding the calculation range (overflow) ($\pm 1 \times 10^{-99} \sim \pm 9.999999999 \times 10^{99}$) causes an OV ERROR.
3. Calculation range of powers

$(x > 0, y > 0)$	$0 \wedge 0$	→ MA ERROR
	$(\pm x) \wedge 0$	→ 1
	$0 \wedge y$	→ 0
	$0 \wedge (-y)$	→ MA ERROR
	$(-x) \wedge (\pm y)$	→ Only when y is an integer. In other cases an MA ERROR will be caused.

Accuracy will be ± 1 at the 7th digit when $1 - 10^{-6} < x < 1 + 10^{-6}$ in $x \wedge y$.

* See page 320 for error displays.

■ Number of Input/Output Digits and Calculation Digits

Up to 79 characters can be entered per line in a calculating expression while internal calculations are performed with 12-digit mantissa plus a 2-digit exponent.

Calculation result will be displayed with a 10-digit mantissa (11th digit rounded to the nearest whole number) plus a 2-digit exponent. This will change to exponential displays automatically if it is 10^{10} or more, or less than 10^{-3} .

Examples:

1.2345678912

12345678912 100

1 0.0001

1.234567891
1.234567891E12
1E-04

■ Operation Levels

Numerical values 8 levels

Relational operators 20 levels

* Triple parentheses will be equal to a 1-level operator.

{ { { 123+456*{ (789+123*{ 456+-----

1 level

1 level

1 level

2-2 Manual Calculations and Input Corrections

Before starting the manual calculation, press the [CLS] key to clear the display.



The cursor is blinking at the leftmost side. This means the computer is waiting for data input. The DEG mode is set for calculations requiring specification of angle units (trigonometric functions, etc.). But such a DEG display will not affect other calculations.

Let's first perform a simple manual calculation.

Example:

$$8 + 4.2 \div 5 - 2.5 \times 2 = 3.84$$

Press the keys according to the numerical expression. Use the [*] key for multiplication, the [/] key for division and the [ENTER] key for “=”.

Operation:

$$8 \text{ [+] } 4.2 \text{ [/] } 5 \text{ [-] } 2.5 \text{ [*] } 2$$

[ENTER]

8+4.2/5-2.5*2_
3.84

Calculations can also be continued after performing a manual calculation.

Operation:

$$123 \text{ [*] } 456 \text{ [ENTER]}$$

(Continue to press) $\text{[+] } 789$

[ENTER]

56088
56088+789_
56877

If, during a calculation, you wish to use the result of the preceding calculation, simply press the [ANS] key.

Operation:






$$1.2 \text{ [+] } 3.4 \text{ [ENTER]}$$



$$11.5 \text{ [/] } \text{[ANS]}$$

[ENTER]



4.6
11.5 / 4.6_
2.5

■ **Modification of Inputs (Correction, deletion, insertion)**

Use , ,  to modify input contents. Although partial modification is possible before completing a manual calculation, modification will be impossible once the  key is used and input operation should be performed from the beginning. It will therefore be desirable that you make it a habit to recheck your entry before pressing the  key.

To modify a program that has already been entered, correct the line and press the  key after setting to the EDIT mode ().

1) **Correction**



Move the cursor with the  or  key to the place to be corrected and enter a correct character or symbol.

Example:


Correct SIN 40 to SIN 30.



Operation:


  (move cursor to 4).



Press .

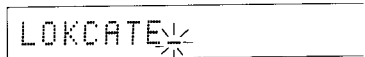


2) **Deletion**






Move the cursor to the character or symbol to be deleted. Pressing the  key will delete the character and all characters/symbols to the right of the cursor will shift one space to the left to readjust the format.

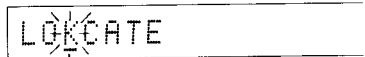
Example:


Correct LOKCATE to LOCATE.



Operation:



     (move cursor to K).



Press .



3) Insertion











Move the cursor to the right of the place where the insertion is to be made and press  . One space will then be opened so that a new character or symbol can be inserted.

Example:

Correct IF S < 0 to IF S <= 0.

40IF S < 0 THEN 90


Operation:

(Move cursor to 0.)

40IF S < 0 THEN 90

Press .

40IF S <= 0 THEN 90

■ Calculation Using the Memory Function

Various mathematical calculations can be conveniently performed by using the memory functions.

Example:

$$50000 \times \underline{0.75} =$$



$$80000 \times \underline{0.75} =$$

These calculations can be performed easily by storing the value, to be repeatedly entered, in the memory and recalling them when needed. Here, we shall set A = 0.75.

In this case, A is a variable. A numerical value or a character is assigned to a variable. It should be noted here that A = 0.75 is an assignment statement and “=” is an instruction to assign 0.75 to A. So the “=” sign here must not be confused with the equal sign in a calculation expression.

Key operations will be as follows:

Operation:

A  0.75 

You will note that this is not entered with the **ENTER** key but the **↵** key as this is not a calculation expression.

Now, let's check whether the correct assignment has been made.

Operation:

A **ENTER**

0.75

Next, let's solve the above calculation expressions.

Operation:

50000 * A **ENTER**

37500

80000 * A **ENTER**

60000

* Variables will be explained in detail in Chapter 3.

Have a clear understanding of the usage of the **↵** and **ENTER** keys since erroneous operation of these keys will cause SN ERROR (syntax error).

ENTER

This key is used to display manual calculation results and also to confirm a variable content.

↵

This key is used to assign numerical values and expressions to variables, execute BASIC program commands and to specify status such as the print mode.

2-3 Numerical Functions

The numerical functions shown in the table below are built into the FX-750P. Although inputs are possible by alphabetical letters, complex equations can be input easily by using the one-key function with the **[F]** key and difficult technical computations can be processed speedily.

Function	Mathematical expression	Format	Meaning/remarks
Trigonometric functions	sin	SIN (Numerical expression) *hereafter X	$-5400^\circ < X < 5400^\circ$ (30π rad, 6000 gra)
	cos	COS (X)	$-5400^\circ < X < 5400^\circ$ (30π rad, 6000 gra)
	tan	TAN (X)	$-5400^\circ < X < 5400^\circ$ (30π rad, 6000 gra) (excludes the cases when $ X = (2n - 1) \times 90^\circ$) * n is an integer.
Inverse trigonometric functions	\sin^{-1}	ASN (X)	$ X \leq 1, -90^\circ \leq \text{ASN}(X) \leq 90^\circ$
	\cos^{-1}	ACS (X)	$ X \leq 1, 0^\circ \leq \text{ACS}(X) \leq 180^\circ$
	\tan^{-1}	ATN (X)	$ X < 10^{100},$ $-90^\circ \leq \text{ATN}(X) \leq 90^\circ$
Hyperbolic functions	sin h	HYP SIN (X)	$ X \leq 230$
	cos h	HYP COS (X)	$ X \leq 230$
	tan h	HYP TAN (X)	$ X \leq 10^{100},$ $-1 \leq \text{HYP TAN}(X) \leq 1$
Inverse hyperbolic functions	$\sin h^{-1}$	HYP ASN (X)	$ X < 5 \times 10^{99}$
	$\cos h^{-1}$	HYP ACS (X)	$1 \leq X < 5 \times 10^{99}$
	$\tan h^{-1}$	HYP ATN (X)	$ X < 1$
Exponential function	e^X	EXP (X)	$-227 \leq X \leq 230$
Natural logarithm	$\log_e X$	LOG (X)	$X > 0$ Accuracy will be ± 1 at the 8th digit when $1 - 10^{-6} < X < 1 + 10^{-6}$
Common logarithm	$\log_{10} X$	LGT (X)	$X > 0$ Accuracy will be ± 1 at the 8th digit when $1 - 10^{-6} < X < 1 + 10^{-6}$
Square root	$\sqrt{\quad}$	SQR (X)	$X \geq 0$
Absolute value	$ x $	ABS (X)	Provides absolute value of X.
Sign		SGN (X)	Provides $\begin{cases} -1 & \text{when } X < 0, \\ 0 & \text{when } X = 0, \\ 1 & \text{when } X > 0. \end{cases}$
Integer		INT (X)	Gaussian function: provides maximum integer not exceeding the value of X.

Function	Mathematical expression	Format	Meaning/remarks
Fraction Rounding		FRAC (X) ROUND(x,y[,f] $\left\{ \begin{matrix} + \\ - \end{matrix} \right\}$) x,y: X f: 0 or 1	Provides fraction of X. Rounds the value of x at 10 ^y position. $\left\{ \begin{matrix} f \text{ is omitted:} \\ f = 0: \\ f = 1: \end{matrix} \right\}$ Specifies 10 ^y position Specifies the yth position of a significant number
Degree /minute /second	Sexagesimal – Decimal	DEG (d [,m [,s]]) d, m, s: X	Provides decimal-converted values.
Circular constant	π	PI	Provides an approximate value of π (3.1415926536)
Random number		RND	0 < random number < 1 A 10 digit random number is generated.

X → Numerical expression

* Parentheses cannot be omitted for ROUND and DEG, however, they can be omitted as to the others when numerical values or variables are used for (numerical expression).

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

Always specify angle units when using trigonometric or inverse trigonometric functions. (There will be no need to specify if the angle units are not to be changed.) DEG (degree) is specified with ANGLE 0 $\left[\text{↵} \right]$, RAD (radian) with ANGLE 1 $\left[\text{↵} \right]$, and GRA (grade) with ANGLE 2 $\left[\text{↵} \right]$. These modes will be displayed at the upper part of the display.

* The relation between DEG, RAD and GRA is $90^\circ = \frac{\pi}{2}$ (rad) = 100 grades.

● **DEG**

Example: $\sin 45^\circ$

Operation: $\left[\text{F} \right] \left[\text{ANGLE} \right] 0 \left[\text{↵} \right]$
 $\left[\text{F} \right] \left[\text{SIN} \right] 45 \left[\text{ENTER} \right]$

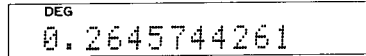
DEG
0.7071067812

Numerical functions

Since using the alphabet keys **SIN** for SIN will be the same as using **F SIN** for one-key function, we shall omit **F** from hereon and use alphabets only.

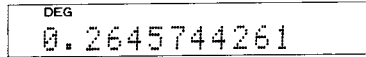
Example: $\sin 15.34166667$

Operation: SIN 15.34166667 **ENTER**



Example: $\sin 15^\circ 20' 30''$

Operation: SIN DEG (15) (') (20) (') (30) (') **ENTER**



• RAD

Example: $\cos(\frac{\pi}{3} \text{ rad})$

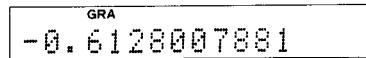
Operation: ANGLE 1 (π) (/) (3) **ENTER**



• GRA

Example: $\tan(-35 \text{ gra})$

Operation: ANGLE 2 (-) (35) **ENTER**

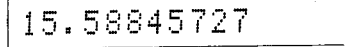
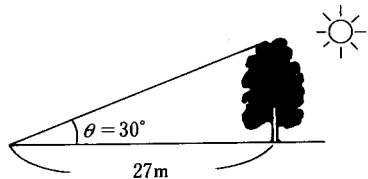


Let's take a break here and have some fun by measuring the height of a tree. The height of trees as shown in the diagram can be easily obtained if we know the length of the shadow and angle θ .

As $\tan \theta = \frac{\text{height}}{\text{base}}$, height will be $\text{base} \times \tan \theta$. We then specify angle unit as DEG and enter base = 27 m and $\theta = 30^\circ$.

Operation:

F ANGLE (27) * TAN 30 **ENTER**



In other words, the height of the tree is about 15.6 m.

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

Example: $\sinh 3.6$

Operation: HYP SIN3.6

18.28545536

Example: $\cosh 1.5 - \sinh 1.5$

Operation: HYP COS 1.5 HYP SIN
1.5

0.2231301602

Example: The value of x when

$$\tanh 4x = 0.88$$

$$x = \frac{\tanh^{-1} 0.88}{4}$$

Operation: HYP ATN0.88 4

0.3439419141

■ Logarithm Functions (\log_{10} , \log_e) and Exponential Functions (e^x , x^y)

Example: $\log_{10} 1.23 (= \log 1.23)$

Operation: LGT 1.23

0.08990511144

Example: $\log_e 90$

Operation: LOG90

4.49980967

Example: $\log_{10} 456 \div \log_e 456$

Operation: LGT 456 LOG 456

0.4342944819

Example: $e^{4.5}$ (To obtain the anti-logarithm of natural log 4.5)

Operation: EXP 4.5

90.0171313

Example: $10^{1.23}$ (To obtain the anti-logarithm of common log 1.23)

Operation: 10 1.23

16.98243652

Example: $\log_{10}\sin 40^\circ + \log_{10}\cos 35^\circ$

Also, what is the anti-logarithm of the answer?
(Logarithm calculation of $\sin 40^\circ \times \cos 35^\circ$)

Operation:

ANGLE 0

LGT SIN40 LGT COS35

10

-0.2785679838

0.5265407845

■ Other Functions

Square root < SQR >

Example: $\sqrt{2} + \sqrt{5}$

Operation: SQR2 SQR5

3.65028154

Signs < SGN >

Example:

“1” is provided if a positive number, “-1” if a negative number and “0” if a 0.

Operation:

SGN6

1

SGN0

0

SGN 8

-1

Absolute value < ABS >

Example:

Absolute value of $1 - 78.9 \div 5.61$

Operation:

ABS 1 78.9 5.61

13.06417112

* Use the absolute value of x in the ABS function. Since a negative number assigned to x in SQR x or LGT x will cause MA ERROR.

Example:

Obtain the square root of absolute value of -26.

Operation:

SQR ([ABS ([26]) ENTER

5.099019514

Integer < INT >**Example:**Obtain the integer portion of $\frac{7800}{96}$ **Operation:**

INT ([7800] [/] 96] ENTER

81

Fractions < FRAC >**Example:**Obtain the fraction of $\frac{7800}{96}$ **Operation:**

FRAC ([7800] [/] 96] ENTER

0.25

Rounding < ROUND >**Example:**Discard 16666 to the 10^2 position (= 100)**Operation:**ROUND ([1 6 6 6 6] [>] 2 [>] [-]
] ENTER

16000

Example:Obtain the quotient of $147 \div 13$, rounded up to the 3 significant digits.**Operation:**

ROUND ([147] [/] 13, 4, 1, +] ENTER

11.3

Angle < DEG >

(Sexagesimal → decimal)

Example:

Convert 15°20'30" to decimals

Operation:

DEG (F) 15 (D) 20 (C) 30 (B) (ENTER)

15.34166667

Circular constant (π) < PI >

Example:

Obtain the area of a circle with a radius of 14 m

Operation:

PI (X) 14 (SHIFT) (C) 2 (ENTER)

615.7521601

Random numbers < RND >

Example:

Generate a random number

Operation:

RND (ENTER)

0.3413855482

2-4 Character Functions

The FX-750P is provided with the character functions shown in the table below.

Name	Format	Function
ASC	ASC(X\$)	Obtains the ASCII code of the first character of character expression.
CHR\$	CHR\$(I)	Obtains the character corresponding to the ASCII code through a numerical expression. $0 \leq I < 256$
DMSS\$	DMSS(I)	Converts a decimal into a sexagesimal. $II < 10^{100}$
HEX\$	HEX\$(I)	Obtains the character string of the hexadecimal converted from a decimal. $-32769 < I < 65536$
INKEY\$	INKEY\$	Reads 1 character from the keyboard.
LEN	LEN(X\$)	Obtains the number of characters in character expression. $0 \leq X\$ \leq 79$
VAL	VAL(X\$)	Converts a character expression to a numerical value.
LEFT\$	LEFT\$(X\$, I)	Fetches the number of characters specified by a numerical expression from the left of a character expression. $0 \leq I < 256$
MID\$	MID\$(X\$, I, J)	Fetches the number of characters specified by the numerical expression 2 from the position specified by the numerical expression 1.
RIGHT\$	RIGHT\$(X\$, I)	Fetches the number of characters specified by a numerical expression from the right of a character expression. $0 \leq I < 256$
STR\$	STR\$(I)	Converts a value of numerical expression to a character string.

X\$ → Character expression.

I → Numerical expression or Numerical expression 1.



J → Numerical expression 2.

< CHR\$ >

Example:



Display graphic characters

Operation:



 223  ENTER



Operation:

 235  ENTER



- * It is not necessary to press the  key for character functions such as CHR\$ that require parentheses because they already include the left parenthesis “(”. However, do not forget to press the  key to close the parenthesis.

Since using the one-key function or alphabet keys for the character function will be the same, we shall indicate by alphabetical letters only from hereon.

< DMSS >

Example:

Convert 15.34166667 to sexagesimal number

Operation:

DMS\$ ([15.34166667]) ENTER

15° 20' 30

< HEXS >

Example:

Convert 250 to a character string expressed in hexadecimals.

Operation:

HEX\$ ([250]) ENTER

00FA

For usage of other character functions, refer to Chapter 5 Command Reference.

■ Hexadecimal Constant < &H >

Convert a 1 to 4 digit hexadecimal number to decimals.

Example:

Convert 00FA obtained by < HEXS > to decimals.

Operation:

SHIFT ([H00FA]) ENTER

250

Example:

Display an alphabet character M.

Operation:

CHR\$ ([SHIFT ([H4D])]) ENTER

M

2-5 Statistics Calculations

It is needless to say that statistical calculation is indispensable in office and technical work to analyze and obtain new data from data collection. Since the FX-750P is provided with the functions shown in the table below, burdensome statistical calculations can be performed easily and correlative coefficients and estimated values can be readily obtained.

- * Since these are not one-key functions, press the keys according to the format.
- * The name of the basic statistics with ★ sign attached and each value are displayed in the sequential order by entering STAT LIST $\left[\text{STAT} \right]$. Press $\left[\text{END} \right]$ or $\left[\text{ENTER} \right]$ if you wish to stop displaying and press again if you wish to continue displaying. Enter STAT LLIST $\left[\text{STAT} \right]$ for printouts.

Format		Function
CNT★	n	Number of statistical data processed
SUMY★	Σy	Sum of the y data
SUMX★	Σx	Sum of the x data
SUMXY★	$\Sigma x y$	Sum of the x data by y data
SUMX2★	Σx^2	Sum of the squares of the x data
SUMY2★	Σy^2	Sum of the squares of the y data
SDX	$x\sigma_{n-1}$	Sample standard deviation value of the x data $\sqrt{\frac{n\Sigma x^2 - (\Sigma x)^2}{n(n-1)}}$
SDY	$y\sigma_{n-1}$	Sample standard deviation value of the y data $\sqrt{\frac{n\Sigma y^2 - (\Sigma y)^2}{n(n-1)}}$
SDXN	$x\sigma_n$	Population standard deviation value of the x data $\sqrt{\frac{n\Sigma x^2 - (\Sigma x)^2}{n}}$
SDYN	$y\sigma_n$	Population standard deviation value of the y data $\sqrt{\frac{n\Sigma y^2 - (\Sigma y)^2}{n}}$
LRA	A	Linear regression constant term $\frac{\Sigma y - \text{LRB} \cdot \Sigma x}{n}$
LRB	B	Linear regression coefficient $\frac{n \cdot \Sigma x y - \Sigma x \cdot \Sigma y}{n \cdot \Sigma x^2 - (\Sigma x)^2}$
COR	r	Correlation coefficient $\frac{n\Sigma x y - \Sigma x \cdot \Sigma y}{\sqrt{\{n\Sigma x^2 - (\Sigma x)^2\}\{n\Sigma y^2 - (\Sigma y)^2\}}}$
EOX (X)	\hat{x}	Estimated value (value of x estimated from the value of y) $\text{EOX}(y_n) = \frac{y_n - \text{LRA}}{\text{LRB}}$
EOY (X)	\hat{y}	Estimated value (value of y estimated from value of x) $\text{EOY}(x_n) = \text{LRA} + x_n \cdot \text{LRB}$

(X) → Numerical expression

As data will remain in the memory unless STAT CLEAR $\left[\text{C} \right]$ is entered, always press these keys to clear the previous data before inputting new data.

■ **Inputting Statistical Data**

● **1 Variable statistical data**

- individual data
 STAT data $\left[\text{C} \right]$
- multiple data of the same value
 STAT data $\left[\text{SHIFT} \right] \left[\text{F} \right]$ Frequency $\left[\text{C} \right]$

● **2 Variables statistical data**

- individual data
 STAT x data $\left[\text{,} \right]$ y data $\left[\text{C} \right]$
- multiple data of the same value
 STAT x data $\left[\text{,} \right]$ y data $\left[\text{SHIFT} \right] \left[\text{F} \right]$ Frequency $\left[\text{C} \right]$

Data is entered with the $\left[\text{C} \right]$ key but the answer to statistical calculations such as standard deviations is obtained with the $\left[\text{ENTER} \right]$ key. Be sure to use the correct key as SN ERROR will occur if the wrong key is pressed. The initial data can be set for the basic statistics by using the assignment statement.

Example: SUMX = 123 $\left[\text{C} \right]$

Obtain the standard deviations and determine the variance in shipments of product x and product y from the state of shipments shown in the table.

Data	4	5	6	7	8	
Product	x	2	2	5	8	8
	y	1	5	5	5	9

Operation:

STAT CLEAR $\left[\text{C} \right]$
 STAT2 $\left[\text{,} \right]$ 1 $\left[\text{C} \right]$ STAT2 $\left[\text{,} \right]$ 5 $\left[\text{C} \right]$ STAT5 $\left[\text{,} \right]$ 5 $\left[\text{C} \right]$
 STAT8 $\left[\text{,} \right]$ 5 $\left[\text{C} \right]$ STAT8 $\left[\text{,} \right]$ 9 $\left[\text{C} \right]$
 STAT LIST $\left[\text{C} \right]$ (Basic statistics will be automatically displayed.)

(Number of data)

CNT 5

(Sum of the y data)

SUMY 25

(Sum of the x data)

SUMX 25

(Sum of the x data by y data)

SUMXY 149

(Sum of the square of the x data)

SUMX2 161

(Sum of the square of the y data)

SUMY2 157

Subsequently

READY P0

(Mean of the x data) $SUMX/CNT$

5

(Mean of the y data) $SUMY/CNT$

5

(Standard deviation of x) $SDXN$

2.683281573

(Standard deviation of y) $SDYN$

2.529822128

Although the total and mean values of products x and y are the same according to the calculation results, variations in shipments will be larger for product x as its standard deviation is larger.

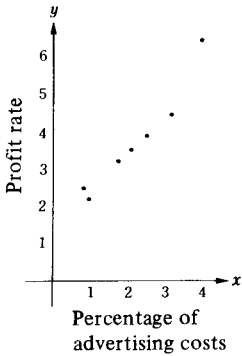
Next, carry out a regression calculation on the pair of data and obtain the correlation coefficient and estimated values.

Example:

The table below indicates the percentage of advertising costs (advertising costs/operating costs $\times 100$) and the percentage of operating profits (operating profits/sales $\times 100$) of 7 chain super markets for the previous fiscal year. Now, let's see if this advertising was effective.

	1	2	3	4	5	6	7
Percentage of advertising costs	0.8	2.1	2.5	1.8	3.1	4.0	1.0
Percentage of operating profits	2.5	3.4	3.7	3.2	4.3	6.3	2.3

Let's prepare a scatter diagram from this table.



The scatter diagram shows that profits are increasing in line with advertising costs. A line connecting the plotted points in the diagram is called the regression curve and, in this case, it is a linear regression as the line is almost straight. In regression curves, we also have logarithm curves, exponential curves and powers curves that can be appropriately selected according to the relation between the x and y data.

It is also known that correlation coefficient (r) is within the range of $-1 \leq r \leq 1$ and that correlation will be positive when $0 < r \leq 1$, negative when $-1 \leq r < 0$, and no correlation when $r = 0$.

Knowing this, let's enter the data for the 7 stores and obtain the statistical data for each.

Operation:

```

STAT CLEAR
STAT0.8 2.5 STAT2.1 3.4 STAT2.5 3.7
STAT1.8 3.2 STAT3.1 4.3 STAT4.0 6.3
STAT1.0 2.3
    
```

(Linear regression constant term: A) LRA	ENTER	1.174221646
(Linear regression coefficient: B) LRB	ENTER	1.142512973
(Correlation coefficient: r) COR	ENTER	0.9628252383

From the value of r , we now know that x and y are in positive correlation. Let's now try to determine what the percentage of advertising cost should be for an operating profit rate of 5.7%, and also try to estimate the operating profit rate with advertising costs of 4.5%.

Operation:

EOX5.7	ENTER	3.961248986
EOY4.5	ENTER	6.315530022

The answer is that the percentage of advertising costs for an operating profit rate of 5.7% will be 3.96%, and that the operating profit rate will be about 6.32% with advertising costs at 4.5%.

■ Logarithm Regression, Exponential Regression and Powers Regression Calculations

Using the data in the table, perform the various regression calculations. Denote x as the fiscal year and y as the achievements.

Fiscal year	Achievements
54	5,810
55	5,637
56	6,736
57	7,938
58	8,169

● Logarithm Regression Calculation

Regression formula is $y = A + B \cdot \ln x$. Input the log (ln) of x for data x , and data y as in linear regression. $\Sigma \ln x$ will be calculated for Σx , $\Sigma (\ln x)^2$ for Σx^2 and $\Sigma \ln xy$ for Σxy .

Operation:

STAT CLEAR

STAT LOG54 5810 STAT LOG55 5637

STAT LOG56 6736 STAT LOG57 7938

STAT LOG58 8169

(Regression constant term A) LRA

-151086.8602

(Regression coefficient B) LRB

39240.6409

(Correlation coefficient r) COR

0.9461867989

(Decision coefficient r^2) 2

0.8952694584

● **Exponential Regression Calculation**

Regression formula is $y = A \cdot e^{B \cdot x}$ ($\ln y = \ln A + B \cdot x$). Input $\log y$ (\ln) for data y , and data x as in linear regression. $\ln A$ will be calculated for decision term A , $\Sigma \ln y$ for the sum SUMY , and $\Sigma x \cdot \ln y$ for sum of the squares SUMY^2 .

Operation:

STAT CLEAR

STAT 54 LOG5810 STAT 55 LOG5637

STAT 56 LOG6736 STAT 57 LOG7938

STAT 58 LOG8169

(Regression constant term A) EXP LRA

21.93154256

(Regression coefficient B) LRB

0.102384121

(Correlation coefficient r) COR

0.9442661562

● **Powers Regression Calculation**

Regression formula is $y = A \cdot x^B$ ($\ln y = \ln A + B \cdot \ln x$). Input logarithms for both data x and y . $\ln A$ will be calculated for constant term A , $\Sigma \ln x$ for Σx , $\Sigma (\ln x)^2$ for Σx^2 , $\Sigma \ln y$ for Σy , $\Sigma (\ln y)^2$ for Σy^2 and $\Sigma (\ln x \cdot \ln y)$ for Σxy .

Operation:

STAT CLEAR

STAT LOG54 LOG5810 STAT LOG55 LOG5637

STAT LOG56 LOG6736 STAT LOG57 LOG7938

STAT LOG58 LOG8169

(Regression constant term A) EXP LRA

6.651154824E-07

(Regression coefficient B) LRB

5.725355325

(Correlation coefficient r) COR

0.9433168782

2-6 Scientific Constants

The FX-750P is provided with 10 scientific constants which can be recalled by means of the number keys (\boxed{F} ~ $\boxed{9}$) in the function mode.

Operation	Name	Symbol	Numerical value	Unit
$\boxed{F}\boxed{0}$	Acceleration of free fall	g	9.80665	ms^{-2}
$\boxed{F}\boxed{1}$	Speed of light (in space)	c	299792458	ms^{-1}
$\boxed{F}\boxed{2}$	Plank's constant	h	6.626176×10^{-34}	J s
$\boxed{F}\boxed{3}$	Gravitational constant	G	6.672×10^{-11}	$\text{Nm}^2\text{kg}^{-2}$
$\boxed{F}\boxed{4}$	Elementary charge	e	$1.6021892 \times 10^{-19}$	C
$\boxed{F}\boxed{5}$	Electron mass	m_e	9.109534×10^{-31}	kg
$\boxed{F}\boxed{6}$	Atomic mass	u	$1.6605655 \times 10^{-27}$	kg
$\boxed{F}\boxed{7}$	Avogadro constant	N_A	6.022045×10^{23}	mol^{-1}
$\boxed{F}\boxed{8}$	Boltzmann's constant	k	1.380662×10^{-23}	JK^{-1}
$\boxed{F}\boxed{9}$	Volume of 1 kg-mole under STP	V_m	0.02241383	$\text{m}^3\text{mol}^{-1}$

Example:

Assume that an ultraviolet light source with a vibration frequency of $\nu = 1.16 \times 10^{15}$ (H_2) is radiating photons of a fixed frequency and calculate the energy of the photons.

If we denote photon energy as E and Plank's constant as h , we obtain $E = 1.16 \times 10^{15} \times h$ (press $\boxed{F}\boxed{2}$) from the photon energy equation $E = h\nu$.

Operation:

1.16E15 * $\boxed{F}\boxed{2}$ $\boxed{\text{ENTER}}$

7.68636416E-19

Example:

Obtain the average value of the translational movement energy possessed by 1 molecule of an ideal gas at 0°C. If we denote the average value of kinetic energy as $\langle K \rangle$, the absolute temperature as T (0° = 273) and Boltzmann's constant as k, we obtain $\langle K \rangle = \frac{3}{2} \times 273 \times k$ ($\frac{3}{2}$ to obtain this) from equation $\langle K \rangle = \frac{3}{2}kT$, which is the average value of the kinetic energy of thermal movement.

Operation:

3 $\frac{3}{2}$ * 273 * $\frac{3}{2}$ ENTER

5.65381089E-21

Learning Science with the FX-750P

• The planet Jupiter, which is the largest planet in the solar system, is said to be 778,300,000 km from the sun. Let's determine how long it takes for light to reach Jupiter. If we denote t as time, S as distance and c as speed of light, we derive $t = \frac{S}{c}$ from the formula of speed of light $c = \frac{S}{t}$. Now, if we substitute with numerical values, we derive

$$t = \frac{778300000000 \text{ [m]}}{\text{Speed light (} \frac{3}{1} \text{)}} = \frac{7.783 \times 10^{11} \text{ [m]}}{\frac{3}{1}}$$

Operation:

7.783E 11 $\frac{3}{1}$ ENTER

2596.129353

The time is therefore about 2,596 seconds or about 43 minutes and 16 seconds.

• Speed and Revolution Cycle of a Satellite

Let's calculate the speed and cycle of revolution of a satellite following a circular orbit around the earth at a distance of 5,500 km. Assume that the radius of earth is 6,400 km and its mass 6×10^{24} kg.

v = Speed of the satellite T = Revolution cycle of the satellite

m = Mass of the satellite M = Mass of earth ($= 6 \times 10^{24}$)

r = Radius of revolution of the satellite ($= (6400 + 5500) \times 10^3 = 1.19 \times 10^7$)

G = Gravitational constant (press $\boxed{F}\boxed{3}$ to obtain).

Since the satellite is carrying out a uniform circular motion, we can set up a balance of forces equation where universal gravity = centrifugal force. In other words, we can set up the equation as $G \frac{mM}{r^2} = \frac{mv^2}{r}$.

From this equation, we derive $v = \sqrt{\frac{GM}{r}}$ $T = \frac{2\pi r}{v}$

We can now obtain velocity v by substituting numerical values and calculate

$$v = \sqrt{\frac{G \cdot 6 \times 10^{24}}{1.19 \times 10^7}}$$

Operation:

SQR $\boxed{}$ $\boxed{F}\boxed{3}$ $\boxed{\times}$ 6E24 $\boxed{/}$ 1.19E7 $\boxed{)}$ ENTER

5800.028977

As shown on the display, the answer is a speed of about 5800 m/sec.

Next calculate T .

$$T = \frac{2\pi \cdot 1.19 \times 10^7}{v}$$

If key operation is continued from the velocity calculation, the answer can be obtained with the \boxed{ANS} key.

Operation:

2 $\boxed{\times}$ PI $\boxed{\times}$ 1.19E7 $\boxed{/}$ ANS ENTER

12891.29855

CHAPTER 2 CALCULATION FUNCTIONS

The answer is therefore a revolution cycle of about 12,891 seconds, from which the hour can be obtained.

Operation:

DMS\$ () ANS / 3600 () ENTER

3° 34' 51.3

The answer is about 3 hours, 34 minutes and 51 seconds.

CHAPTER

3

"BASIC" REFERENCE

In the previous chapter you learned calculating functions so you now understand that the FX-750P can also be used as a scientific calculator.

In this chapter you will now learn programming for full utilization of the many and varied functions of the FX-750P.

3-1 Program

When we consider recent scientific achievements, the Space Shuttle will probably be the first item to come to mind. Compared to the huge rockets that were launched in the past, it may be said that the light and compact Space Shuttle is a machine befitting this age of conservation of resources.

As we are all well aware, the Space Shuttle is controlled by computers and some of the controls carried out are as follows.

- The Shuttle is launched in an upright posture and its 2 boosters (initial launching rockets) are detached from the orbiter at an altitude of 44 km (27.3 miles).
- The detached boosters are dropped in a computed area of the sea from where they are recovered for reuse.
- The main engine burns for 8 minutes and the outer tank is detached at an altitude of 109 km (67.7 miles). This occurs just prior to the Shuttle entering its orbit.
- Upon entering its orbit, the Space Shuttle circles the earth at a speed of 7,743 m/sec. (25,404 ft/sec.).
- The rate of deceleration when leaving its orbit is 91 m/sec. (299 ft/sec.).

A host of other complex controls are processed by computers and these computers are controlled by precision programs. It may therefore be said that programs are accurate and, moreover, detailed work instructions to enable the computer to execute its task correctly. Once the program is memorized in the computer, the computer functions faithfully in compliance with the program. Now, if we wish the computer to carry out some of our daily work, how can we go about in giving correct instructions to the computer.

3-2 The Flow Chart

To give accurate instructions to the computer, we must first analyze what we wish the computer to do.

Simple work may be analyzed in our mind but, if we attempt to analyze complicated work in this manner, it will be extremely difficult, if not impossible, to arrive at a work procedure; or even if we think that an analysis is complete, an important procedure may be missing.

We therefore often write down the important points or use diagrams to arrange the processes. In this manner, it will be possible to view the overall procedure on a systematic basis. We can also look for any errors or omissions in the analysis.

■ Preparing the Flow Chart

There are a number of methods of explaining work procedures easily and efficiently and one of these is the flow chart method. In the flow chart method, the various procedures of work are explained in simple block form with each block showing the meaning of that procedure. The procedural relations and the work flow are indicated by connecting these blocks with lines.

We shall now use a simple example to explain how to prepare the flow chart.

When you use the FX-750P, you must always switch on the power.

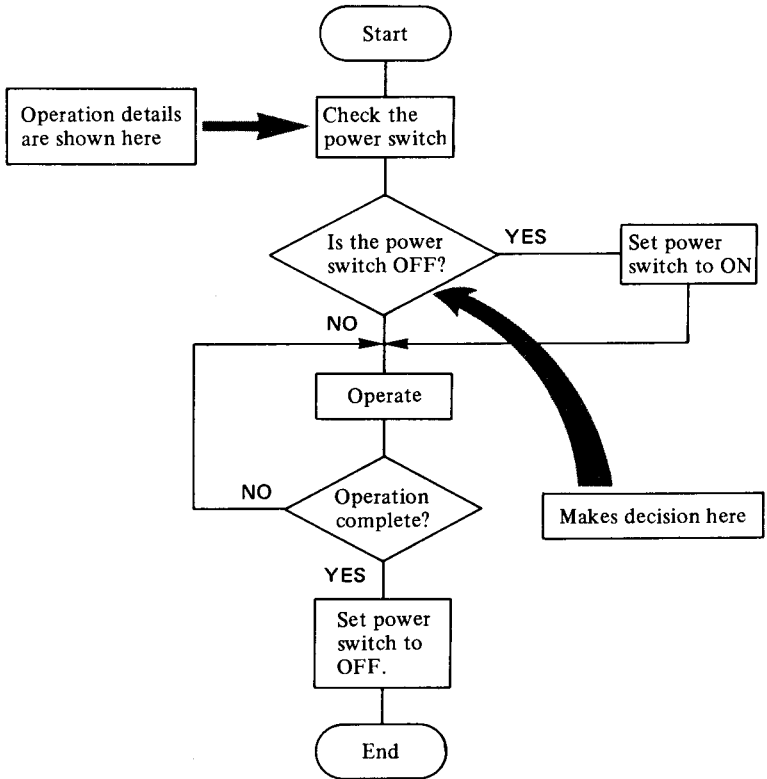
Let's then analyze the operation of switching on the power and prepare a flow chart.

When we use the computer, we turn the switch "ON". When we are finished, we turn the switch "OFF". If the switch is already ON, we use in that state. We do not turn the switch "OFF" while carrying out an operation.

From the above items, we derive the following procedure.


1. Check the power switch.
2. Set to ON if OFF. If already ON, proceed to the next step.
3. Carry out the operation.
4. If the operation is incomplete, return to Step 3. If the operation is complete, proceed to the next step.
5. Set power switch to OFF.

Now, prepare a flow chart based on Steps 1 to 5.




Complex operations can therefore be in easily understood form by using flow charts. Programs used in computers are prepared based on the same type of procedure used in preparing flow charts.

Some of the symbols frequently used when preparing flow charts are shown below. (See page 327)

Terminal 


This terminal symbol is used at the beginning and end of a flow chart.

Input 

Indicates input by manual operation

Process 

Indicates execution of process in the block.


Display 

This symbol is used when displaying processed results, etc.

Flow line 

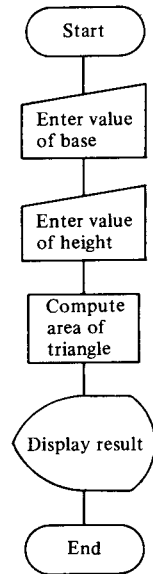
This is an arrow line connecting 2 symbols and the work proceeds according to the direction of the arrow.

As flow charts are basically written from top to bottom or from left to right, the work will proceed down or to the right if the arrows are omitted.

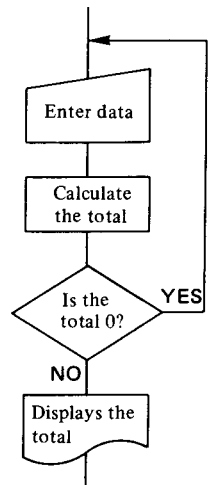
Decision 

Decisions become necessary as the operation proceeds. Decisions, queries or comparisons are made in relation to the items in the blocks. There are generally 2 or more exits (branches).

Example



Example



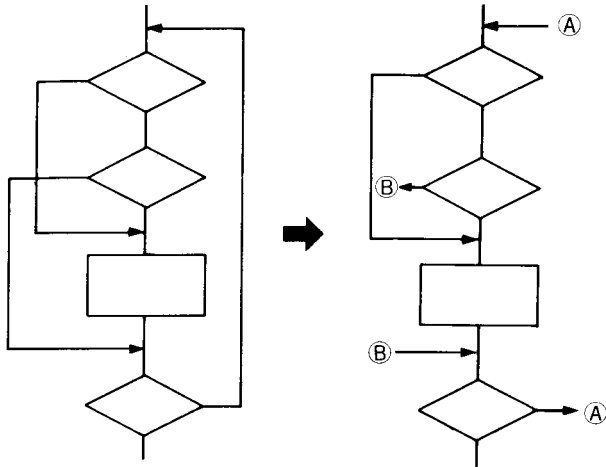
Document output



This symbol is used when outputting calculation results and various documents to the printer.

Connector ○

This indicates exit to a place outside the flow chart or entry to the flow chart.

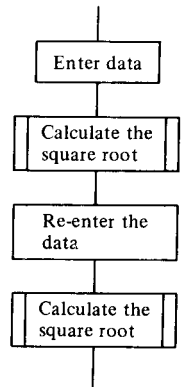


Defined symbol



Indicates process defined at a different place such as subroutine.

Example



3-3 Programming

Explanations up to this point were in relation to analyzing the work and on preparing a flow chart. We shall now proceed with actually preparing a program and operating the computer.

There are many different types of computer languages. Large-scaled computers generally use the COBOL or FORTRAN as a language while BASIC is most widely used in personal computers. BASIC is used in the FX-750P. The BASIC language was developed for the beginners with simplicity of programming in mind. So beginners who are working with computers for the first time will have no trouble in learning this language.

