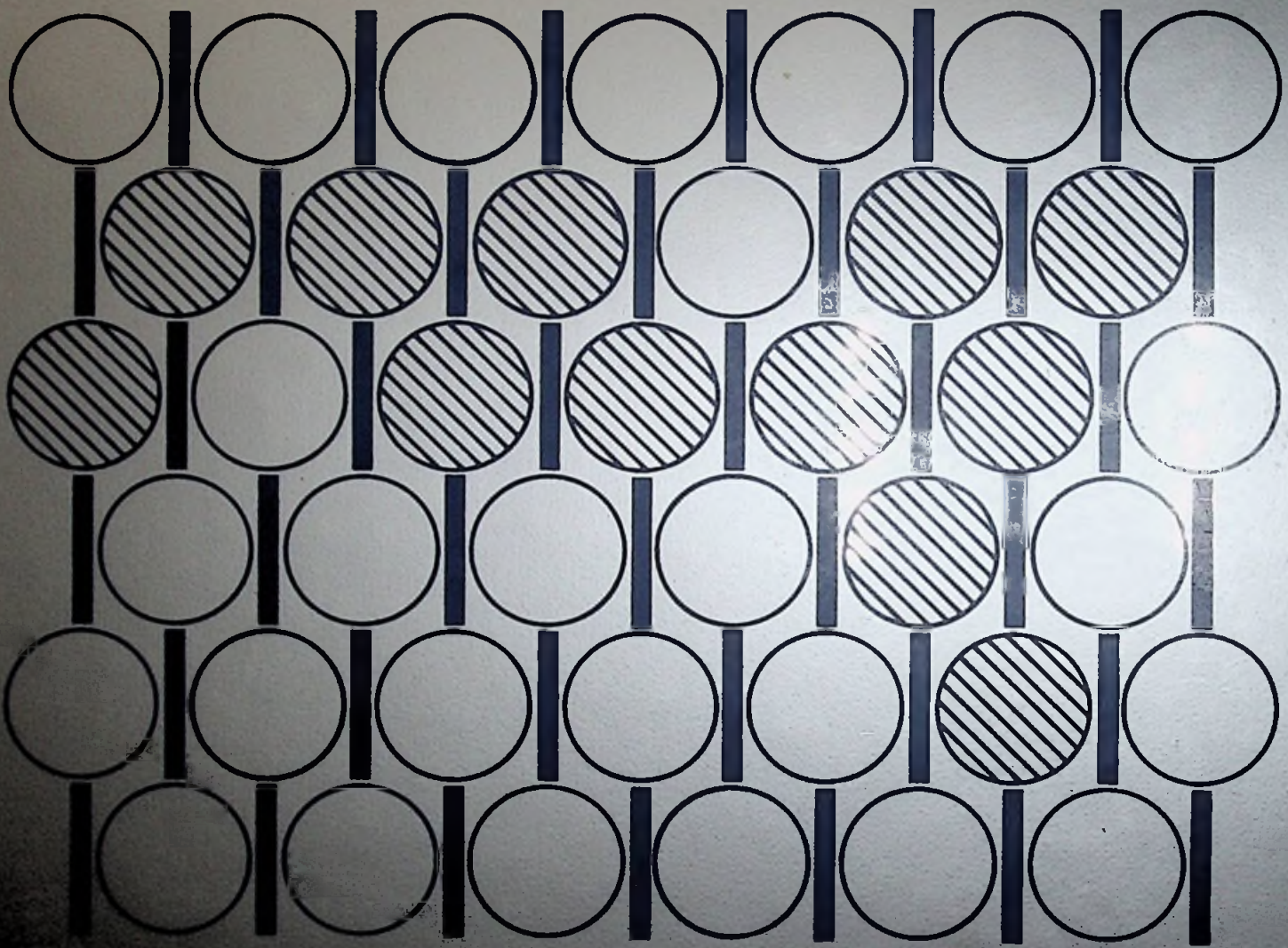


SANYO

MBC-550 SERIES

USER'S GUIDE

PREAFACE

Please read this manual thoroughly before you begin to install and use Sanyo MBC-550 series computers.