■ Preparing a Sample Program

Create a program with the simple example of “obtaining the cube root x of numerical value y ”.

1. Task Analysis

- 1) Enter a certain number y from the keyboard.
- 2) The equation for obtaining cube root x will be as follows.

$$x = e^{\frac{\log_e y}{3}}$$

$$y = x^3$$

If we take the log of both sides, we derive

$$\log_e y = 3 \log_e x$$

Therefore

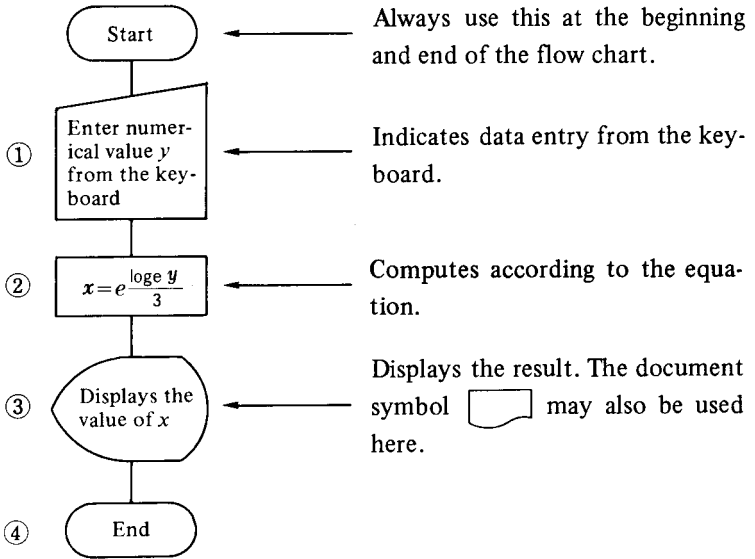
$$\log_e x = \frac{\log_e y}{3}$$

$$x = e^{\frac{\log_e y}{3}}$$

- 3) Display cube root x obtained.

2. Creating the Flow Chart

If we put this analysis results in flow chart form, it will appear as follows.



Since there are four procedures ①, ②, ③ and ④ that will be the object of the programming, create a program for these procedures.

3. Programming

Write a program according to Steps 1 to 4 of the flow chart.

- ① 10 INPUT Y ← Enter a numerical value from the keyboard.
- ② 20 X=EXP (LOG Y/3) ← Computes based on the equation.
- ③ 30 PRINT X ← Displays computed results.
- ④ 40 END ← Execution terminates.

Programming therefore enables troublesome cube root computations to be made in only 4 lines.

We shall now explain a bit further about this program.

The numbers 10, 20, 30 and 40 at the head of each line are called line numbers and indicate the sequence of processing in the computer. The sequence normally starts from the smaller number as in the above example but it is not particularly necessary to start from line 10. The sequence can start from any number large or small but it is important that the program is written in the order of the flow chart.

The word INPUT on line 10 is one of the commands in BASIC and means the computer is ready for an input. In this program, it means to input y on which the cube root is to be obtained.

The word PRINT on line 30 is also a command in BASIC meaning to display and, in this program, means to display the results of the computation.

In the case of line 20, the formal method of writing this is

```
20 LET X=EXP( LOG Y /3 )
```

LET is also a command in BASIC and is called an assignment statement as it means to assign the content of the right side to the variable on the left side. As this command can be omitted, it is normally left out as in the previous example.

This statement is to carry out a function computation and EXP is to give the exponential function e^x to x and LOG is to give the natural logarithm $\log_e x$ to x . The slash (\diagdown) here means \div . In the BASIC language, characters such as X and Y are called variables and numerical values such as 3 are called constants.

Refer to the Command Reference Chapter 5, for the meanings and the input range of the functions.

Programs created with the BASIC language not only use easily understood commands but also use equations that we use in our daily activities.

3-4 Variables

You have now learned how to create a program based on a flow chart so we shall now explain the functions of the characters X and Y (variables) used in the program.

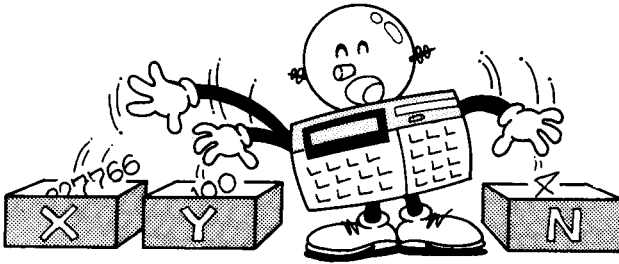
General equations such as

$$x = e \frac{\log y}{n}$$

mean that the values on the left and right are equal but, in the BASIC language, the meaning is different; it means to assign the content of the right side to the left side. The expression may be $x = e \frac{\log y}{n}$ but it cannot be expressed as

$$e \frac{\log y}{n} = x$$

and naturally equations such as $1 = x$ cannot be used. The reason for this is that when variables such as X and Y are used in computers, they indicate the place of storage of numerical values or characters. For a clearer understanding, just imagine that the computer has boxes marked X and Y in which the numerical values are placed.



Therefore, when computing $x = e \frac{\log y}{n}$, the numerical values in boxes N and Y are taken out and computed and the result is placed in box X. By using this method, it may actually be expressed as

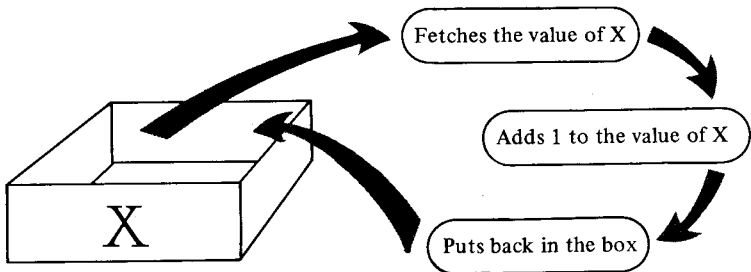
$$X = X + 1$$

Although this cannot be established as an identity, the following will be performed in the computer as the content of on the right side is to be assigned to the left side.

- 1) Adds 1 to a numerical value taken from box X.
- 2) Puts the result in box X again.

This method such as adding 1 to the value of X is widely used in computers.

The makeup of $X = X + 1$



■ Using the Array Variable

An array variable is a variable whose function is expanded. This array variable is highly effective when many variables are used for the same purpose.

For example, if we wish to process data for 1 month, a maximum of 31 variables will be required. The program becomes complex when we attempt to prepare these by using such variables as

A1, A2, A3, ..., A0, B1, B2, ..., B0, ..., C9, D1

so we simply use an array variable which consists of a variable with a number attached.

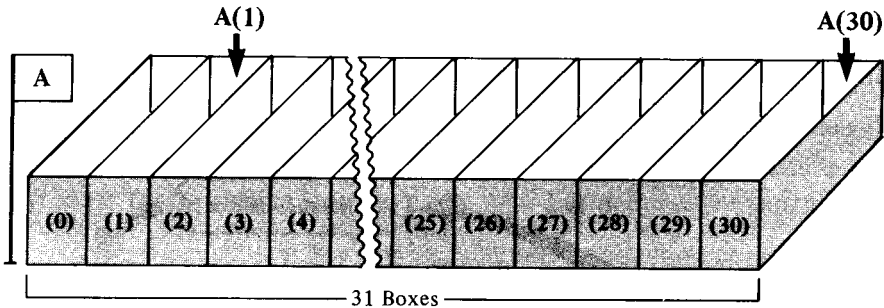
Example:

A(30)

This variable can be used with the DIM command.

10 DIM A(30)

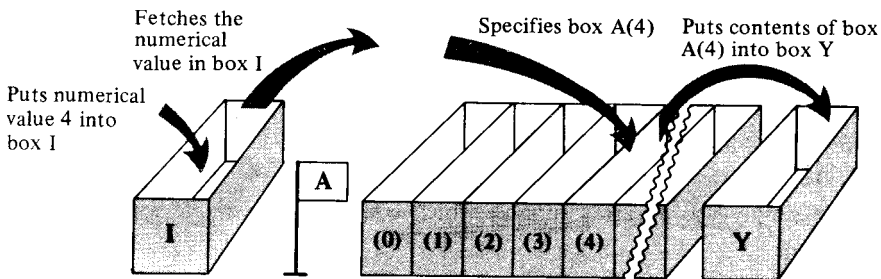
In this example, we will create an array variable called A which has 31 boxes from 0 to 30. Simply assume that there are 31 boxes called A and that they are differentiated by assigning numbers (parameters) from 0 to 30.



The numerical values in these boxes can be taken out or put in by simply changing the parameters such as A(0), A(30), etc. Program creation is simplified since variables can also be used in parenthesis instead of numbers.

Example:

Fetch from the box of the same number as when I = 4, Y = A(I).



We believe you now understand that large amounts of data can be easily handled by using the array variables.

Further explanations in relation to variables are given in the Chapter 4, APPLICATIONS, together with actual program examples.

■ Types of Variables

Variables consist of the numerical variables which contain numerical values, and character variables which contain character strings. The variables we have discussed up to this point are numerical variables in which numerical values are stored for computations. Now we come to the character variable which consists of an alphabetical letter with a \$ sign attached such as A\$ ~ Z\$.

The types of variables in the FX-750P are as shown below.

Numerical variables

- Fixed variables: A, B, etc. (up to 12 digits)
- Registered variables: A1, X9, etc. (up to 12 digits)
- Array variables:
 - Half-precision numerical array
 - A! (10), etc. (up to 5 digits)
 - Single-precision numerical array
 - A(10), etc. (up to 12 digits)

Character variables

- Fixed variables: A\$, B\$, etc. (up to 7 characters)
Example: A\$ = "FX-750P"
- Registered variables: A1\$, X9\$, etc. (up to 16 characters)
Example: B1\$ = "DEVIATION"
- Array variables: Number of characters can be specified, from 0 to 79 characters.


Fixed Variables (A to Z or A\$ to Z\$)

There are 26 numerical fixed variables from A to Z and 26 character fixed variables from A\$ to Z\$. Either the numerical value box or the character string box only can be used for the fixed variable. A UV ERROR will occur if we attempt to use a numerical fixed variable and a character fixed variable of the same name (e.g. A and A\$).

Registered Variables

Variable names with 2 characters can be used by using capital alphabetical letters and numerical values. However, if variables with 3 or more characters (excluding \$) are used, an SN ERROR will occur during program execution.

The following rules must be observed in relation to registered variables.

1. The first character must be a capital alphabetical letter.
 2. A reserved word (IF, ON, TO, PI, etc., see page 326) cannot be used.
 3. A numerical registered variable (AB, A1, etc.) can store 12 digits for the mantissa and 2 digits for the exponents.
 4. A maximum of 16 characters can be stored in the character registered variable.
 5. A total of 40 registered variables can be used, including the array variables. However, as a VA ERROR will occur and execution of the program will stop if 40 variables are exceeded, the excess variables must be erased with CLEAR (see page 223) or ERASE (see page 227)
- * The variable name registered can be confirmed by pressing LIST V .

Half-precision and single-precision

Usually internal calculations are performed by a 12-digit mantissa. The attachment of ! to an array variable provides half-precision (up to 5 digits) allowing more data to be stored in the memory in case 5-digit calculation precision is sufficient for data storage. The specification of single precision requires 8 bytes for one numerical variable, but only 4 bytes are required when half-precision is defined. In another word, the specification of half-precision permits effective use of limited memories.

3-5 Program Input and Execution

Explanations up to this point were in relation to the theoretical aspects of the program. Was it difficult? It may have been a bit difficult for some of you but, as they say, practice makes perfect so you will be surprised at how easy it is once you actually start using the computer.

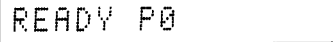
Let's execute the program, created previously to obtain cube root, with the FX-750P. However, it will first be necessary to have the computer store the program. Various methods have recently been developed in relation to storing programs such as the voice input (tone recognition) method and the written character input method (pattern recognition) in the field of large-scaled computers. In personal computers, however, the orthodox method of entering from the keyboard is used.

■ Prior to Inputting Program

Enter the cube root computing program from the keyboard. However, the following operations are necessary in preparation for program input.

Operation:

NEW



NEW is a command to erase the program which is stored in the currently designated program area. This operation is carried out first as it is necessary that the area in which the new program is to be input be cleared. Care should be taken to differentiate NEW and NEW ALL as NEW ALL will erase the programs and variable contents in all program areas.

READY P0 on the display indicates the program area P0 to be used.

The FX-750P has 10 program areas from P0 to P9. Independent programs can be input in the respective program area. This is a highly convenient dividing function which enables free usage of each area within the range of the memory capacity. However, the variables used must be common to all program areas.

■ Program Input





Operation:



10  INPUT Y

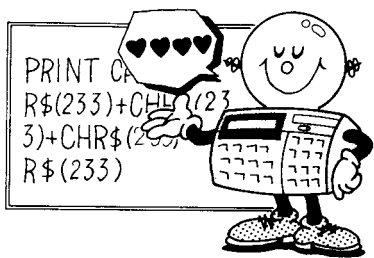
10INPUT Y_



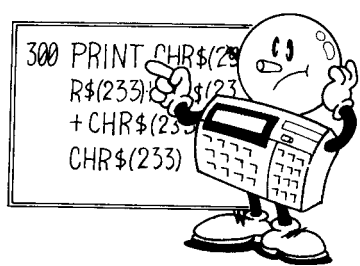
10 INPUT Y

As using one-key command with the  key (e.g.  ) is the same as entering **I****N****P****U****T** with the alphabet keys,  will be omitted from hereon and indications will be with alphabetical letters only.

The  key functions to store the program in the computer at this time. This differs from the function of the  key when executing the NEW command as it has meaning in relation to the number "10" entered in advance of the command. This number is called the line number and programs are stored in sequence from the small line numbers.



If there is no line number, this will be executed immediately.



If there is a line number, this will be stored as a program.

Input the remaining lines in the same manner.

Operation:

20 X=EXP(LOG Y / 3)

20 X=EXP(LOGY/3)

30 PRINT X

30 PRINT X

40 END

40 END

■ Checking the Input Program

To view the input program, enter LIST .

LIST is the command to “display the program”.

All programs can be checked since the programs from line 10 to line 40 will be displayed in successive order.

Press the or key to hold the display stationary. The display will remain still until the or key is pressed again.

■ Correcting the Program

If an input error is discovered when reviewing the program with the LIST command, set to the EDIT mode and correct. EDIT is a command used to edit a program and the program will be displayed from the first line by pressing EDIT . In this instance, the line 10 will be displayed.

Operation:

EDIT

10 INPUT Y_

Differing from the display with the LIST command, you will note that the cursor is blinking at the end of the line. The method of editing in the EDIT mode is the same as the method explained in Chapter 2 for the operation function. Simply move the cursor to the place to be edited and correct, delete (use) or insert (use). Then press the key and the corrected line will be

written and the next line will be displayed. If **SHIFT** **↵** is pressed after correcting, the corrected line will be written and the previous line will be displayed. Press the **↵** key for lines having no error. Press the **BRK** key to cancel the EDIT mode.

■ Program Execution

Now that we have finally input an error free program, let's actually operate the computer and execute this program. We must perform the following operations to execute this program.

Operation:

RUN **↵**

?

RUN is a command to "execute the program". When the program is executed, a question mark (?) will be displayed. This is a display requesting input as the **INPUT** command on line 10 has been executed. Let's enter 64 for example.

Operation:

64 **ENTER**

(**↵** can also be used. Same hereafter)

4 **WAIT**

The cube root of 64 is computed on line 20 and the result is displayed by means of the **PRINT** command on line 30. The word **WAIT** on the upper part of the display is a sign that the **PRINT** command has been executed and that the computer is waiting for entry from the **↵** key or **ENTER** key.

Operation :

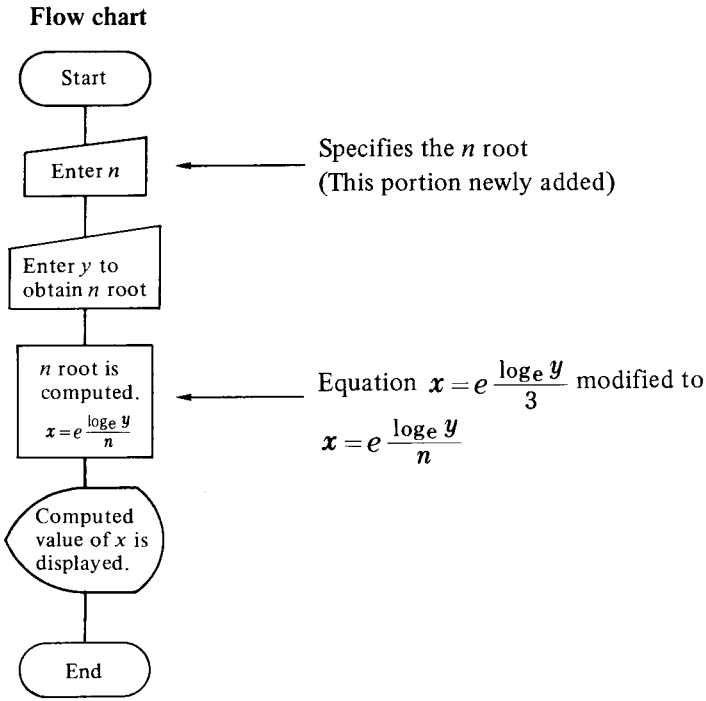
ENTER

READY P0

This command is to return the executed program to its original status. We suggest you execute a few more programs by changing the values.

3-6 Program Modification

There will be cases when it will be desirable to modify the program to correspond with work requirements. As the program currently stored is to compute cube roots, let's modify the program to obtain n roots. To accomplish this, we simply change the constant 3 for a cube root to a variable N and then compute the n root.



The program for this flow chart will be as follows.

```

5 INPUT "N=" ; N
10 INPUT "Y=" ; Y
20 X=EXP ( LOG Y/N )
30 PRINT X
40 END
  
```

This is called a character string and, if enclosed with quotation marks, will be handled as a character even if numerical values.

When a character string is used, a semicolon will be required in front of the variable.

Let's review the portion modified.

First, line 5 was added. This is to input N for the n root. The reason that 5 was selected for the line number is that it was necessary to execute this line ahead of line 10 so a number smaller than 10 was needed. Any number under 10 (1 to 9) could have been used here.

5 INPUT "N=" ; N

"N =" ; has been entered following the INPUT command. This is a character string to show what the computer is querying, and N = ? will be displayed indicating that the computer is in waiting mode for n root instructions. If we use a comma in place of the semicolon, a question mark (?) will not be displayed.

A "Y =" ; portion has also been added to line 10.

As /3 on line 20 has been changed to /N, computation has been changed from cube root to n root.

■ Program Modification Method

1) Add the entry portion of N for n root.

Operation:

5 INPUT "N=" ; N

```
5 INPUT "N=" ; N
```

2) Recall line 10 with the EDIT mode

Operation:

EDIT 10

```
10 INPUT Y_
```

3) Move the cursor to position Y. Enter "Y =" ; Y anew.

Operation:

Y "Y=" ; Y

```
20 X=EXP(LOGY/3)_
```

4) As line 20 will be displayed, move the cursor to position 3 and enter N.

Operation:

3 N

```
30 PRINT X_
```

5) Line 30 displayed but no modification required here.

Press the **BRK** key and cancel the EDIT mode.

Operation:

BRK

READY P0

After the completion of modification, display the program again by using the LIST command and confirm that the modification was correctly carried out.

■ Executing the Modified Program

Execute the modified program using the RUN command.

Example:

Obtain the 4th root of 100.

The comment $N = ?$, meaning N of the n root, will be displayed.

Operation:

RUN **↵**

$N = ?$

Enter the number 4 since we are looking for the 4th root.

Operation:

4 **ENTER**

$Y = ?$

Now, enter 100.

Operation:

100 **ENTER**

3.16227766 ^{WAIT}

The 4th root of 100 is computed and displayed. A WAIT mark will also appear on the upper part of the display.

* The Program obtaining the n th root would be of some assistance for you to be accustomed to programming. Manual calculation using “ \wedge ” gives the same calculation result.

Example: $100 \wedge (1/4)$ **ENTER**

Press the **ENTER** key (or **↵** key) after viewing the computation results and execution of the program will be terminated.

3-7 Precautions for Practical Programs in Chapter 4

Practical programs are given and BASIC is explained by actual use in the applications appendix of Chapter 4. As inputting long programs accurately is difficult, input operations and precautions, and error displays and their corrective measures are listed to enable smooth input of the application programs in Chapter 4.

■ Input Operations and Precautions

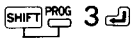
1. Specifying the program area

As the FX-750P is provided with program dividing functions, first specify where the input is to be made in the 10 areas (P0 ~ P9).

Example:

Specify P3 as the program area.


Operation:




READY P3

2. Clear the program in the specified program area and all variables.

Operation:











NEW  (Clears the program)

CLEAR  (Clears all variables)

3. Use the one-key command for inputting and be sure to remember the position.

4. Always press the key and store after completing input of each line.

5. If an entry error is noticed while inputting a program, proceed as follows.

- Before pressing the  key Use the   keys and move the cursor to the place to be corrected and correct the input.
- After pressing the  key Recall the line to be corrected using EDIT, Line No.  and move the cursor by using   keys to the place to be corrected, correct the entry and press the  key.
- Forgot to enter one line Enter at a convenient place during the input operation.
- Delete a whole line Enter DELETE Line No.  or Line No. .

6. Checking the printing

```

200 PRINT USING"###
    #.###";R;" OHM"
210 PRINT "X=";
220 PRINT USING"###
    #,###";X;" OHM"
230 END


```




Space automatically inserted (No need to press **SPC**)

Press the **SPC** key for a space enclosed with double quotations ("").

Do not mistake a semicolon (;) this for a colon (:).

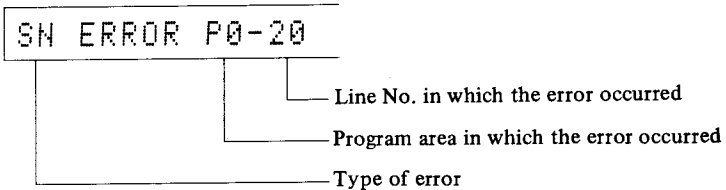
Be careful of periods (.) and commas (,) and use the decimal key for periods.

7. After completing input, check the program list with LIST .

Press  or **ENTER** to stop the display and press these again to continue the display. If an input error is noticed when checking, press the **BRN** key, recall the line to be corrected with EDIT Line No.  and, after correcting the input, press the  key.

■ Error Displays and Corrective Measures

There will be cases when an error will be displayed and program execution stopped, or, even if the program is executed, the result will be incorrect. This will be due to errors, or "bugs" in the program and correcting these errors is called "debugging".



Error displays during program execution will show the type of error, the program area and the line No.

SN ERROR is a syntax error which is displayed when there is a format error in the program. As there are 21 errors including OV ERROR and BS ERROR, check the type of error from the list of error messages on page 320 and determine the cause.

Enter EDIT 20 to display the line to be corrected and, after correcting, press the key. Pressing the key will release the EDIT mode and READY Pn will be displayed enabling continuation of program execution.

Errors that violate the BASIC grammar will be displayed. However, pay close attention when inputting since ERROR will not be displayed if the wrong expression was set up or if the wrong equation was entered, but the results obtained will be incorrect.

4

CHAPTER

APPLICATIONS

With the explanations given in the previous chapter in relation to the BASIC language and programming, we believe you now have a general knowledge of computers.

Practical programs to utilize the full capabilities of the FX-750P will be given in this chapter together with comments in relation to BASIC commands.

4-1 Foreign Currency Conversions

We are often confronted with the bothersome task of converting foreign currencies during trips abroad for sightseeing or for business.

This program displays the amount in each currency immediately upon entering the exchange rates against a specified foreign currency. Each amount will also be displayed when currencies are converted to a specified foreign currency.

Mutual conversions among U.S. dollars, Japanese yen, German marks, French francs, and English pounds will be made here.

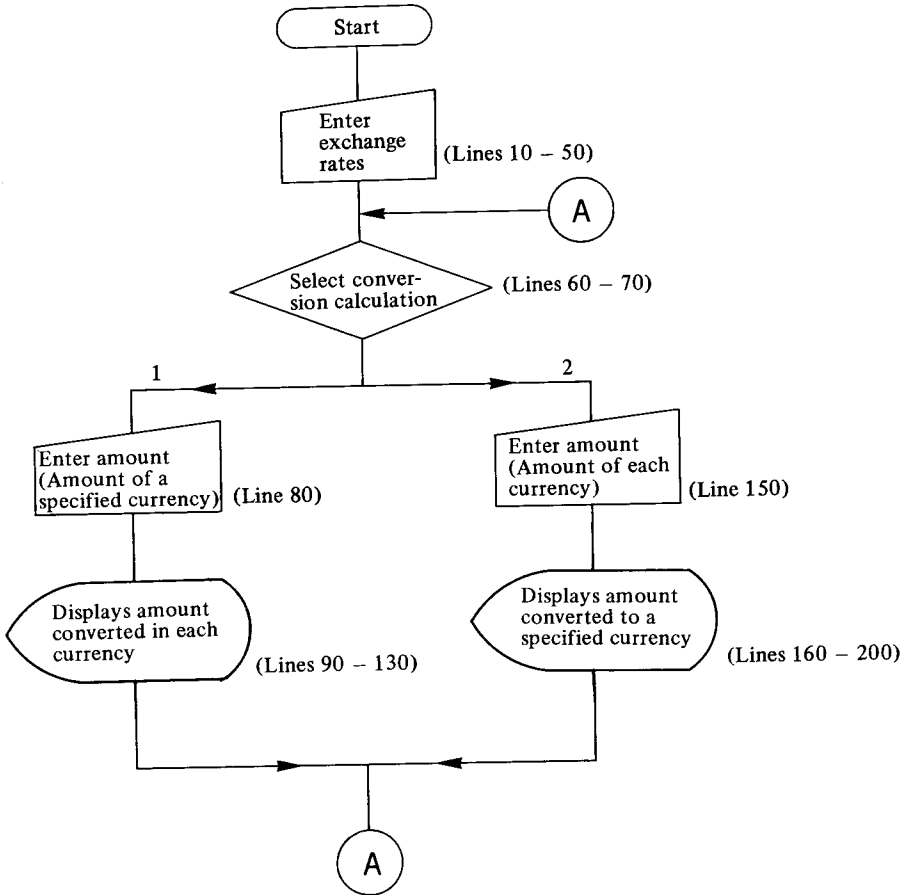
Commands

PRINT, INPUT, GOTO, IF ~ THEN ~ ELSE

Program List

```
10 INPUT "DOLLAR="
   ;A
20 INPUT "YEN=";B
30 INPUT "MARK=";C
40 INPUT "FRANC=";
   D
50 INPUT "POUND=";
   E
60 INPUT "1 OR 2";
   X
70 IF X=2 THEN 150
80 INPUT "AMOUNT="
   ;Y
90 PRINT Y*A; "DOLL
   AR"
100 PRINT Y*B; "YEN"
110 PRINT Y*C; "MARK
   "
120 PRINT Y*D; "FRAN
   C"
130 PRINT Y*E; "POUN
   D"
140 GOTO 60
150 INPUT "AMOUNT="
   ;Y
160 PRINT Y; "DOLLAR
   ";Y/A
170 PRINT Y; "YEN";Y
   /B
180 PRINT Y; "MARK";
   Y/C
190 PRINT Y; "FRANC"
   ;Y/D
200 PRINT Y; "POUND"
   ;Y/E
210 GOTO 60
```

Flow chart



CHAPTER 4 APPLICATIONS

Enter the exchange rates for a specified currency and convert to your own or other country currencies. We will use 1 U.S. dollar as the standard and make approximate calculations here since the rates fluctuate daily. Accurate conversions can be made by setting the base currency as 1 and entering the exchange rates for that day.

Data input

Unit currency	Exchange rate
U.S. dollar	\$ 1
Japanese yen	¥ 225
German Mark	DM2.62
French franc	FF7.94
English pound	ST. £. 0.66

Operation:

RUN ↵

DOLLAR=?

Set U.S. dollars as 1 since conversions will be made against dollars.

1 ↵

YEN=?

Enter the exchange rate of yen against a dollar.

225 ↵

MARK=?

Enter the exchange rate of mark against a dollar.

2.62 ↵

FRANC=?

Enter exchange rates for francs and pounds in the same manner.

0.66 ↵

1 OR 2?

The necessary data have now been entered. Assume we bought an item for US\$3,675. Then select 1 if you wish the converted amount in each currency.

1

AMOUNT=?

Enter the dollar amount.

3675

3675DOLLAR

826875YEN

The converted Japanese yen amount will be displayed.

9628.5MARK

Amounts for francs and pounds will be displayed in the same manner.

1 OR 2?

Next, select 2 if you wish to know what a certain amount in each currency will be when those are converted to dollars.

2

AMOUNT=?

The amount converted to dollars of 80,000 in each currency will be displayed.

80000

80000DOLLAR 80000

80000YEN 355.5555556

80000MARK 30534.35115

1 OR 2?

Since this program is endless, press to terminate.

■ Let's Display Characters (PRINT command)

90 PRINT Y*A; "DOLLAR"

PRINT command Display contents

The PRINT command on line 90 allows the display of the result of A times Y and DOLLAR.

The PRINT command may be used as a single command only but it is normally used with numerical expressions and character strings.

Following are examples of various methods of use.

Erases the display with a single PRINT statement only.

10 PRINT

RUN ↵

READY P0

Displays a numerical value if it follows a PRINT statement.

10 PRINT 18

RUN ↵

18

Displays a character string if it is enclosed with "" (double quotations).

10 PRINT "CASIO"

RUN ↵

CASIO

Displays a calculated result of a numerical expression, if any.

10 PRINT 18*21

RUN ↵

378

Character strings will be displayed by linking with the + symbol.
Use the **SPC** key to insert space.

```
10 PRINT "CASIO "+
  "FX-750P"
```

RUN ↵

CASIO FX-750P

Displays an assigned value if a variable exists.

```
10 A=3.19
20 PRINT A
```

RUN ↵

3.19

Displays messages enclosed with " " if a character variable exists.

```
10 A1$="COMPUTER"
20 PRINT A1$
```

RUN ↵

COMPUTER

When displaying numerical values, a space will be given ahead for a positive number and a - sign will appear ahead for a negative number.

If the PRINT statement is executed, the data will be displayed and the program execution will stop with WAIT displayed. If time is set by the WAIT command (see page 250), the program execution is temporarily stopped as specified.

If not set, the program execution is being stopped unless **ENTER** or ↵ is pressed.

DEG 3.19

WAIT

When displaying multiple data with a single PRINT statement, punctuate the data with ,(comma) or ;(semicolon).

If , is used, the data will be displayed one at a time.

```
10 FOR I=1 TO 3
20 PRINT I, I*5
30 NEXT I
```

RUN ↵

↵
⋮
↵

1
5
⋮
15

If ; is used, the data will be displayed in a row. ; will not comply with the WAIT command at this time.

```
10 FOR I=1 TO 3
20 PRINT I; I*5
30 NEXT I
```

RUN ↵

↵
↵

1 5
2 10
3 15

The PRINT command has the function of displaying calculation results and characters. (See page 239)

■ Let's Input Data (INPUT command)

```
10 INPUT "DOLLAR=" ; A
      INPUT command  Message  Variable
```

This program line displays the message, DOLLAR=?, waiting for the data input from the keyboard through the INPUT statement on line 10. The input data is assigned to variable A.

The INPUT command can be used by combining with a variable only such as 10 INPUT A. However, as a ? mark only will be displayed, a message statement is required to facilitate the clarification of the data to be entered.

```
10 INPUT "AMOUNT="
;A
```

The message statement AMOUNT requests input of the amount.

```
RUN
```

```
AMOUNT=?
```

The message and the variable must be separated with a ; or , in such a case. Punctuate with a ; to display a ? mark after a message. Punctuate with a , if a ? mark is not necessary. (A message will be required in this case. If no message, an SN ERROR will occur.)

The data format and type of variable must be the same. That is, characters can not be stored in a variable where numerical values are stored. TM ERROR will be displayed if different types of data are entered. Press the or key to make a new entry. (Note: Inputs will still be possible even in TM ERROR state.)

```
ABCDE
```

```
↵
```

```
1200
```

```
TM ERROR
```

```
AMOUNT=?
```

```
READY P0
```

As calculation equations can be used for data inputs, the input data need not be initially calculated.

Calculation equations can be entered as they are.

```
5 B=20
10 INPUT "DATA=";A
20 PRINT A
```

```
RUN
```

```
10+25+37*3
```

```
DATA=?
```

```
146
```

Inputs of functions and variables are also possible. (However, the variables must be preregistered. **Example:** $10 + \sqrt{81} + \sin 30^\circ + B = 10 + 9 + 0.5 + 20$)

RUN ↵

10+SQR81+SIN30+B ↵

DATA=?
39.5

Data can be input into multiple variables through the INPUT statement. The variables are punctuated with a , in this case. The variables should be entered one after another.

```
10 INPUT "AMOUNT="
   ;A,B
```

RUN ↵

100 ↵

150 ↵

AMOUNT=?
?
READY P0

Pressing 100, 150 ↵ will cause a TM ERROR.

4 variables can also be entered consecutively. A minus sign (-) is required when entering a negative number.

```
10 INPUT "AMOUNT="
   ;A,B,C,D
20 PRINT "DATA=";A
   ;B;C;D
```

RUN ↵

4 ↵

-8 ↵

5 ↵

7 ↵

AMOUNT=?
?
?
?
DATA= 4-8 5 7

Flexible character displays are possible because character variables can be used as the message through the INPUT command.

```

10 A$="(APPLE)"
20 INPUT "PRICE"+A → Change the names of the products with the
   $;X             same message (PRICE).
30 A$="(GRAPE)"
40 INPUT "PRICE"+A
   $;Y
50 A$="(PEACH)"
60 INPUT "PRICE"+A
   $;Z

```

RUN ↵

120 ↵

140 ↵

250 ↵

PRICE(APPLE)?

PRICE(GRAPE)?

PRICE(PEACH)?

READY P0

The item name only can be changed and displayed as shown.

The INPUT command has the function of assigning data having been manually input to a variable. (See page 234)

■ To Change the Program Flow (GOTO command)

140 GOTO 60

GOTO command Specifies line number to which the program jumps

When the program is executed up to line 140, it is forced to jump to line 60 by the GOTO command.

Execution of the program jumps to line 10.

```
10 PRINT "CASIO, ";
20 GOTO 10
RUN ↵
```

CASIO, CASIO, CASIO, CASIO,

CASIO will be displayed endlessly. To stop the execution, press **BRK**.

Displays 1.5 times the number entered.

```
10 INPUT "AMOUNT="
   ;X
20 Y=X*1.5
30 PRINT Y
40 GOTO 10
RUN ↵
100 ↵
```

AMOUNT=?
150

In this manner, the GOTO command can perform the same process any number of times.

A UL ERROR will occur if a line number that does not exist in the program is specified.

```
20 GOTO 200
30 GOTO 10
RUN ↵
```

UL ERROR P0-20

The GOTO command will increase in number and become confusing if the structure of the program itself is not carefully considered when creating a program. The structure of the created program will also become difficult to understand at a later date even by the author of the program.

The GOTO command has the function of jumping execution of a program to a specified line number. (See page 232)

■ Conditional Expression (IF ~ THEN)

```
70 IF X=2 THEN 150
```

Condition

The line number to be jumped to (or process to be executed) when the condition is true.

Jumps program execution to line 150 if 2 is assigned to the variable X on line 70. The next line will be executed if X is not 2. Conditional control of program flow is possible by using the IF statement.

Usage of the IF statement

1) IF (condition) THEN (command)

```
10 A=20:B=100
20 IF A=20 THEN B=
   B/2
30 PRINT B
```

If the content of A is 20, the content of B will be one half.

RUN ↵

50

2) IF (condition) THEN (command 1) ELSE (command 2)

```
10 INPUT A
20 IF A<100 THEN P
   RINT "ABC" ELSE
   PRINT "XYZ"
```

Displays ABC (executes command 1) if A is smaller than 100 and displays XYZ (command 2) if it is 100 or larger.

RUN ↵

?

120 ↵

XYZ

A line number can also be used in place of a command.

```

10 INPUT A
20 IF A<100 THEN 3  → Jumps to line 30 if A is smaller than 100 and
   ELSE 50           jumps to line 50 if it is 100 or larger.
30 PRINT "ABC"
40 END
50 PRINT "XYZ"
60 END
    
```

RUN ↵

?

90 ↵

ABC

The command on line 50 will be executed if a value equal to or larger than 100 is entered.

140 ↵

XYZ

These can also be used in combination.

```

10 INPUT A
20 IF A<100 THEN B
   =A*2: GOTO 30 E
   LSE B=A/2: GOTO
   50
30 PRINT B
40 END
50 PRINT B
60 END
    
```

If A is smaller than 100 on line 20, the execution will jump to line 30 with B as twice the value of A.

RUN ↵

?

68 ↵

136

If A is 100 or larger, the command will jump to line 50 with B being 1/2 the value of A.

RUN ↵

134 ↵

?
67

There are 9 relational operators for conditional expressions:

=, <>, ><, >, <, >=, =>, <=, =<

```
10 A=50
20 IF A<>20 THEN P → END is displayed if A is not 20.
   PRINT "END"
30 END
```

The AND and OR conditions can be programmed as follows.

```
10 A=10:B=20
20 IF A=10 THEN IF → HELLO! will be displayed when A = 10 and B =
   B=20 THEN PRIN 20.
   T "HELLO!"
```

```
10 INPUT A
20 IF A=1 THEN 40 → GOOD will be displayed if the input value is 1 or 7.
   ELSE IF A=7 THE
   N 40
30 GOTO 10
40 PRINT "GOOD"
```

IF ~ THEN can be used together in this manner.

The IF ~ THEN ~ ELSE command has the function of making a judgement on specified conditions. (See page 233)

4-2 Circuit Calculations (Impedance calculation)

Calculating circuit impedance is a troublesome task when designing electrical circuits.

This program eases the task by performing the following.

1. Displays impedance and declination if the resistance value and reactance are entered.
2. Displays resistance value and reactance if impedance and declination are entered.

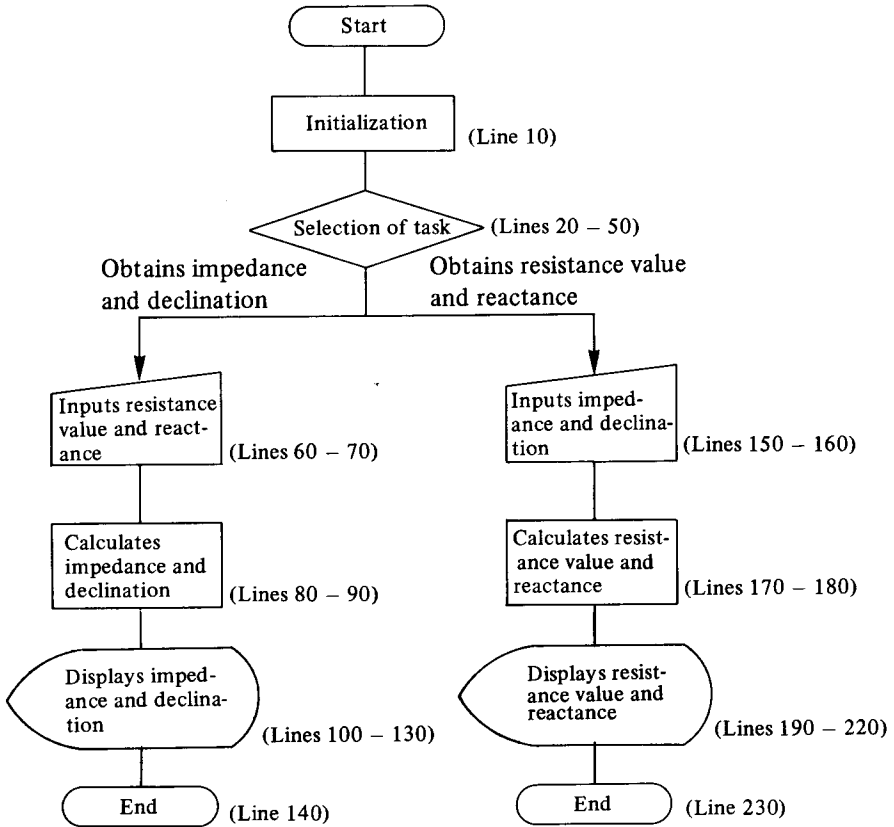
Commands and Functions

SQR、SIN、ATN、USING、ANGLE、LET

Program List

```
10 ANGLE 0
20 INPUT "(1)..Z,S
   (2)..R,X":N
30 IF N<1 THEN BEE
   P : GOTO 20
40 IF N>2 THEN BEE
   P : GOTO 20
50 IF N=2 THEN 150
60 INPUT "R=":R
70 INPUT "X=":X
80 LET Z=SQR(R^2+X
   ^2)
90 LET S=ATN(X/R)
100 PRINT "Z=":
110 PRINT USING"###
   #.###":Z:" OHM"
120 PRINT "S=":
130 PRINT USING"###
   #.###":S
140 END
150 INPUT "Z=":Z
160 INPUT "S=":S
170 LET R=Z*COSS
180 LET X=Z*SINS
190 PRINT "R=":
200 PRINT USING"###
   #.###":R:" OHM"
210 PRINT "X=":
220 PRINT USING"###
   #.###":X:" OHM"
230 END
```

Flow Chart



Impedance and declination can be obtained by entering the resistance value and reactance or, conversely, the resistance value and reactance can be obtained by entering the impedance and declination.