The contents of this manual are subject to change. If you wish to receive revisions and updates, please contact the place of purchase.

Sanyo assumes no responsibilities for errors in this manual or their consequences.

TABLE OF CONTENTS

CHAPTER 1
GETTING STARTED

INTRODUCTION

The MBC-550 series computer is 16 bit component type personal computer. It features an Intel 8088 (3.6 MHz no-wait) central processing unit (CPU). The main 128K memory may be expanded up to 256K.

The MBC-550 series computer comes wih a standard 5 1/4 inch single-sided, double density floppy disk drive (160K bytes in its formatted state). An additional drive may be added.

High resolution color or monochrome monitors may be used with your computer. The color monitor is 14" 640 x 200 dots one-dot-eight-color type and the green monitor is 12" 640 x 200 dots type. We recommend Sanyo brand monitors for the best results.

The operating system used is MS-DOS (tm) by Microsoft (tm). A wide range of software and applications is available. Please contact your authorized Sanyo computer dealer for more information on these programs.

The MBC-550 series computer has a parallel printer that allows compatibility with Sanyo brand printers as well as most parallel printers. An optional RS-232C interface is available for connections with serial printers.

SETTING UP

Computer Location

The placement of your Sanyo computer is important. Just as you would not place your refrigerator near the furnace, you should locate your computer where it will work with maximum efficiency.

Please follow these guide lines in selecting a suitable place for your computer.

1. Do not place the computer in the direct sunlight or near heaters or coolers.

2. Do not locate the computer in areas of dirt and dust or where vibration may take place.

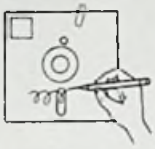
3. Do not use in areas of intense electromagnetic fields.

4. Keep the computer away from walls and place it in a well ventilated room.

5. Use only specified local power lines.

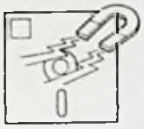
6. Do not drink or eat near the computer or place it where crumbs and liquid can enter the computer cabinet.

Diskette Handling

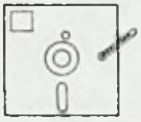


Do not pin or clip the diskette (disk) jacket.

Do not write anywhere on the disk except on the label. Always write with a felt tip pen; do not use a pencil or ink pen.

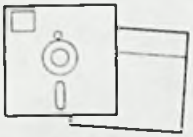


Do not touch the magnetic recording face.
Never clean with alcohol or similar material.



Never expose disks to magnetic or electric fields.

Do not store disks in direct sunlight, extreme heat or cold.



Keep the diskettes inside their envelopes to protect against dust or dirt.

Do not bend or fold diskettes.

Always make copies of original disks for safe keeping.

Recommended Diskettes

Sanyo recommends the use of the Sanyo MD1DD diskettes (single sided, double density, soft sectored).
Sanyo assumes no responsibility for trouble arising from the use of other diskette types.

INSTALLATION

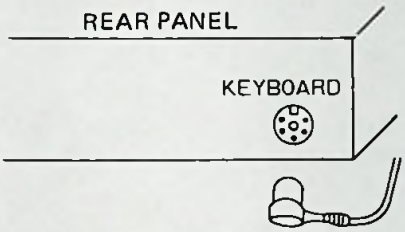
Use local AC power outlets. Remove diskettes before switching the power **on** or **off.**

System Connection

Please read the following instructions before connecting the system.

Check each box [] as you complete each step.

Connect the keyboard as shown below.



[] Remove the vinyl cover from the plugs and connectors.

[] Join the keyboard cable to the computer socket labeled KEY BOARD.

Monitor Connection

Sanyo CRT-36 Data Display Monitor

The CRT-36 monitor may be placed on the top of the computer cabinet or in a convenient location nearby.

[] Connect the video cord to the monitor in the rear socket labled VIDEO OUTPUT.

[] Connect the other end to the rear of the Sanyo 550 series computer in the socket labeled MONO CHROME.

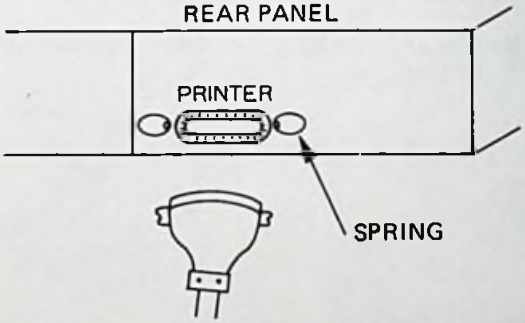
[] Connect the power cord to a local outlet. The power switch is located below the screen.

NOTE: FOR OTHER MONITORS - SOME MONITORS PROVIDE POOR EMI PERFORMANCE. TO ELIMINBATE THE NOISE PROBLEM, CONNECT THE MONITOR POWER CORD TO THE MBC 550 SERIES COMPUTER POWER OUTLET LOCATED IN THE REAR OF THE COMPUTER.

Printer Connection(optional)

Parallel Printers

Connect the printer as shown below.



[] Connect the printer cable to the computer connector labeled PRINTER. Lock by pressing springs into the slots on the printer plug. Note that this is a parallel (Centronics) printer port and will support only a parallel printer such as the Sanyo PR5000/5500. Please contact your Sanyo computer dealer for other Sanyo printers.

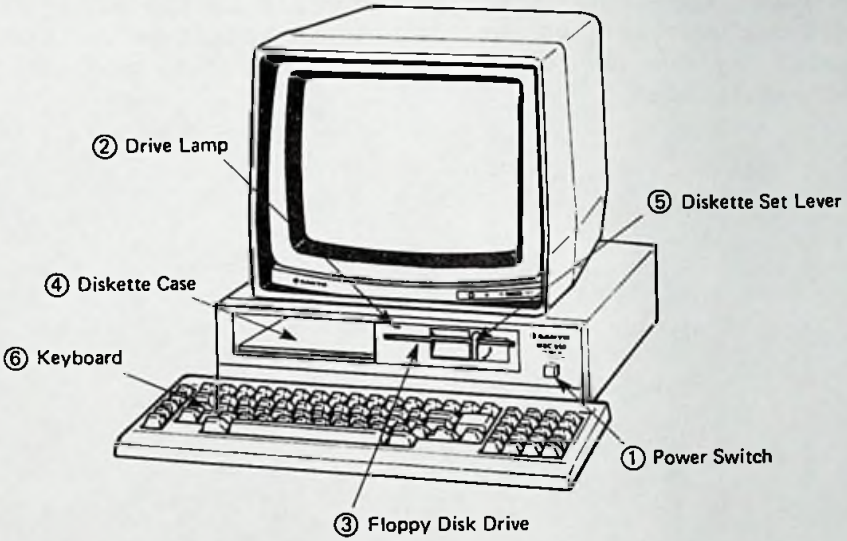
[] Connect the printer cable to the printer.

CAUTION: When connecting peripherals to the MBC-550 series computers, be sure to use the specified shield cables (by the peripherals' manufacturers) to meet the FCC regulations.

NAMES AND FUNCTIONS OF THE PARTS

Names

Front



Functions

1 Power Switch

Press the switch in to turn the computer on. A red light will glow on the A: drive indicating full power. Never turn the power on with a diskette in the drive.

2 Drive Lamp

This red lamp glows when power is first turned on and indicates full power received. With two drives, the drive lamp indicates which drive is being used.

3 Floppy Disk Drive

This is a built-in 5 1/4" single sided, double density, floppy disk drive. One more drive may be added in the space now used for diskette storage.

4 Diskette Case

This compartment may be used to store disks. Please note the position of the diskette as it enters the compartment. Do not force entry, try again with disk rotated till it matches the figure in the illustration.

5 Diskette Set Lever

Insert the diskette into the drive and move the lever down to set the diskette. Move the lever up to remove the diskette. Never start the computer with the diskette in the drive.