Data 1		Data 2	
Resistance value	5 Ω	Impedance	16 Ω
Reactance	15 Ω	Declination	72°

Operation:

RUN ↵

(1)..Z,S (2)..R,X?

Obtains the impedance and declination first.

1 ↵

R=?

The resistance value is 5 Ω.

5 ↵

X=?

Impedance will be displayed if the 15 Ω reactance is entered.

15 ↵

Z= 15.811 OHM

Obtains the declination.

↵

S= 71.565

↵

READY P0

RUN ↵

(1)..Z,S (2)..R,X?

Obtains resistance value and reactance.

2 ↵

Z=?

Impedance is 16 Ω.

16 ↵

S=?

Displays resistance value if an declination of 72° is entered.

72 ↵

R= 4.944 OHM

Displays reactance if ↵ is pressed subsequently.

↵

X= 15.217 OHM

■ SQR Function

```
80 LET Z=SQR(R^2+X^2)
      SQR function Argument
```

J. G. Kerzinger
Gartenstraße 2
D 8789 Hammelburg

The square root of the sum of R^2 and X^2 is being obtained and assigned to variable Z on line 80. The SQR function is equivalent to the square root sign ($\sqrt{\quad}$) used in mathematics.

Following is an example using the SQR function.

Obtain the square root of 25.

```
10 A=25
20 B=SQRA
30 PRINT A;B
RUN ↵
```

25 5

In regard to numerical functions such as the SQR function, if the argument is a numerical expression, it must be enclosed in parentheses.

```

10 A=SQR200+25
20 B=SQR(200+25)
30 X=200
40 C=SQRX+25
50 D=SQR(X+25)
60 PRINT "A=";A
70 PRINT "B=";B
80 PRINT "C=";C
90 PRINT "D=";D
    
```

On lines (lines 10 and 40) where the numerical expressions are not enclosed in parenthesis, square root calculations will be performed on the first numerical value or variable only. Where numerical expressions are enclosed in parentheses, square root calculations will be performed after calculating these expressions.

RUN ↵

↵

↵

↵

A= 39.14213562
B= 15
C= 39.14213562
D= 15

The square root of a negative number cannot be calculated. That is, the value of numerical expression must be 0 or larger. An MA ERROR will be displayed if we attempt to obtain the square root of a negative number.

```

10 A=25
20 B=-25
30 PRINT SQRA
40 PRINT SQRB
    
```

RUN ↵

↵

5
MA ERROR P0-40

The SQR function obtains the square root of numerical values. (See page 266)

■ SIN, COS, TAN Functions

```
170 LET R=Z*COSS      COS function
180 LET X=Z*SINS      SIN function
```

The resistance value and reactance are being obtained on lines 170 and 180 respectively by obtaining the COS and SIN of the numerical value stored in variable S and multiplying these by Z. Trigonometric functions such as SIN and COS can therefore be used in a program. Although parentheses are not required for SIN, COS and TAN of numerical values stored only in variables, they are required when obtaining SIN, COS and TAN with numerical expressions.

```
10 X=30:Y=15
20 A=SINX+Y
30 B=SIN(X+Y)
40 PRINT "SIN X+Y="
   ":A
50 PRINT "SIN(X+Y)
   "=:B
```

Calculated as $\text{SIN } 30^\circ + 15 = 0.5 + 15 = 15.5$.

RUN ↵

```
SIN X+Y= 15.5
```

Calculated as $\text{SIN } (30 + 15)^\circ = \text{SIN } 45^\circ = 1/\sqrt{2} = 0.7071067812$.

↵

```
SIN(X+Y)= 0.7071067812
```

The following 3 types of angle units can be used for trigonometric functions.

1. DEGREE (DEG)
2. RADIAN (RAD)
3. GRADE (GRA)

These are specified by the ANGLE command.

ANGLE 0 DEG

ANGLE 1 RAD

ANGLE 2 GRA

The numerical values that can be given are restricted to the following range.

DEG $-5400^\circ < \text{Numerical expression} < 5400^\circ$

RAD $-30\pi < \text{Numerical expression} < 30\pi$

GRA $-6000 < \text{Numerical expression} < 6000$

A BS ERROR will be displayed if numerical values outside these ranges are given.

```
10 ANGLE 0:REM DEG
   REE
20 A=SIN(10000)
30 PRINT A
```

This is clearly outside the range of $-5400^\circ < \text{Numerical expression} < 5400^\circ$.

RUN ↵

BS ERROR P0-20

TAN function is also available for obtaining tangent x of given numerical values. If the angle is an odd multiple of 90° ($\pi/2$ RAD, 100GRA), the calculation results will be infinite when using this function. An MA ERROR will be displayed.

The SIN function obtains sine x of a given numerical value.
 The COS function obtains cosine x of a given numerical value.
 The TAN function obtains tangent x of a given numerical value.
 (See page 263)

■ ASN, ACS, ATN Functions

```
90 LET S=ATN(X/R)
```

ATN function Argument

The declination is being obtained on line 90 by calculating the arctangent ($\tan^{-1}x$) of the numerical value obtained by dividing reactance (X) with resistance value (R).

ATN is a function to obtain arctangents. It is called an inverse trigonometric function as angles are obtained by giving numerical values as against trigonometric functions in which values are calculated on given angles.

$$y = \sin x \longleftrightarrow x = \sin^{-1} y \quad \text{Arcsine (ASN)}$$

$$y = \cos x \longleftrightarrow x = \cos^{-1} y \quad \text{Arccosine (ACS)}$$

$$y = \tan x \longleftrightarrow x = \tan^{-1} y \quad \text{Arctangent (ATN)}$$

In the case of ASN and ACS, the argument is restricted to $|\text{argument}| \leq 1$. The results can be obtained within the following range.

$$-90^\circ \left(-\frac{\pi}{2} \text{ RAD}, -100 \text{ GRA}\right) \leq \text{ASN} \leq 90^\circ \left(\frac{\pi}{2} \text{ RAD}, 100 \text{ GRA}\right)$$

$$0^\circ (0 \text{ RAD}, 0 \text{ GRA}) \leq \text{ACS} \leq 180^\circ (\pi \text{ RAD}, 200 \text{ GRA})$$

$$-90^\circ \left(-\frac{\pi}{2} \text{ RAD}, -100 \text{ GRA}\right) \leq \text{ATN} \leq 90^\circ \left(\frac{\pi}{2} \text{ RAD}, 100 \text{ GRA}\right)$$

The ASN has the function of obtaining the arcsine ($\sin^{-1}x$) of a given numerical value, ACS of obtaining the arccosine ($\cos^{-1}x$) and ATN of obtaining the arctangent ($\tan^{-1}x$). (See page 264)

■ Angle Command

```
10 ANGLE 0
    ANGLE command | Specifies degree (DEG)
```

The following 3 angle units can be specified with the ANGLE command when performing trigonometric and inverse trigonometric calculations.

- ANGLE 0 DEGREE (DEG)
- ANGLE 1 RADIAN (RAD)
- ANGLE 2 GRADE (GRA)

```
10 REM DEGREE
20 ANGLE 0
30 A=SIN30
40 REM RADIAN
50 ANGLE 1
60 B=SIN30
70 REM GRADIENT
80 ANGLE 2
90 C=SIN30
100 PRINT "SIN30(DE
    G)=";A
110 PRINT "SIN30(RA
    D)=";B
120 PRINT "SIN30(GR
    A)=";C
```

Results will differ according to the specified angle unit even with the same SIN 30.

```
RUN ↵
↵
↵
```

SIN30(DEG)= 0.5
SIN30(RAD)=-0.9880316241
SIN30(GRA)= 0.4539904997

The ANGLE command has the function of specifying angle units (DEG, RAD, GRA). (See page 220)

■ Rearranging Display Formats with the USING Function

```

110 PRINT USING"####.###";Z;" OHM"
130 PRINT USING"####.###";S

```

USING function
Format

USING “####.###” is written after the PRINT command on lines 110 and 130. The USING function is used here to rearrange the format for easy reading if the value to be displayed such as Z or S has as many as 10 digits after the decimal point. The desired number of digits can be displayed with this function.

Following is a comparison when using and not using the USING function.

1) When USING is not used.

```

10 A=SQR15
20 B=SQR30
30 PRINT A;
40 PRINT B

```

RUN ↵

```
3.872983346 5.477225575
```

2) When USING is used.

```

10 A=SQR15
20 B=SQR30
30 PRINT USING"##.
##";A;
40 PRINT USING"##.
##";B

```

RUN ↵

```
3.87 5.48
```

USING thus rounds the displayed value to the specified digits.

The USING function can also be used to make the display easier to read as follows.

```

10 A=SQR15
20 B=SQR30
30 PRINT USING"##.
##";"A=";A;
40 PRINT USING"##.
##";"B=";B

```

→ The message, A =, is added.

RUN ↵

A= 3.87, B= 5.48

We still have the symbols # and . after USING.

.##

This type of character string is called a format character string and is used to display numerical values. In addition, the & symbol is used to display characters.

```

10 A=SQR30
20 PRINT USING"###"      → Displays numerical values
   .###";A
30 B$="ABCDEF6"
40 PRINT USING"&&&"      → Displays characters
   &&&";B$
    
```

The display will consist of 3 integers and 3 decimal places.

RUN ↵

5.477

Displays characters equal to the number of & symbols.

↵

ABCDE

When displaying multiple values with the same format, USING need not be specified for each value.

```

10 A=SQR15
20 B=SQR40
30 C=SQR90
40 PRINT USING"##.###";A;B;C
    
```

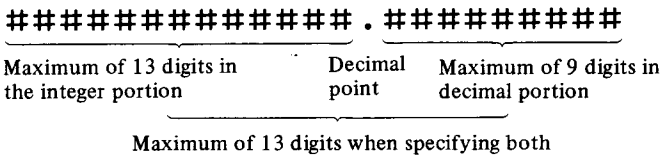
The values of A, B and C are displayed according to a single format character string.

RUN ↵

3.873 6.325 9.487

The following points should be noted when using the USING function.

1. An SN ERROR will occur if characters other than #, ., and & are written in the format character string.
2. # and . for numerical values and & for characters cannot be used in the same format character string. An SN ERROR will occur.
3. Format character string for numerical values.
 - # Specifies numerical value digits
 - Specifies decimal point
 - ^ Specifies exponents
- A maximum of 13 # can be specified for the integer portion and 9 for the decimal portions. A total of 13 (including a decimal point) can be specified when specifying for both integer and decimal portions.



- The minus sign uses 1 digit of #.
- ^ is written at the end of the format character string.
- If the digits in the decimal portion exceeds the length specified by the format character string, the digit following the specified digit will be rounded to the nearest whole number.

```
10 A=PI
20 PRINT A
30 PRINT USING"###.
###";A
```

RUN ↵

↵

3.141592654
3.142

- If the digits in the integer portion exceeds the length specified by the format character string, a % sign is placed at the beginning without allowing output by the format.

```

10 A=12345.6789
20 PRINT A
30 PRINT USING"###
   ##.##";A
40 PRINT USING"##.
   ###";A
    
```

RUN ↵

↵

↵

12345.6789
12345.68
% 12345.6789

4. Format character strings for characters.

& . . . Character position specification

- Any number of & symbols can be written.
- If the number of & symbols is less than the character string output, the characters will be displayed from the beginning by the number of & symbols.

```

10 A1$="ABCDEFGH1"
20 PRINT USING"###
   ###";A1$
    
```

Although 9 characters are assigned to A1\$, since there are only 6 & symbols in the format, only the first 6 characters will be displayed from the beginning.

RUN ↵

ABCDEF

- If the number of & symbols is greater than the character string output, the remaining symbols will be output as spaces.
 - Character strings will be displayed with left justification.
5. A USING is effective in only one PRINT or LPRINT command.
 6. The USING function is also used to change the format.
 7. A USING specification is canceled by USING " " ; .

The USING function is used together with the PRINT command (or LPRINT command) and has the function of displaying (printing) numerical values and character strings according to specified formats. (See page 261)

■ LET Command

```
80 LET Z=SQR(R^2+X^2)
```

```
90 LET S=ATN(X/R)
```

└─ LET command

Since assignment statements such as on line 80 are used frequently in programs, the LET command can be omitted.

```
10 LET A=10
20 LET B=25
30 PRINT A;B } (A)
```

(A) and (B) have the same meaning.

```
10 A=10
20 B=25
30 PRINT A;B } (B)
```

Note:

An assignment statement must be a variable on the left side of the = sign and a constant, variable or expression on the right. An SN ERROR will occur if there is a constant or an expression on the left. A TM ERROR will occur if the correspondence of the left and right sides is incorrect; e.g., characters are assigned to numerical variables.

The LET command placed at the beginning of an assignment statement shows that an assignment will be performed. (See page 235)

4-3 Analysis of Variance

Many data groups will be formed when market surveys or sales surveys are conducted (by area and by branch). Since there is very little chance that any two data will be the same, how many we determine the overall trend from these different figures?

This program is to make significance test by inputting data on these groups.

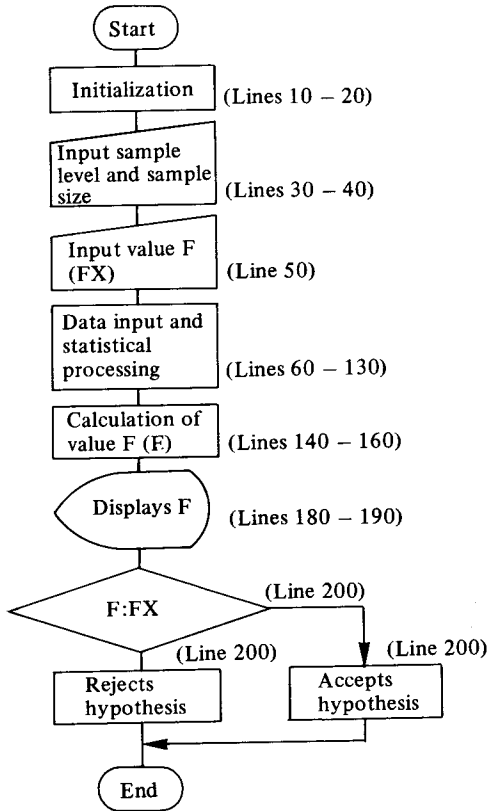
Commands and Functions

STAT CLEAR、STAT、SUMX(Y)、SUMX2(Y2)、
SDX(Y)、SDXN(YN)

Program List

```
10 CLEAR
20 STAT CLEAR
30 INPUT "LEVEL=":
  A
40 INPUT "SAMPLE S
  IZE=":N
50 INPUT "F=":FX
60 FOR I=1 TO A:B=
  0
70 FOR J=1 TO N
80 INPUT X
90 STAT X
100 B=B+X
110 NEXT J
120 C=C+B^2
130 NEXT I
140 D=SUMX^2/(A*N)
150 E=SUMX2-C/N
160 F=(C/N-D)*(A*N-
  A)/(A-1)/E
180 REM PRINT OUT
190 PRINT "FO=":F
200 IF F>FX THEN PR
  INT "REJECT!" E
  LSE PRINT "ACCE
  PT!"
```


Flow Chart



Input data of each group and check for any significance between the groups.

Input Data

Number of groups (level) \ Number of samples	1	2	3	4
1	4	6	5	6
2	5	3	4	5

$$F_{0.05} \left(\frac{3}{4} \right) = 6.59$$

Operation:

RUN ↵

LEVEL=?

Number of levels is requested. Since there are 4 levels here, enter 4.

4 ↵

SAMPLE SIZE=?

Number of samples is requested. Since there are 2 samples, enter 2.

2 ↵

F=?

Enter 6.59 for the value of $F_{0.05(3/4)}$ from the F distribution chart.

6.59 ↵

?

Input data of each level in sequential order.

4 ↵

?

5 ↵

?

6 ↵

?

⋮

5 ↵

F0 = 0.3333333

↵

ACCEPT!

The value of F will be calculated and displayed after inputting the final data. Since the value of FO is smaller than the value of $F_{0.05(3/4)}$, the data prove to belong to the same source (population).

■ Preparation for Statistical Calculations (STAT CLEAR Command)

20 STAT CLEAR

└─ STAT CLEAR command

Preparations are being made on line 20 to perform statistical calculations. The CLEAR command erases all variable contents on line 10, but the STAT CLEAR command erases only the memories relating to statistical calculations, i.e., clears the basic statistics (CNT, SUMX, SUMY, SUMXY, SUMX2, SUMY2).

The STAT CLEAR command has the function of initializing for calculating the basic statistics (CNT, SUMX, SUMY, SUMXY, SUMX2, SUMY2).
(See page 247)

■ STAT Command

90 STAT X

STAT command └─ Numerical expression

On line 90 data are input. Bothersome programs to obtain standard deviations, estimated values and correlation coefficients need not be prepared since the FX-750P has built-in statistical functions. Although only one-variable data, X have been entered at this time, paired data can also be entered.

```
10 STAT CLEAR
20 FOR I=1 TO 10
30 STAT I
40 NEXT I
50 PRINT "XBAR=";S
   UMX/CNT
```

→ Built-in STAT command facilitates the summation and mean calculation with regard to the numbers 1 to 10.

RUN ↵

XBAR= 5.5

```

10 STAT CLEAR
20 FOR I=1 TO 5
30 READ X,Y
40 STAT X,Y
50 NEXT I
60 PRINT "XBAR=";S
   UMX/CNT
70 PRINT "YBAR=";S
   UMY/CNT
80 DATA 2,4,3,6,5,
   11,9,20,13,27
    
```

Reading in 5 sets of data X, Y and obtaining the mean value of the respective X and Y values are easy with the STAT command.

RUN ↵

↵

XBAR= 6.4
YBAR= 13.6

* Mean values can be obtained by dividing sum of data by number of data (SUMX/CNT, SUMY/CNT).

The STAT command has the function of inputting statistical data.
(See page 246)

■ SUMX and SUMX2 Functions

140 $P = \frac{\text{SUMX}^2}{A * N}$ Basic statistic SUMX

150 $E = \frac{\text{SUMX2} - C}{N}$ Basic statistic SUMX2

Basic statistics SUMX and SUMX2 are used on lines 140 and 150 as means of calculating the F value.

SUMX obtains the sum of the one-variable data which are input by the STAT command and SUMX2 obtains the sum of the squares of those data.

$$\text{SUMX} = \sum_{i=1}^n x_i$$

$$\text{SUMX2} = \sum_{i=1}^n x_i^2$$

```
(1) 10 CLEAR
      20 FOR I=1 TO 10
      30 A=A+I:B=B+I^2
      40 NEXT I
      50 PRINT A;B
```

```
(2) 10 STAT CLEAR
      20 FOR I=1 TO 10
      30 STAT I
      40 NEXT I
      50 PRINT SUMX;SUMX
          2
```

Programs (1) and (2) are both obtaining the sum and the sum of the squares 1 to 10. Program (2) uses the statistical functions.

SUMY and SUMY2 are used in the case of paired data.

```
(1) 10 CLEAR
      20 FOR I=1 TO 5
      30 READ X,Y
      40 A=A+X:B=B+X^2
      50 C=C+Y:D=D+Y^2
      60 NEXT I
      70 PRINT A;B
      80 PRINT C;D
      90 END
    1000 DATA 3,5,4,9,2,
           1,6,4,6,9
```

```
(2) 10 STAT CLEAR
      20 FOR I=1 TO 5
      30 READ X,Y
      40 STAT X,Y
      50 NEXT I
      60 PRINT SUMX;SUMX
          2
      70 PRINT SUMY;SUMY
          2
      80 END
    1000 DATA 3,5,4,9,2,
           1,6,4,6,9
```

In both programs (1) and (2), data X and Y are read and the sum and the sum of the squares are calculated. Program (2) using the SUMX and SUMY functions is clearly easier to prepare.

SUMX and SUMX2 have the function of obtaining the sum and the sum of the squares of one-variable data entered with the STAT command respectively. (See page 273)

* SUMY and SUMY2 can be used in the case of paired data.

■ SDX(Y) and SDXN(YN) Functions

In addition to sum and mean values, standard deviations (sample standard deviations and population standard deviations) are frequently used in statistical data.

```

10 STAT CLEAR
20 READ X
30 IF X=-99 THEN 6
   0
40 STAT X
50 GOTO 20
60 PRINT "SUM=";SU      → Displays the sum.
   MX
70 PRINT "MEAN=";S     → Displays the mean value.
   UMX/CNT
80 PRINT "SD=";SDX     → Displays the sample standard deviation.
90 END
100 DATA 63,87,71,6
    5,63
110 DATA 67,73,63,8
    5,78
120 DATA -99
    
```

RUN ↵ (Sum)

↵ (Mean value)

↵ (Sample standard deviation)

SUM = 715
MEAN = 71.5
SD = 9.107384062

Bothersome standard deviation calculations can be performed easily in this manner. Also, when the data have been sampled out from a large group, the standard deviation of the group (population standard deviation) can be obtained immediately.

```

10 STAT CLEAR
20 READ X
30 IF X=-99 THEN 6
   0
40 STAT X
50 GOTO 20
60 PRINT "SUM=";SU      → Displays the sum.
   MX
70 PRINT "MEAN=";S     → Displays the mean value.
   UMX/CNT
80 PRINT "SD=";SDX     → Displays the sample standard deviation.
90 PRINT "SDN=";SD     → Displays the population standard deviation.
   XN
95 END
100 DATA 63,87,71,6
     5,63
110 DATA 67,73,63,8
     5,78
120 DATA -99

```

RUN (Sum)

SUM= 715

(Mean value)

MEAN= 71.5

(Sample standard deviation)

SD= 9.107384062

(Population standard deviation)

SDN= 8.640023148

SDY and SDYN can be used in the case of paired data.

```

10 STAT CLEAR
20 READ X,Y
30 IF X=-99 THEN 6
   0
40 STAT X,Y
50 GOTO 20
60 PRINT " SUM(X)=
   ";SUMX
70 PRINT "MEAN(X)=
   ";SUMX/CNT
80 PRINT " SD(X)=      → Displays sum, mean value, sample standard deviation
   ";SDX              and population standard deviation on data X.
90 PRINT " SDN(X)=
   ";SDXN

```

```

100 PRINT " SUM(Y)=
      ";SUMY
110 PRINT "MEAN(Y)=
      ";SUMY/CNT
120 PRINT " SD(Y)"
      ;SDY
130 PRINT " SDN(Y)=
      ";SDYN
140 END
200 DATA 929,50,925
      ,59
210 DATA 854,51,818
      ,48
220 DATA 931,51,849
      ,52
230 DATA -99,-99
    
```

→ Displays sum, mean value, sample standard deviation and population standard deviation on data Y.

RUN ↵ (Sum)

↵ (Mean value)

↵ (Sample standard deviation)

↵ (Population standard deviation)

↵ (Total)

↵ (Mean value)

↵ (Sample standard deviation)

↵ (Population standard deviation)

SUM(X) =	5306
MEAN(X) =	884.3333333
SD(X) =	49.79022663
SDN(X) =	45.45205045
SUM(Y) =	311
MEAN(Y) =	51.83333333
SD(Y) =	3.763863265
SDN(Y) =	3.435921356

The SDX and SDY functions are used to obtain the sample standard deviation of statistical data. SDXN and SDYN permit the calculation of the population standard deviation of the statistical data. (See page 275)

Input data x and y in sets and obtain the regression line conforming closest to the aggregate data. Obtain the estimated value of y for given data x and the estimated value of x for given data y .

Input data

x	258.2	370.8	442.3	501.6
y	190.2	240.7	303.1	412.8

Operation:

RUN ↵

DATA # 1, X=?

Enter data in sequence from the left. Enter data x first.

258.2 ↵

DATA # 1, Y=?

Enter data y .

190.2 ↵

DATA # 2, X=?

Enter the next data x and data y .

370.8 ↵

DATA # 2, Y=?

240.7 ↵

DATA # 3, X=?

⋮

⋮

Enter the last data y .

412.8 ↵

DATA # 5, X=?

Enter 0 and the linear regression equation will be displayed.

0 ↵

Y=-54.44575618+ 0.867558

⋮
Screen scrolls

44575618+ 0.8675586654*X

Next, obtain the estimated value.

↵

X=?

Enter 100 to x and then the estimated value of y will be displayed.

100 ↵

Y= 32.31011036

Next, enter 100 to y to obtain the estimated value of x.

↵

Y=?

100 ↵

X= 178.0234148

↵

X=?

⋮

Since this program is endless, press the **BRK** key to terminate the program.

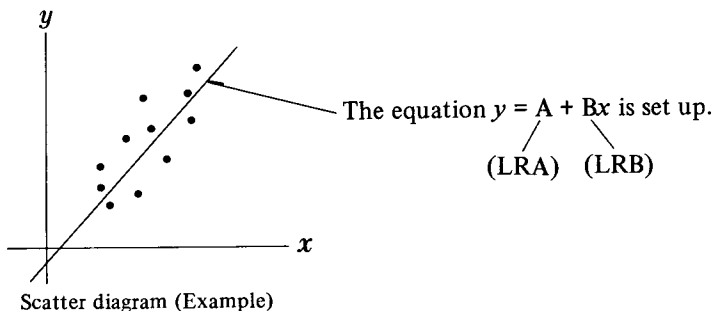
■ **LRA and LRB functions**

```
100 PRINT "Y=" ; LRA ; "+" ; LRB ; "*X" : GOTO 120
```

Obtains linear regression constant term.

Obtains linear regression coefficient

A linear regression equation closely fitting the input data will be displayed on line 100.



Constant **A** and coefficient **B** in the linear regression equation $y = A + Bx$ are obtained by the following expressions.

(Constant term of the regression expression)
$$A = \frac{\sum y - \text{LRB} \cdot \sum x}{n}$$

(Coefficient of the regression expression)
$$B = \frac{n \sum xy - \sum x \cdot \sum y}{n \sum x^2 - (\sum x)^2}$$

The LRA and LRB functions obtain the the straight line that conforms most closely to paired data entered by the STAT command. (See page 276)

■ CNT Function

```
20 PRINT "DATA #"; CNT+1; ", X=";
```

└ Statistical function CNT

Lines 20 shows the order of the x data currently being entered. CNT (n), which is a basic statistic, increases the number of the statistical data by one each time a data is entered.

This is a highly convenient function for processing a large number of data since it is not necessary to precount the number of data.

The CNT function stores the number of data which are statistically processed. (See page 273)

■ EOX and EOY Functions

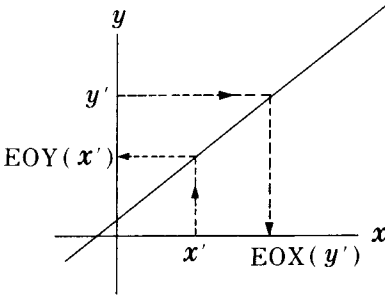
```

130 PRINT "Y=" ; EOY(X0)
  ⋮
150 PRINT "X=" ; EOX(Y0)
    
```

Obtains the value of y for the coordinate x in a linear equation.

Obtains the value of x for the coordinate y in a linear equation.

On line 130, the value of y is obtained for a given x in the linear equation $y = A + Bx$. On line 150, the value of x is obtained for a given y .



$$y = A + Bx$$

The values of EOX and EOY are obtained by the following equations:

$$EOX(y_n) = \frac{y_n - LRA}{LRB}$$

$$EOY(x_n) = LRA + x_n \cdot LRB$$

The EOX and EOY functions allow the calculation of the estimated value of y for a given x , and x for a given y based on a regression line obtained by LRA and LRB. (See pages 277, 278)

■ COR Function

Correlation coefficients (r) often become a problem when obtaining regression line and they may be expressed as follows:

$$r = \frac{n \cdot \sum x \cdot y - \sum x \cdot \sum y}{\sqrt{\{n \cdot \sum x^2 - (\sum x)^2\} \{n \cdot \sum y^2 - (\sum y)^2\}}}$$

Programming this will be extremely troublesome. With the FX-750P, the COR function is built-in, so correlation coefficient (r) can be calculated immediately, thus eliminating the trouble of preparing complex calculating formulas to obtain correlation coefficients.

```

10 STAT CLEAR
20 READ X,Y
30 IF X=0 THEN 60
40 STAT X,Y
50 GOTO 20
60 PRINT "COR=";COR
  R
70 END
80 DATA 10,21,15,2
  9,20,42
90 DATA 47,92,12,2
  2,0,0

```

} → Reads data and executes statistical processing.
 } → Displays correlation coefficient immediately.

RUN ↵

COR= 0.9983837729

The correlation coefficient of this data will therefore be about 0.998.

The COR function obtains the correlation coefficient of paired data entered by the STAT command. (See page 277)

4-5 Management of the Household Budget

When our weekly shopping at the supermarket is viewed from an overall standpoint, what type of goods do we purchase most frequently? If we did wasteful shopping this week, we must be a bit more conservative next week. If the type of item and the price are entered, this program will compute the total amount and the percentage of each item.

This program is handy as it can also be used when totalizing data by groups.

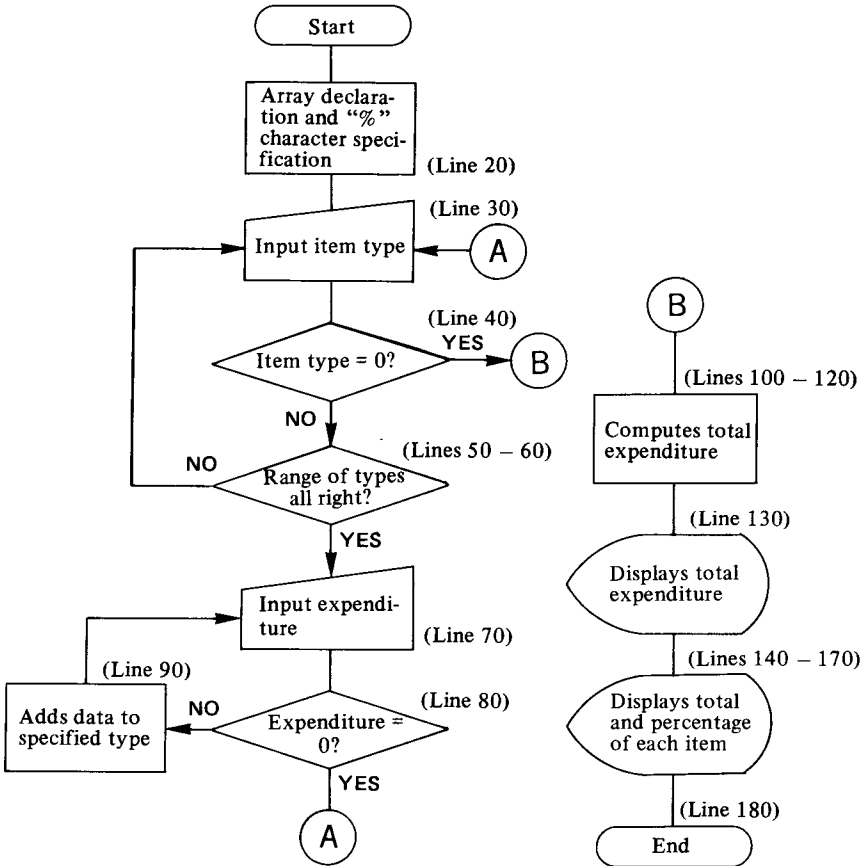
Commands and Functions

CLEAR、DIM、CHR\$、ROUND、FOR ~ NEXT ~ STEP/NEXT

Program List

```
5 CLEAR
10 N=5
20 DIM A(N):P$=CHR$(37)
30 INPUT "ITEM(0,1,2,3,4,5)=":X
40 IF X=0 THEN 100
50 IF X<0 THEN BEE
P : GOTO 30
60 IF X>N THEN BEE
P : GOTO 30
70 INPUT "EXPENSE? (END=0)",Y
80 IF Y=0 THEN 30
90 A(X)=A(X)+Y: 60
TO 70
100 FOR I=1 TO N
110 A(0)=A(0)+A(I)
120 NEXT I
130 PRINT "TOTAL=":
A(0)
140 FOR I=1 TO N
150 B=ROUND(A(I)/A(0)*100,-3)
160 PRINT "NO.":I:
="":A(I):"("):B:P$:""
170 NEXT I
180 END
```

Flow Chart



CHAPTER 4 APPLICATIONS

Classify the items purchased by type (itemize) and obtain the percentage by entering each amount. Although 5 items are used here, this program may be used for 10 items by modifying N = 5 to N = 10 on line 10.

Input Data

Item	Grocery (1)	Sundry (2)	Stationery (3)	Clothing (4)	Others (5)
Amount	\$15	35	21	250	110
(Data)	5	18	16	44	85
			2	6	

Operation:

RUN ↵

ITEM(0, 1, 2, 3, 4, 5) = ?

Select the type of item from among item 1 to item 5.

1 ↵

EXPENSE?(END=0)

Item 1 is groceries. Today's purchases were \$15.00 and \$5.00.

15 ↵

EXPENSE?(END=0)

5 ↵

EXPENSE?(END=0)

Since the purchases of item 1 is completed, enter 0.

0 ↵

ITEM(0, 1, 2, 3, 4, 5) = ?

Specify item 2.

2 ↵

⋮

Continue to enter the data for each item in this manner. Enter 0 after completing entry of the last data.

0 ↵

↵

↵

↵

↵

↵

TOTAL=607
NO.1= 20(3.29%)
NO.2= 53(8.73%)
NO.3= 39(6.43%)
NO.4= 300(49.42%)
NO.5= 195(32.13%)

The percentage and the total amount of each item will be displayed as shown.

■ Clear All Data (CLEAR Command)

5 CLEAR

└── CLEAR command

All of the contents stored in variables are erased on line 5. Since all the stored data are erased with the CLEAR command, do not use this command when executing a program based on stored data.

The following program obtains the total amount of input data.

```
10 CLEAR
20 INPUT "DATA=";X
30 IF X=0 THEN 50
40 Y=Y+X: GOTO 20
50 PRINT "TOTAL=";
Y
```

Executes this program at first.

Enter 0 to obtain the total amount.

RUN ↵

10 ↵

20 ↵

0 ↵

DATA=?
DATA=?
DATA=?
TOTAL=30

Next, delete line 10.

```
20 INPUT "DATA=";X
30 IF X=0 THEN 50
40 Y=Y+X: GOTO 20
50 PRINT "TOTAL=";
  Y
```

Correct answers cannot be obtained in calculations performed if the CLEAR statement on line 10 is deleted. This is because the numerical value of 30 has already been assigned to Y in the previous calculation.

RUN ↵

10 ↵

20 ↵

0 ↵

DATA=?
DATA=?
DATA=?
TOTAL= 60

If the CLEAR command is executed during the FOR ~ NEXT loop, an FO ERROR will be displayed since the FOR loop stack (place to return) will also be cleared and the FOR ~ NEXT loop cannot be maintained.

```
10 FOR I=1 TO 10
20 PRINT I
30 CLEAR
40 NEXT I
```

RUN ↵

↵

1
FO ERROR P0-40

The CLEAR command has the function of erasing all stored values in the variables. (See page 223)

■ What is an Array? (DIM)

```

20 DIM A(N) : P$=CHR$(37)
  DIM command  |  Subscript
                  |
                  |  Array variable
  
```

The DIM command declares array variables such as on line 20 and is capable of declaring multiple variables at one time.

Example: DIM A(5), B(7), C(30)

1) Subscript

On line 20, DIM declares use of the one-dimensional array called A. The maximum value of the subscript will be N (In this case it will be 5 since 5 is assigned to N on line 10 (page 134)). It means that the 6 boxes of A(0), A(1), A(2), A(3), A(4) and A(5) have been prepared on line 20 since subscripts start with 0 in BASIC. Variables can also be used as subscripts of array variables in addition to numerical values.

However, if there are any fractions in the variable, these fractions will be automatically discarded.

Example:

If the data stored in subscript 3 of array variable A is assigned to X, it may be expressed as follows:

```
X=A(3) , I=3:X=A(I) and J=3.5:X=A(J)
```

These three types of programming results in the same memory assignment.

A UV ERROR will occur if the specified subscript exceeds the subscript range of the array variable declared by the DIM command.

```

10 DIM A(5)
20 A(0)=0
30 A(1)=10
40 A(2)=20

```

—————> Subscript range by the DIM command is $0 \leq \text{subscript} \leq 5$.

```
50 A(3)=30
60 A(4)=40
70 A(5)=50
80 A(6)=60
```

→ The specified subscript range on line 10 is exceeded here as 6 is specified.

RUN ↵

UV ERROR P0-80

Furthermore, the maximum value of the subscript that can be declared by the DIM command is 255. Subscripts exceeding this value ($0 \leq \text{subscript} < 256$) cannot be used.

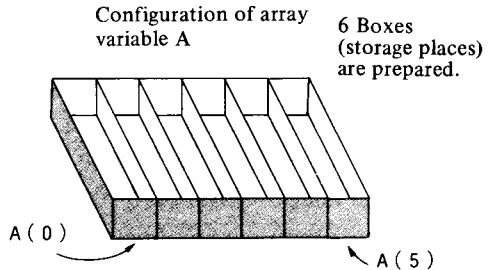
2) Dimension

Up to 2 subscripts can be specified for an array variable by punctuating with a comma (.). This is called a 2-dimensional array.

Example:

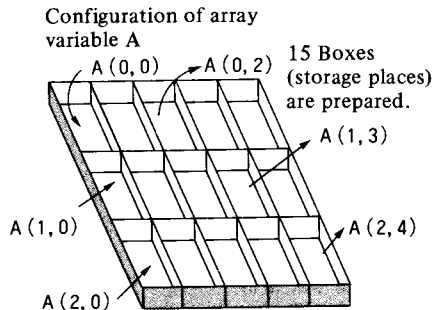
1-dimensional array

```
DIM A(5)
```



2-dimensional array

```
DIM A(2,4)
```



3) Array Variable Names

Capital alphabetical letters from A to Z may be used for the variable names. The symbol \$ or ! can be added after A ~ Z when specifying character arrays or half-precision arrays.

Single-precision array

A(i) ~ Z(i) Precision: 12-digit mantissa and 2-digit exponent.
 A(i, j) ~ Z(i, j) Memory: 8 bytes each

Half-precision array

A!(i) ~ Z!(i) Precision: 5-digit mantissa and 2-digit exponent.
 A!(i, j) ~ Z!(i, j) Characters beyond 6 digit-mantissa will be discarded when executing assignment statements.
 Memory: 4 bytes each. Storage capacity can be saved if 5 digits or less will be sufficient.