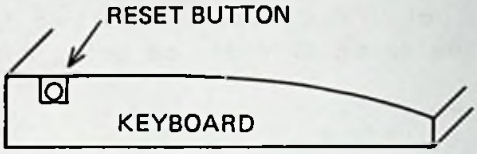
6 Keyboard

The keyboard contains ten function keys, cursor control keys, numeric keys, ten key pad, math function keys, and editing keys in ASCII specifications.

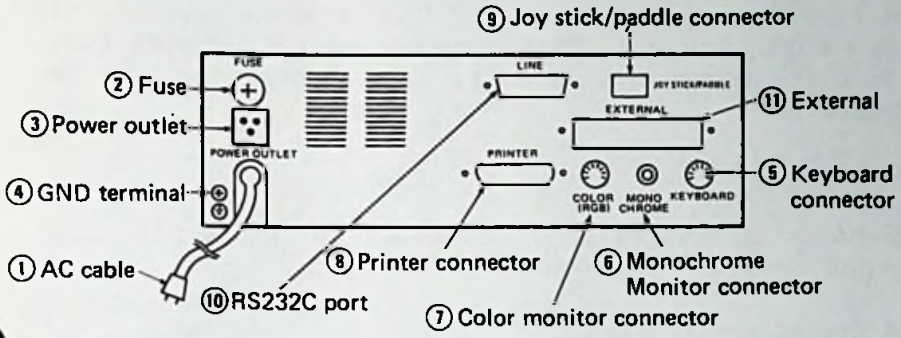
7 Reset Button

The RESET button is located on the left rear edge of the keyboard. The RESET button is used to start the system (cold boot) again after it has been running.

Never depress the button when a program is running.



Back


⑨ Joy stick/paddle connector
FUSE
LINE
② Fuse
③ Power outlet
POWER OUTLET
EXTERNAL
⑪ External
PRINTER
④ GND terminal
COLOR (RGB)
MONO CHROME
KEYBOARD
⑤ Keyboard connector
① AC cable
⑧ Printer connector
⑥ Monochrome Monitor connector
⑩ RS232C port
⑦ Color monitor connector

Functions

1 AC Cable

This connects the computer to the AC outlet (50/60Hz).

2 Fuse

A fuse is provided to protect the internal circuits. If you must change a fuse, use only a 1.5 ampere fuse. Turn the power off. An extra fuse is provided with the computer.

3 Power Outlet

A power outlet is provided to allow you to plug your monitor into the computer.

4 Grounding Terminal

An electrical may be attached here.

5 Keyboard Connector

This socket connects the keyboard with the computer.

6 Monochrome Monitor Connector

This socket connects high resolution composite monitors.

7 Color Monitor Connector

This socket connects high resolution color monitors.

8 Printer Connector

This connects a parallel printer to the computer.

9 Joy Stick/Paddle Connector

This plate is removed for installing a joystick or paddle.

10 RS232C Port

This plate is removed for installing an RS232C (serial) interface.

11 EXTERNAL

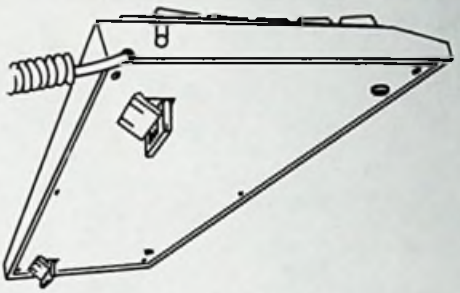
This plate may be removed for installing future interfaces provided by Sanyo.

Tilt Legs

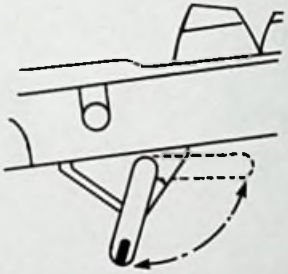
The keyboard of the Sanyo MBC 550 series computer is equiped with adjustable tilt legs to provide you with maximum comfort when using the keyboard.

Snap the legs toward the rear of the keyboard. Push firmly against the back rests. Make sure both legs are at the same angle. Refer to the illustration below.

For a lesser slope, the keyboard may be operated without the tilt legs in the extended position.



Back View of Keyboard



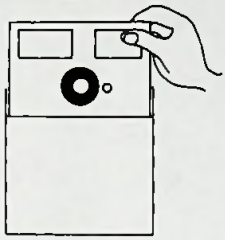
leg

OPERATIONS

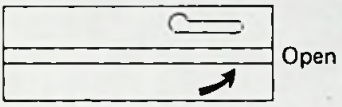
Disk Drive

Never insert or withdraw a disk while the drive is running. Except for initial start up, The input symbols, A: or B: (second drive) for DOS mode and > for Sanyo-Basic mode will be displayed when the drive unit is not running.

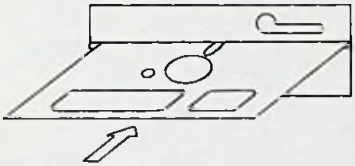
Inserting the Diskette



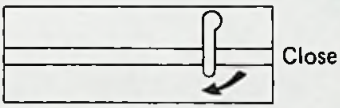
Withdraw the diskette from its envelope.



Open the drive door by moving the gate lever up.



Hold the diskette by the label and insert into the drive. The write protect cut out on the disk edge will face left and the read/write slot will lead into the drive.

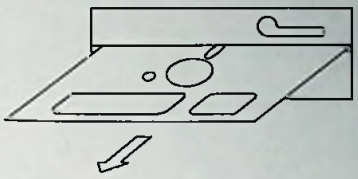


Insert the diskette till the end of its passage.

Close the door by moving the gate lever down.

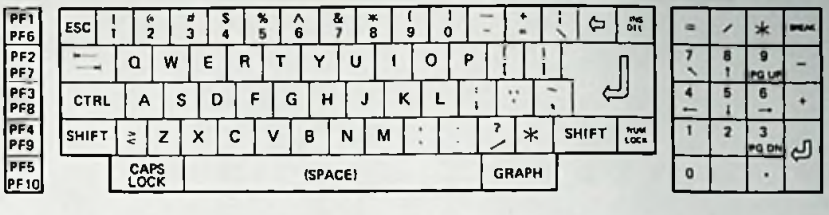
Withdrawing the Diskette

Open the drive door by moving the gate lever up. Withdraw the diskette and close the door by moving the diskette set lever down.



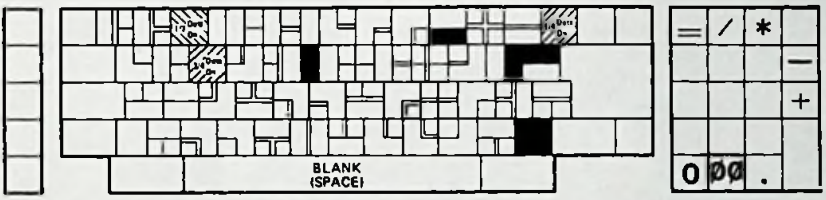
Keyboard

ASCII Keyboard



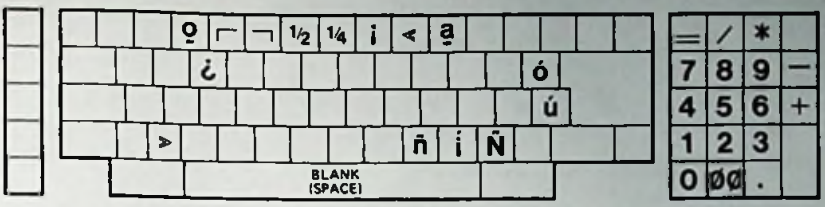
Keyboard with Graphic Symbols

1) GRAPH



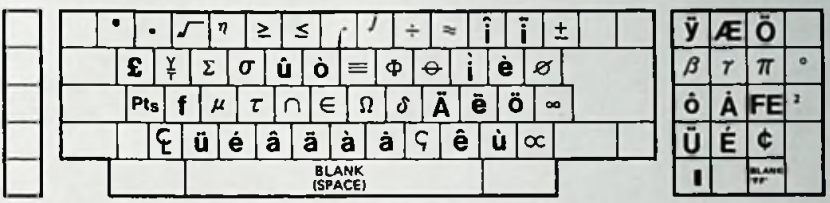
ØØ 30H is generated twice.