Character array

A\$(i) ~ Z\$(i) Capacity: Can store up to 16 characters each.
 A\$(i, j) ~ Z\$(i, j) The number of characters that can be specified for storage is within the range of $0 \leq \text{number of characters} < 80$.
 Memory: (Number of stored characters + 1) byte(s)

Example:

When desiring to store a maximum of 20 characters:

```
DIM A$(5)*20
```

A(i) and A(i, j), A!(i) and A!(i, j), and A\$(i) and A\$(i, j) cannot be used together. Also, A(j) cannot be declared again once A(i) is declared. DD ERROR will be displayed (duplicate declaration error) in either case.

```
10 DIM A(10)           —————> Array A(i) is declared.
20 A(0)=12:A(1)=35
   :A(2)=27
30 DIM A(3,4)         —————> A(i, j) of the same array name is declared.
40 A(0,1)=75:A(1,1)
   :)=19:A(2,1)=34
```

```
RUN
```

```
DD ERROR P0-30
```

```

10 DIM A(10)      —————> Array A(i) is declared.
20 FOR I=0 TO 10
30 A(I)=I*I
40 NEXT I
50 DIM A(20)      —————> Array A(i) is declared again as a slightly larger
60 FOR I=0 TO 20  array is desired.
70 A(I)=I*I
80 NEXT I

```

RUN ↵

DD ERROR P0-50

The same array cannot be defined again. If it must be redefined, first use the CLEAR or ERASE command. Be careful when using these commands since all the values stored in variables will be cleared by the CLEAR command and data stored in specified variables will be erased by the ERASE command.

CLEAR — Clears all stored data.

ERASE A — Erases all stored data in array variable A. (See page 227)

```

10 DIM A(3),B(5) —————> Declares array A and B.
20 FOR I=0 TO 2
30 A(I)=I+100:B(I)
   =I+200
40 NEXT I
50 FOR I=0 TO 2
60 PRINT A(I);B(I)
70 NEXT I
80 ERASE A      —————> Erases array A, as a larger array A is desired.
90 DIM A(5)     —————> Declares array A again.
100 FOR I=1 TO 2
110 A(I)=I+300
120 NEXT I
130 FOR I=0 TO 2
140 PRINT A(I);B(I)
150 NEXT I

```

The following values are stored in arrays A and B.

RUN ↵

↵

↵

A	B
100	200
101	201
102	202

Array A(0) has no data due to erasure by the ERASE command.



0	200
---	-----

Numerical values entered into array A(1) and A(2) are displayed.



301	201
-----	-----



302	202
-----	-----

The DIM command has the function of declaring the dimension and the maximum subscripts of an array variable. This declaration is always required before using the array variable. (See page 225)

■ CHR\$ Function

```
20 DIM A(N):P$=CHR$( 37 )
      CHR$ function   | Character code
```

On line 20, the 37th character in the character code is being assigned to P\$ to display the % symbol.

- 10 PRINT CHR\$(37) → %
- 20 PRINT CHR\$(39) → Single quotation mark
- 30 PRINT CHR\$(64) → Unit price symbol
- 40 PRINT CHR\$(91) → Left bracket
- 50 PRINT CHR\$(93) → Right bracket
- 60 PRINT CHR\$(95) → Cursor (underline)
- 70 PRINT CHR\$(135) → Cursor (Square)

RUN ↓



%



'



[



[



]
-
■

Characters that are not on the keyboard can be displayed in this manner by using the CHR\$ function as long as they are defined in the Character Code Table. (See page 325)

The numerical values that can be used with the CHR\$ function are within the range of $0 \leq \text{numerical value} < 256$.

The CHR\$ function converts a given numerical value to its corresponding character. (See page 251)

* These corresponding numerical values and characters are listed in the internationally used ASCII (American Standard Code of Information Interchange) code.

■ ROUND Function

$$150 \quad B = \text{ROUND} \left(\frac{A(1)}{A(0)} * 100, -3 \right)$$

ROUND function
Number to be rounded
Digit position

On line 150, the total of each class is divided by the grand total and then multiplied by 100. The percentage in relation to the total for each item is then displayed.

A numerical value is rounded off at the third decimal place.

Since there are a number of other methods of using the ROUND function, some of these will be explained with simple programs.

```

10 A=1234567
20 B=4.56789
30 C=ROUND(A,2)
40 D=ROUND(B,-2)
50 PRINT C
60 PRINT D
    
```


The numerical value 1234567 is assigned to A and 4.56789 is assigned to B. The numerical value of A is rounded at the 10^2 position and the result is assigned to C. The numerical value of B is rounded at the 10^{-2} position (1/100 position) and the result is assigned to D.

The display will therefore be numerical values rounded at the 100 and 1/100 positions respectively.

RUN ↵

↵

1235000
4.6

When ROUND (x , y) is specified in this manner, the calculation results of the given numerical value or numerical expression will be rounded at the 10^y position.

```
10 A=123456789
20 B=ROUND(A,3,0)
30 C=ROUND(A,3,1)
40 PRINT B
50 PRINT C
```

The numerical value 123456789 is assigned to A. The numerical value of A will be rounded at the 10^3 position and the result is assigned to B. The 3rd significant digit of the numerical value of A will also be rounded and the result is assigned to C. B is displayed with the numerical value rounded at the 10^3 position and C is displayed with the effective digits rounded to 2 digits.

RUN ↵

↵

123460000
120000000

When ROUND (x , y , 0 or 1) is performed in this manner, x will be rounded at the 10^y position of x in the case of 0 and the y th significant digits from the left will be displayed for x in the case of 1.

```
(1) 10 A=123456789
    20 B=ROUND(A,3,0,+
      )
    30 C=ROUND(A,3,0,-
      )
    40 D=ROUND(A,3,0)
    50 PRINT A
    60 PRINT B
    70 PRINT C
    80 PRINT D
```

```
(2) 10 A=123456789
    20 B=ROUND(A,3,1,+ ..... Rounded up at the
      ) specified digit if the
    30 C=ROUND(A,3,1,- ..... sign is +.
      ) Discarded at the speci-
    40 D=ROUND(A,3,1) fied digit if the sign is
    50 PRINT A
    60 PRINT B
    70 PRINT C
    80 PRINT D
```

(1)

The numerical value 123456789 is assigned to A.

Round up the numerical value of A at the 10^3 position and assign the result to B.

Also, discard the numerical value of A at the 10^3 position and assign the result to C.

Then, round the numerical value of A at the 10^3 position and assign the result to D.

```
RUN ↵
    ↵
    ↵
    ↵
```

123456789
123460000
123450000
123460000

(2)

Processing in this case will be at the yth-digit significant digit, whereas processing will be at the 10^y position in the case of (1).

```
RUN ↵
    ↵
    ↵
    ↵
```

123456789
130000000
120000000
120000000

Note:

When specifying digits on which to perform rounding, the numerical value must be |digit specifying value| < 100. A BS ERROR will be displayed if a value over 100 is specified.

The ROUND function rounds a numerical value at a specified digit.
(See page 270)

■ **Repeating the Operation (FOR ~ NEXT Command)**

```
100 FOR I=1 TO N
110 A(0)=A(0)+A(I)
120 NEXT I
```

The processing $A(0) = A(0) + A(I)$ is being executed repeatedly (N times) until the value of I exceeds N on line 100 ~ 120.

In the same manner, the statements on lines 150 and 160 are being repeatedly executed on lines 140 to 170 until the value of I advances from 1 to N (N times).

```
140 FOR I=1 TO N
150 B=ROUND(A(I)/A(0)*100,-3)
160 PRINT "NO";I;"=";A(I);"(";B;P$;"
170 NEXT I
```

This method of repeating execution a fixed number of times is often used in the process of creating a program.

The FOR ~ NEXT command generally has the following format.

FOR control variable = initial value TO final value STEP increment
NEXT control variable

The value of the control variable is first set to its initial value and the increment specified with STEP is then added to the control variable.

The command between FOR and NEXT will then be executed if the value of control variable is below the final value. (This is called a loop.)

Execution will jump to the next statement of NEXT when the value of the control variable exceeds the final value. (This means the loop is completed.) If STEP is omitted, increment will automatically be set to 1.

```
10 FOR I=1 TO 10 STEP 2
20 PRINT "*";
30 NEXT I
```

If the increment is set as 2, the value of I changes in steps of 2 such as 1, 3, 5, 7, 9 and the loop will be completed when it reaches 11 as this exceeds the final value 10. Five * symbols will be displayed at this time.

RUN ↵

```
*****
```

```
10 FOR I=1 TO 10
20 PRINT "*";
30 NEXT I
```

If increment is omitted, the value of I advances in single steps such as 1, 2, 3, 4, 5 ... and the loop will be completed when it reaches 11 as this is greater than the final value 10. So ten * symbols will be displayed at this time.

RUN ↵

```
*****
```

The initial value, final value and increment may be negative numbers or fractions but care is required as to size.

This is, the loop is executed only once when the increment is positive even if the final value is smaller than the initial value or negative even if the final value is larger than the initial value.

```

10 FOR I=1 TO 5
  20 FOR J=1 TO 3
  30 PRINT I*J;
  40 NEXT J
  50 PRINT ""
  60 NEXT I

```

FOR ~ TO ~ STEP ~ NEXT commands are often used in nested form.

The specification of multiple FOR ~ NEXT loops is called nesting.

Operations at this time will be as follows:

When the value of I is 1, J = 1, 2, 3 I*J = 1, 2, 3
 When the value of I is 2, J = 1, 2, 3 I*J = 2, 4, 6
 When the value of I is 3, J = 1, 2, 3 I*J = 3, 6, 9
 When the value of I is 4, J = 1, 2, 3 I*J = 4, 8, 12
 When the value of I is 5, J = 1, 2, 3 I*J = 5, 10, 15

```

10 FOR I=1 TO 5
  20 FOR J=1 TO 3
  30 PRINT "I=";I;";",
    J="";J
  40 NEXT I
  50 NEXT J

```

Care should be taken in relation to the positions of the FOR and NEXT commands. An FO ERROR will be displayed if incorrect.

RUN ↵

⋮

↵

↵

I = 1, J = 1

⋮

I = 5, J = 1

FO ERROR P0-50

The FOR ~ NEXT command can be nested up to 6 levels. A NO ERROR will be displayed if more than 6 levels of nesting is specified.

The FOR ~ TO ~ STEP ~ NEXT command has the function of repeating execution of the program between the FOR and NEXT commands a specified number of times. (See page 228)

4-6 Weekly Schedule

We often hear of cases where a person taking a leisurely noon recess is startled by a phone call from a client inquiring as to the time of the appointment for that day, since the appointment had completely slipped that person's mind. In all probability, you have also experienced this once or twice.

This program is to assist those who are looking for a quick method of setting up a schedule even if for the current week.

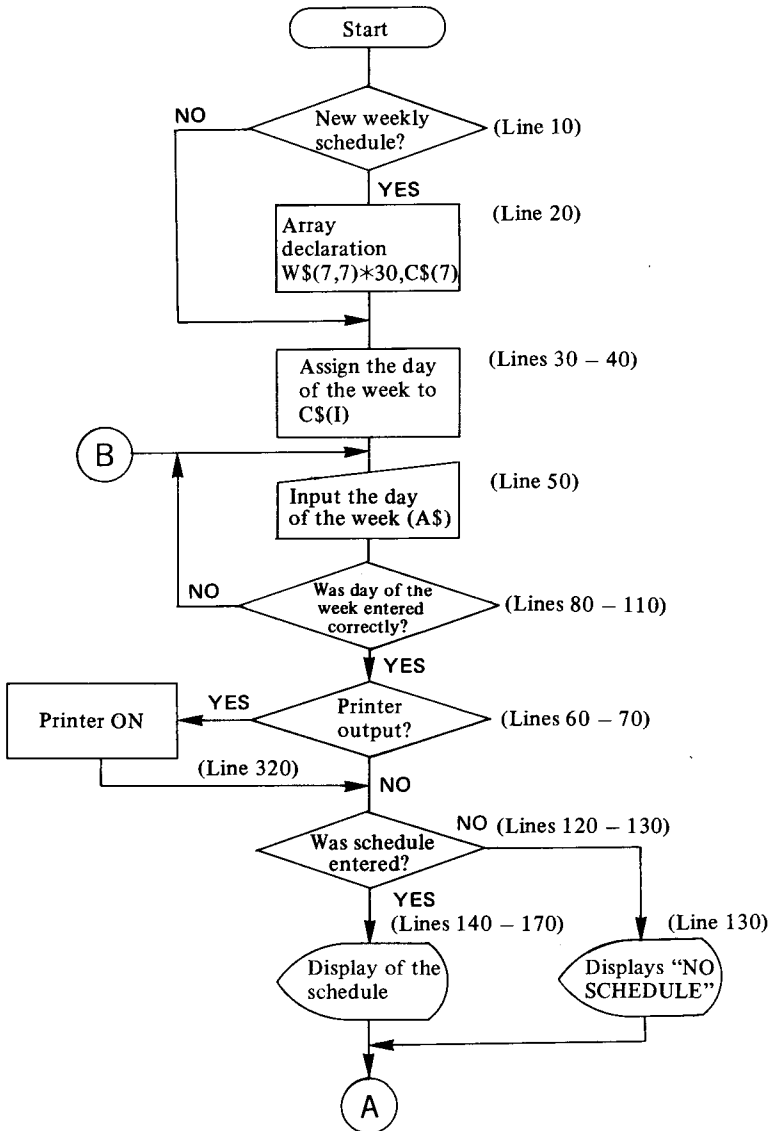
Commands and Functions

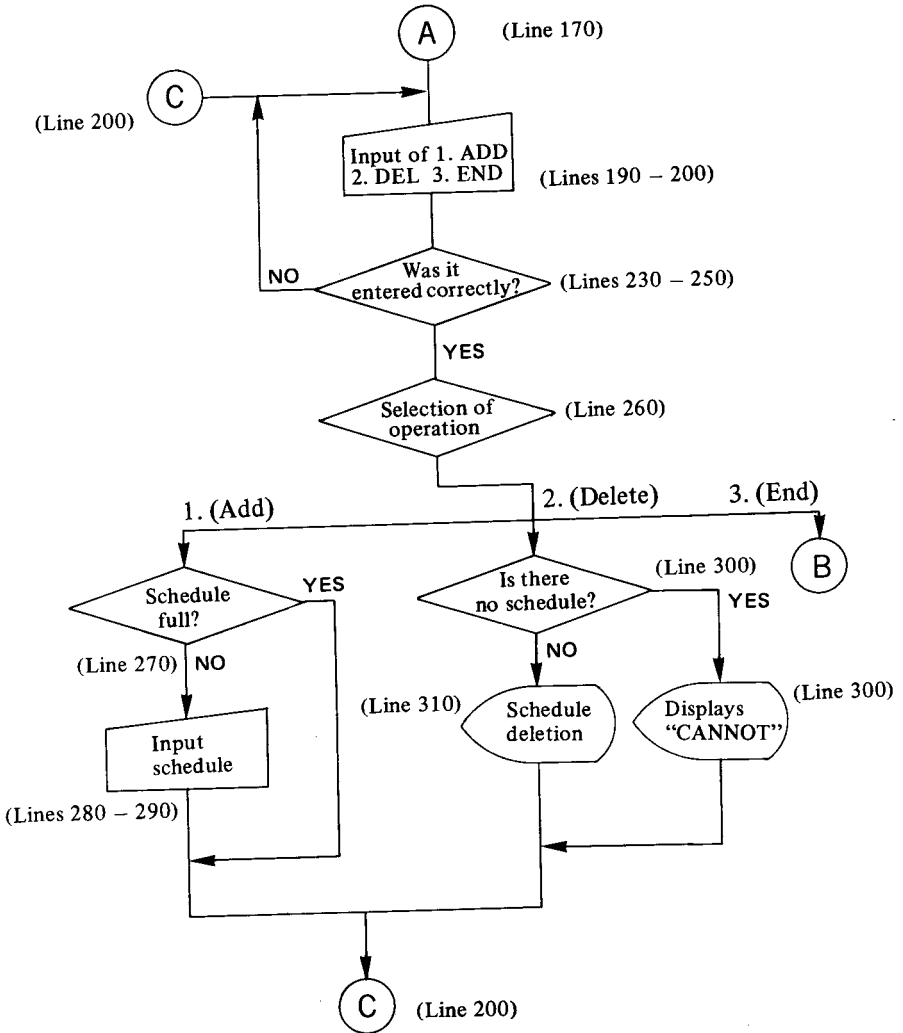
READ、DATA、RESTORE、VAL、STR\$、INKEY\$、ASC

Program List

```
10 INPUT "FIRST (Y
/N)";X$:IF X$="
N" THEN 30
20 CLEAR :DIM W$(7
,7)*30,C$(7)
30 FOR I=1 TO 7:RE
AD C$(I):NEXT I
40 DATA SUN,MON,TU
E,WED,THU,FRI,S
AT
50 INPUT "DAY=":A$
60 INPUT "PRINTER
(Y/N)";N$
70 IF N$="Y" THEN
GOSUB 320
80 X=0:FOR I=1 TO
7
90 IF A$=C$(I) THE
N X=I
100 NEXT I
110 IF X=0 THEN BEE
P : GOTO 50
120 MA=VAL(W$(X,0))
130 IF MA=0 THEN PR
INT "NO SCHEDUL
E": GOTO 180
140 FOR I=1 TO MA
150 PRINT W$(X,I);
160 A$=INKEY$:IF A$
="" THEN 160
170 PRINT :NEXT I
180 PRINT OFF
190 PRINT "1..ADD 2
..DEL 3..END";
200 A$=INKEY$:IF A$
="" THEN 200
210 Y=ASC(A$)
220 Y=Y-48
230 IF Y<0 THEN 200
240 IF Y>3 THEN 200
250 PRINT
260 ON Y GOTO 270,3
00,50
270 IF MA=7 THEN BE
EP :PRINT "NO S
PACE!": GOTO 19
0
280 MA=MA+1:INPUT W
$(X,MA)
290 W$(X,0)=STR$(MA
): GOTO 190
300 IF MA=0 THEN BE
EP :PRINT "CANN
OT!": GOTO 190
310 MA=MA-1:W$(X,0)
=STR$(MA): GOTO
190
320 PRINT ON :PRINT
A$
330 RETURN
```

Flow Chart





Prepare a week's schedule by inputting daily data. Check that the printer (FA-20) is correctly connected before printing the data.

- Use 3 alphabetical characters to enter the days of the week.
SUN, MON, TUE, WED, THU, FRI, SAT
- You can use up to 30 characters per schedule and can enter 7 schedules per day.

Input Data

Monday	10 a.m.	Meeting with Mr. Jones
	2 p.m.	Conference
	4 p.m.	Go to Company A
Tuesday	1 p.m.	Conference
	3 p.m.	Meeting with Mr. Smith

Operation

RUN 

FIRST (Y/N)?

The display is asking if data is for a new week. Press the key to enter the first data for the week. Press the key to check or modify (add, delete) a previously entered schedule. Be careful in using the key. All the previously entered data will be erased when this key is pressed.

Since a new week's schedule is to be entered here, press the following keys.

Y 

DAY?

Enter the day of the week.

MON 

PRINTER (Y/N)?

This display is asking if the input schedule is to be printed out. Since nothing has been entered yet, press the **N** key.

N

NO SCHEDULE

The display shows there is no schedule for Monday.

1..ADD 2..DEL 3..END

- 1 ADD.....Inputs data
- 2 DEL.....Deletes data
- 3 END.....Terminates operation

Select one of the following processes and press the corresponding key.

Since data is to be entered here, select 1.

1

?

Enter the schedule for Monday within 30 characters per item in sequential order.

10AM JONES

1

1..ADD 2..DEL 3..END

?

2PM CONFERENCE

1

1..ADD 2..DEL 3..END

?

4PM A CO.

1..ADD 2..DEL 3..END

When the schedule for Monday has been entered, press 3.

3

DAY?

Next, enter the schedule for Tuesday.

TUE

PRINTER (Y/N)?

N

NO SCHEDULE

1..ADD 2..DEL 3..END

1

?

1PM CONFERENCE

1..ADD 2..DEL 3..END

1

?

3PM SMITH

1..ADD 2..DEL 3..END

Stop data entry for a while and print out the data that has been entered so far. The data will be displayed and printed each time the key is pressed.

3

DAY?

MON

PRINTER (Y/N)?

Although the key is normally pressed here, press the key if a printer is not used. Display and key operation are the same.

Y

10AM JONES

2PM CONFERENCE

4PM A CO.

Printing will be done as shown below.

```
MON
10AM JONES
2PM CONFERENCE
4PM A CO.
```

(Next operation)

↵

1..ADD 2..DEL 3..END

3

DAY?

The days of the week and data are entered in this manner. However, since this program is endless, press (BRK) to terminate the program.

(BRK)

READY P0

Next, we'll call the stored data and modify the schedule.

RUN ↵

FIRST (Y/N)?

N ↵

DAY?

Modify the schedule for Tuesday.

TUE ↵

PRINTER (Y/N)?

We will not print out at this time.

N ↵

1PM CONFERENCE

↵

3PM SMITH

↵

1..ADD 2..DEL 3..END

Modify after all of the schedule for Tuesday is displayed.

Change the meeting with Mr. Smith to 4 p.m. To accomplish this, first delete the current data and re-enter the new data.

2

1..ADD 2..DEL 3..END

The display is the same as before except that 3 PM SMITH has been erased. Input data will be erased one at a time from the end each time the **[2]** key is pressed. You cannot specify any particular data to be deleted. Enter the new data with the following operation.

1	?
4 PM [SPC] SMITH [↵]	1..ADD 2..DEL 3..END
3	DAY?

Press **[BRK]** when you wish to terminate the program.

■ Entering Data in a Program. (READ Command)

```
30 FOR I=1 TO 7:READ C$(I):NEXT I
    |
    | Reads data specified by the data statement into
    | array variable C$(I). (READ Command)
40 DATA SUN,MON,TUE,WED,THU,FRI,SAT
    |
    | Data statement           Data
```

SUN, MON . . . SAT is read into array variable C\$(I) on line 30. A DATA statement will always be necessary as shown on line 40 when executing the READ command. One data in the data statement will be read in each time the READ command is executed.

```
10 FOR I=1 TO 4
20 READ A$
30 PRINT A$
40 NEXT I
50 DATA MOUNT,SKY,
   RIVER,SEA
```

→ The READ command will be executed 4 times by the FOR-NEXT statement.

RUN **[↵]**

[↵]

[↵]

[↵]

MOUNT
SKY
RIVER
SEA

Although the variable remains the same A\$, data will continue to change. (The data in the DATA statement on line 50 will be read in sequential order.)

The READ command is also capable of reading data into multiple variables at one time.

```

10 READ A$,B$,C$ → Reads data into character variables A$, B$ and C$.
20 PRINT A$
30 PRINT B$      } → Displays the data read in.
40 PRINT C$
50 DATA SUNDAY,MON
   DAY,TUESDAY
    
```

RUN ↵

↵

↵

SUNDAY
MONDAY
TUESDAY

If the number of the data in the DATA statement exceeds that of the variables read in with the READ command, the excess data will be disregarded.

```

10 READ A$,B$
20 PRINT A$
30 PRINT B$
40 DATA RED,BLUE,G
   REEN,YELLOW
    
```

RUN ↵

↵

↵

RED
BLUE
READY P0

If READ command is executed, remaining data will be read in.

To display GREEN and YELLOW as well, modify the program as follows.

```

10 READ A$,B$
20 PRINT A$
30 PRINT B$
40 READ C$,D$ → Adds a new READ command.
50 PRINT C$
60 PRINT D$
70 DATA RED,BLUE,G
  REEN,YELLOW

```

RUN ↵

↵

↵

↵

RED
BLUE
GREEN
YELLOW

A DA ERROR will occur if the number of the data specified by the data statement is smaller than that of the variables entered with the READ command.

```

10 READ A$ }
20 READ B$ } → The READ command is executed 4 times.
30 READ C$ }
40 READ D$ }
50 DATA BLACK,WHIT → There are only 3 data available.
  E,GRAY

```

A DA ERROR will be displayed if read-in of a 4th data is attempted when executing this program.

RUN ↵

DA ERROR P0-40

Use the RESTORE command when wishing to read data on a particular line from among the many DATA statements.

X can therefore be used for calculation.

RUN ↵

12345 24690

All characters other than numerical characters will be processed as 0 with the VAL function. However, hexadecimal notations (&HOO) are exceptions (example: VAL ("&HF") 15). If a numerical character is at the beginning of a mixed string of numerical and non-numerical characters, these characters will be converted to numerical values. If a non-numerical character is at the beginning, the characters will be processed as 0.

```
10 A$="ABCDEF6" → Alphabetical character string.
20 X=VAL(A$)
30 PRINT X
```

Alphabetical character string is converted to the numerical value of 0.

RUN ↵

0

```
10 A$="246XYZ" → The first 3 characters assigned to A$ are numerical
20 B$="XYZ789" → characters.
30 C$="45LMN69" → The last 3 of the characters assigned to B$ are
40 X=VAL(A$) → numerical characters.
50 Y=VAL(B$) → Numerical characters are at the beginning and end
60 Z=VAL(C$) → of the C$.
70 PRINT "X=";X
80 PRINT "Y=";Y
90 PRINT "Z=";Z
```

In character strings beginning with a numerical character, the first numerical characters are converted to numerical values.

RUN ↵

↵

↵

X= 246

Y= 0

Z= 45

This is a convenient function to use.

```

10 INPUT X$
20 Y=VAL(X$)
30 IF Y<1 THEN 10
40 IF Y>10 THEN 10
50 PRINT Y

```

The above is a program that can only accept data within the range of $1 \leq \text{data} \leq 10$.

In an other word, if a non-numerical data is entered into X\$, Y will be 0 by the VAL function. Also, since numerical data entered will be converted to numerical values by the VAL function, Y can be judged by this value when numerical characters are entered.

The VAL function converts a given character string to a numerical value.
(See page 252)

■ STR\$ Function

```
290 W$(X,0)=STR$(MA):GOTO160
```

STR\$ function
└ Converts numerical values stored in MA to characters.

The numerical values stored in variable MA are being converted to character strings and are being stored in character array W\$ on line 290. The STR\$ function converts numerical values to character strings in this manner while the VAL function converts character strings to numerical values.

One space will be attached at the beginning of a character string converted from a numerical value with the STR\$ function if the numerical value is a positive number or a 0. A (-) symbol will be attached at the beginning in the case of a negative number.

Care will also be required on the type of character variables to be stored in the character string when handling decimal fractions or large numbers.

A maximum of 7 characters can be used for A\$ ~ Z\$. (The symbol will occupy one character space at this time.)

A maximum of 16 characters can be used for A1\$ ~ Z9\$. (The symbol will occupy one character space at this time).

```

10 A=37.169
20 B=-34.885
30 X$=STR$(A)
40 Y$=STR$(B)
50 PRINT "DATA=";X
   $;"(X$)"
60 PRINT "DATA=";Y
   $;"(Y$)"

```

One space will be attached at the beginning if the number is positive. A (-) symbol will be attached at the beginning if the number is negative.

RUN ↵

↵

```
DATA= 37.169(X$)
```

```
DATA=-34.885(Y$)
```

```

10 A=548.3271
20 X$=STR$(A)
30 PRINT "DATA=";X
   $;"(X$)"

```

→ X\$ is basically "548.3271". But only 7 characters can be stored in X\$ when executing the STR\$ function.

An ST ERROR therefore occurs on line 20.

RUN ↵

```
ST ERROR P0-20
```

Modifying X\$ to X1\$ in this program permits storage of up to 16 characters.

```

10 A=548.3271
20 X1$=STR$(A)
30 PRINT "DATA=";X
   1$;"(X1$)"

```

RUN ↵

```
DATA= 548.3271(X1$)
```

```

10 A=-3.1415926535
   *10^-99
20 X1$=STR$(A)
30 PRINT "DATA=";X
   1$;"(X1$)"

```

The following is an example of using all 16 characters of X1\$.

RUN 

DATA=-3.141592654E-99(X1

Screen scrolls

TA=-3.141592654E-99(X1\$)

The STR\$ function converts given numerical values to character strings.
(See page 253)



■ INKEY\$ Function

```
200 A$=INKEY$: IF A$="" THEN 200
```




INKEY\$ function

Character is assigned from the keyboard.

One character is assigned from the keyboard into A\$ and if the result is null (" "), the command returns to line 200. That is, there will be no execution until a key on the keyboard is pressed.

This method is used for key operation in games or when wishing to omit the  or  key after a key entry.

The difference between INKEY\$ and INPUT is shown below.

Use	INKEY\$	INPUT
Display during command execution	Nothing displayed	? (Input request message. It is possible not to display "?".)
Data entry	Key pressed when executing command. (No entry without pressing key.)	Data entered until  is pressed
Kinds of data	1 character as a character	Number of characters and digits within the range of a variable such as a character or numerical value.
Execution of next command	Immediate execution ( Unnecessary)	Suspend until  is pressed.

```

10 PRINT "CONTINUE
(Y/N)?";
20 A$=INKEY$:IF A$
   =" " THEN 20 ELS
   E BEEP 0
30 IF A$="Y" THEN
   60
40 IF A$="N" THEN
   80
50 GOTO 20
60 PRINT :PRINT "C
ASIO"
70 GOTO 10
80 PRINT :PRINT "E
ND"

```

RUN ↵

CONTINUE(Y/N)?

The next process will be performed when the **Y** or **N** key is pressed.

Y

CASIO

↵

CONTINUE(Y/N)?

CASIO will be displayed if the **Y** key is pressed again and display of CONTINUE(Y/N) ? will be repeated when ↵ is pressed. Pressing the **N** key terminates the program.

N

END

↵

READY P0

```

10 PRINT "PRESS AN
Y KEY..";
20 R=INT(RND*10)+1
30 A$=INKEY$:IF A$
   =" " THEN 20
40 PRINT
50 PRINT "NUMBER="
   :R
60 GOTO 10

```

When this program is executed, PRESS ANY KEY ... is displayed, and a random number will be displayed if a key is pressed. This type of programming can also be used for games.

The INKEY\$ function assigns a character with the ASCII code when a key is pressed. Null (" ") is assigned when a key is not pressed. (See page 259)

■ ASC Function

210 Y=ASC(A\$)

└─ ASC function
 └─ Paired ASCII code stored in A\$ is assigned to Y.

The ASCII code of a character stored in character variable A\$ will be assigned to variable Y in this manner. Only the first character of the ASCII code will be assigned if a character string is stored in A\$. 0 will be assigned if nothing is stored in A\$.

```
10 FOR I=1 TO 6
20 READ A$
30 S=ASC(A$)
40 PRINT A$;"=";S
50 NEXT I
60 DATA A,B,C,0,1,
  2
```

Reads characters A, B, C, 0, 1, and 2 and displays the respective ASCII codes.

RUN ↵

↵

↵

↵

A=	65
B=	66
C=	67
0=	48

↵

1 = 49

↵

2 = 50

```

10 A$="ABCDEFGG"
20 B$=""
30 X=ASC(A$)
40 Y=ASC(B$)
50 PRINT "A$=";X
60 PRINT "B$=";Y

```

RUN ↵

A\$ = 65

↵

B\$ = 0

The ASCII code of the first character (A) of the character string stored in A\$ is displayed. 0 will be assigned to B\$ because nothing is stored here.

The ASC function converts an assigned character to its corresponding numerical value (ASCII code). (The CHR\$ function is available as a corresponding function. See page 251.)

The ASC function converts the first character of a given character string to its corresponding numerical value. (See page 251)

* Refer to CHARACTER CODE TABLE. (See page 325)

4-7 \bar{X} -R Control

It is well known that highly exacting qualities are demanded for the cars that we drive daily. Quality control is playing an extremely important role in industrial products and other fields.

The \bar{X} -R control method, which is one of the methods to maintain high quality of products, is used in this program. In \bar{X} -R Control chart, \bar{X} control is a method of obtaining the allowable mean range based on the mean of several measured values (production amount per specified period of time) sampled out of each group at random and R control to obtain the permissive data dispersion range based on R_i ($R_i = \text{maximum} - \text{minimum}$) of each group.

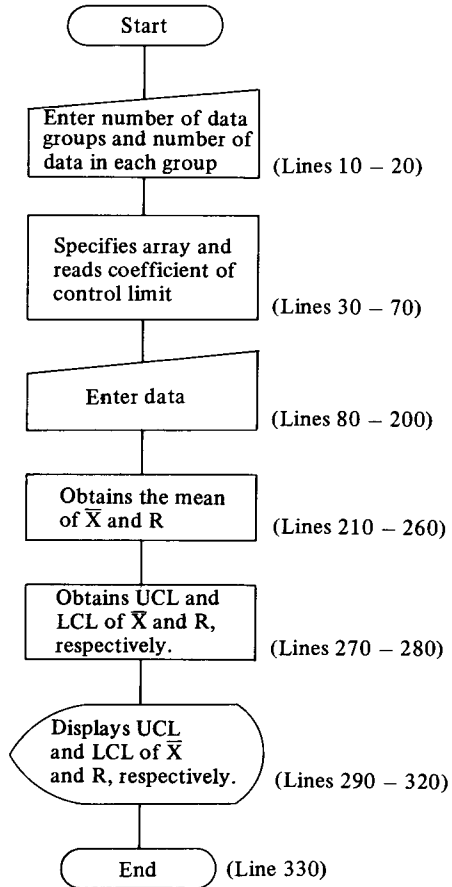
Command

CLS

Program List

```
5 CLEAR
10 INPUT "NO. OF G
  ROUP=";N
20 INPUT "NO. OF D
  ATA/GROUP=";M
30 DIM A(N),R(N),Q
  (10,3)
40 PRINT "READING
  DATA...";
50 FOR I=1 TO 10:F
  OR J=1 TO 3
60 READ Q(I,J)
70 NEXT J:NEXT I
90 CLS
90 FOR I=1 TO N
100 STAT CLEAR :MA=
  0:MI=10000
110 FOR J=1 TO M
120 PRINT "GRP:";I;
  ",DATA":J;"=";
130 INPUT X
140 STAT X
150 IF X>MA THEN MA
  =X
160 IF X<MI THEN MI
  =X
170 NEXT J
180 A(I)=SUMX/CNT
190 R(I)=MA-MI
200 NEXT I
210 STAT CLEAR
220 FOR I=1 TO N
230 STAT A(I),R(I)
240 NEXT I
250 XB=SUMX/CNT
260 RB=SUMY/CNT
270 U1=Q(N,3)*RB:L1
  =Q(N,2)*RB
280 U2=XB+Q(N,1)*RB
  :L2=XB-Q(N,1)*R
  B
290 BEEP :PRINT "XB
  AR=":ROUND(XB,-
  3)
300 PRINT "LCL=";RO
  UND(L1,-3);",UC
  L=":ROUND(U1,-3
  )
310 BEEP :PRINT "RB
  AR=":ROUND(RB,-
  3)
320 PRINT "LCL=";RO
  UND(L1,-3);",UC
  L=":ROUND(U1,-3
  )
330 END
340 DATA0,0,0
350 DATA1.88,0,3.26
  8
360 DATA1.023,0,2.5
  74
370 DATA0.729,0,2.2
  88
380 DATA0.577,0,2.1
  14
390 DATA0.483,0,2.0
  04
400 DATA0.419,0.076
  ,1.924
410 DATA0.373,0.136
  ,1.864
420 DATA0.337,0.184
  ,1.816
430 DATA0.308,0.223
  ,1.777
```


Flow Chart



Enter sampling data and obtain control limits.

Entered Data

Sample \ Group	1	2	3
1	10.3	10.2	10.2
2	10	10.1	10
3	10.3	10.2	10.3
4	10.4	10.3	10.3

Operation

RUN ↵

NO. OF GROUP=?

Enter the number of groups.

3 ↵

NO. OF DATA/GROUP=? ↵

Enter the number of data.

4 ↵

READING DATA...

DATA has been read in.

GRP: 1, DATA 1=?

Enter data of group 1.

10.3 ↵

GRP: 1, DATA 2=?

10 ↵

GRP: 1, DATA 3=?

10.3 ↵

GRP: 1, DATA 4=?

10.4 ↵

GRP: 2, DATA 1=?

⋮

⋮

Displays mean value of X when the 4th data of group 3 is entered.

10.3 ↵

XBAR= 10.22

Displays the lines of LCL and UCL. (\bar{X} control – upper control limit and lower control limit)



```
LCL= 9.91,UCL= 10.52
```

Displays the mean value of R. (Average of Rs (maximum – minimum))



```
RBAR= 0.3
```

Displays the lines of LCL and UCL. (R control – upper control limit and lower control limit)



```
LCL=0,UCL= 0.77
```

The quality of this group may be considered excellent since the data entered is within the range of $9.91 \leq \text{data} \leq 10.52$.

■ CLS Command

```
80 CLS
```

└── CLS command

The CLS command is used on line 80 to erase display content. The CLS command can therefore be used in this manner to clear the display.

```
10 FOR I=0 TO 22
20 LOCATE I:PRINT
   "#";
30 FOR J=1 TO 5:NE
   XT J
40 CLS
50 NEXT I
```

In this program, the symbol # moves from left to right.

(Each time # moves 1 character to the right, the display is cleared. This process is repeated 23 times.)

The CLS command has the function of erasing the display contents.
(See page 224)

4-8 ABC Analysis

The products handled by the retail sales industry are extremely wide in variety. Compiling sales by type and grasping the sales channels of these many products become a tremendous task. With this program, it will be possible to compile sales in descending order and also assign A, B and C rankings by simply entering the product name and the sales amount.

The ranking in this program is set up as follows:

A Rank Products that amount to 80% of the total and are in the top 10 in ranking.

B Rank Products that amount to 15% of the total excluding the A rank products.

C Rank All remaining products.

Therefore, when using for scientific analysis, the data may be ranked as follows based on the probability distributions approximated by a normal curve.

Within the range of $\pm 1\sigma$ (A Rank)

Within the range of $\pm 2\sigma$ (B Rank)

Within the range of $\pm 3\sigma$ (C Rank)

Commands and Functions

REM, LEFT\$, MID\$, RIGHT\$

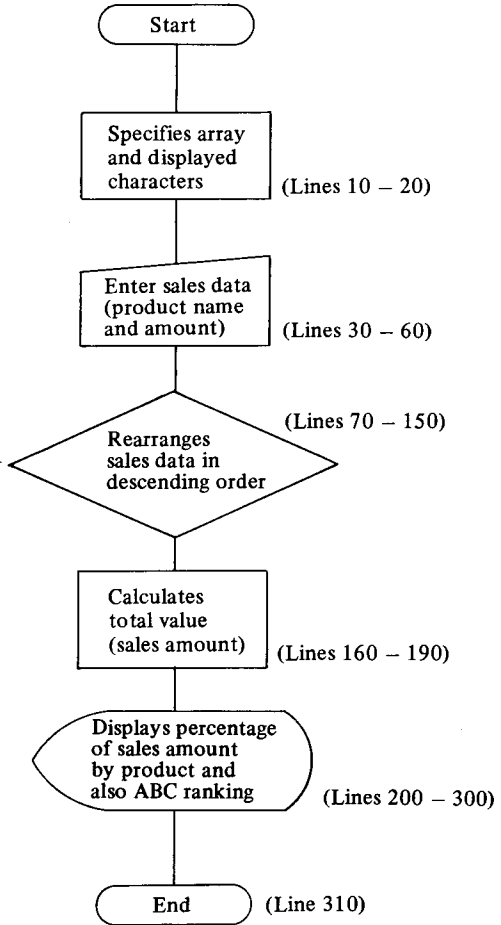
Program List

```
10 INPUT "NO. OF I          80 F=0
   TEM=";N                 90 FOR I=1 TO N-1
20 DIM S(N),B$(N):        100 IF S(I)>=S(I+1)
   P$=CHR$(37):C$=        THEN 140
   "ABC"                   110 X=S(I):S(I)=S(I
30 FOR I=1 TO N            +1):S(I+1)=X
40 PRINT "NO.":I:;        120 X$=B$(I):B$(I)=
   INPUT " NAME=";        B$(I+1):B$(I+1)
   B$(I)                  =X$
50 PRINT "NO.":I:;        130 F=1
   INPUT " SALES="       140 NEXT I
   :S(I)                  150 IF F=1 THEN 70
60 NEXT I                 160 REM CALC & DISP
70 REM SORT DATA        170 G=0
```

Flow Chart

```

180 FOR I=1 TO N:G=
    G+S(I):NEXT I
190 T=0
200 PRINT "NAME SAL
    ES TOTAL (*;P$;
    ")
210 FOR I=1 TO N
220 T=T+S(I)
230 PRINT LEFT$(B$(
    I),5);
240 PRINT S(I);T;
250 R=T/G*100
260 PRINT ROUND(R,-
    2);
270 IF R<80 THEN IF
    I<=10 THEN PRI
    NT LEFT$(C$,1):
    GOTO 300
280 IF R<95 THEN PR
    INT MID$(C$,2,1
    ): GOTO 300
290 PRINT RIGHT$(C$
    ,1)
300 NEXT I
310 END
    
```



Perform ABC analysis by entering product name (or product number) and overall sales amount.

Product number	Sales amount
001	5703
002	4328
003	3984
004	8905
005	2305
006	2996
007	4848
008	2636
009	2095
010	5510

(Do not use 3-digit positional representation.)

Operation

RUN ↵

NO. OF ITEM=?

Enter the total number of items.

10 ↵

NO. 1 NAME=?

Enter the product number (or name).

001 ↵

NO. 1 SALES=?

Enter the sales amount for 001.

5703 ↵

NO. 2 NAME?

⋮

Enter in the same manner down to product No. 10.

After data are entered, the FX-750P will sort the data in about 20 seconds and display the results.

	NAME	SALES	TOTAL	(%)
┌	004	8905	8905	20.6A
┌	001	5703	14608	33.7A
┌	010	5510	20118	46.5A
⋮				
┌	005	2305	41215	95.2C
┌	009	2095	43310	100C

The product numbers will be sorted in sequential order from the high sales amount and will be displayed together with accumulation, percentages and rankings.

Order	Product number	Sales amount	Accumulation	Accumulating percentages to the total	Rank
1	004	8905	8905	20.6	A
2	001	5703	14608	33.7	A
3	010	5510	20118	46.5	A
4	007	4848	24966	57.6	A
5	002	4328	29294	67.6	A
6	003	3984	33278	76.8	A
7	006	2996	36274	83.8	B
8	008	2636	38910	89.8	B
9	005	2305	41215	95.2	C
10	009	2095	43310	100	C

■ For Easy Understanding of the Program (REM Command)

```

70 REM SORT DATA
160 REM CALC & DISP
    └─ REM command
    
```

The commands on lines 70 and 160 do not directly affect execution of the program itself. The reason these commands are used is to make the program clearer and easier to understand. That is, the longer a program becomes, the more difficult will it be for other people to understand the program. Comments are therefore inserted in the program with REM commands to facilitate quicker understanding of the program.

```

10 REM AVERAGE PRO
    GRAM
20 FOR I=1 TO 5
30 READ X:REM READ
    DATA
40 S=S+X:REM CALCU
    LATE TOTAL
50 NEXT I
60 REM CALCULATE A
    VERAGE
70 AV=S/5
80 REM PRINT OUT
90 PRINT "AVE=";AV
100 END
110 REM ORIGINAL DA
    TA
120 DATA 54,48,68,4
    2,59
    
```

The program will become very easy to understand if comments of this type are inserted.

The REM command has the function of inserting comments in the program.
(See page 245)

■ LEFT\$ Function

```
270 IF R<80 THEN IF I<=10
    THEN PRINT LEFT$(C$,1):GOTO 300
```

LEFT\$ function: Fetches the first character from the left of the character string stored in C\$.

As shown on line 270, the LEFT\$ function can fetch the specified number of characters from the left side of the character string stored in a character variable.

```
10 A$=LEFT$("ABCDE
    FG",4) → Fetches 4 characters from the left side of character
20 PRINT A$ string "ABCDEFGG".
RUN ↵
```

ABCD

```
10 A$="SUMMER"
20 B$=LEFT$(A$,4) → Fetches 4 characters from the left side of the
30 PRINT B$ character string in A$.
RUN ↵
```

SUMM

```
10 A$=LEFT$("ABCD"
    ,8) → If the character string to be fetched is longer than
20 PRINT A$ the original character string, the original character
string itself will be assigned to A$.
RUN ↵
```

ABCD

```
10 A$=LEFT$("",5) → If there is no character string, null (" ") will be
20 PRINT A$ assigned to A$.
RUN ↵
```

(Null is displayed.)

The LEFT\$ function fetches a character string of the specified length from the left side of a given character string. (See page 256)

■ MID\$ Function

```
280 IF R<95 THEN PRINT MID$(C$,2,1)
```

MID\$ function: Fetches one character towards the right from the 2nd character of the character string stored in C\$.

As shown on line 280, the MID\$ function can fetch the specified number of characters from the specified position in the character string stored in a character variable.

```
10 A$=MID$("ABCDEF" → Fetches 2 characters from the 4th character (D)
   6",4,2)           from the left of character string "ABCDEF".
20 PRINT A$
```

RUN ↵

DE

```
10 A$=MID$("ABCDEF" → If the number of characters to the right of the
   6",3,7)           specified position is less than that of characters
20 PRINT A$         to be fetched, all characters from that position
                    will be fetched.
```

RUN ↵

CDEFG

```
10 A$=MID$("ABCDEF" → If the specified position exceeds the length of
   6",10,3)          the given character string, null (" ") will be
20 PRINT A$         assigned to A$
```

RUN ↵

(Null is displayed.)

```
10 A$=MID$("ABCDEF" → If the length of the character string to be fet-
   6",3)             ched is omitted, all characters after the speci-
20 PRINT A$         fied position will be fetched.
```

RUN ↵

CDEFG

The MID\$ function fetches the specified number of characters towards the right of the specified position in a given character string. (See page 258)

■ RIGHTS Function

```
290 PRINT RIGHT$(C$,1):GOTO 300
```

RIGHT\$ function: Fetches 1 character from the right side of the character string stored in C\$.

As shown on line 290, the RIGHTS function can fetch the specified number of characters from the right side of a character string stored in a character variable.

```
10 A$=RIGHT$("ABCD
   EFG",4)
20 PRINT A$
```

Fetches 4 characters from the right of character string "ABCDEFG".

RUN ↵

```
DEFG
```

```
10 A$=RIGHT$("ABCD
   EFG",10)
20 PRINT A$
```

If the length of the character string to be fetched is longer than the original character string, the original character string itself will be assigned to A\$.

RUN ↵

```
ABCDEFG
```

```
10 A$=RIGHT$("",5)
20 PRINT A$
```

If there is no character string, null (" ") will be assigned to A\$.

The RIGHTS function fetches a character string of the specified length from the right side of a given character string. (See page 257)

4-9 Chemical Calculations

(Radiation attenuation calculation)

All countries have recently become hypersensitive in relation to radioactivity. Instead of simply fearing radiation, however, shouldn't we try to learn the true nature of radiation?

The type of radiation is selected and the following calculations are performed and the results are displayed in this program.

1. How many hours have already elapsed?
2. What will be the amount of radiation at a certain time from now.
3. What was the initial intensity of radiation?

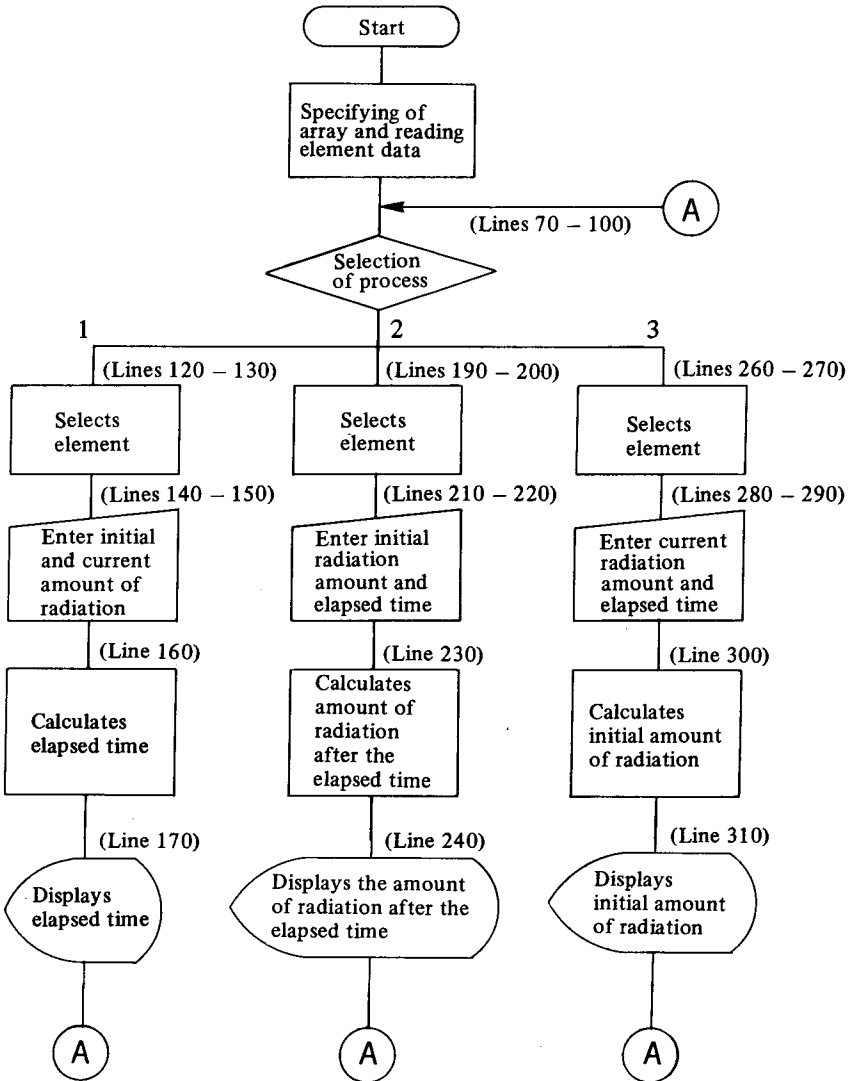
Commands and Functions

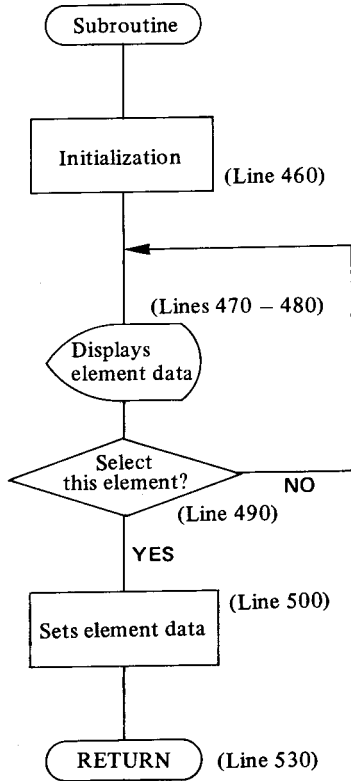
ON ~ GOTO、LOG、GOSUB / RETURN

Program List

```
10 CLEAR
20 N=10
30 DIM M$(N),T(N)
40 FOR I=1 TO N
50 READ M$(I),T(I)
60 NEXT I
70 INPUT "1=TIME,2
=AFTER,3=BEFORE
",X
75 IF X>3 THEN BEE
P : GOTO 60
80 IF X<1 THEN BEE
P : GOTO 70
90 IF X>3 THEN BEE
P : GOTO 70
100 ON X:GOTO 120,1
90,260
110 END
120 REM TIME
130 GOSUB 440
140 INPUT "INITIAL=
":B
150 INPUT "PRESENT=
":C
160 T=Y*LOG(C/B)/LO
G(1/2)
170 PRINT "TIME=":T
180 GOTO 70
190 REM AFTER
200 GOSUB 440
210 INPUT "INITIAL=
":B
220 INPUT "TIME=":T
230 C=B*(1/2)^(T/Y)
240 PRINT "AFTER=":
C
250 GOTO 70
260 REM BEFORE
270 GOSUB 440
280 INPUT "PRESENT=
":C
290 INPUT "TIME=":T
300 B=C/(1/2)^(T/Y)
310 PRINT "BEFORE="
:B
320 GOTO 70
330 REM DATA FOR I
SOTOPE
340 DATA H3,1.07415
12E5
350 DATA C14,5.0194
8E7
360 DATA C060,46103
.88
370 DATA C057,6400
380 DATA I125,1440
390 DATA I131,193.2
400 DATA CS137,2628
00
410 DATA CR51,667.2
420 DATA F18,1.8283
430 DATA P32,342.72
440 REM SELECT DATA
450 PRINT "SELECT I
SOTOPE!"
460 Y=0
470 FOR I=1 TO N
480 PRINT M$(I);
490 Z$=INKEY$:IF Z$
="" THEN 490
500 IF Z$=CHR$(13)
THEN PRINT :Y=T
(I): GOTO 530
510 PRINT :NEXT I
520 GOTO 460
530 RETURN
```

Flow Chart





Select the element and process. Enter 2 from among the 3 factors of initial value, current value and elapsed time to obtain the value of the remaining factor.

Input Data

Data 1

Element name	CR51
Initial amount of radiation	100 curies
Current amount of radiation	50 curies

Data 2

Element name	CR51
Initial amount of radiation	100 curies
Elapsed time	2 weeks

Data 3

Element name	CR51
Current amount of radiation	70.5 curies
Elapsed time	1 week

Radioactivity and Half-life Period (Unit: hour)

* Element names will be displayed in the following order.

H3 (Heavy hydrogen) 1.0741512E5

C14 (Carbon 14) 5.01948E7

CO60 (Cobalt 60) 64103.88

CO57 (Cobalt 57) 6480

I125 (Iodine 125) 1440

I131 (Iodine 131) 193.2

CS137 (Cesium 137) 262800

CR51 (Krypton 51) 667.2

F18 (Iron 18) 1.8283

P32 (Phosphorous 32) 342.72

Operation

RUN ↵

1=TIME, 2=AFTER, 3=BEFORE

Specifying 1 estimates the elapsed time from the current amount of radiation.
 Specifying 2 estimates the amount of radiation after the elapsed time.
 Specifying 3 estimates the initial amount of radiation from the current amount of radiation and the elapsed time.
 Let's select 1.

1 ↵

SELECT ISOTOPE!

Press the **ENTER** key until element name CR51 is displayed.

ENTER

H3

⋮

⋮

ENTER

CR51

After CR51 is displayed, press ↵ and proceed to the next operation.

↵

INITIAL=?

Enter 100 curies for the initial amount of radiation of krypton 51.

100 ↵

PRESENT=?

The present amount of radiation is 50 curies.

50 ↵

TIME= 667.2

It is estimated that the elapsed time for radiation to drop from 100 curies to 50 curies is about 27.8 days $(667.2 \text{ (hours)} \div 24)$.

↵

1=TIME, 2=AFTER, 3=BEFORE

Next, obtain the amount of radiation after elapse of a specified period of time from the current amount of radiation.

2 ↵

ENTER

ENTER

SELECT ISOTOPE!

H3

C14

Select krypton 51.

ENTER

CR51

To proceed to the next process.

↵

INITIAL=?

The initial amount of radiation is 100 curies.

100 ↵

TIME=?

Here we set the elapsed time as 2 weeks (24 hours × 14 days).

24*14 ↵

AFTER= 70.53459231

The display shows that the estimated amount of radiation after 2 weeks is approximately 70.5 curies.

Press keys as follows to obtain the initial amount of radiation from the current amount of radiation and the elapsed time.

↵

3 ↵

ENTER

⋮

ENTER

↵

1=TIME, 2=AFTER, 3=BEFORE

SELECT ISOTOPE!

H3

⋮

CR51

PRESENT=?


```

40 GOTO 10
100 PRINT "BOOK"
110 GOTO 10
200 PRINT "WATCH"
210 GOTO 10
300 PRINT "RADIO"
310 GOTO 10

```

If the value of X is greater than the number of the destination, or is a fraction, the ON ~ GOTO command causes the next line to be executed.

The ON ~ GOTO command has the function of specifying the destination by the content of the variable following ON. (See page 238)

■ Perform Similar Operations Together. (GOSUB/RETURN)

```

130 GOSUB 440
  ⋮
200 GOSUB 440
  ⋮
270 GOSUB 440
  ⋮
440 REM SELECT DATA ← Subroutine executed with GOSUB
450 PRINT "SELECT ISOTOPE!"
  ⋮
530 RETURN
      RETURN command

```

The statement GOSUB 440 is written in each of the line 130, 200 and 270. There is actually 3 times when radioactive isotopes are selected in this program. This is an operation to select 1 element from a vast number of elements. An extremely long program will therefore result if we attempt to create a separate program for each case.

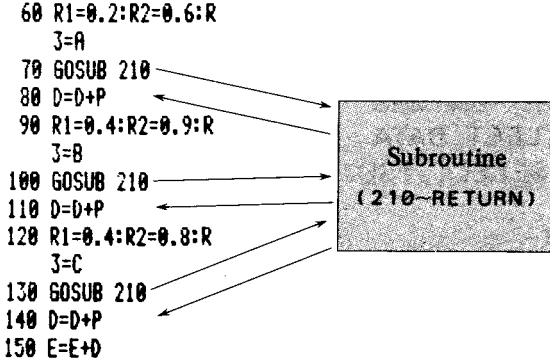
It would therefore be convenient if the programs could be combined and also be stored so they can be called when necessary. The program on lines 440 – 530 is a subroutine created in this manner.

```

440 REM SELECT DATA
450 PRINT "SELECT I
      SOTOPE!"
460 Y=0
470 FOR I=1 TO N
480 PRINT M$(I);
490 Z$=INKEY$:IF Z$
      =" " THEN 490
500 IF Z$=CHR$(13)
      THEN PRINT :Y=T
      (I): GOTO 530
510 PRINT :NEXT I
520 GOTO 460
530 RETURN
    
```

} Subroutine

Programs created for this purpose always have a RETURN command at the end. In other words, executions from the line specified with GOSUB to the RETURN command are considered a separate program called a subroutine (subprogram).



The RETURN command returns execution to the statement or line next to the line in which GOSUB command has been executed. Execution jumps to the next line in the above example since there is no statement corresponding with GOSUB 210.

```

10 A=7
20 GOSUB 100
30 PRINT "C=";C
40 END
100 C=A*2
110 B=A/2
120 PRINT "B=";B
130 RETURN
    
```

Both values of B and C will be displayed here as there is a RETURN command.

RUN ↵

B = 3.5

↵

C = 14

```

10 A=100
20 B=250
30 RETURN
    
```

If only a RETURN command exists and there is no GOSUB command, a GS ERROR will occur.

RUN ↵

GS ERROR P0-30

A GOSUB command cannot only specify line numbers but can also specify program areas as well.

For example, let us assume that the following program is in program area 0.

```

10 A=10
20 GOSUB PROG 1 ——— Jumps to program area 1 (subroutine) and return
30 PRINT B              after executing the program.
40 END
    
```

Also, that the following program is in program area 1.

```

10 B=A*2/5
20 RETURN
    
```

The value of B will be calculated and displayed if program execution is performed from program area 0.

RUN ↵

4

A subroutine can also be called from among the different subroutines by the GOSUB command.

```

10 A=15
20 GOSUB 100 —————> Calls the subroutine starting from line 100.
30 PRINT "C=";C
40 END —————> Programs following line 100 will be executed if
                    there is no END command here and GS ERROR
                    will occur.
100 REM SUBROUTINE-
    1
110 B=A*A*A
120 GOSUB 1000 —————> Calls subroutine starting from line 1000.
130 RETURN
1000 REM SOBROUTINE-
    2
1010 C=SQR(B)
1020 RETURN

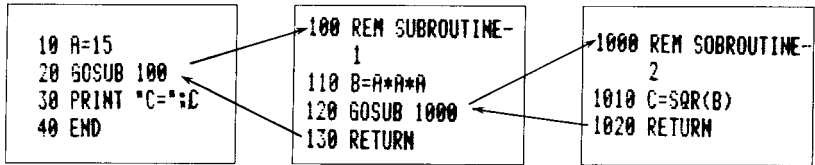
```

$\sqrt{15^3}$ will be calculated and displayed here.

RUN ↵

C = 58.09475019

Putting in easy-to-understand form, the program will be as follows.



The subroutines called with the GOSUB command are linked successively in this manner. Subroutines can be specified up to 12 levels with the FX-750P. An NO ERROR will occur if over 12 levels.

```

10 GOSUB 20 — 1
15 END
20 GOSUB 30 — 2
25 RETURN
30 GOSUB 40 — 3
35 RETURN
40 GOSUB 50 — 4
45 RETURN
50 GOSUB 60 — 5
55 RETURN
60 GOSUB 70 — 6
65 RETURN

```

```

70 GOSUB 80— 7
75 RETURN
80 GOSUB 90— 8
85 RETURN
90 GOSUB 100—9
95 RETURN
100 GOSUB 110—10
105 RETURN
110 GOSUB 120—11
115 RETURN
120 GOSUB 130—12
125 RETURN
130 GOSUB 140—13:
135 RETURN
140 GOSUB 150—14:
145 RETURN

```

→ GOSUB cannot be used from here on.

The GOSUB/RETURN command has the function of causing program execution to be branched to the statement or line next to the line including the GOSUB command, after calling and executing a subroutine.

(See page 231)

■ END Command

```

110 END
      |
      | END command

```

If you wish to terminate execution of a program, enter END at the desired place. Also, enter END at the end of a program such as on line 110. (Although the lack of END here will not affect execution, it will be desirable to make it a habit to enter END.)

The END command is similar to the STOP command in stopping execution of a program. However, program execution can be resumed with the CONT command in the case of the STOP command but this is not possible in the case of the END command.

Following is an example showing how the END command is used.

```

10 PRINT "ABC";
20 GOTO 50
30 PRINT "GHI"
40 END
50 PRINT "DEF";
60 GOTO 30
    
```

RUN ↵

↵

ABCDEFGHI
READY P0

If this program is executed with 40 END deleted, DEFGHI will continue to be displayed each time the ↵ key is pressed. Press [BRK] to terminate the program.

RUN ↵

↵

ABCDEFGHI
DEFGHI

The END command has the function of terminating execution of a program. (See page 227)

■ LOG Function

```

160 T=Y*LOG(C/B)/LOG(1/2)
           └── LOG function ─┘
    
```

On line 160, the elapsed time is being obtained from the initial and current amount of radiation of a radioactive body. The LOG function is used here for this purpose. The LOG function permits the calculation of natural logarithms and is expressed in the form of $\log_e x$ in mathematics.

```

10 FOR I=1 TO 10
20 X=LOGI
30 PRINT "LOG(";I;
   ")=":X
40 NEXT I
    
```



```
RUN ↵
```

```
LOG( 1)= 0
```

```
↵
```

```
LOG( 2)= 0.6931471806
```

```
⋮
```

```
⋮
```

```
↵
```

```
LOG( 10)= 2.302585093
```

The values to be calculated by the LOG function must be $0 < \text{numerical expression}$. An MA ERROR will be displayed in the following case.

```
10 A=-5
20 X=LOGA
30 PRINT X
```

```
RUN ↵
```

```
MA ERROR P0-20
```

A similar function to the LOG function calculating natural logarithm, $\log_e x$ is the LGT function which obtains common logarithm, $\log_{10} x$.

```
10 FOR I=1 TO 10
20 X=LGTI
30 PRINT "LGT(*;I;
   ")=";X
40 NEXT I
```

```
RUN ↵
```

```
LGT( 1)= 0
```

```
↵
```

```
LGT( 2)= 0.3010299957
```

```
⋮
```

```
⋮
```

```
↵
```

```
LGT( 10)= 1
```

The LOG function obtains the natural logarithm of given numerical values.
(See page 267)

4-10 Number Guessing Game

This may be a game you can enjoy during a break from study or work. The game is comparatively simple in that the player guesses a number between 1 and 20 which the FX-750P has picked at random.

The program uses a random number function to have the computer pick a number. Once you acquire the knack of this game, you should be able to guess the number within 4 tries.

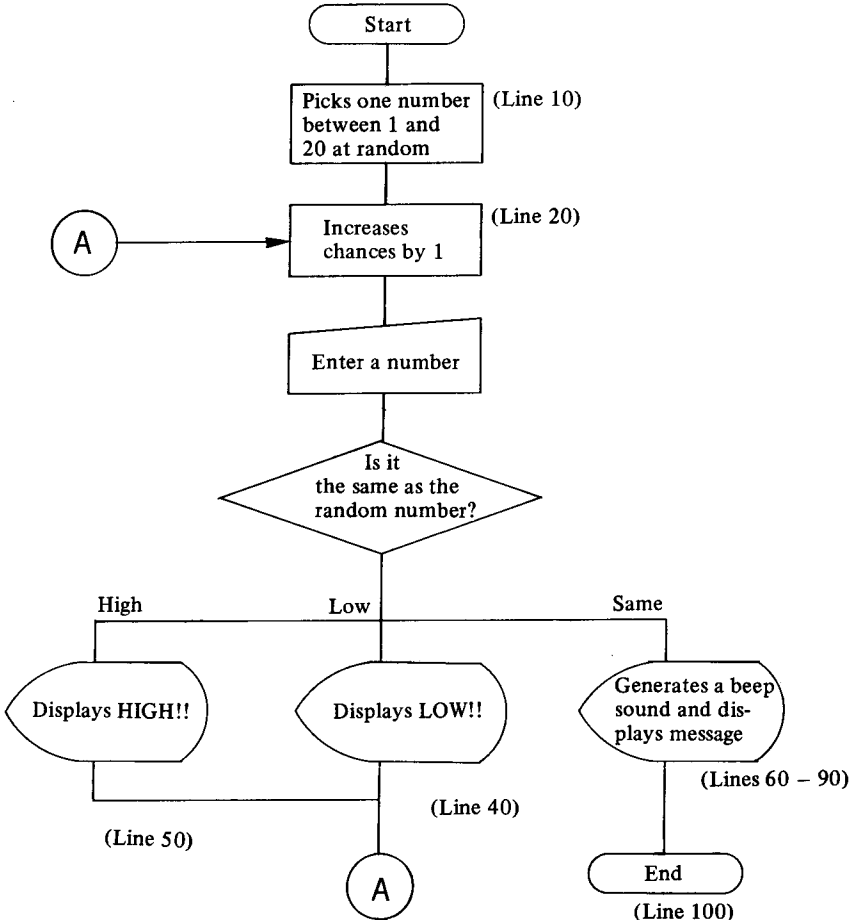
Commands and Functions

RND、INT、BEEP、WAIT

Program List

```
10 A=INT(RND*20+1)
   :C=0
20 C=C+1
30 INPUT "NUMBER!,";B
40 IF B<A THEN PRINT "LOW!!:";GOTO 20
50 IF B>A THEN PRINT "HIGH!!:";GOTO 20
60 BEEP
65 WAIT 30
70 PRINT "CONGRATULATION!!"
80 PRINT " YOU GUESSED IT!!"
90 PRINT "YOU GOT IT IN";C;"TRIES"
100 END
```

Flow Chart



Guess and enter a number between 1 and 20 picked by the FX-750P.

Operation:

RUN ↵

NUMBER!..?

Guess number 10 for example.

10 ↵

HIGH!!! NUMBER!..?

Since 10 is too high, let's enter 3.

3 ↵

LOW!!! NUMBER!..?

Since 3 is too low, the correct number will be between 3 and 10 so we'll try 5.

5 ↵

CONGRATULATION!!

↵

YOU GUESSED IT!!!

You got it in 3 tries.

YOU GOT IT IN 3 TRIES

■ RND Function

```
10 A=INT(RND*20+1):C=0
```

└ RND function

Although the RND function can be used for various purposes without being modified, it can also be used to generate random numbers within a desirable range (0 to 100, 100 to 200, etc.) by creating a numerical expression as shown on line 10. A random number between 1 and less than 21 is being obtained by generating random numbers from 0 to less than 20 and adding 1.

We'll use a simple program here to show how the random number is generated.

```
10 A=RND → Generates a random number and stores in A.
20 PRINT A → Displays the random number stored in A.
30 GOTO 10
```

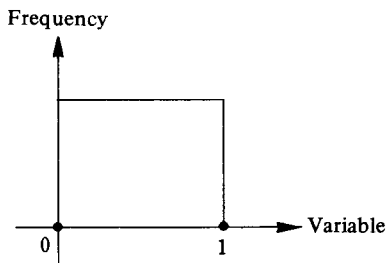
RUN ↵

↵

0.9828790089
0.4737026837

⋮

The random numbers displayed successively as mentioned above are called uniform random numbers and are used widely in games and simulations. (Uniform random numbers are random numbers that are generated uniformly.)



For reference purposes, a check was made of the FX-750P with the following results.

```

10 STAT CLEAR
20 FOR I=1 TO 1000
30 STAT RND
40 NEXT I
50 PRINT "MEAN=";S
   UMX/CNT
60 PRINT "SD=";SDX
    
```

RUN ↵

MEAN= 0.490399328 ↵

The mean of 1000 random numbers

↵

SD= 0.2837680059 ↵

Standard deviation

Use the following expression to create random numbers within the range of 15 ≤ numerical value ≤ 50.

```

10 R=INT(RND*35+1)+15
20 PRINT R
    
```

The RND function generates random numbers of up to 10 digits that are larger than 0 but smaller than 1 ($0 < X < 1$). (See page 272)

■ INT Function

```

10 A=INT(RND*20+1):C=0
    
```

INT function

Integer portion is to be extracted.

On line 10, a random number including fractions from 1 to 20 are generated and converted to an integer.

In addition to numerical expressions, numerical values and variables can also be used in the parenthesis.

Now, we'll see how the INT function works.

```
10 A=RND*10
```

```
20 PRINT A;" ";INT  
(A)
```

```
30 GOTO 10
```

The RND function generates a random number smaller than 1 but larger than 0 and the result is multiplied by 10.

Displays the random number multiplied by 10 and the numerical value after the INT function is executed.

```
RUN ↵
```

```
↵
```

```
5.737355616, 5
```

```
1.375856229, 1
```

You will note that the integer portion of the numerical value has been extracted and displayed at the right.

When the numerical value is a negative number, the INT function provides an integer that is 1 less than the integer of the numerical value.

Modify the previous program so a negative random number will appear on line number 10 and execute the program.

```
10 A=-RND*10
```

```
20 PRINT A;" ";INT  
(A)
```

```
30 GOTO 10
```

```
RUN ↵
```

```
↵
```

```
-4.314976866, -5
```

```
-9.389450762, -10
```

The display shows that the integer at the right is 1 less than the integer portion of the numerical value.

The INT function provides the maximum integer that does not exceed the given numerical value. (See page 268)

■ BEEP Command

```
60 BEEP
    └ BEEP command
```

On line 60, a buzzer sound is generated when the correct number is entered. Interesting sound effects can be enjoyed with the BEEP command as the tone changes when set to BEEP (or BEEP 0) or BEEP 1.

Modifying line 60 as shown below, for example, will give an interesting sound effect.

```
60 BEEP 0:BEEP 1:8
    EEP 0:BEEP 1
```

The BEEP command has the function of generating high and low tones. BEEP (or BEEP 0) generates a low tone and BEEP 1 generates a high tone. (See page 221)



■ Controlling the Display Time (WAIT Command)

```
65 WAIT 30
    └ Time specifying number (n)
```

The time interval of data display is set by the WAIT command on line 65. The WAIT command is capable of setting optional display time of the data to be output through PRINT command in this manner.

The time interval set with the WAIT command will be about 0.05 seconds x time specifying number (*n*).

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
```

In the above program, the  key or  key must be pressed each time a numerical value is displayed.

CHAPTER

COMMAND REFERENCE

5

Symbols

The following notations are used to describe the grammar here.

- **Bold letter words** Must always be written in this form.
- $\left\{ \begin{array}{c} \times \times \times \times \\ \circ \circ \circ \circ \end{array} \right\}$ Must select one from inside the $\left\{ \quad \right\}$ and write.
- $\left[\begin{array}{c} \times \times \times \times \\ \circ \circ \circ \circ \end{array} \right]$ Elements in $\left[\quad \right]$ can be omitted.
- $\circ \circ \circ \circ *$ Elements with * superscripted at the right can be written repeatedly.
- Numerical expression Numerical values, calculation expressions and numerical variables such as 10, 2 + 3, A, S*Q, etc.
- Character string Character constants and character variables such as "ABC", X\$, N\$ + M\$, etc.
- Parameter Element accompanying commands.
- **Ⓟ** Can be executed in a program.
- **Ⓜ** Can be executed manually only.
- **Ⓐ** Can be executed manually or in a program.
- **ⓕ** Function that can be executed in a program or manually.

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,"

Example:

GOTO $\left\{ \begin{array}{l} \text{line number} \\ \text{PROG program area number} \end{array} \right\}$

This expresses the following 2 formats.

- 1) GOTO line number
- 2) GOTO PROG program area number

5-1 Manual Commands


CONT



Function:

Continues execution of a temporarily stopped program.

Explanation:

Reexecutes from the line number next to the line that stopped with the STOP command or  key.

Example

Inserting STOP commands in strategic places when creating a program to simplify debugging.

Assume we create a program such as the following to obtain the volume of a cylinder.

```
10 INPUT "R":R
20 INPUT "H":H
30 D=R*2*PI
40 STOP
50 A=D*H
60 PRINT "A=":A
70 END
```

Execute this program with the RUN command.

RUN 

R?

First, enter 10 for example as the computer is asking for the radius of the cylinder.

10 

H?

The computer is now requesting the height H, so we enter 20 for example.

20 

STOP P0-40

Line 30 is executed and the display shows that execution has been stopped with the STOP command on line 40.


Check whether the program lines previous to line 40 have been correctly executed.

Display the value of variable D which is the circular area of a cylinder calculated on line 20.

D 

314.1592654

Check correctness by comparing the value of variable D with the area of a circle with a radius of 10 which has been verified by manual calculation.



Restart execution of the program after line 40 by pressing CONT .

The volume of the cylinder will be displayed as the final results of the calculation and the program will terminate.

CONT 

A= 6283.185307 ^{WAIT}

The STOP command is inserted between sections in the program for use in checking and debugging the program.

The CONT command can be used to restart execution of a program stopped with the  key. However, FC ERROR will occur if the CONT command is used when the program is being corrected or a line being added or deleted with the EDIT command while the program is stopped with the STOP command or  key, and the program will not restart. (See section on STOP)

DELETE { $\left[\frac{\text{starting line number}}{\text{line number}} \right] - \left[\frac{\text{ending line number}}{\text{line number}} \right] \}$ M

Function:

Deletes program lines.





Parameter:

Line number: $1 \leq \text{line number} < 10000$

Explanation:

Cannot be used while specifying a password

Example

DELETE 20  (Deletes line 20 only)
 DELETE 50-100  (Deletes from line 50 to 100)
 DELETE 200-  (Deletes lines from 200 on)
 DELETE -80  (Deletes from the first line to line 80)

EDIT [line number]



Function:

Puts the currently specified program area into mode permitting modification and correction of the program contents and displays the program.

Parameter:

Line number: $1 \leq \text{line number} < 10000$

Explanation:

- 1) This cannot be used while specifying a password.
- 2) The EDIT mode will be cancelled in the following cases.
 1. When the **BRN** key or **CLS** key is pressed.
 2. When an erroneous operation has been performed.
 3. When the program to be displayed no longer exists.
 4. When power is off. (including auto power off)

Examples

EDIT (Displays from the first line.)

EDIT 50 (Displays from line 50.)

LIST	{	{ [starting line number] [-[ending line number]] }	}	Ⓐ
		ALL		
		V		
LLIST	{	{ [starting line number] [-[ending line number]] }	}	
		ALL		
		V		

Function:

LIST: Displays contents of the program in the currently specified program area.

LLIST: Prints out contents of the program in the currently specified program area.




Parameter:

Line number: $1 \leq \text{line number} < 10000$


ALL: Displays (or prints) all program contents from program area P0 to P9 sequentially.


V: Displays (or prints) registered variable names and declared array variable names.


Explanation:


- 1) If the parameter is omitted, the program list will be displayed (or printed) from the first line.
 - 2) The program contents will be automatically displayed (or printed) in sequential order by executing the LIST command. Press  when you wish to stop. Press  again when you wish to display from the next line on.
 - 3) Press the  key to stop the LIST or LLIST command.
- * This cannot be used while specifying a password.


Examples

LIST  (Displays from the first line.)

LIST 20  (Displays line 20)

LIST 50—100  (Displays from line 50 to 100)

LIST 200—  (Displays from line 200 on)

LIST —80  (Displays from the first line to line 80)

LOAD [**ALL**] ["file name"] Ⓐ

LOAD ["file name"] [[, A] [, M]]

Function:

Loads a program stored on a cassette tape into the computer.

Parameter:

- ALL: Reads programs in all program areas into the computer.
- File name: Character string (can be omitted) of $1 \leq \text{file name} \leq 8$ characters.
- ,A: Reads in programs stored with ASCII code format.
- ,M: Merges programs stored with ASCII code format with the main program.
The difference with ,A is that, though the old program will be lost with a regular LOAD operation, the old program will remain with LOAD, M or LOAD "file name", M as long as the line numbers are different.

- Reads the program below into P0 with LOAD, A

Program in P0 when read in with LOAD, A

```
15 X=100
70 Y=200
110 PRINT X;Y
```

Program presently stored in P0

```
10 A=1
20 B=2
30 C=A*B
100 PRINT A;B;C
```

Program stored on the tape by SAVE, A

```
15 X=100
70 Y=200
110 PRINT X;Y
```

Program in P0 when read in with LOAD, M

```
10 A=1
15 X=100
20 B=2
30 C=A*B
70 Y=200
100 PRINT A;B;C
110 PRINT X;Y
```

- Reads above program into P0 with LOAD, M

Explanation:

- 1) Reads program with the same format that appears first after the SAVE command execution if file name is omitted.
- 2) Loads only programs with the same format as during SAVE.

Commands of the same format correspond such as LOAD for programs stored with SAVE and LOAD ALL for programs stored with SAVE ALL. LOAD and SAVE combinations are shown in the following table.

	LOAD	LOAD "file name"	LOAD ALL	LOAD ALL "file name"	LOAD ,A	LOAD "file name" ,A	LOAD ,M	LOAD "file name" ,M
SAVE	○	×	×	×	×	×	×	×
SAVE "file name"	○	○	×	×	×	×	×	×
SAVE ALL	×	×	○	×	×	×	×	×
SAVE ALL "file name"	×	×	○	○	×	×	×	×
SAVE ,A	×	×	×	×	○	×	○	×
SAVE "file name" ,A	×	×	×	×	○	○	○	○

○ – can be loaded. × – cannot be loaded.

3) Password

If a program with a password is stored on a cassette tape, the password will also be stored. Following is reference information for loading programs with a password.

1. LOAD can be performed when the FX-750P is not provided with a password. In this case, a loaded password becomes the password of the FX-750P.
2. When the password provided in the FX-750P is the same as the password of the program to be stored, the program can be loaded as it is. The password will also remain in this case.
3. If the password of the FX-750P and that of the program to be loaded differ, PR ERROR is displayed as soon as loading starts and loading cannot be performed.

Example

LOAD "ABC" 

LOAD ALL 

NEW (ALL)

A

Function:

Erases the program in the currently specified program area.

Erases all programs and variables.

Parameter:

ALL: Erases all programs and variables (including statistical variables) in all program areas (P0 ~ P9).

Explanation:

- 1) The NEW command erases the program in the currently specified program area but does not erase the variables. This cannot be used while a password is being assigned.
- 2) The NEW ALL command performs the following.
 - 1) Clears all numerical variables to 0.
 - 2) Clears all character variables to " " (null).
 - 3) Releases arrays declared by the DIM command.
 - 4) Clears all registered variables.
 - 5) Sets ANGLE to 0 (DEG).
 - 6) Sets program area to P0.

NEW ALL can also be used while assigning a password.

(See ANGLE, DIM and PASS sections)

Example

NEW ↵

NEW ALL ↵

PASS "Password"**Function:**

Specifies or cancels a password.

Parameter:

Password: Character string of $1 \leq \text{password} \leq 8$ characters.

Explanation:

- 1) A password is set in all program areas and can only be released when the password assigned later matches the password presently assigned. A PR ERROR (protect error) will occur if they do not match.
- 2) Creating, editing, deleting and erasing of programs are not possible when the password has been set by this command. (PR ERROR will occur if the DELETE, EDIT, LIST, LLIST, NEW, or SAVE, A command is used.)
- 3) The password will be saved even if the power is turned off.
- 4) If a program is stored on a cassette tape with the SAVE or SAVE ALL command with a password attached, the password will also be stored. If a program with a password is read in from a cassette tape with the LOAD or LOAD ALL command, this password will also be read in. If a program with a password different to that in the computer is read in, a PR ERROR will occur. (See section on SAVE, LOAD and NEW.)

Note:

If you are unable to remember an assigned password, execute the NEW ALL command and clear all programs and variables.

PROG

Program area number

(A)

Function:

Specifies one of the program areas from P0 to P9.


Parameter:

Program area number: $0 \leq \text{numerical value (numerical expression)} < 10$

Explanation:

If a specified program area number (other than $0 \leq \text{numerical value (numerical expression)} < 10$) does not exist, a BS ERROR will occur.

Example

PROG 2 
(Specifies program area P2.)

READY P2

RUN

[Starting line number]

(M)

Function:

Executes a program.

Parameter:

Run starting line number: $1 \leq \text{line number} < 10000$

Explanation:

- 1) Executes from the first line of the program if the line number is omitted.
- 2) Executes from the closest larger line number than that specified if the specified line number does not exist.
- 3) This cannot be used in a program.

Example RUN 
 RUN 1000 

SAVE [ALL] ["file name"] (A)

SAVE ["file name"] ,A

Function:

Stores a program on a cassette tape.

Parameter:

ALL: Stores programs of all program areas on a cassette tape.

File name: Character string of $1 \leq \text{file name} \leq 8$ characters. (This can be omitted.)

,A: Stores the program in the currently specified program area with ASCII code format.

Although more time will be required to store with the ASCII format, programs must be stored with this format to execute the LOAD,M command (see LOAD). This cannot be used when a password is assigned. If used, a PR ERROR will occur.

(See LOAD, VERIFY.)

Example

SAVE 'ABC' ↵

SAVE ALL ↵

STAT LIST



A

STAT LLIST

Function:

STAT LIST:	Displays basic statistics.
STAT LLIST:	Prints out basic statistics.

Explanation:

- 1) Basic statistics are displayed in the following order.
CNT, SUMY, SUMX, SUMXY, SUMX2, and SUMY2 (see page 273)
- 2) When desiring to stop display temporarily, press  or . Press again to restart.
- 3) Use the STAT LLIST command when in PRINT ON mode since the STAT LIST command cannot be used in this mode.

ExampleSTAT LIST STAT LLIST 

SYSTEM ({ P })



Function:

Displays program area status, number of remaining bytes and the number of RAM bytes used.

Parameter:

P: Displays the number of bytes used for the program.

V: Displays number of bytes used for the variables.

Explanation:

SYSTEM