2) GRAPH + SHIFT



ØØ 30H is generated twice.

3) GRAPH + CTRL



The keys are imprinted with only the upper case characters they represent. They can be used to input commands in both upper case and lower case.

When a letter key is depressed, the corresponding lower case letter will be input.

To input upper case letters, depress the SHIFT key.

Depress the CAPS LOCK key to continually enter upper case letters. The CAPS LOCK lamp set in the key will glow red indicating upper case continual entry.

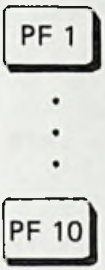
By depressing the CAPS LOCK key again, lower case letters are again continually entered. The CAPS LOCK lamp will be off.

Repeat

The keyboard is equipped with the auto-repeat function. If a key is depressed for about 0.5 seconds, that character will be input again and again.

If the GRAPH key is depressed in Sanyo-Basic mode, a beep sounds and the Graph Lamp in the key lights indicating that graphic mode is on. The keys on the keyboard now represent graphic characters and may be input.

To exit the graphic mode, depress the GRAPH key again. A beep sounds and the Graph Lamp goes out.



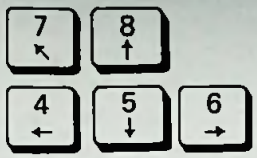
Each function key may be assigned a character string while in Sanyo-Basic. Depressing the function key then will have the same effect as entering successively the character keys corresponding to the string.

In the DOS (Disk Operating System) mode, the keys are assigned through the software.

Numeric Keys

These keys are intended for business operations. The SHIFT, and GRAPH keys do not effect their operation.

This key is an on/off control for the cursor (edit) keys. In Sanyo Basic or through software applications, the cursor keys are operable. Depressing this key will cause the number keys to direct the cursor, another depression and the keys will enter only numerals. This also allows line feeds (LF) during applications.



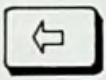
These "arrow" keys move the cursor one line or character per stroke and are accessed through software. The cursor keys are located on the ten key pad: key 7 - cursor diagonally, key eight - cursor up, key four - cursor left, key five - cursor down, and key six - cursor right.

The red BREAK key causes the interruption of a Sanyo Basic Program.

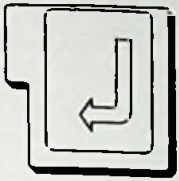
This key is depressed to reset the system. It is located on the left rear edge of the key board.

This key changes the system to control mode.

The INS/ DEL key, in Sanyo Basic, inserts a character to the left of the cursor, or erases the character to the left of cursor. The INS function is accessed depressing the SHIFT key and the INS/DEL key. The DEL function is accessed by directly depressing the INS/DEL key.

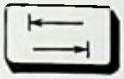


The back space key (indicated by arrow left key to the left of the INS/DEL key) erases the character to the left of the cursor as it moves left.



The RETURN key, indicated by a down left arrow, checks for valid commands and returns the cursor to home position. It starts new lines in basic mode.

This key is used to create the escape sequence in the basic mode.



The tab key, indicated by these symbols, ← →

outputs the tab code. The key may be used with some application software, e.g. WordStar, Easy Writer I, II, etc.

RUNNING THE OPERATING SYSTEM

The following prompts will appear after you insert a system disk and close the drive lever:

```
MS-DOS Version x.xx
Copyright xxxx Microsoft Corp.

Command V. x.xx
Current date is xx-xx-xxxx
Enter new date:
```

This is referred to as a cold start. You are now in the DOS (Disk Operating System). You must now enter today's date and the time.

Type today's date in an **mm-dd-yy** form where:

mm is a one or two digit number form 1-12 and represents the month.

dd is a one or two digit number for 1-31 and represents the day of the month.

yy is a two or four digit number from 80-99(the 19 is assumed), or from 1980-2099. **yy** represents the year.