```
P #123#56789 1089 BYTES
```

Program area status
Number of remaining bytes

Displays a heart symbol in an area where a program is written. A number or a heart symbol blinks at the currently specified program area.

SYSTEM P

```
PROG 1185 BYTES USED
```

Displays the number of bytes being used for the program (including the program management area).

SYSTEM V

```
VAR 1021 BYTES USED
```

Displays the number of bytes being used for the variables (including the variable management area).

VERIFY ["file name"]

**Function:**

Checks status of a program or data stored on a cassette tape.

Parameter:

File name: Character string of $1 \leq \text{file name} \leq 8$ characters.
(This can be omitted.)

Explanation:

Checks the first file that appears after execution of the command if the file name is omitted. RW ERROR will occur if not recorded correctly.

Example

VERIFY ↵

VERIFY "PRG.5" ↵

5-2 Program Commands

ANGLE Numerical expression

(A)

Function:

Specifies angle units.

Parameter:

Numerical expression: $0 \leq \text{numerical expression} < 3$

Explanation:

ANGLE 0 – DEGREE Expressed by degrees such as 90° , 360° , etc.

ANGLE 1 – RADIAN Expressed by radians such as $\pi/2$, 2π , etc.

ANGLE 2 – GRADE Expresses 90° as 100 such as 100, 400, etc.

The ranges are as follows.

Degree: $|\text{angle}| < 5400^\circ$

Radian: $|\text{angle}| < 30\pi$ radians

Grade: $|\text{angle}| < 6000$ grades

* $360^\circ = 2\pi$ radians = 400 grades

ANGLE 0 is set when power is first turned on.

Example

```
10 INPUT R
20 FOR I=0 TO 2
30 ANGLE I
40 PRINT SINR
50 NEXT I
60 END
```

BEEP ({ 0 })

Ⓐ

Function:

Generates a buzzing sound.

Parameter:

0: Low sound

1: High sound

Explanation:

The same as 0 if this parameter is omitted.

BEEP Generates a low buzzer sound.

BEEP 0 Ditto

BEEP 1 Generates a high buzzer sound.

Example

```
10 FOR I=1 TO 10
20 BEEP 0:BEEP 1
30 NEXT I
40 END
```

CHAIN [“file name”]

A

Function:

Loads a specified program and executes from the first line.

Parameter:

File name: Character string of $1 \leq \text{file name} \leq 8$ characters.
(This can be omitted.)

Explanation:

- 1) Loads and executes first program that appears after this command is executed if the file name is omitted.
- 2) The program in the currently specified program area for loading will be cleared.
- 3) Programs stored with the SAVE ALL or SAVE, A command cannot be read in with the CHAIN command.
- 4) If the program being loaded is provided with a password, this password will also be read in.
- 5) The execution of this command does not clear variable contents.

Example

```
900 PRINT "PUSH 1-3  
";  
910 K$=INKEY$  
920 IF K$="1" THEN  
    CHAIN "PROG 1"  
930 IF K$="2" THEN  
    CHAIN "PROG 2"  
940 IF K$="3" THEN  
    CHAIN "PROG 3"  
950 GOTO 910
```

CLEAR

A

Function:

Clears all variables.

Explanation:

Clears contents of fixed variables (A ~ Z, A\$ ~ Z\$) to 0 or "" (null).

Also cancels registration of registered variables and releases defined array variables.

CLEAR cannot be used in a FOR ~ NEXT loop since the FOR ~ NEXT nesting stacks will be cleared and it will not be possible to return to the FOR command.

Example

```
10 CLEAR
20 DIM A(19)
30 FOR I=0 TO 19
40 INPUT A(I)
50 T=T+A(I)
60 NEXT I
70 FOR I=0 TO 19
80 PRINT A(I)/T*10
   0
90 NEXT I
100 END
```

* Input 20 data and obtain the percentage of each.

CLS



A

Function:

Clears all displays and moves the cursor to the home position (left end).

Example

```
10 X=10
20 CLS
30 LOCATE X:PRINT
   "*"
40 K$=INKEY$
50 IF K$="4" THEN
   X=X-1:BEEP 0
60 IF K$="6" THEN
   X=X+1:BEEP 1
70 GOTO 20
```

- * If the  key is pressed, the “*” moves to the left and if the  key is pressed, it moves to the right.

A

DIM Array variable name [!] (subscript) [,array variable name [!] (subscript)]*

DIM Array variable name \$ (subscript) [* numerical expression] [,array variable name \$ (subscript) [* numerical expression]]*

Function:

Declares definition of an array variable.

Parameters:

Array variable name: Uses 1 alphabetical letter from A to Z.

! : Specifies a half-precision numerical array.

\$: Specifies a character array.

Subscript: Can define up to a 2-dimensional array.

$0 \leq \text{subscript} < 256$

* Numerical expression: Provided to a character array only and specifies a defined-length character array.

$0 \leq \text{numerical expression} < 80$.

Explanation:

Numerical arrays

1) Single-precision numerical array

Uses 8 bytes per a variable and can store a 12 digit mantissa plus a 2 digit exponent.

2) Half-precision numerical array

Requires ! after an array variable name. One variable uses 4 bytes and can store a 5 digit mantissa plus a 2 digit exponent.

Character arrays

1) Fixed-length character array

Requires \$ after an array variable name and “* numerical expression” is omitted. An array of 16 characters can be stored in a variable and each variable uses 17 bytes.

2) Defined-length character array

Specifies the maximum length of the character string that can be stored by the “* numerical expression” placed after the subscript, requiring a number of characters + 1 byte per a variable.

Note:

- A DD ERROR (duplicate defined error) will occur if the number of variables declared with the same array variable name is different. If this occurs, cancel the previous definition by inserting an ERASE or CLEAR statement before the DIM statement.
- If array specification differs in form such as D!(5), D\$(5) or D(5), the variables will be classified separately even if the array variable names are the same.
- Always declare with the DIM statement when using an array.

END

⒫

Function:

Terminates execution of a program.

Explanation:

Even if there is a program after this command, it will not be executed. This command clears the GOSUB command loop and the FOR ~ NEXT nesting stacks.

Any number of END commands can also be used in a program.

ERASE Variable name [, variable name]*

Ⓐ

Function:

Releases and erases defined registered variables and array variables.

Parameter:

Variable name: Registered variable name or array variable name.

Explanation:

This command can erase registered variables or array variables, confirmed by LIST V, with one variable name as a unit.

If a specified variable name is unregistered, the command will skip this line and execute the next line.

This command cannot be used in a FOR ~ NEXT loop since the FOR ~ NEXT nesting stack will be cleared.

Example

```
10 ERASE D,E$
20 DIM D(10),E$(15
)
```

FOR control variable name = initial value (P)

TO final value [**STEP** increment]

NEXT control variable name

Function:

Repeats execution of statements between FOR and NEXT while changing the control variable from the initial to the final value in the specified increment.

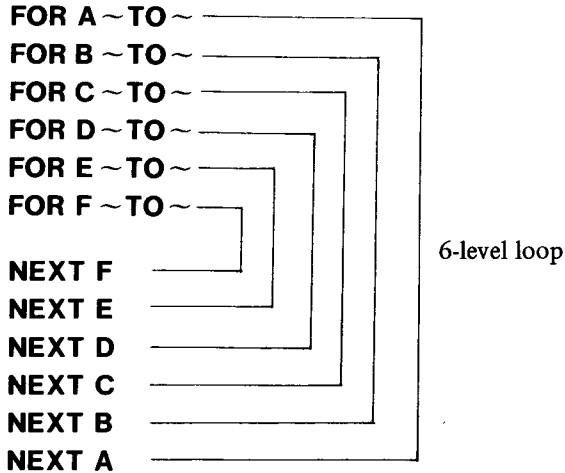
Parameters:

- Control variable name: Numerical variable name. Cannot use array variables.
- Initial value: Numerical expression or numerical variable. Negative number is possible.
- Final value: Numerical expression or numerical variable. Negative number is possible.
- Increment: Numerical expression or numerical variable. Negative number is possible. (This can be omitted.)

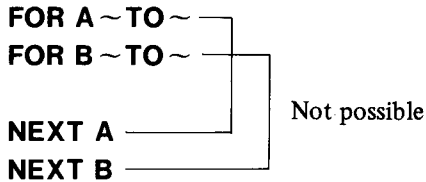
Explanation:

- 1) 1 is set as increment if STEP is omitted.
- 2) If the initial value is greater than the final value, statements between FOR and NEXT will be executed only once.
- 3) Nesting of the FOR ~ NEXT loop is possible up to 6 levels.

Nesting structure



4) The FOR ~ NEXT loop must be completely nested.



FO ERROR will occur if the loops cross as in the above.

- 5) Multiple NEXT commands can exist for one FOR command but an FO ERROR will occur if the NEXT command is executed before the FOR command or there is no NEXT command.
- 6) A jump out of the FOR ~ NEXT loop is possible but a jump into the loop with a GOTO statement, etc. is not possible. Nesting will continue even after jumping out of the loop as long as the statement is not ended with NEXT command.
- 7) CLEAR and ERASE commands cannot be executed in the FOR ~ NEXT loop.
(See CLEAR, ERASE.)

GET [“file name”] variable [,variable]*

A

Function:

Reads data stored on a cassette tape into variables with the PUT command.

Parameter:

File name: Character string of $1 \leq \text{file name} \leq 8$ characters.
(This can be omitted.)

Explanation:

- 1) Reads in first data file that appears on a cassette tape after execution of the command if the file name is omitted.
- 2) The variable name used when storing with the PUT command need not match the variable name used when reading in with this command.
- 3) If the number of data read by the GET command exceeds the stored data, DA ERROR will occur due to insufficient data.


Example

GET A: Reads data into variable A.
GET “TEST” A,B,C: Reads data from “TEST” file into variables A, B and C.

```
P0
10 A=1:B=2:C=3
20 PUT "TEST" A,B,C
30 END

P1
10 GET "TEST" X,Y,Z
20 PRINT X,Y,Z
30 END
```

- * Stores data of variables A, B and C on a cassette tape with P0. The data is then read into variables X, Y and Z with P1 and is then displayed.

<p>GOSUB { Line number PROG program area number }</p> <p>RETURN</p>	
---------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------

**Function:**

GOSUB: Jumps to a subroutine of a specified line number or program area.

RETURN: Returns from the subroutine to the statement immediately after the GOSUB statement.

Parameters:

Destination line number: $1 \leq \text{line number} < 10000$

Program area number: $0 \leq \text{number} < 10$

Explanation:

- 1) A subroutine can be referenced inside another subroutine and up to 12 nestings are possible.
 - 2) A UL ERROR will occur if the destination line number does not exist or there is no program in the specified program area.
 - 3) Although there may be multiple RETURN commands in a subroutine for one GOSUB command, a minimum of 1 must always exist.
 - 4) If the RETURN command is executed before the GOSUB command, a GS ERROR will occur.
- * Numerical expressions can be used for the line number and the program area number for destination.

Example

```
GOSUB 1000
GOSUB N
GOSUB PROG 9
GOSUB PROG P
```

```

10 PRINT "START"
20 GOSUB 100
30 PRINT "END"
40 END
100 PRINT "LINE 100"
"
110 RETURN

```

<b style="font-size: 2em;">GOTO { Line number PROG program area number }	P
---------------------------------------------------------------------------------	---------------------------------------------------------------------------------------

Function:

Jumps unconditionally to the specified destination.

Parameters:

Destination line number: $1 \leq \text{line number} < 10000$

Program area number: $0 \leq \text{number} < 10$

Explanation:

A UL ERROR will occur if the destination line number does not exist or if there is no program in the specified program area.

* Numerical expressions can be used for the destination line number and the program area number.

```

GOTO 150
GOTO Z
GOTO S*10
GOTO PROG 5
GOTO PROG R

```

Example

<pre> 10 INPUT "PUSH 1-3" " : A 20 IF A<1 THEN 10 30 IF A>3 THEN 10 40 GOTO A*100 50 END 100 PRINT "LINE 100" " : GOTO 10 </pre>	<pre> 200 PRINT "LINE 200" " : GOTO 10 300 PRINT "LINE 300" " : GOTO 10 </pre>
----------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

INPUT [“message statement” { ; }]

P

variable name [, [“message statement” { ; }] variable name]*

Function:




Inputs data from the keyboard into a variable.

Parameters:



Message statement: A character expression starting with a character constant

Variable name: All variables can be used.

Explanation:

- 1) If there is a message, this message will be displayed. If there is no message, only “?” will be displayed.
- 2) If a “;” is attached after a message, a “?” will be displayed but a “?” will not be displayed if “,” is attached instead.
- 3) **ENTER** and  will have the same effect if pressed while executing an INPUT statement.
- 4) If there are multiple variables in an INPUT statement, the data are entered one at a time with the **ENTER** or  key.
- 5) If  or **ENTER** is pressed without entering a data, null will be assigned to the variable (character string of 0 length).
- 6) TM ERROR will be displayed and a new entry requested if a character or null is assigned to a numerical variable.
- 7) Numerical expressions (A * 2, etc.) can be used to input numerical values.
- 8) Character expressions starting with a character constant can also be used for an input message.

Example: INPUT “MOUNT” + A\$; N

- 9) Up to 23 characters, including the “?”, can be used in the input message.
- 10) Inputs will be incorrect if the INPUT statement message is displayed with shift or with a 24th character (including “?” only). If this occurs such as during execution of program trace, press   (an FC ERROR is displayed) and correct the message to within 23 characters.

Example

<pre> 1) 10 INPUT A 20 B=2*A 30 PRINT B </pre>	<pre> 2) 10 INPUT "H=";H,"N =" ;N 20 S=N*H/2 30 PRINT S </pre>
------------------------------------------------	-------------------------------------------------------------------

LET

<pre> numerical variable = numerical expression character variable = character expression </pre>

Ⓐ

Function:

Assigns the value of the expression on the right to the variable on the left.

Explanation:

- 1) The LET command may be omitted. (Assignment statements are normally used with this command omitted.)
- 2) Numerical expressions are assigned to numerical variables, and character expressions (character strings) to character variables. TM ERROR will occur if this correspondence is not correct.

Example

```

10 LET A=123
20 B=456
30 LET C$="CASIO"
40 D$="FX-750P"
50 PRINT A:B
60 PRINT C$:D$

```

LOCATE position

└ numerical expression

(P)

Function:

Specifies the cursor position.

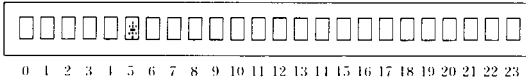
Parameter:

Position: Horizontal position on the display screen.
 $0 \leq \text{numerical expression} < 24$

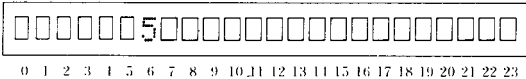
Explanation:

- 1) Specifies the cursor position and displays from that position if there is a PRINT statement.
- 2) A question mark “?” or a message will be displayed at the specified position if there is an INPUT statement after the LOCATE statement.
- 3) As shown below, position specifications will be from 0 on the left end to 23 on the right end.

In the case of **LOCATE 5 :PRINT “*”**



In the case of **PRINT 5 :LOCATE 5**



└ For positive numerical value display, one space is left open in the first display location as the “+” sign is omitted.

```

Example  10 X=11:CLS          50 IF K$="4" THEN
          20 LOCATE X:PRINT  BEEP 1:X=X-1:IF
          " *";              X<0 THEN X=23
          30 LOCATE X:PRINT  60 IF K$="6" THEN
          " ";                BEEP 0:X=X+1:IF
          40 K$=INKEY$        X>23 THEN X=0
                               70 GOTO 20
    
```

* Pressing the key moves “*” to the left and pressing the key moves it to the right.

ON numerical expression **GOSUB** destination [, destination]* (P)

★ Destination { Destination line number
PROG program area number }

Function:

The GOSUB command causes a jump to a destination specified by the value of a numerical expression.

Parameter:

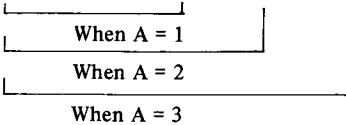
Destination line number: $1 \leq \text{line number} < 10000$

Program area number: $0 \leq \text{number} < 10$

Explanation:

- 1) The numerical expressions and destinations correspond in that a jump will be caused to the first destination if the numerical value is 1, to the second destination if the numerical value is 2, and so on.

ON A GOSUB 100, 200, 300



- 2) Any number of destinations may be written as long as it is within the range of 1 line (79 characters).
- 3) A UL ERROR will occur if the destination line number does not exist or if there is no program in the specified program area.
- 4) If the value of the numerical expression is smaller than 1, or if there is no destination, a jump will not be performed and the next statement will be executed. (see GOSUB)

Example

```

10 INPUT X
20 ON X GOSUB 200,
   300,400
30 GOTO 10
200 PRINT "SUB 200"
   :RETURN
300 PRINT "SUB 300"
   :RETURN
400 PRINT "SUB 400"
   :RETURN
  
```

ON numerical expression **GOTO** destination [, destination]* (P)

★ Destination { destination line number
 PROG program area number }

Function:

Makes an unconditional jump to the destination specified by the value of a numerical expression.

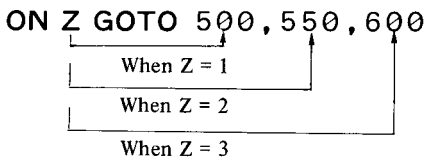
Parameters:

Destination line number: $1 \leq \text{line number} < 10000$

Program area number: $0 \leq \text{number} < 10$

Explanation:

- 1) The numerical expressions and destinations correspond in that a jump will be caused to the first destination if the numerical value is 1, to the second destination if the numerical value is 2, etc.



- 2) Any number of destinations may be written as long as it is within the range of 1 line (79 characters).
- 3) A UL ERROR will occur if the destination line number does not exist or if there is no program in the specified program area.
- 4) If the value of numerical expression is less than 1 or the destination line number is not specified, the next program line is executed without being branched.

Example

```

10 INPUT A
20 ON A GOTO 100,1
   50,200
30 PRINT "OTHER"
40 GOTO 10
100 PRINT "A=1": 60
   TO 10
150 PRINT "A=2": 60
   TO 10
200 PRINT "A=3": 60
   TO 10
    
```

PRINT [output data] [| ; | output data]*

Ⓐ

LPRINT [output data][| ; | output data]*



Function:

PRINT: Displays output data
LPRINT: Outputs data to the printer

Parameter:

Output data: Output control functions (TAB, USING), numerical expression and character expression

Explanation:

- 1) A plus or minus sign is attached before the value if the output data is a numerical expression. The plus sign, however, will appear as a space.
- 2) If the output data is a numerical expression with over 10 digits in its mantissa portion, it will be rounded off at the 11th digit and displayed. If it is $|\text{the numerical expression}| \geq 10^{10}$ or $|\text{numerical expression}| < 10^{-3}$, the exponential sign (E) and 2 exponential digits will be displayed.
- 3) If the output is a character expression, the number of characters between the quotation marks will be a maximum of 79 including line number, commands and spaces.
- 4) Continuous display of 2 or more expressions or character strings is possible by punctuating with a comma (,) or a semicolon (;). If a comma is used, program execution is caused to pause for a time specified by the WAIT command and the WAIT symbol lights. If it is desired to resume execution of the program during the wait period, simply press  or  . If the semicolon is used, the respective output data will be displayed continuously and there will be no pause in program execution. (see WAIT)
- 5) When all output data are omitted (PRINT statement only), it simply clears the display.

- 6) Display will not stop even if the PRINT statement is executed when printing in the print mode (PRTON).
- 7) The display will scroll to the left when it fills the screen (25 characters or more).
- 8) LPRINT outputs to the printer the same data as the PRINT command but does not cause program execution to pause.

Note:

TAB can only be used in the LPRINT command. It will cause an SN ERROR if used in the PRINT command.

Example

```
PRINT 5 ^ 3 ⏎  
PRINT B ⏎  
PRINT "X=" ; X ⏎  
LPRINT TAB ( 3 ) ; "ANSWER=" ; A ⏎
```

PRINT ON

PRINT OFF

A

Function:

PRINT ON: Sets the print mode.
 PRINT OFF: Releases the print mode.

Explanation:

1) Printing out of manual operations

Numerical expressions and results of manual calculations are printed out. In addition, the execution results by manual commands can be printed out.

2) Printing out by the PRINT command

The PRINT command execution in the PRINT-ON mode permits the printing of the displayed contents.

(Quite a short period of time is allowed for such a display.)

3) Printing out by the INPUT command

The message and the “?” mark are printed for facilitating the input.

4) Printing out in trace mode

The printing in the traced order of statement execution is possible, providing a convenient method of debugging logic errors through the printed record of program execution.

5) During program execution, error messages and the STOP command are printed out.

The PRTON symbol will appear when set in the PRINT mode.

An NR ERROR will occur if the PRINT mode is set without connecting a printer.

Note:

The STAT LIST command cannot be used in the PRINT ON mode. Use the STAT LLIST command instead.

Example 10 PRINT ON 20 ANGLE 0 30 FOR I=0 TO 360 STEP 15	40 PRINT "SIN";I;" =";SINI 50 NEXT I 60 PRINT OFF
-----------------------------------------------------------------------------	------------------------------------------------------------

PUT ["file name"] variable [, variable]*

(A)

Function:

Stores variable contents on a cassette tape.

Parameter:

File name: Character string of $1 \leq \text{file name} \leq 8$ characters.
(This can be omitted.) (See GET)

Example

PUT A Stores the content of variable A on a cassette tape.

PUT "DATA" X , Y , Z Stores the contents of variables X,Y and Z on a cassette tape with file name "DATA".

```
P0
10 FOR I=1 TO 10
20 PUT I
30 NEXT I
40 END

P1
10 CLEAR :DIM A(9)
20 FOR I=0 TO 9
30 GET A(I)
40 NEXT I
50 FOR I=0 TO 9
60 PRINT A(I)
70 NEXT I
80 END
```

* The program in P0 stores 10 data on a cassette tape and the program in P1 reads in 10 data from a tape to array variable A () and displays. The file name is omitted when used in this manner.

READ variable [, variable name]*

(P)

RESTORE [line number]

DATA [data] [, [data]]*

Function:

READ: Reads in the contents of DATA statement to the specified variable.

RESTORE: Specifies the position of the DATA statement read with the READ statement.

DATA: Stores data

Parameters:

Variable name: All variables are usable.

Line number: Numerical expression. $1 \leq \text{line number} < 10000$
(fractions will be discarded)

Data: Numerical value or character string and character constant.

Explanation:

- 1) The READ statement assigns data in a currently specified DATA statement in successive order to specified variables.
- 2) Data of a DATA statement is read in ascending order of line numbers. Data in the same DATA statement is read in order from the beginning of the statement.
- 3) If the number of data in the DATA statement is less than the number of variables read in with the READ statement, a DA ERROR will occur.
- 4) Omitting a line number of the RESTORE statement causes subsequent READ statements to be executed from the beginning of the DATA statement that appears first in the program.
- 5) If a line number is specified with a RESTORE statement, subsequent READ statements to be executed are read from the DATA statement following the specified line.

- 6) UL ERROR occurs if the line number specified with the RESTORE statement does not exist or, and a DA ERROR occurs if there is no DATA statement after the specified line number.
- 7) Multiple data can be written in a DATA statement by punctuating with a comma.
- 8) Nothing happens when a DATA statement only is executed.
- 9) When the data itself includes commas or a space at the beginning, enclose the data with double quotation marks.
- 10) If data is omitted in a DATA statement, the character strings will be considered 0 length (null string).
 Example: DATA , , will have the same meaning as DATA " " , " " , " " .
- 11) A DATA statement may be written in any place of the program.

Note:

If a line number is specified by a RESTORE statement, data will be read from the DATA statement in the previous program area even if this program execution has been branched with the GOTO or GOSUB statement. Insert a RESTORE statement specification at the beginning of the destination program in the case of preventing data read-in from the previous program area.

```

P0
 10 CLEAR
 20 RESTORE 100
 30 GOTO PROG 1
100 DATA1,2,3
P1
 10 READ A,B,C
 20 PRINT A;B;C
 30 END
 40 DATA4,5,6
    
```

* The result here will be 1,2,3.

```

P0
 10 CLEAR
 20 RESTORE 100
 30 GOTO PROG 1
100 DATA1,2,3
P1
 5 RESTORE
 10 READ A,B,C
 20 PRINT A;B;C
 30 END
 40 DATA4,5,6
    
```

The result here will be 4,5,6.

REM [comment]

Ⓟ

Function:

Inserts comments in a program.

Parameter:

Comment: Character string

Explanation:

All characters or symbols written after **REM** will be considered comments and will not be executed. Commands will not be executed even if continued with a multistatement in the same line.

Example

```
10 REM TEST : A = 50
```

└─ This will not be executed.

```
1000 REM *SUBROUTINE
```

```
  :
```

STAT *x* data value [;frequency] (A)

STAT *x* data value, *y* data value [;frequency]

Function:

Inputs statistical data and frequency of the data.

Parameters:

<i>x</i> data value:	Numerical expression
<i>y</i> data value:	Numerical expression
Frequency:	Numerical expression

Explanation:

- 1) Input a 1-variable statistical data with the **STAT** *x* ; *n*. If ; *n* is omitted here, *n* will be considered equal to 1.
- 2) Input a 2-variable statistical data with the **STAT** *x* , *y* ; *n* statement. If ; *n* is omitted here, *n* will be considered equal to 1.
- 3) The variable contents must always be defined before executing this command if a variable is used for the value or frequency of each data.

Example

```

10 STAT CLEAR
20 INPUT "DATA NO."
   ;N
30 FOR I=1 TO N
40 INPUT X,Y
50 STAT LGT(X),Y
60 NEXT I
70 STAT LIST

```

* Performs logarithmic regression calculation using a statistical function.

STAT CLEAR

Ⓐ

Function:

Performs initialization of basic statistics.

Explanation:

- 1) Clears CNT, SUMX, SUMY, SUMXY, SUMX2 and SUMY2 to 0.
- 2) Initializes the basic statistics and prepares for next calculation. Always execute this command when performing new statistical calculations. (See STAT.)

STOP

Ⓟ

Function:

Stops execution of a program temporarily.

Explanation:

- 1) The STOP command causes a stop message to be displayed and the STOP symbol to light.
- 2) If the following operations are performed when stopped, restart of execution will be possible with CONT.
 - Manual calculations
Example: $A * 3$
 - Manual assignments to variables
Example: $X = 123$

- Manual check of variable contents.

Example: P

- Execution of the following commands.
ANGLE, BEEP, CLEAR, CLS, DIM, ERASE,
PRINT, LPRINT, PRINT ON, PRINT OFF,
STAT, STAT CLEAR, TRON, TROFF, WAIT
- 3) If an error is caused when stopped, restart of execution will not be possible with CONT.
 - 4) If the following operations are performed when stopped, always press the key and execute.
 - Execution of manual commands (such as RUN, LOAD, LIST, etc. excluding CONT.)
 - Execution of the PUT and GET commands
 - Execution of a program in program areas to .

Example

```

10 CLEAR :DIM D$(6
    ,2)
20 FOR I=0 TO 6
30 READ D$(I,1)
40 NEXT I
50 STOP
60 FOR I=1 TO 6
70 PRINT D$(I,1)::
    INPUT D$(I,2)
80 NEXT I
90 DATA SUNDAY,MON
    DAY,TUESDAY,MED
    NESDAY,THURSDAY
    ,FRIDAY,SATURDA
    Y
    
```

- * Since execution of a program will stop after reading in data from the DATA statement, press CONT to restart execution.

TRON

Ⓐ

TROFF**Function:**

TRON: Specifies the trace mode.

TROFF: Releases the trace mode.

Explanation:

If the trace mode is specified, the program area number (0 – 9) and the line number currently being executed will be displayed in successive order as shown below.

```
[0:10] [0:20] [0:30]
```

The display scrolls to the left when a line reaches the right end of the screen.

Example

TRON ↵

TROFF ↵

WAIT numerical expression

P

Function:





Specifies a pause time during execution of a PRINT statement.

Parameter:

Numerical expression: $0 \leq \text{numerical expression} < 999$

The mode will be the same as without WAIT specification if the value is over 999.

Explanation:

- 1) The WAIT symbol will appear during the pause after execution of the PRINT statement.
- 2) Program execution can be resumed by pressing  or  during the pause.
- 3) If the numerical expression is over 999, program execution will not comply with the WAIT statement if there is a semicolon at the end of the PRINT statement. If it is a comma, program execution will remain stopped until the  or  key is pressed.
- 4) The set time will be effective from execution of the WAIT command to termination of a program. This time will be automatically released after termination of the program.
- 5) Input WAIT 999 to release the set time in a program.
- 6) The stopping time is about 0.05 sec. $\times n$ (set number).

Example

```
10 WAIT 20
20 PRINT "1SECOND"
30 WAIT 400
40 PRINT "20SECOND"
"
50 WAIT 999
60 PRINT "STOP"
70 END
```


5-3 Character Functions

ASC ({ "character string" }
character variable)

Ⓕ

Function:

Gives the ASCII code of a specified character.

Explanation:

Gives the ASCII code of the first character if the content of a character string or a character variable has 2 or more characters. (See CHARACTER CODE TABLE on page 325.)

Example

```
10 INPUT A$  
20 X=ASC(A$)  
30 PRINT X
```

CHR\$ ($\frac{\text{ASCII code}}{\text{numerical expression}}$)

Ⓕ

Function:

Gives the character of a specified ASCII code.

Parameter:

ASCII code: Numerical expression within the range of
 $0 \leq \text{numerical expression} < 256$

Explanation:

A variable may also be used as a parameter. Any fractions contained in the value of the numerical expression will be discarded. A null will be displayed if the specified character is not in the CHARACTER CODE TABLE.

Example

```
10 PRINT CHR$(34);  
"GOOD";CHR$(34)
```

Display

"GOOD"

VAL ("character string" character variable)	F
---------------------------------------------------------	---------------------------------------------------------------------------------------

Function:

Converts a character string into a numerical value.

Explanation:

- 1) If a character string starts with a character other than the numeral (including symbol, decimal point, hexadecimal notation, exponential sign E), a "0" is provided.
- 2) If characters other than numerals are used in the middle of a character string, the character and after will be ignored.
- 3) If there is a space at the beginning of a character string, this will be ignored.
- 4) An SN ERROR will occur if 3 or more numerals follow the exponent sign "E".

Example

VAL(" 1 2 3 ") ENTER → 1 2 3
 VAL("&HFF") ENTER → 2 5 5

STR\$ (numerical expression)

F

Function:

Converts a numerical value into numerals as a character string.

Parameter:

Numerical expression: Numerical value, calculation expression, numerical variable, numerical array variable.

Explanation:

A space will be provided if a numerical value is positive and a “-” sign will be provided if the numerical value is negative.

Example

```
10 A=10/3
20 A1$=STR$(A)
30 PRINT A1$
```

DMS\$ (numerical expression)

F

Function:

Converts a decimal to a sexagesimally expressed character string.

Parameter:

Argument: Numerical expression within the range of
 $|\text{argument}| < 10^{100}$

Explanation:

1) The input range is $|\text{argument}| < 10^{100}$ but the range of conversion to units of degrees, minutes and seconds (hours, minutes, seconds) is actually $|\text{numerical expression}| < 1E6$. Numerical values outside this conversion range are used in original form as character strings.

Example

- Manual

DMS\$ (1 . 2 3 4) 1° 14' 2.4

- Program

```
10 T1$=DMS$(1.234)
20 PRINT T1$
```

HEX\$ (argument)

F

Function:

Converts a decimal to a hexadecimally expressed 4 digit character string.

Parameter:

Argument: Numerical expression within the range of
 $-32769 < \text{argument} < 65536$.

Explanation:

- 1) Expresses a negative number as a complement of 2.
- 2) Numerical values 32768 and more are converted by subtracting 65535.

Example

```
10 A=127
20 PRINT HEX$(A)
```

LEFT\$ ({ $\left. \begin{array}{l} \text{character expression} \\ \text{"character string"} \\ \text{character variable} \end{array} \right\}$, $\frac{\text{number of characters}}{\text{numerical expression}}$) [Ⓕ]

Function:

Fetches the specified number of characters from the left side of the character string.

Parameter:

Number of characters: Numerical expression. Can also use variables.
 $0 \leq \text{number of characters} < 256$

Explanation:

If the number of characters to be fetched is specified as "0", null will be displayed. Also, if the specified number exceeds the number of existing characters, the assigned character string itself will be fetched.

Example

```
10 A$="ABCDEF"
20 PRINT LEFT$(A$,
4)
```

RIGHT\$ ({ $\left. \begin{array}{l} \text{character expression} \\ \text{"character string"} \\ \text{character variable} \end{array} \right\}$, $\frac{\text{number of characters}}{\text{numerical expression}}$) [ⓕ]

Function:

Fetches a specified number of characters from the right side of a character string.

Parameter:

Number of characters: Numerical expression. Can use variables.
 $0 \leq \text{number of characters} < 256$

Explanation:

If the number of characters to be fetched is specified as "0", null will be displayed. Also, if the specified number exceeds the number of existing characters, the assigned character string itself will be fetched.

Example

```
10 INPUT B$
20 PRINT RIGHT$(B$
   ,2)
30 GOTO 10
```

MID\$	({ character expression "character string" character variable } ,	<u>position</u> numerical expression) (F)
		(<u>number of characters</u> numerical expression)	

Function:

Fetches the specified number of characters from a specified position in a character string.

Parameters:

- Position: Numerical expression. Specifies the character position in a character string as counted from the left. Can also use variables.
 $1 \leq \text{position} < 256$
- Number of characters: Numerical expression. Can also use variables.
 $0 \leq \text{number of characters} < 256$

Explanation:

- 1) All characters after the specified position will be fetched if specification of the number to be fetched is omitted.
- 2) If the specified number of characters is "0", null will be fetched.
- 3) If the specified number of characters is greater than the number of characters after the specified position, all remaining characters will be fetched.
- 4) If the specified position is greater than a character string, null will be fetched.
- 5) Any fractions in a parameter will be discarded.

Example

```
10 A$="ABCDEF"
20 INPUT M,N
30 PRINT MID$(A$,M
    ,N)
```


LEN ({ character expression "character string" character variable })	<div style="border: 1px solid black; border-radius: 50%; width: 20px; height: 20px; display: flex; align-items: center; justify-content: center;">F</div>
-------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------

Function:

Provides the length (number of characters) of a character string.

Parameter:

Character expression: $0 \leq \text{number of characters} \leq 79$

Explanation:

The length of the character expression, etc. provided must be within the range of 0 ~ 79 characters.

Example


```
10 A$="BASIC"
20 PRINT LEN(A$)
```

INKEY\$	<div style="border: 1px solid black; border-radius: 50%; width: 20px; height: 20px; display: flex; align-items: center; justify-content: center;">F</div>
----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------

Function:

Inputs 1 character from the keyboard into a character variable.

Explanation:

- 1) Key input of numerals, alphabetical characters or symbols is possible with no need to press the  key.
- 2) Neither the "?" nor the cursor will be displayed and there will also be no input wait state. Nothing will be input (null state) if the key is not pressed and the following command will be executed. INKEY\$ is generally used in combination with the IF statement to provide an input wait state.

Example

```
10 A$=INKEY$
20 IF A$="" THEN 1
   0
30 BEEP
40 PRINT A$
```

- * When a key is pressed, the input will be confirmed with a beep sound and will be displayed.

5-4 Control Functions

TAB (numerical expression)

Ⓐ

Function:

Prints a specified number of spaces.

Parameter:

Numerical expression: $0 \leq \text{numerical expression} \leq 80$ (fractions discarded)

Explanation:

1) If the left side of the current printing position is specified, the TAB of the next line will be set after a line change. When making continuous printouts of characters punctuated with a semicolon, the TAB cannot be used after a line change.

* The TAB function can only be used in an LPRINT statement. An SN ERROR will occur if used in a PRINT statement.

Example

```
10 X$="BASIC"  
20 LPRINT TAB(10);  
   X$
```

USING "format character string"

Function:

Specifies the output format of the PRINT statement and LPRINT statement.

Parameter:

List of format character strings

- # ————— For numerical values. Can use within 13 digits before the decimal point and within 9 digits after the decimal point but the total must be within 13 digits including decimal point.
- ————— Decimal point position
- ^ ————— Exponent specification placed at the end.
- & ————— For characters. Cannot be used together with those for numerical values.

Explanation:

1) Numerical values will be output with right justification and characters with left justification.

2) Numerical values

If the fractions are longer than the specified value, it will be rounded off within the specified value.

If the integers are longer than the specified value (i.e., in such a case that a numerical value cannot be expressed in terms of specified format), a % sign will be added at the left and the output will not comply with the format.

A minus sign will also be counted as a digit.

3) Characters

If an output character string is longer than the specified length, the excess characters will be discarded.

If the output character string is less than the specified number of characters, spaces will be inserted to make up the difference.

Example

```

10 A$="ABCDE"           50 PRINT USING"###
20 B=12.35              .#";B;"=DATA"
30 PRINT USING"###     60 PRINT USING"###
   ";A$;"=NAME"         .###";B;"=DATA"
40 PRINT USING"###
   ###";A$;"=NAME
   "

```

Operation

RUN ↵

ABC=NA

↵

ABCDE =NAME

↵

12.4=DATA

↵

12.350=DATA

5-5 Numerical Functions

SIN argument

(F)

COS argument

TAN argument

Function: **SIN:** Calculates the sine of x , $\text{Sin } x$.
 COS: Calculates the cosine of x , $\text{Cos } x$.
 TAN: Calculates the tangent of x , $\text{Tan } x$.

Parameter:

Argument: Numerical expression. Match to selected angle unit. (see ANGLE)
 $-5400^\circ < \text{argument} < 5400^\circ$ (degree)
 $-30\pi < \text{argument} < 30\pi$ (radian)
 $-6000 < \text{argument} < 6000$ (grade)

Explanation:

- 1) Variables or numerical expressions can be used as the argument in addition to a real number.
- 2) The argument need not be enclosed in parenthesis if it is a single real number or variable. If the calculation result is the argument, however, a parenthesis will be required.

$\text{SIN } X + Y$ Add Y to the value of $\text{SIN } X$.

$\text{SIN } (X + Y)$ Calculates the sine of the sum of X and Y .

* In regard to **TAN**, however, the input range excludes

$|\text{argument}| = (2n - 1) * 1 \text{ right angle}$.

(1 right angle = $90^\circ = \pi/2 \text{ rad} = 100 \text{ grad}$)

Example

```
10 INPUT A
20 PRINT SINA
```

ASN argument

ACS argument

ATN argument

Ⓕ

Function:

These are inverse trigonometric functions to obtain the angle when a trigonometric function value is given.

ASN: Calculates the arcsine, $\text{Sin}^{-1}x$.

ACS: Calculates the arccosine, $\text{Cos}^{-1}x$.

ATN: Calculates the arctangent, $\text{Tan}^{-1}x$.

Parameter:

Arguments: Numerical expression. ASN and ACS are within the range of $|\text{argument}| \leq 1$
 ATN is within the range of $|\text{argument}| < 10^{100}$

Explanation:

The output ranges are as follows.

$$-90^\circ \leq \text{ASN}X \leq 90^\circ$$

$$0^\circ \leq \text{ACS}X \leq 180^\circ$$

$$-90^\circ \leq \text{ATN}X \leq 90^\circ$$

Example

```
10 INPUT X
20 PRINT ASNX
```

HYP SIN argument

Ⓕ

HYP COS argument**HYP TAN** argument**Function:**

These are functions to obtain the hyperbolic functions.

$$\text{HYP SIN} : \sinh x = (e^x - e^{-x})/2$$

$$\text{HYP COS} : \cosh x = (e^x + e^{-x})/2$$

$$\text{HYP TAN} : \tanh x = (e^x - e^{-x})/(e^x + e^{-x})$$

Parameter:

Argument: Numerical expression.

HYP SIN and HYP COS are within the range of $|\text{argument}| \leq 230$. HYP TAN is within the range of $|\text{argument}| < 10^{100}$.

HYPASN argument (F)

HYPACS argument

HYPATN argument

Function:

These are functions to obtain inverse hyperbolic functions.

$$\text{HYP ASN} : \sinh^{-1} x = \log(x + \sqrt{x^2 + 1})$$

$$\text{HYP ACS} : \cosh^{-1} x = \log(x + \sqrt{x^2 - 1})$$

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

Parameter:

Argument: Numerical expression.

HYP ASN is within the range of $|\text{argument}| < 5 \times 10^{99}$. HYP ACS is within the range of $1 < \text{argument} < 5 \times 10^{99}$. HYP ATN is within the range of $|\text{argument}| < 1$.

SQR argument (F)

Function:

Obtains the square root of an argument.

Parameter:

Argument: Numerical expression. Argument ≥ 0

Example

```
10 A=256
20 B=SQRA
30 PRINT B
```


LGT argument ⓕ

LOG argument

Function:

LGT: Calculates the common logarithm of x , $\log_{10}x$.

LOG: Calculates the natural logarithm of x , $\log_e x$.

Parameter:

Argument: Numerical expression. Argument > 0 .

* Accuracy will be ± 1 at the 8th digit when the value of x is $1 - 10^{-6} < x < 1 + 10^{-6}$.

Example

```
10 A=5
20 PRINT LGTA
```

EXP argument ⓕ

Function:

Calculates the value of the exponent function of an argument (argument = x : e^x)

Parameter:

Argument: Numerical expression.

$-10^{100} \leq \text{argument} \leq 230.2585092$

Example

```
10 Y=EXP2.7
20 PRINT Y
```

ABS argument

ⓕ

Function:

Obtains the absolute value of an argument.

Parameter

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

Example

```
10 A=ABS-68
20 PRINT A
```

* Obtains the absolute value of -68.

INT argument

ⓕ

Function:

Gives the maximum integer that does not exceed an argument. (When the argument is positive, the fractions will be discarded.)

Parameter:

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

Example

- Manual calculation

```
INT 4.4 [ENTER] → 4
```

```
INT -2.5 [ENTER] → -3
```

- Program calculation

```
10 A=2468.135
20 PRINT INTA
```

FRAC argument

Ⓕ

Function:

Gives the fractions of an argument. (Discards the integers.)

Parameter:

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

Example

```
10 A=3.141592
20 B=FRAC A
30 PRINT B
```

SGN argument

Ⓕ

Function:

Obtains a value corresponding to the sign of the argument.

Parameter:

Argument: Numerical expression.

Explanation:

1 will be provided if the argument is greater than 0 (plus).

0 will be provided if the argument is 0.

-1 will be provided if the argument is smaller than 0 (minus).

Example

```
10 INPUT X
20 Y=SGNX
30 PRINT Y
```

ROUND (numerical expression 1, numerical expression 2) (F)

[, { 0 }] [, { + }]

Function:

Obtains the value of numerical expression 1 which is rounded by a specified method at a digit specified by numerical expression 2.

Parameter:

Numerical expression 1: |Numerical expression 1| < 10¹⁰⁰

Numerical expression 2: |Numerical expression 2| < 10¹⁰⁰

Explanation:

1) ROUND (x, y)

x is rounded at the 10^y position.

2) ROUND (x, y, 0)

x is rounded at the 10^y position.

ROUND (x, y, 1)

x is rounded at the significant digit y.

3) ROUND (x, y, { 0 } , sign)

When the sign is "+", it will be a rounded up value. (See above 2.)

When the sign is "-", it will be a discarded value. (See above 2.)

Note:

All values except 0 of a specified digit will be rounded up during a rounding up operation.

Example

The first digit after the decimal point has been specified in the following.

12.30 → 13

12.03 → 12

Example

```

10 A=12345.67
20 X=ROUND(A,2)
30 Y=ROUND(A,2,1)
40 Z=ROUND(A,2,0,+
)
50 PRINT X;Y;Z

```

$$\text{DEG} \left(\underbrace{\quad}_{\text{degree}} \left[\underbrace{\quad}_{\text{numerical expression}} \left[\underbrace{\quad}_{\text{minute}} \left[\underbrace{\quad}_{\text{numerical expression}} \left[\underbrace{\quad}_{\text{second}} \left[\underbrace{\quad}_{\text{numerical expression}} \right] \right] \right] \right] \right] \right] \right) \text{ (F)}$$

Function:

Converts sexagesimal numbers to decimal numbers.

Parameter:

Degree, minute, second: Numerical expression.
 $|\text{Numerical expression}| < 10^{100}$

Explanation:

In the case of a negative angle, a minus sign “-” will be added to each degree, minute and second.

Example

- Manual calculation.

DEG (12 , 34 , 56)

12.58222222

- Program calculation

```
10 INPUT X,Y,Z
20 PRINT DEG(X,Y,Z)
  )
30 END
```

PI

Ⓣ

Function:

Provides the value of π (ratio of the circumference of a circle to its diameter – 3,1415926536).

Note:

The display will become 3.141592654.

Example

RND

Ⓣ

Function:

Generates a random number of within 10 digits in which $0 < \text{random number} < 1$.

Example

```
10 A=RND
20 PRINT A
```

5-6 Statistical Functions

CNT

Ⓡ

Function:

Obtains the number (n) of statistically processed data.

SUMX

Ⓡ

SUMY

Function:

Obtains sum total of the data.

SUMX : $\sum x$

SUMY : $\sum y$

SUMXY

Ⓕ

Function:

Obtains the sum of the products of the x and y data.

$$\text{SUMXY} : \sum xy$$

SUMX 2

Ⓕ

SUMY 2

Function:

Obtains the sum of the squares of a specified data.

$$\text{SUMX2} : \sum x^2$$

$$\text{SUMY2} : \sum y^2$$

SDX

ⓕ

SDY**Function:**

Obtains the sample standard deviation.

$$\text{SDX} : \sqrt{\frac{n \cdot \sum x^2 - (\sum x)^2}{n(n-1)}} \quad (x \delta_{n-1})$$

$$\text{SDY} : \sqrt{\frac{n \cdot \sum y^2 - (\sum y)^2}{n(n-1)}} \quad (y \delta_{n-1})$$

*(n is the number of data)***SDXN**

ⓕ

SDYN**Function:**

Obtains the population standard deviation.

$$\text{SDXN} : \sqrt{\frac{n \cdot \sum x^2 - (\sum x)^2}{n^2}} \quad (x \delta_n)$$

$$\text{SDYN} : \sqrt{\frac{n \cdot \sum y^2 - (\sum y)^2}{n^2}} \quad (y \delta_n)$$

(n is the number of data)

LRA

Ⓡ

Function:

Obtains linear regression constant term.

$$\text{LRA} : \frac{\sum y - \text{LRB} \cdot \sum x}{n} \quad (n \text{ is the number of data})$$

Example

10 Y=X*LRB+LRA

LRB

Ⓡ

Function

Obtains the linear regression coefficient.

$$\text{LRB} : \frac{n \cdot \sum xy - \sum x \cdot \sum y}{n \sum x^2 - (\sum x)^2} \quad (n \text{ is the number of data})$$

COR

ⓕ

Function

Obtains the correlation coefficient (r).

$$\text{COR} : \frac{n \cdot \sum xy - \sum x \cdot \sum y}{\sqrt{\{n \cdot \sum x^2 - (\sum x)^2\} \{n \cdot \sum y^2 - (\sum y)^2\}}}$$

(n is the number of data)

EOX (argument)

ⓕ

Function

Obtains an estimated value of x in relation to y .

$$\text{EOX} : \frac{y - \text{LRA}}{\text{LRB}} \quad (\hat{x})$$

Parameter

Argument: Numerical expression

EOY (argument)

ⓕ

Function

Obtains an estimated value of y in relation to x .

$$\text{EOY} : \text{LRA} + x \cdot \text{LRB} \quad (\hat{y})$$

Parameter

Argument: Numerical expression

5-7 Others

■ Hexadecimal Conversion

Converts 1 to 4 digit hexadecimal numbers following &H to decimal numbers.

Example

- Manual calculation

&H2 5 A 0

9632

- Program calculation

```
10 A=&H6F00  
20 PRINT A
```


CHAPTER

PRACTICAL LIBRARY

6

6-1 Achievement Processing 2 RC-4 RAM cards required.

This is an achievement processing program to reduce time and labor for the teachers. This program can be processed easily anywhere.

Features and Processing Details

1. Achievements of up to 61 students can be processed and up to 10 different subjects can be freely set.

This program computes the total score and deviation value for each student.

2. Sorting (ranking) by individual subjects and by overall subjects can be performed.

3. Computes deviation value by individual subjects and by overall subjects.

4. Attendance number and subject names are initially registered and processing is carried out by entering score only.

(The attendance number is used since entering the student names is quite troublesome.)


Example:

3 subjects and 5 students

Score data

Student's attendance No.	Subject		
	English	Mathematics	Science
1	75	58	80
2	63	70	78
3	81	72	61
4	48	51	80
5	53	63	40

Note 1.

When executing a program for the first time, enter CLEAR  to erase all data in the RAM card.

Note 2.

The RAM card in slot 0 is the program card and the RAM card in slot 1 is the data card. The program card can also be used by other teachers by simply replacing the data card. Use the RC-4 RAM cards (4K bytes) for both of the program and data.

OperationRUN

ACHIEVEMENT PROCESSING

NUMBER OF SUBJECTS?

Enter the number of subjects

3

NUMBER OF STUDENTS?

There are 5 students. (Input possible up to a maximum of 61 students.)

5

SUBJECT INPUT

This is for input of subject names

SUBJECT # 1=?

Enter the characters of English, mathematics and science in successive order.
(Input of subject names with up to 20 characters possible.)ENGLISH

SUBJECT # 2=?

MATH.

SUBJECT # 3=?

SCIENCE

1IN 2ED 3CAL 4SORT 5END

The processing menu will be displayed after completing input of the subjects.

- 1 IN Select this when entering the scores.
- 2 ED Select this when checking and correcting the scores entered.
- 3 CAL Select this when confirming the scores and calculating the average, etc.
- 4 SORT Select this when ranking by specified subjects or by overall subjects.
- 5 END Select this when terminating the program or when changing the data card.

1

ENGLISH ATTEND. 1=?

This is a request for score input on English. Enter 75, 63, 81, 48 and 53 in order of attendance number.

75 ↵

ENGLISH ATTEND. 2=?

⋮

⋮

53 ↵

MATH. ATTEND. 1=?

This is a request for score input on mathematics. Enter 58, 70, 72, 51 and 63 in order of the attendance number.

58 ↵

MATH. ATTEND. 2=?

⋮

⋮

63 ↵

SCIENCE ATTEND. 1=?

Also enter 80, 78, 61, 80 and 40 in order of the attendance number for the score of science. When the final score is entered, the processing menu will be displayed again.

80 ↵

SCIENCE ATTEND. 2=?

⋮

⋮

40 ↵

1IN 2ED 3CAL 4SORT 5END

Although calculations can be performed immediately, we shall check the scores entered since an input error may be caused when scores of the large number of students are processed.

2

1.LIST 2.EDIT 3.END

Select 1 to check scores and select 2 to make corrections. If an input error is discovered while checking the scores, jot down the subject number and the attendance number.

1

SUBJECT NUMBER 1 ENGLISH

↵

ATTEND. 1= 75

↵

ATTEND. 2= 63

⋮

Press the ↵ key to check
in successive order.

⋮

Scores will be displayed in the order of
English, mathematics and science.

↵

SUBJECT NUMBER 2 MATH.

⋮

⋮

↵

ATTEND. 5= 40

↵

1.LIST 2.EDIT 3.END

After completing the check, select 2 if there was an input error, and select 3 to return to the processing menu if there was no error. For practice sakes, we shall change a score of 51 in mathematics to 52 for a student with an attendance No. of 4.

2

SUBJECT NUMBER=?

2 ↵

ATTEND. NO.=?

4 ↵

MATH. ATTEND. 4=51

↵

MODIFY (Y/N)?

Change to 52

Y ↵

MATH. ATTEND. 4=?

52 ↵

1.LIST 2.EDIT 3.END

3

1IN 2ED 3CAL 4SORT 5END

Select 3 and calculate after completing checks and corrections.

3

⋮

WAIT!!...

⋮ Calculation will take about 2 minutes for
61 students.

ENGLISH

AVERAGE 64.000

↵

The class average for English is displayed.

↵

DEVIATION 14.035

The display shows that the standard deviation is 14.035. The scores and deviation values will then be displayed in order of attendance number each time the ↵ key is pressed.

↵

ATTEND. 1= 75 57.84

⋮ Attendance No. Score Deviation value
⋮

⋮

↵

MATH.

⋮ The score and the deviation value in
mathematics will be displayed.

⋮

↵

SCIENCE

⋮ The score and the deviation value in
science will be displayed.

⋮

After completing display of each subject, the average and the individual scores and deviation value will next be displayed with regard to overall subjects.

↵

TOTAL

↵

AVERAGE 194.800

↵

DEVIATION 25.917

⋮

⋮

⋮

⋮

↵

ATTEND. 5=156 35.03

↵

1IN 2ED 3CAL 4SORT 5END

We shall next sort the data.

4

SORT SUBJECT NO.=?

If we input the number corresponding to a subject to be sorted, the individual scores and deviation values will be displayed in sequential order from the high values. However, sorting will be performed here in relation to all subjects.

Enter 11 for sorting all subjects. (This is because 11 is registered for sorting of all subjects.)

11 ↵

SORTING...

⋮

The computer is sorting the data.

↵

TOTAL

↵

NO. 1 3 214 57.41

↵

NO. 2 1 213 57.02

⋮

Ranking	Attendance No.	Score	Deviation value
NO. 5	5	156	35.03

↵

NO. 5 5 156 35.03


↵

1IN 2ED 3CAL 4SORT 5END

This processing completes.

5

1..CHANGE CARD 2..END

Press the  key to stop the execution of this program. The computer will be in READY mode.

If you desire to restart processing by changing the data card, select 1.

1

POWER OFF & CHANGE CARD

Variables

Variable name	Variable content
A! ()	Scores
B\$ ()	Subject name
C! ()	Average, Standard deviation
I	Counter
J	Counter
U\$	Wait for menu input
W	Menu input judgement
E	Edit subject No.
D	Edit student No.
X	Number of subjects
Y	Number of students
K	Sort subject No.
H	Sorting

Program List

```

10 REM CHECK RAM C
ARD
20 PRINT "*ACHIEVE
MENT PROCESSING
*"
30 IF Z$="DATA OK"
THEN 130
40 INPUT "NUMBER O
F SUBJECTS":X
50 IF X<11 THEN IF
X>0 THEN 70
60 BEEP : GOTO 40
70 INPUT "NUMBER O
F STUDENTS":Y
80 IF Y<62 THEN IF
Y>0 THEN Y=Y-1
: GOTO 100
90 BEEP : GOTO 70
100 DIM A!(12,60),B
$(11)*10,C!(11,
1)
110 Z$="DATA OK":PR
INT "WAIT...";
120 PRINT
130 REM CHECK SUBJE
CT
140 IF B$(11)="TOTA
L" THEN 210
150 BEEP :PRINT "SU
BJECT INPUT"
160 B$(11)="TOTAL"
170 FOR I=1 TO X
180 PRINT "SUBJECT
#:I; "=";
190 INPUT B$(I)
200 NEXT I
210 REM SELECT JOB
220 PRINT "1IN 2ED
3CAL 4SORT 5END
";
230 V$=INKEY$:IF V$
="*" THEN 230
240 BEEP :W=VAL(V$)
:IF W=0 THEN 23
0 ELSE IF W>5 T
HEN 230
250 PRINT
260 ON W GOTO 280,3
50,610,1110,990
270 GOTO 220
280 REM INPUT DATA
290 FOR J=1 TO X
300 FOR I=0 TO Y
310 PRINT B$(J);" A
TTEND.":I+1;"="
;
320 INPUT A!(J,I)
330 NEXT I:NEXT J
340 GOTO 220
350 REM EDIT DATA
360 PRINT "1.LIST 2
.EDIT 3.END";
370 V$=INKEY$:IF V$
="*" THEN 370
380 PRINT
390 BEEP :W=VAL(V$)
:IF W=0 THEN 36
0 ELSE IF W>3 T
HEN 360
400 IF W=3 THEN 220
410 ON W GOSUB 430,
500
420 GOTO 360
430 REM LIST DATA
440 FOR J=1 TO X
450 PRINT "SUBJECT
NUMBER":J;" ":B
$(J)
460 FOR I=0 TO Y
470 PRINT "ATTEND.
":I+1;"=":A!(J,I
)
480 NEXT I:NEXT J
490 RETURN
500 REM EDIT DATA
510 INPUT "SUBJECT
NUMBER=":E:IF E
>X THEN BEEP :
GOTO 510
520 IF E<1 THEN BEE
P : GOTO 510
530 INPUT "ATTEND.
NO.=":D:IF D>Y+
1 THEN BEEP : G
OTO 530
540 IF D<1 THEN BEE
P : GOTO 530
550 PRINT B$(E);" A
TTEND.":D;"=":A
!(E,D-1)
560 INPUT "MODIFY (
Y/N)":V$:IF V$=
"Y" THEN 580
570 RETURN
580 PRINT B$(E);" A
TTEND.":D;"=";
590 INPUT A!(E,D-1)
600 RETURN
610 REM CALC DATA
620 PRINT "WAIT!!..
.";
630 FOR J=1 TO X
640 STAT CLEAR
650 FOR I=0 TO Y
660 STAT A!(J,I)
670 NEXT I
680 C!(J,0)=SUMX/CN
T:C!(J,1)=SDX
690 NEXT J
700 STAT CLEAR
710 FOR I=0 TO Y
720 A!(11,I)=0:FOR
J=1 TO X
730 A!(11,I)=A!(11,
I)+A!(J,I)
740 NEXT J

```

```

750 STAT A!(11,I)
760 NEXT I
770 C!(11,0)=SUMX/C
    NT:C!(11,1)=SDX
780 PRINT
790 FOR J=1 TO X
800 BEEP :BEEP
810 PRINT B$(J)
820 PRINT "AVERAGE
    ";;PRINT USING"
    ###.###";C!(J,0
    )
830 PRINT "DEVIATIO
    N";:PRINT USING
    "###.###";C!(J,
    1)
840 FOR I=0 TO Y
850 PRINT "ATTEND."
    ;;PRINT USING"#
    #";I+1;
860 PRINT "=";:PRIN
    T USING"###";A!
    (J,I);" ";
870 PRINT USING"###
    .###";50+10*(A!(
    J,I)-C!(J,0))/C
    !(J,1)
880 NEXT I:NEXT J
890 PRINT
900 PRINT B$(11)
910 PRINT "AVERAGE
    ";;PRINT USING"
    ###.###";C!(11,
    0)
920 PRINT "DEVIATIO
    N";:PRINT USING
    "###.###";C!(11
    ,1)
930 FOR I=0 TO Y
940 PRINT "ATTEND."
    ;;PRINT USING"#
    #";I+1;
950 PRINT "=";:PRIN
    T USING"###";A!
    (11,I);" ";
960 PRINT USING"###
    .###";50+10*(A!(
    11,I)-C!(11,0))
    /C!(11,1)
970 NEXT I
980 PRINT : GOTO 22
    0
990 REM ANOTHER CAR
    D
1000 PRINT "1..CHANG
    E CARD 2..END";
1010 V$=INKEY$:IF V$
    ="" THEN 1010
1020 BEEP :W=VAL(V$)
    :IF W=0 THEN 10
    10 ELSE IF W>2
    THEN 1010
1030 ON W GOTO 1050,
    1100
1040 GOTO 1010
1050 PRINT
1060 BEEP :BEEP :BEE
    P
1070 PRINT "POWER OF
    F & CHANGE CARD
    ";
1080 V$=INKEY$:IF V$
    ="" THEN 1080
1090 GOTO 1000
1100 END
1110 REM SORT DATA
1120 INPUT "SORT SUB
    JECT NO.=";K
1130 IF K<=X THEN IF
    K>=1 THEN 1160
1140 IF K=11 THEN 11
    60
1150 BEEP : GOTO 112
    0
1160 PRINT "SORTING.
    ..";
1170 FOR I=0 TO Y:A!
    (12,I)=I+1:A!(0
    ,I)=A!(K,I):NEX
    T I
1180 FOR I=0 TO Y-1:
    FOR J=I+1 TO Y
1190 IF A!(0,J)<A!(0
    ,I) THEN 1220
1200 H=A!(0,I):A!(0,
    I)=A!(0,J):A!(0
    ,J)=H
1210 H=A!(12,I):A!(1
    2,I)=A!(12,J):A
    !(12,J)=H
1220 NEXT J:NEXT I
1230 BEEP :BEEP :PRI
    NT :PRINT B$(K)
1240 FOR I=0 TO Y
1250 PRINT "NO.":PR
    INT USING"###";I
    +1;
1260 PRINT USING"###
    ";A!(12,I);" "
    ;
1270 PRINT USING"###
    ";A!(0,I);" ";
1280 PRINT USING"###
    .###";50+10*(A!(
    0,I)-C!(K,0))/C
    !(K,1)
1290 NEXT I
1300 GOTO 220


```


6-2 Tabulation Program

2 RC-4 RAM cards required.

Horizontal and vertical totalization forms the basis of all account books. This program, which turns a feature of the FX-750P to good account, uses a program card and data card. Just by changing data cards, the total for each card can be performed. Data cards also permit monthly addition of new data. Therefore, this program can be widely used in all fields. Its format can be chosen freely within the following range: up to 32 lines showing a person's name, product name, group name, etc. and up to 12 columns showing the months of the year.

Attention:

- This program uses program cards and data cards. Use them in pairs by inserting a program card in slot 0 and a data card in slot 1.
- Enter CLEAR  when executing this program with new data card.

Example:

Mr. A's sales in January and February are input for each product. The totals are displayed and then the data for March are added to renew the totals.

Card name		3 columns			Until Dec.
Item name	Jan. (1)	Feb. (2)	Mar. (3)	Total	
A	# 0 1	100	110	80	290
B	# 0 2	105	50	190	345
C	# 0 3	120	75	105	300
D	# 0 4	150	200	95	445
E	# 0 5	195	150	150	495
Total		670	585	620	1875

Max. 7 characters

5 lines (rows)

Permits input of up to 7 characters.

Max. 32 lines

Permits input of up to 7-digit numerical values.

Add new data here.

Total is renewed each time new data are added.

RUN

(1FORMAT 2INPUT 3END)

When the program is executed, the above menu is displayed first of all.

- 1 FORMAT Choose this for format preparation.
- 2 INPUT Choose this for data input or correction.
- 3 END Choose this when terminating the program.

Now let us start format preparation according to the example.

1

* FORMAT *

<<< NEW CARD >>>

CARD NAME ?

The card name is asked. Since the data concern Mr. A, let us input A.

A

NUMBER OF LINES (1-32) ?

The number of lines is asked. We have to decide on the number of lines to be used. Any number within the range of 1 to 32 can be chosen freely. In this case, we need 5 lines.

5

NUMBER OF COLUMNS(1-12)?

The number of columns – i.e., months – is then chosen. In this case, 3 months. If you choose a 12-month format, it is possible to enter the data from a desired month.

3

ITEM NAME 1: ?

In the item name column, the names of a person, product, group, district, etc. are input. In this case, we input # (number).

# 01 ↵	ITEM NAME 2:?
⋮	⋮
# 05 ↵	(1FORMAT 2INPUT 3END)

Having input the card name, line and column specifications, and the item name, we have completed the format preparation. Display returns to the menu. Now data are to be input.

2	* INPUT *
↵	<< A'S CARD >>
↵	INPUT (1- 3)?

Data are input for each month. In this case, we begin by inputting data for January.

1 ↵ (January is specified.)	1:#01	?
100 ↵ (Input data from #01.)	2:#02	?
105 ↵	3:#03	?

Data are then input in sequence.

195 ↵	INPUT (1- 3)?	
2 ↵ (February is specified.)	1:#01	?
110 ↵ (Input data from #01.)	2:#02	?
⋮	⋮	
150 ↵	INPUT (1- 3)?	

Data for January and February have been input, but specification of the next month is asked because the format covers March as well.

In this example, data for March are input as an addition. To stop the data input for the column (i.e., January and February), enter E (END).

E

PRINT OUT Y/N

Here the display shows whether a printer is used or not. Vertical and horizontal tabulation automatically performs an internal calculation by pressing E after entering data. Therefore, if the main frame is connected to the FA-20 and Y is pressed, the totals are printed out.

Here we press the N key.

N

RECALL (1- 3)?

This menu specifies the month for which data calculation results are to be displayed.

1-1 (Specifies January.)

A<< 1 >>

(Displays data for January in order.)

#01: 100

⋮

(Displays the column total for January.)

1TOTAL 670

(Displays the line total for January.)

#01: 100

⋮

(Displays grand total for January and February.)

G. TOTAL 1255

(Returns to the menu.)

(1FORMAT 2INPUT 3END)

Note:

When the specified months end in the middle of the period covered by the format, the line total is displayed for the specified months.

When a particular month is specified, data for that month are first displayed, followed by column total and then line total. Finally, the grand total is displayed. The grand total of all data currently stored will be displayed regardless of the month specified.

■ Addition of Data

Choose 2 on the menu.

2

↵

↵

(1FORMAT 2INPUT 3END)

* INPUT *

<< A'S CARD >>

INPUT (1- 3)?

The month for which data are to be input is asked.

Specify the month to be added. For example, in the case of a 12-month format, INPUT (1-12)? is displayed. Just specify the month for which data are to be added.

In this case, the month is March.

3 ↵

1:#01

?

Input the data for March.

80 ↵ (Input data from #01.)

2:#02

?

When inputting is completed.

⋮

E ↵

PRINT OUT Y/N

N

RECALL (1- 3)?

Entering of 3-3 ↵ causes the data and totals to be displayed, indicating the addition of the data.

Note:

It sometimes happens that after recording the card name and data on a card, data have to be updated. In such a case, press **Ⓔ** (INPUT) for data modification. When format 1 is selected, DATA CLEAR Y/N is displayed after card name display. If you want to add data, press the N key. Be careful not to press the Y key since all data on the card are erased.

■ To Take a Look at Data

Select 2 from the menu.

2

(Check of input data.)

(Check of Mr. A's card.)

E (Input is not performed.)

N (Printout is not performed.)

(1FORMAT 2INPUT 3END)
* INPUT *
<< A'S CARD >>
INPUT (1- 3)?
PRINT OUT Y/N
RECALL (1- 3)?

Specify the desired month by - . Press 1 - 1 when you wish to see data and total for January only and press 1 - 3 if you wish to see data and total for January to March.

When all work terminates, select 3 from the menu to indicate the END.

Variables

Variable name	Variable content
A ()	Data array
C\$ ()	Item name
Z\$ ()	Wait for input
F ()	Column total
G ()	Line total
E\$ ()	Card name
A0 - D4	Blanks
D5\$	END variable
P	Number of lines
U	Number of columns
D\$	Data input
V	Conversion of data into numerical values
I	Counter
J	Counter
Q	Counter start
T	Counter end
K	Counter
L\$	Wait of menu input
M	Conversion of menu into numerical values

Program list

```

P0
10 CLS :PRINT "(IF
  ORMAT 2INPUT 3E
  ND)";
20 L$=INKEY$:IF L$
  =" " THEN 20
30 CLS :M=VAL(L$)
40 IF M=0 THEN 10
  ELSE IF M>3 THE
    N 10
50 IF M=1 THEN GOT
  O PROG 1
70 IF M=2 THEN GOT
  O PROG 2
80 IF M=3 THEN PRI
  NT "*** END OF
  JOB ***":END

P1
10 PRINT "* FORMAT
  *"
20 IF E$="" THEN P
  RINT "<<< NEW C
  ARD >>>": GOTO
  70
30 PRINT "<< ";E$;
  CHR$(39);"S CAR
  D >>"
40 PRINT "DATA CLE
  AR Y/N ";
50 Z$=INKEY$:IF Z$
  =" " THEN 50 ELS
  E CLS
60 IF Z$(">Y" THEN
  GOTO PROG 0
70 CLEAR :DIM A(32
  ,11),C$(32)*7
80 A0=0:A1=0:A2=0:
  A3=0:A4=0:A5=0:
  A6=0:A7=0:A8=0:
  A9=0
90 B0=0:B1=0:B2=0:
  B3=0:B4=0:B5=0:
  B6=0:B7=0:B8=0:
  B9=0
100 C0=0:C1=0:C2=0:
  C3=0:C4=0:C5=0:
  C6=0:C7=0:C8=0:
  C9=0
110 D0=0:D1=0:D2=0:
  D3=0:D4=0
120 DIM F(32),G(12)
130 D5$="END"
140 INPUT " CARD NA
  ME ";E$
150 PRINT "NUMBER O
  F LINES (1-32)
  ";
155 INPUT P
160 PRINT "NUMBER O
  F COLUMNS(1-12)
  ";
165 INPUT U
170 CLS :FOR I=1 TO
  P
180 PRINT "ITEM NAM
  E";I;" ";:INPUT
  C$(I)
190 NEXT I
200 GOTO PROG 0

P2
10 PRINT "* INPUT
  *"
20 PRINT "<< ";E$;
  CHR$(39);"S CAR
  D >>"
30 PRINT "INPUT (1
  -";U;" )";:INPUT
  D$:N=VAL(D$)
40 IF D$="E" THEN
  100
50 IF N<1 THEN 30
  ELSE IF N>U THE
    N 30 ELSE N=N-1
60 FOR I=1 TO P
70 PRINT I;" ";C$(
  I)::LOCATE 15:I
  NPUT D$:A(I,N)=
  VAL(D$)
80 IF D$="E" THEN
  I=P
90 NEXT I: GOTO 30
100 PRINT "PRINT OU
  T Y/N ";
110 Z$=INKEY$:IF Z$
  =" " THEN 110
120 CLS :PRINT "REC
  ALL (1-";U;" )";
  :INPUT D$
130 IF D$="E" THEN
  GOTO PROG 0
140 IF Z$="Y" THEN
  PRINT ON
150 V=LEN(D$)
160 IF V<3 THEN 120
  ELSE IF V>5 TH
    EN 120
170 IF V=4 THEN Q=2
  ELSE IF V=5 TH
    EN Q=2 ELSE Q=1
180 S=VAL(LEFT$(D$,
  2)):T=VAL(RIGHT
  $(D$,Q)):IF T>U
  THEN 120
190 FOR J=5 TO T
200 CLS :PRINT E$;"
  <<";J;" >>";:O=
  50: GOSUB 500
210 FOR I=1 TO P
220 CLS :PRINT C$(I
  );";":A(I,J-1);
  : GOSUB 500

```

```
230 NEXT I:NEXT J
240 FOR J=S TO T
250 G(0)=0
260 FOR I=1 TO P
270 G(0)=G(0)+A(I,J
    -1)
280 NEXT I
290 G(J)=G(0)
300 CLS :PRINT J;"T
    OTAL";G(J);:0=2
    00: GOSUB 500
310 NEXT J
320 A1=0
330 FOR J=1 TO U
340 FOR I=1 TO P
350 A1=A1+A(I,J-1)
360 NEXT I
370 NEXT J
380 FOR J=1 TO P:F(
    0)=0
390 FOR I=1 TO T
400 F(0)=F(0)+A(J,I
    -1)
410 NEXT I
420 CLS :PRINT C$(J
    );":":F(0):: 60
    SUB 500
430 NEXT J
440 CLS :PRINT "G.
    TOTAL";A1:: GOS
    UB 500
450 CLS :PRINT OFF
460 GOTO PROG 0
470 NEXT J
480 CLS :PRINT OFF
490 GOTO PROG 0
500 IF Z$="V" THEN
    PRINT :RETURN
510 FOR K=1 TO 0:NE
    XT K:RETURN
```


6-3 Memo Card

Mr. A is a healthy person but he goes to a hospital every day. It's no wonder since he is a special salesman for X Medicine. Today, he must make the rounds of 5 hospitals. This is where the FX-750P with its RAM card comes into play. Before departing on his rounds, he carefully studies methods of approach by recalling the previous order list and the previous month's sales from the RAM card in which the data is stored.

50 memo can be used with up to 30 characters in each and these electronic memos can be handily used in a car or at the customer's site.

Example

N Hospital

Item No. (Item Name)	Quantity ordered
001	2 doz.
005	25 doz.
090	4 doz.
101	20 doz.
Previous month's sales	\$7,000

O Hospital

Item No. (Item Name)	Quantity ordered
003	5 doz.
005	30 doz.
081	2 doz.
101	15 doz.
Previous month's sales	\$8,900

RUN

```
      ** MEMO CARD **
INITIAL (Y/N)?
```

The first message is a query as to whether this is an initial data input or a continued usage of data. (Be careful when answering this query as all data entered so far will be cleared if the key is entered.)

Y

```
SELECT (A,E,D,S,L,P)?
```

- A Input and addition of data (ADD)
- E Editing of data (EDIT)
- D Deletion of data (DELETE)
- S Searching of data (SEARCH)
- L Displays data in order of input (LIST)
- P ON/OFF of printer (PRINT)

We will select A and enter the data.

A ↵

DATA # 1=?

The first data entry is the order list of the N hospital.

N-001-2D ↵

DATA # 2=?

N-005-25D ↵

DATA # 3=?

N-009-4D ↵

DATA # 4=?

N-101-20D ↵

DATA # 5=?

N-\$7,000 ↵

DATA # 6=?

Next, enter the data for the O hospital.

O-003-5D ↵

DATA # 7=?

⋮

⋮

O-\$8,900 ↵

DATA # 11=?

Enter END after all data have been entered.

END ↵

SELECT (A, E, D, S, L, P)?

We will now recall the list for the O hospital. Since this is a search operation, select S.

S ↵

SEARCH DATA=?

Since search is executed with the initial letter of the data, enter O for O hospital.

O ↵

SEARCHING...

↵	# 6=0-003-5D
↵	# 7=0-005-30D
↵	# 8=0-081=2D
↵	# 9=0-101=15D
↵	# 10=0-#8,900
↵	SELECT (A,E,D,S,L,P)?

Next, we edit the data.

E ↵	EDIT #=?
-----	----------

The display is asking what number data to correct. If you are not certain as to what number it is, check this by pressing L ↵ and displaying the data list.

Here we will change the item No. of the 6th data to 004.

6 ↵	0-003-5D
-----	----------

If the data is to be corrected, press the ↵ key.

↵	UP-TO-DATE=?
---	--------------

The computer is asking for up-to-date data input.

0-004-5D ↵	SELECT (A,E,D,S,L,P)?
------------	-----------------------

We will now delete a data.

D ↵	DELETE #=?
-----	------------

The computer is asking what number data to delete. Let's delete the 8th data.

8 ↵	DATA=0-081-2D
-----	---------------

Requesting confirmation of deletion.

↵

OK?

If deletion OK, press the Y key.

Y ↵

** WAIT **
SELECT (A, E, D, S, L, P)?

The 8th data will be deleted and the 9th data will move up to the 8th position. All data after the 9th will also move up by one position. However, data cannot be inserted in between.

(Since data search is executed with the initial letter, the necessary data can be displayed even if data input has been performed at random.)

Variables

Variable name	Variable contents
C\$	INITIAL (Y/N)
C\$	SELECT (A, E, D, S, L, P)
D\$ ()	Data
D\$ ()	Data storage variable
N	Number of data
P	Data No.
PP	Printer control variable
R	Data deletion variable
D\$ ()	Search data
I	FOR ~ NEXT loop variable
Y	Y/N discrimination variable

Program List

```

5 PRINT "    **
  MEMO CARD **"
10 INPUT "INITIAL
  (Y/N)";C$:IF C$
  ="N" THEN 30
20 CLEAR :N=50:DIM
  D$(N)*30:PP=0
30 INPUT "SELECT (
  A,E,D,S,L,P)";C
  $
40 IF C$="A" THEN
  100
50 IF C$="E" THEN
  170
60 IF C$="D" THEN
  230
70 IF C$="S" THEN
  360
80 IF C$="L" THEN
  440
90 GOTO 30
100 REM ADD
110 P=P+1:IF P<=N T
  HEN 130
120 BEEP :PRINT "SO
  RRY! NO ROOM!":
  P=P-1: GOTO 30
130 PRINT "DATA #":
  P:="":
40 INPUT D$(0):IF
  D$(0)="END" THE
  N P=P-1: GOTO 3
  0
50 D$(P)=D$(0)
60 GOTO 110
70 REM EDIT
80 INPUT "EDIT #="
  :R
90 IF R>P THEN BEE
  P : GOTO 100
00 PRINT D$(R)

210 INPUT "UP-TO-DA
  TE=";D$(R)
220 GOTO 30
230 REM DELETE
240 INPUT "DELETE #
  =";R
250 IF R>P THEN BEE
  P : GOTO 240
260 PRINT "DATA=";D
  $(R)
270 INPUT "OK";Y$
280 IF Y$<>"Y" THEN
  30
290 PRINT "    **
  WAIT **":
300 IF R=N THEN 340
310 FOR I=R+1 TO P
320 D$(I-1)=D$(I)
330 NEXT I
340 P=P-1:CLS
350 GOTO 30
360 REM SEARCH
370 INPUT "SEARCH D
  ATA=";D$(0)
380 O=LEN(D$(0)):PR
  INT "  SEARCH
  ING...":
390 FOR I=1 TO P
400 IF LEFT$(D$(I),
  O)<>D$(0) THEN
  420
410 CLS :PRINT "#":
  I:="":D$(I):PRI
  NT "SEARCHING..
  .":
420 NEXT I:CLS
430 GOTO 30
440 FOR I=1 TO P
450 PRINT "DATA#":I
  :="":D$(I)
460 NEXT I
470 GOTO 30
480 REM PRINTER
490 IF PP=1 THEN PP
  =0:PRINT OFF :
  GOTO 510
500 IF PP=0 THEN PP
  =1:PRINT ON
510 GOTO 30

```

6-4 Testing the Method of Appeal and Effectiveness of DM

DM (direct mail) is used widely to promote product sales but the principal concern is the reaction upon receiving the DM. This program decides whether the new DM, prepared and mailed with a different appeal from the previous DM, is effective based on the number of orders received. The percentage testing method (one-sided test), which is one of the statistical methods, is used in this program. (This is effective only in relation to orders exceeding past percentages).

Example:

Order percentage in the past (Number of orders/number of DMs) = 2.8%

Number of DMs sent this time = 6000

Number of orders received this time = 186

Although it appears that DM is effective with an order percentage of 3.1%, how will this appear with a percentage test?

RUN ↵

DIRECT MAIL

↵

BEFORE(%)?

Enter the past percentage of orders.

2.8 ↵

SAMPLE SIZE=?

Enter 6000 for the DMs sent this time.

6000 ↵

ORDERS RECEIVED=?

Enter the number of orders received this time.

186 ↵

RELIABILITY (90, 95, 99%)?

Enter either 90%, 95% or 99% for the appeal reliability of the DM.

90 ↵

BEFORE(%)= 2.8

Test results of the input data at a 90% reliability will be displayed.

↵

SAMPLE SIZE= 6000

↵

ORDERS RECEIVED (%)= 3.1

↵

RELIABILITY= 90%

↵

CRITERION 1.408590425

↵

EFFECTIVE!

From the percentage of reliability of 90%, the computer is judging that the appeal of the new DM is effective.

Then, how will this appear with a reliability of 95%?

↵

NEXT RELIABILITY (Y/N)?

Y ↵

RELIABILITY (90, 95, 99%)?

95 ↵

BEFORE(%)= 2.8

↵

SAMPLE SIZE= 6000

↵

ORDERS RECEIVED (%)= 3.1

↵

RELIABILITY= 95%

↵

CRITERION 1.408590425

↵

INEFFECTIVE!

The computer has judged that the appeal is not effective at 95% reliability.

If we obtain the results in a similar manner with a reliability of 99%, it will be as follows.

Criterion $Z = 1.409$ (1.408590425)

Will be effective when reliability is 90% as

$Z = 1.409 > Z_{0.1} = 1.282.$

Will not be effective when reliability is 95% as

$Z = 1.409 < Z_{0.05} = 1.645.$

Will not be effective when reliability is 99% as

$Z = 1.409 < Z_{0.01} = 2.326.$

($Z_{0.1}$, $Z_{0.05}$ and $Z_{0.01}$ are based on a normal distribution table.)

Variables

Variable name	Variable content
A	Selection of reliability
C ()	Criterion
D	Criterion Flag
F	Criterion quantity
E\$	Change or no change
N	Number of DMs sent
P	Percentage of order received up to now
PE\$	%
X	Orders received this time

Program List

```

5 PRINT " **DI
RECT MAIL**"
10 CLEAR :DIM C(3)
:PE$=CHR$(37)
20 C(1)=1.282:C(2)
=1.645:C(3)=2.3
26
30 INPUT "BEFORE("
+PE$+)"":P=P/
100
40 INPUT "SAMPLE S
IZE=":N:IF N<=0
THEN 40
50 INPUT "ORDERS R
ECEIVED=":X:X=X
/N
60 PRINT "RELIABIL
ITY (90,95,99"+
PE$+)"":INPUT
A
70 IF A=90 THEN D=
1: GOTO 110
80 IF A=95 THEN D=
2: GOTO 110
90 IF A=99 THEN D=
3: GOTO 110
100 GOTO 60
110 F=(X-P)/SQR(P*(
1-P)/N)
120 PRINT "BEFORE("
:PE$:)="":P*100
130 PRINT "SAMPLE S
IZE=":N
140 PRINT "ORDERS R
ECEIVED (":PE$:
")=":X*100
150 PRINT "RELIABIL
ITY=":A:PE$
160 PRINT "CRITERIO
N":F
170 IF F<-C(D) THEN
BEEP :PRINT "E
FFECTIVE!": G0T
O 200
180 IF F>C(D) THEN
BEEP :PRINT "EF
FFECTIVE!": GOTO
200
190 BEEP :BEEP :PRI
NT "INEFFECTIVE
!"
200 PRINT "NEXT REL
IABILITY (Y/N)"
:INPUT E$
210 IF E$="Y" THEN
60
220 END

```

6-5 Monster Road Game

You are on your way home without a worry on your mind when you suddenly see a monster lurking along the way. You're in trouble since there is only 1 road leading home.

However, the monster has weak points, i.e., he has poor vision and will catch moving objects only. As long as you don't move, you're safe.

Follow the advice and work your way safely home.

Good luck!

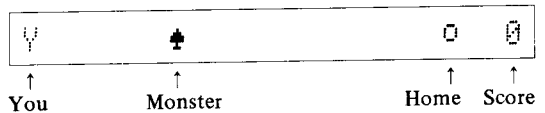
Characters That Appear

Y (You) Manipulate the **1** key (slow down) and the **3** key (speed up) and try to make it home without being caught by the monster.

♣ (Monster) . . . Appears and disappears and moves randomly on the screen.

0 (Home) 10 points added each time you reach home safely.

Game Display



Game Rule

RUN

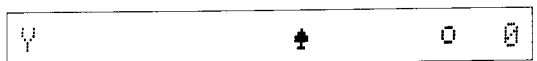
*** MONSTER ROAD ***

The game starts after the title is displayed.

Head for home while watching the movement of the monster.

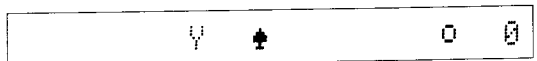
3 (speed-up)

⋮



The monster approaches. Slow down by pressing the **1** key and watch the monster closely.

1 (slow-down)



There will be no fear of being caught by the monster as long as you are motionless but, if the monster becomes aware of your movement, it will charge you relentlessly. The monster has moved behind you while you were standing motionless. Speed up and rush for home. Good luck!

⋮

♣ Y	0 0
-----	-----

Just a bit more.

♣ Y 0 0

You've escaped the monster and made it home.

0 10

Your score is displayed and the next game starts.

Y ♣	0 10
-----	------

This time you were unlucky and was caught.

X
CAUGHT...!!

The score up to now will be displayed with a beep sound and the computer asks if you wish to try again.

Press the key if you wish to continue and the key if you wish to stop.

SCORE 10
CONTINUE (Y/N)

Variables

Variable name	Variable content
X	Your position
Y	Monster's position
Z	Score
P	Speed
A	Your previous position
B	Monster's previous position
C	Input control
D\$	Input

Program list

```

10 CLS :WAIT 60:PR
   INT " *** MONS
   TER ROAD ***":Z
   =-10
20 CLS :Z=Z+10:LOC
   ATE 20:PRINT CH
   R$(237):USING"#
   ##":Z::BEEP
30 A=0:B=0:P=0:X=0
   :Y=INT(RND*15+3
   )
40 LOCATE A:PRINT
   " ";;LOCATE B:P
   RINT " ";
50 LOCATE X:PRINT
   "Y";;LOCATE Y:P
   RINT CHR$(232);
60 A=X:B=Y:IF X=Y
   THEN IF P>0 THE
   N 200
70 FOR C=0 TO 5:D$
   =INKEY$:IF D$="
   " THEN NEXT C
80 IF D$="3" THEN
   P=P+1:IF P=4 TH
   EN P=3
90 IF D$="1" THEN
   P=P-1:IF P=-1 T
   HEN P=0
100 X=X+P:IF X>19 T
   HEN 20
110 Y=Y+SGN(A-Y-.1)
   *SGN(RND-.1)*IN
   T(RND*4+1)
120 IF Y<1 THEN Y=1
   ELSE IF Y>18 T
   HEN Y=18
130 GOTO 40
200 BEEP 1:BEEP :LO
   CATE X:PRINT "X
   ":BEEP
210 PRINT :PRINT "
   CAUGHT...!!"
220 PRINT " SCOR
   E":Z
230 PRINT " CONTI
   NUE (Y/N)";
240 D$=INKEY$:IF D$
   =" " THEN 240
250 IF D$="Y" THEN
   10 ELSE END

```

CHAPTER

7

LIST OF FUNCTIONS AND STANDARDS

7-1 NUMERICAL FUNCTIONS

Function	Mathematical expression	Format	Meaning/remarks
Trigonometric functions	sin	SIN (Numerical expression) *hereafter X	$-5400^\circ < X < 5400^\circ$ (30π rad, 6000 gra)
	cos	COS (X)	$-5400^\circ < X < 5400^\circ$ (30π rad, 6000 gra)
	tan	TAN (X)	$-5400^\circ < X < 5400^\circ$ (6000 gra) (excludes the cases when $ X = (2n - 1) \times 90^\circ$)
Inverse trigonometric functions	\sin^{-1}	ASN (X)	* n is an integer. $ X \leq 1, -90^\circ \leq \text{ASN}(X) \leq 90^\circ$
	\cos^{-1}	ACS (X)	$ X \leq 1, 0^\circ \leq \text{ACS}(X) \leq 180^\circ$
	\tan^{-1}	ATN (X)	$ X < 10^{1.00}$, $-90^\circ \leq \text{ATN}(X) \leq 90^\circ$
Hyperbolic functions	sin h	HYP SIN (X)	$ X \leq 230$
	cos h	HYP COS (X)	$ X \leq 230$
	tan h	HYP TAN (X)	$ X \leq 10^{1.00}$, $-1 \leq \text{HYP TAN}(X) \leq 1$
Inverse hyperbolic functions	$\sin h^{-1}$	HYP ASN (X)	$ X < 5 \times 10^{99}$
	$\cos h^{-1}$	HYP ACS (X)	$1 \leq X < 5 \times 10^{99}$
	$\tan h^{-1}$	HYP ATN (X)	$ X < 1$
Exponential function	e^x	EXP (X)	$-227 \leq X \leq 230$
Natural logarithm	$\log_e x$	LOG (X)	$X > 0$ Accuracy will be ± 1 at the 8th digit when $1 - 10^{-6} < X < 1 + 10^{-6}$
Common logarithm	$\log_{10} x$	LGT (X)	$X > 0$ Accuracy will be ± 1 at the 8th digit when $1 - 10^{-6} < X < 1 + 10^{-6}$
Square root	$\sqrt{\quad}$	SQR (X)	$X \geq 0$
Absolute value	$ x $	ABS (X)	Provides absolute value of X.
Sign		SGN (X)	Provides $\begin{cases} -1 & \text{when } X < 0, \\ 0 & \text{when } X = 0, \\ 1 & \text{when } X > 0. \end{cases}$
Integer		INT (X)	Gaussian function: provides maximum integer not exceeding the value of X.

Function	Mathematical expression	Format	Meaning/remarks
Fraction Rounding		FRAC (X) ROUND(x,y,[f] {+, -}) x,y: X f: 0 or 1	Provides fraction of X. Rounds the value of x at 10 ^y position. { f is omitted: } Specifies 10 ^y position { f = 0: } { f = 1: } Specifies the yth position of a significant number { Sign omitted: rounds off. { +: rounds up. { -: discards.
Degree /minute /second	Sexagesimal → Decimal	DEG (d [,m [,s]]) d, m, s: X	Provides decimal-converted values.
Circular constant	π	PI	Provides an approximate value of π (3.1415926536)
Random number		RND	0 < random number < 1 A 10 digit random number is generated.

(X) → Numerical expression

* Parentheses cannot be omitted for ROUND and DEG, however, they can be omitted as to the others when numerical values or variables are used for (numerical expression).

7-2 STATISTICS FUNCTIONS

Format		Function
CNT*	n	Number of statistical data processed
SUMY*	Σy	Sum of the y data
SUMX*	Σx	Sum of the x data
SUMXY*	Σxy	Sum of the x data by y data
SUMX2*	Σx^2	Sum of the squares of the x data
SUMY2*	Σy^2	Sum of the squares of the y data
SDX	$x\sigma_{n-1}$	Sample standard deviation value of the x data $\sqrt{\frac{n\Sigma x^2 - (\Sigma x)^2}{n(n-1)}}$
SDY	$y\sigma_{n-1}$	Sample standard deviation value of the y data $\sqrt{\frac{n\Sigma y^2 - (\Sigma y)^2}{n(n-1)}}$
SDXN	$x\sigma_n$	Population standard deviation value of the x data $\sqrt{\frac{n\Sigma x^2 - (\Sigma x)^2}{n^2}}$
SDYN	$y\sigma_n$	Population standard deviation value of the y data $\sqrt{\frac{n\Sigma y^2 - (\Sigma y)^2}{n^2}}$
LRA	A	Linear regression constant term $\frac{\Sigma y - \text{LRB} \cdot \Sigma x}{n}$
LRB	B	Linear regression coefficient $\frac{n \cdot \Sigma xy - \Sigma x \cdot \Sigma y}{n \cdot \Sigma x^2 - (\Sigma x)^2}$
COR	r	Correlation coefficient $\frac{n\Sigma xy - \Sigma x \cdot \Sigma y}{\sqrt{(n\Sigma x^2 - (\Sigma x)^2)(n\Sigma y^2 - (\Sigma y)^2)}}$
EOX (X)	\hat{x}	Estimated value (value of x estimated from the value of y) $\text{EOX}(y_n) = \frac{y_n - \text{LRA}}{\text{LRB}}$
EOY (X)	\hat{y}	Estimated value (value of y estimated from value of x) $\text{EOY}(x_n) = \text{LRA} + x_n \cdot \text{LRB}$

(X) → Numerical expression

* The name of the basic statistics with ★ sign attached and each value are displayed in the sequential order by entering STAT LIST ↵ .

Press ↵ or ENTER if you wish to stop displaying and press again if you wish to continue displaying. Enter STAT LLIST ↵ for printouts.

7-3 CONTROL FUNCTIONS

Name	Format	Function
TAB	TAB (numerical expression)	Prints a specified number of spaces.
USING	USING	Specifies the output format of the PRINT statement and LPRINT statement.











7-4 CHARACTER FUNCTIONS

Name	Format	Function
ASC	ASC(X\$)	Obtains the ASCII code of the first character of character expression.
CHR\$	CHR\$(I)	Obtains the character corresponding to the ASCII code through a numerical expression. $0 \leq I < 256$
DMS\$	DMS\$(I)	Converts a decimal into a sexagesimal. $ I < 10^{100}$
HEX\$	HEX\$(I)	Obtains the character string of the hexadecimal converted from a decimal. $-32769 < I < 65536$
INKEY\$	INKEY\$	Reads 1 character from the keyboard.
LEN	LEN(X\$)	Obtains the number of characters in character expression. $0 \leq X\$ \leq 79$
VAL	VAL(X\$)	Converts a character expression to a numerical value.
LEFT\$	LEFT\$(X\$, I)	Fetches the number of characters specified by a numerical expression from the left of a character expression. $0 \leq I < 256$
MID\$	MID\$(X\$, I, J)	Fetches the number of characters specified by the numerical expression 2 from the position specified by the numerical expression 1.
RIGHT\$	RIGHT\$(X\$, I)	Fetches the number of characters specified by a numerical expression from the right of a character expression. $0 \leq I < 256$
STR\$	STR\$(I)	Converts a value of numerical expression to a character string.

X\$ → Character expression

I, J → Numerical expression

7-5 SCIENTIFIC CONSTANTS

Operation	Name	Symbol	Numerical value	Unit
	Acceleration of free fall	<i>g</i>	9.80665	ms ⁻²
	Speed of light (in space)	<i>c</i>	299792458	ms ⁻¹
	Plank's constant	<i>h</i>	6.626176×10^{-34}	J s
	Gravitational constant	<i>G</i>	6.672×10^{-11}	Nm ² kg ⁻²
	Elementary charge	<i>e</i>	$1.6021892 \times 10^{-19}$	C
	Electron mass	<i>m_e</i>	9.109534×10^{-31}	kg
	Atomic mass	<i>u</i>	$1.6605655 \times 10^{-27}$	kg
	Avogadro constant	<i>N_A</i>	6.022045×10^{23}	mol ⁻¹
	Boltzmann's constant	<i>k</i>	1.380662×10^{-23}	JK ⁻¹
	Volume of 1 kg-mole under STP	<i>V_m</i>	0.02241383	m ³ mol ⁻¹

7-6 MANUAL COMMANDS

Command	Example of usage	Function
CONT	CONT	Reexecutes program stopped with the STOP Statement or the [STOP] key.
DELETE	DELETE 20	Deletes line 20 only.
	DELETE 200 –	Deletes line 200 and after.
	DELETE – 80	Deletes from the first line to line 80.
	DELETE 50 – 100	Deletes from line 50 to line 100.
EDIT	EDIT	Displays first line of the program and sets to EDIT mode.
	EDIT 50	Displays line 50 and sets to EDIT mode.
LIST	LIST	Displays program contents in the currently specified program area.
	LIST 20	Displays line 20.
	LIST 200 –	Displays line 200 and after.
	LIST – 80	Displays from the first line to line 80.
	LIST 50 – 100	Displays from line 50 to 100.
	LIST ALL	Displays program contents in all program areas.
	LIST V	Displays registered variable names and declared array variable names.
LLIST	LLIST	Prints program contents in the currently specified program area.
	LLIST 20	Prints line 20.
	LLIST 200 –	Prints line 200 and after.
	LLIST – 80	Prints from the first line to line 80.
	LLIST 50 – 100	Prints from line 50 to line 100.
	LLIST ALL	Prints program contents in all program areas.
	LLIST V	Prints registered variable names and declared array variable names.
LOAD	LOAD	Loads a program from a cassette tape to the currently specified program area.
	LOAD ALL	Loads programs to all program areas from a cassette tape.
	LOAD,A	Loads a program with ASCII code format to the currently specified program area.
	LOAD,M	Mixes the program in the currently specified program area with the program read with ASCII code format.
	LOAD “ABC” LOAD ALL “CASIO” LOAD “TEST”, A LOAD “TEST”,M	Performs same function as the above format in relation to the program with a specified file name.

Command	Example of usage	Function
NEW	NEW	Erases programs in the currently specified program area.
	NEW ALL	Erases the programs in all program areas and all variables.
PASS	PASS "ABC"	Sets or releases passwords.
PROG	PROG 3	Specifies program area to be used as P3.
RUN	RUN	Starts execution from the first line of the program in the currently specified program area.
	RUN 1000	Starts execution of program from line 1000.
SAVE	SAVE	Stores the program in the currently specified program area on a cassette tape.
	SAVE ALL	Stores all programs in all program areas on a cassette tape.
	SAVE,A	Stores the program in the currently specified program area on a cassette tape with the ASCII code format.
	SAVE "ABC" SAVE ALL "CASIO" SAVE "TEST",A	Stores a program with its file name on a cassette tape.
SYSTEM	SYSTEM	Displays the program area status and the number of remaining bytes for user's area.
	SYSTEM P	Displays number of bytes used for a program and the number of bytes for a program management area.
	SYSTEM V	Displays number of bytes used for variables and also the number of bytes for a variable management area.
VERIFY	VERIFY	Checks save status of programs and data on a cassette tape.
	VERIFY "ABC"	Performs same operation as the above with regard to the program having a specified file name.

7-7 PROGRAM COMMANDS The commands with a * sign attached can also be used as manual commands.


Command	Example of usage	Function
ANGLE*	ANGLE 0	Sets angle units in degrees.
	ANGLE 1	Sets angle units in radians
	ANGLE 2	Sets angle units in grades.
BEEP*	BEEP	Generates a low buzzer sound.
	BEEP 0	Same as BEEP.
	BEEP 1	Generates a high buzzer sound.
CHAIN*	CHAIN	Loads the program that first appears from a cassette tape, and executes it.
	CHAIN "ABC"	Loads the program with a specified file name from a cassette tape, and executes it.
CLS	CLS	Clears the display and moves the cursor to the left end of the screen.
CLEAR*	CLEAR	Clears all variables.
DATA*	DATA 1,2,3	Stores data to be read with the READ statement.
DIM*	DIM A(3)	Declares one-dimensional single-precision numerical array.
	DIM B (2,3)	Declares two-dimensional single-precision numerical array.
	DIM C! (4)	Declares one-dimensional half-precision numerical array.
	DIM D! (3,4)	Declares two dimensional half-precision numerical array.
	DIM G\$(2)*3	Declares one-dimensional character array and specifies character length 3.
	DIM H\$(4,5)*6	Declares two-dimensional character array and specifies character length 6.
END	END	Terminates execution of a program.
ERASE*	ERASE D	Cancels the definition of registered variable D and array variable D.
FOR ~ TO ~ STEP ~ NEXT	FOR I = 5 TO 20 STEP 0.5 ~ NEXT I	Repeatedly changes the control variables between the FOR statement and the NEXT statement from the initial value to the final value with specified increments.
GET*	GET A	Reads data stored on a cassette type with the PUT command into a variable.
	GET "TEST" A	Reads data with file name "TEST" into variable A.
GOSUB	GOSUB 1000	Jumps to line 1000.
	GOSUB PROG 9	Jumps to program area P9 prepared as a sub-routine.

Command	Example of usage	Function
RETURN	RETURN	Returns to the line immediately after the GOSUB statement.
GOTO	GOTO 150	Unconditional jump to line 150.
	GOTO PROG 5	Unconditional jump to program area P5.
IF conditional expression THEN ~ ELSE ~	IF X > 10 THEN 10 ELSE 80	Executes statements after THEN when a conditional expression is true. Executes statements after ELSE if it is not true.
INPUT	INPUT X	Displays ? and waits for input for variable X.
	INPUT "Message", X	Displays message and waits for input for variable X.
	INPUT "TEST"; X	Displays message and ? and waits for input for variable X.
LET	LET A = B	Assigns the value on the right side of the equal sign to the variable on the left.
LOCATE	LOCATE 5	Specifies cursor position.
ON ~ GOSUB	ON X GOSUB 200, 300	Jumps to subroutine depending upon the value of variable X.
ON ~ GOTO	ON A GOTO 100, 110	Unconditional jump depending upon the value of variable A.
PRINT*	PRINT X,Y	Displays values of X and Y.
	PRINT X;Y	Displays values of X and Y successively.
LPRINT*	LPRINT X,Y	Prints values of X and Y
	LPRINT X;Y	Prints values of X and Y successively.
PRINT ON*	PRINT ON	Sets the print mode.
PRINT OFF*	PRINT OFF	Releases the print mode.
PUT*	PUT A	Stores value assigned to variable A on a cassette tape.
	PUT "DATA" A	Stores value assigned to variable A with a specified file name.
READ	READ X	Reads data stored by a DATA statement into a specified variable.
REM	REM ***	Provides a comment to a program.
RESTORE	RESTORE	Specifies that the first DATA statement should be read by the execution of the following READ statement.
	RESTORE 100	Specifies that the DATA statement on line 100 should be read by the execution of the following READ statement.

Command	Example of usage	Function
STAT*	STAT X,Y,N	Inputs statistical data and the number of the statistical data.
	STAT CLEAR	Initialization of the registers storing basic statistics.
	STAT LIST	Displays basic statistics.
	STAT LLIST	Prints basic statistics.
STOP	STOP	Suspends execution of a program.
TRON*	TRON	Sets the trace mode.
TROFF*	TROFF	Releases the trace mode.
WAIT	WAIT	Specifies a pause time during execution of a PRINT statement.

7-8 ERROR MESSAGES

Error message	Error	Corrective measure
BS error (Bad Subscript)	<p>1) The value of an array variable subscript is negative or is 256 or over. Example: DIM A (256)</p> <p>2) The specified numerical value is outside the argument range. Ex) The LOCATE command range is outside the range of $0 \leq X < 24$.</p>	<p>1) Change the subscript to a value within the specified range. If the subscript is a variable, check the related program.</p> <p>2) Change specifications to be within the argument range.</p>
BV error (Buffer overflow)	1) The input and output buffers are overflowed.	1) Let both operation and program be expressed within 79 characters per line.
DA error (read without Data)	1) A READ statement or a GET statement was executed although there was no data to read.	1) Check the relation of the READ statement and the DATA statement. Create data in the statements to be assigned to the variable.
DD error (Duplicate Definition)	<p>1) Arrays with the same array name but with different subscripts were defined in duplicate. Ex) The following array variable declarations by a DIM statement will cause a DD error. DIM A(1), A(2, 3) DIM A!(1), A!(2, 3) DIM A\$(1), A\$(2, 3) DIM A\$(1)*20, A\$(2, 3)*20</p>	1) Check the variable of the line in which the error occurred and also check the subscripts of the arrays with the same name. If different subscripts are found, change one of the array names and reorganize the program.

Error message	Error	Corrective measure
FC error (illegal Function Call)	1) An attempt was made to execute one of the following manual commands in a program. The commands are CONT, PASS, RUN, EDIT and DELETE. 2) An attempt was made to execute a program command manually. Such program commands are END, LET, REM, STOP, LOCATE, GOTO, GOSUB, RETURN, INPUT, DATA, READ, RESTORE, FOR ~ NEXT, IF ~ THEN ~ ELSESE. 3) A CONT command was executed when successive program execution was not allowed.	1) Delete the incorrect command from the program and reorganize the program. 2) Execute the command with a line number attached in a program. 3) Press the BRK key and re-execute the program from the beginning. If you know what line the program execution is stopped, re-execute from the next line by entering RUN line number  .
FO error (NEXT without FOR)	1) There is no FOR statement corresponding to NEXT.	1) Check the nesting structure.
GS error (RETURN without GoSub)	1) There is a RETURN statement not corresponding to a GOSUB statement.	1) Check the nesting structure specified by the GOSUB statement. Separate the main routine from subroutine.
MA error (MAThematical error)	1) An arithmetic operation or a numerical function operation is undetermined or impossible. Ex) Division by 0.	1) Check the numerical expression on the line where the error occurred. Also check the value of the related variable.
NO error (Nesting Over-flow)	1) Nesting levels exceeded the specified range. Ex) GOSUB/RETURN Max. 12 levels FOR ~ NEXT Max. 6 levels	1) Check the nesting structure and reduce the nesting level to within the permissible range.

Error message	Error	Corrective measure
NR error (device Not Ready)	1) I/O devices are not correctly connected. Ex) A printer is not connected.	1) Check whether the peripheral devices are correctly connected. Also check whether FA-20 is sufficiently charged and that paper is not jammed in the printer.
OM error (Out of Memory)	1) RAM card capacity was insufficient. Ex) An array declaration was made causing deficient memory.	1) Erase unnecessary programs. Expand memory capacity with the RAM card. Use the SYSTEM command to check number of remaining bytes.
OV error (OVERflow error)	1) Operation results or the numerical input value exceeds the range of $ x < 10^{100}$.	1) Check the numerical expression on the line in which the error occurred. Insert a PRINT statement in the program and check the value of the variable.
PR error (PRotected error)	1) Attempted to execute a command that cannot be used in a program having a password. Such commands are DELETE, LIST, LLIST, NEW and EDIT. 2) A different password was entered. 3) An attempt was made to load a program whose password is different from the FX-750P password.	1) Release the locked condition by re-entering the password and execute the program. 2) Input the correct password. 3) Release the FX-750P password before loading. In this case, the newly loaded password becomes the FX-750P password.
RW error (Read Write error)	1) A parity error occurred when executing a LOAD or VERIFY command.	1) Store the program again with the SAVE command.
SN error (SyNtax error)	1) There is an error in the command format. 2) When a fraction is used for a line number. 3) When an array of over 2 dimensions was declared.	1) Use the EDIT command to correct the line in which the error occurred. 2) Correct the line number. 3) Correct the declared array to within 2 dimensions.

Error message	Error	Corrective measure
SO error (Stack Over error)	1) The numerical value stack exceeded 8 levels. 2) The operator stack exceeded 20 levels. 3) The character stack exceeded 10 levels.	1), 2) Simplify or divide the numerical expression so the stack will come within the specified range. 3) Simplify or divide the character expression so the stack will come within the specified number of levels.
ST error (SString error)	1) Attempted to assign a character string, exceeding the permissible character variable length, to a character variable. The permissible character variable lengths are as follows. Fixed character variable: Max. 7 characters Registered character variable: Max. 16 characters Character array variable: Max. 79 characters.	1) Change to a variable with a greater character length. Shorten the character length of the character string to be assigned to the variable. Be careful when concatenating character strings.
TM error (Type Mismatch)	1) The variables on the left and right side of an assignment statement are different types. 2) The argument types do not match during an assignment.	1) Unify the left and right sides of the assignment statement in terms of numerical variables or character variables. 2) Assign a numerical value to a numerical variable and a character string to a character variable.
UL error (Undefined Line number)	1) The line number specified by an IF ~ THEN, GOTO or GOSUB statement does not exist. 2) There is no program in the program area specified by a GOTO statement or a GOSUB statement.	1) Create a line number to which the jump is to be made or change the specified line number to which the jump is to be made. 2) Create a program area for the branching destination or change the branching destination.

Error message	Error	Corrective measure
UV error (Undefined Variable)	1) An array variable was used without a declaration by a DIM statement.	1) Declare the array by a DIM statement at the beginning of a program.
VA error (VARIABLE error)	1) Attempted to register more than 41 variables.	1) A total of up to 40 registered variables and array variables can be used. Check the variable names by LIST V, and delete unnecessary variable names by CLEAR or ERASE.

7-9 CHARACTER CODE TABLE

Hexa- decim- al	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	16	(Space) 32	0 48	@ 64	P 80					(Space) 160	一 176	タ 192	ミ 208	224	240
1	(DEL) 1	! 17	1 33	A 49	Q 65						ア 161	チ 177	ム 193	209	225	241
2	(LINE TOP) 2	(INS) 18	" 34	2 50	B 66	R 82					イ 178	ツ 194	メ 210	226	242	
3	3	19	# 35	3 51	C 67	S 83					ウ 179	テ 195	モ 211	227	243	
4	4	20	\$ 36	4 52	D 68	T 84					エ 180	ト 196	ヤ 212	228	244	
5	5	21	% 37	5 53	E 69	U 85					オ 181	ナ 197	ユ 213	229	245	
6	(LINE END) 6	22	& 38	6 54	F 70	V 86					ラ 182	カ 198	ニ 214	230	246	
7	(ENTER) 7	23	' 39	7 55	G 71	W 87		■			ア 183	キ 199	ヌ 215	231	247	
8	(▲) 8	() 24	() 40	8 56	H 72	X 88					イ 184	ク 200	ネ 216	232	248	♠
9	9	25) 41	9 57	I 73	Y 89					ウ 185	ケ 201	ノ 217	233	249	♥
A	10	26	* 42	: 58	J 74	Z 90					エ 186	コ 202	ハ 218	レ 234	250	♦
B	11	27	+ 43	; 59	K 75	[91					オ 187	サ 203	ヒ 219	ロ 235	251	♣
C	(CLS) (→) 12	28	, 44	< 60	L 76	¥ 92					ヤ 188	シ 204	フ 220	ワ 236	252	●
D	(CR) (←) 13	29	- 45	= 61	M 77] 93					ユ 189	ス 205	ヘ 221	ン 237	253	○
E	14	30	. 46	> 62	N 78	^ 94					ヨ 190	セ 206	ホ 222	バ 238	254	
F	15	31	/ 47	? 63	O 79	_ 95					ツ 191	ソ 207	マ 223	° 239	255	


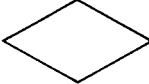
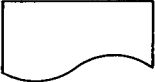
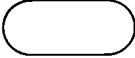
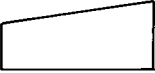
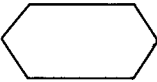


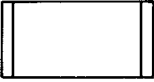
* The characters in parenthesis are control codes and will not be displayed.

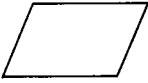


7-10 LIST OF RESERVED WORDS

The words shown below are called reserved words used to express commands or functions. The words with the symbol * cannot be used as registered variables.

&H	ERASE	LRA	SIN
ABS	EXP	LRB	SQR
ACS	FOR	MID\$	STAT
ANGLE	FRAC	NEW	STEP
ASC	GET	NEXT	STOP
ASN	GOSUB	OFF	STR\$
ATN	GOTO	ON*	SUMX
BEEP	HEX\$	PASS	SUMX 2
CHAIN	HYP	PI*	SUMX Y
CHR\$	IF*	PRINT	SUMY
CLEAR	INKEY\$	PROG	SUMY 2
CLS	INPUT	PUT	SYSTEM
CNT	INT	READ	SYSTEM P
CONT	LEFT\$	RESTORE	SYSTEM V
COR	LEN	RETURN	TAB
COS	LET	RIGHT\$	TAN
DATA	LGT	RND	THEN
DEG	LIST	ROUND	TO*
DELETE	LIST V	RUN	TROFF
DMS\$	LLIST	SAVE	TRON
EDIT	LLIST V	SDX	USING
ELSE	LOAD	SDXN	VAL
END	LOCATE	SDY	VERIFY
EOX	LOG	SDYN	WAIT
EOY	LPRINT	SGN	

7-11 LIST OF FLOWCHART SYMBOLS

Symbol	Meaning	Details
	Process	Indicates all types of processing functions.
	Judgement	Makes a judgement for program execution to be branched.
	Document	Indicates printing out on paper.
	Terminal	Indicates start, end and stop of processing.
	Manual input	Indicates inputs from the keyboard.
	Preparation	Preparations for setting initial value, etc.
	Magnetic tape	Indicates external memory devices such as a tape recorder.
	Display	Indicates displaying of messages and data on the display.
	Predefined process	Indicates predefined processes such as subroutines.

Symbol	Meaning	Details
	Input/output	Indicates a general input/output for which the I/O media is not especially specified.
	Connector	Indicates an exit to another place or an entrance from another place of the flowchart.
	Flow line	Connects the symbols mentioned above.

SPECIFICATIONS

Type:

FX-750P

Fundamental calculation functions:

Negative numbers, exponentials, parenthetical addition/subtraction/multiplication/division (with priority sequence judgement function – true algebraic logic).

Built-in functions:

Trigonometric and inverse trigonometric functions (angle units in degrees, radians and grades), hyperbolic and inverse hyperbolic functions, logarithmic and exponential functions, square roots, powers, integers, fractions, absolute values, signs, conversion from decimals to sexagesimals, conversion from decimals to hexadecimal, rounding, π and random numbers.

Statistical calculation functions:

Standard deviation = number of data, sum, sum of squares, standard deviation (2 types)

Linear regression = Number of data, sum of x , sum of y , sum of x^2 , sum of y^2 , sum of products, standard deviation of x (2 types), standard deviation of y (2 types), constant term, regression coefficient, correlation coefficient, estimated value of x , and estimated value of y .

Commands:

CONT, DELETE, EDIT, LIST, LLIST, LOAD, NEW, PASS, PROG, RUN, SAVE, SYSTEM, VERIFY, ANGLE, BEEP, CHAIN, CLEAR, CLS, DIM, END, ERASE, FOR ~ TO ~ STEP, NEXT, GET, GOSUB, RETURN, GOTO, IF ~ THEN ~ ELSE, INPUT, LET, LOCATE, ON ~ GOSUB, ON ~ GOTO, PRINT, LPRINT, PRINT ON, PRINT OFF, PUT, READ, RESTORE, DATA, REM, STAT, STAT CLEAR, STAT LIST, STAT LLIST, STOP, TRON, TROFF, WAIT

Character functions:

ASC, CHR\$, VAL, STR\$, DMS\$, HEX\$, LEFT\$, RIGHT\$, MID\$, LEN, INKEY\$

Display functions:

TAB, USING

Calculation range:

$\pm 1 \times 10^{-99} \sim \pm 9.999999999 \times 10^{-99}$. Internal operation uses 12-digit mantissa.

Program language:

BASIC

RAM capacity:

In case a 4K byte RAM card is inserted.

User area: 2,800 bytes

System area: 1,088 bytes

Fixed variable area: 208 bytes

ROM capacity:

About 27K bytes

Number of program area:

Maximum 10 (P0 – P9)

Number of stacks:

Subroutines: 12 levels

FOR ~ NEXT loop: 6 levels

Numerical values: 8 levels

Operators: 20 levels

Display:

Displays with 10-digit mantissa (minus sign not included) or 10-digit mantissa + 2-digit exponent. Also displays WAIT, DEG, RAD, GRA, TRON, PRTON and STOP symbols.

Display elements:

24-digit, dot matrix liquid crystal display.

Main component:

C-MOS-VLSI, etc.

Power supply:

Main frame ... 2 lithium batteries (CR2032)

RAM card ... 1 lithium battery (CR2016)

Power consumption:

Main frame ... 0.03W

Battery life:

Main frame ... approx. 120 hours

With FA-20 connected ... approx. 60 hours

RAM card battery (when stored separately from the main frame)

RC-2 ... approx. 2 years

RC-4 ... approx. 1 year

Auto power-off:

Approximately 6 minutes after last operation

Ambient temperature range:

0°C to 40°C (32°F to 104°F)

Dimensions and weight:

Main frame — 16mmH × 186mmW × 82mmD, 226g (including batteries)

($\frac{5}{8}$ "H × $7\frac{3}{8}$ "W × $3\frac{1}{4}$ "D, 8oz)

RAM card — 3.8mmH × 60mmW × 50mmD, 26g (including a battery)

($\frac{5}{32}$ "H × $2\frac{3}{8}$ "W × 2"D, 0.9oz)

INDEX

A		Cursor	25
&H	279	D	
ABS	268	DATA	243
ACS	111, 264	Data Storing and Loading	35
ANGLE	112, 220	Debugging	88
Array variable	75	DEG(degree)	24, 220
ASC	166, 251	DELETE	207
ASCII code	251	DIM	139, 225
ASN	111, 264	Direct Mode	26
Assignment	44, 73	DMS\$	254
ATN	111, 264	Double quotation mark	40, 83
Auto Power Off	23		
B		E	
BEEP	200, 221	EDIT	208
C		Editing Keys	28
Calculation Methods	39	END	191, 227
Calculation Priority Sequence	38	Enter key	30, 45
CHAIN	222	EOX	132, 277
CHARACTER CODE TABLE	325	EOY	132, 278
Character Functions	53	ERASE	227
Character variables	77	ERROR MESSAGES	320
Checking the Input Program	81	Error Messages after Inserting or Removing of a RAM Card	21
Checking the printing	87	EXP	267
CHR\$	143, 251	F	
CLEAR	137, 223	Fixed Variables	77
CLS	171, 224	Fixed variable area	13
CNT	131, 273	FOR ~ TO ~ STEP/NEXT	147, 228
Comments	176, 245	FRAC	269
Common system area	13, 14	Function Mode	28
Conditional expression	101, 233	G	
CONT	205, 247	GET	230
Contrast Adjustment	25	GOSUB	187, 231
Control variable name	228	GOTO	99, 232
COR	132, 277	GRADE	24, 220
Correcting the Program	81		
COS	109, 263		

H		Number of Input/Output Digits and calculation digits	41
Half-precision	78	Numerical Functions	46, 47
HEX\$	255	Numerical variables	77
HYPASN, HYPACS, HYPATN	266	O	
HYP SIN, HYPCOS, HYPTAN	265	ON ~ GOSUB	237
I		ON ~ GOTO	238
IF ~ THEN ~ ELSE	101, 233	One-Dimensional array	140
INKEY\$	164, 259	Option	31
INPUT	96, 234	P	
INT	198, 268	PASS	214
L		PI	272
LEFT\$	177, 256	PRINT	239
LEN	259	PRINT ON, PRINT OFF	241
LET	117, 235	PROG	215
LGT	267	Program management area	12, 18
LIST, LLIST	209	Program Storing/Loading	33
LOAD(ALL)	210	PUT	242
LOCATE	236	R	
LOG	192, 267	RADIAN	24, 220
Loop	229	RAM Card	4
LPRINT	239	Removing the RAM Card	9
LRA, LRB	130, 276	Replacing the RAM Card Battery	10
M		Setting the RAM Card in the Computer	7
Memory Map	13	Random numbers	197, 272
Message	96	READ	157, 243
MID\$	178, 258	Registered Variables	78
Modification of Inputs (Correction, deletion, insertion)	43	Relational Operater	38
N		REM	176, 245
Nesting	229, 231	RESERVED WORDS	326
NEW(ALL)	213	RESTORE	243
Null	213	RETURN	187, 231
Number of Bytes Depending on RAM Card Combination	14	Return key	29, 45
		RIGHT\$	179, 257

INDEX

RND	197, 272	U	
ROUND	144, 270	User area	12, 18
RUN	82, 215	USING	113, 261
S			
SAVE(ALL)	33, 216	V	
Scientific Constants	61	Variables	74
SDX, SDY	124, 275	Variable management area	12, 18
SDXN, SDYN	124, 275	VAL	160, 252
Semicolon (;)	83, 95	VERIFY	219
SGN	269	W	
Shift Mode	27	WAIT	200, 250
SIN	109, 263		
Single-precision	78		
Special Keys	28		
SQR	266		
STAT	246		
STAT CLEAR	247		
Statistics Functions	55		
STAT LIST	217		
STAT LLIST	217		
STOP	247		
STR\$	162, 253		
Subroutine	188		
SUMX, SUMY	122, 273		
SUMX2, SUMY2	122, 274		
SUMXY	274		
SYSTEM	13, 218		
System area	13, 18		
T			
TAB	260		
TAN	109, 263		
TRON, TROFF	249		
Two-dimensional array	140		

CASIO®