Each number set is separated by hyphen - or slash /. For example: 05-06-84 or 05/06/84.

Enter the date and depress the RETURN key. A prompt will appear giving you the system time.

```
Current time is hh.mm:ss.cc
```

Press the RETURN key if you do not want to change the time shown. A new time may be entered in the following form:

hh is a one or two digit number from 0-23 and represents the hour.

mm is a one or two digit number from 0-59 and represents the minutes.

You do not have to enter the seconds **ss** or the hundredths of seconds **cc**.

The hour and minute entries must be separated by colons **:**. Depress the RETURN key after you have entered the time.

The system uses the time entered as the new time if entered correctly. If you do not enter the time correctly, the following message will be displayed:

```
Invalid time
Enter new time:
```

You may enter the correct form for the time or depress the RETURN key to use the time displayed previously.

You are now at command level (DOS) and will see the prompt **A:**

You may obtain a listing of the files on your disk by entering **DIR** followed by a depression of the RETURN key. By entering the filenames with a .COM at the end of them, you may access those system commands, or enter the appropriate filename of any software available to you on your disk.

NOTE: THE TIMER OF THE COMPUTER DOES NOT ALWAYS INDICATE THE ACCURATE TIME AS THE CLOCK OF THE COMPUTER STOPS WHILE THE FLOPPY DISK DRIVE IS BEING ACCESSED.

If your application software is on another disk and you have a single drive system, you will have to remove the diskette now in the drive and replace it with the application diskette. If you have two drives, the application software may be on the second drive (drive on left) and you may enter B: to access that drive, then a DIR to obtain a listing for that drive. Enter commands by typing in the filenames just as you would for single drive systems.

FORMATTING YOUR DISKETTE

Blank diskettes must be formatted before data is stored on them. Formatting is a way of arranging the diskette so the MBC550 will know how and where to store and look for information. Just as a file drawer is arranged with file holders and separated into groups, a diskette must be arranged so data may be separated in easy to locate sections and areas.

The FORMAT command accomplishes the task of assigning sectors and work areas on the diskette.

Single Drive

With the system disk in drive A, type in the format command after the system prompt:

A:**FORMAT /S**

/S is an option that allows you to copy the operating system onto the disk as you format it.

The following message will appear:

Remove disk now in drive A
Insert a blank disk in drive A and depress any key

Remove the system disk. Place a blank disk in drive A and depress any key. Formatting will begin. The following prompt will appear as long as the disk is being formatted:

```
Formatting...
```

Formatting is complete when this prompt is displayed:

```
Formatting completed
Format another disk (Y/N)?
```

If you wish to format another blank disk, remove the formatted disk, insert a blank disk in the drive and enter **Y** and depress the RETURN Key. IF you do not wish to format another disk, enter **N** and depress the RETURN key. You will see the system prompt A: indicating you are back to command mode.

CAUTION FORMATTING A DISK WITH INFORMATION ON IT WILL DESTROY ALL THE CONTENTS OF THE DISK. PLEASE BE CAREFUL TO CHECK THAT THE DISK IS BLANK OR THE DATA ON THE DISK IS NO LONGER NEEDED.

If you have copied the system onto your formatted disk with the **/S** option, you must also copy the file COMMAND.COM. You will do this with the DISKCOPY command explained in the section titled DISKCOPY Command.

Double Drives

With the system disk, or disk containing FORMAT.COM in drive A, type in the format command with the B drive designation:

```
A:FORMAT B: /S
```

/S is an option that allows you to copy the operating system onto the disk as you are formatting it.

The following message will appear:

```
Insert a blank disk in drive B: and depress any key
```

Place a blank disk in drive B and depress any key. The following prompt will appear as long as the disk is being formatted:

```
Formatting...
```

Formatting is complete when this prompt is displayed:

```
Formatting completed
Format another disk (Y/N)?
```

If you wish to format another blank disk, remove the now formatted disk from drive B, replace with a blank disk and enter **Y** followed then depress the RETURN key. If you do not want to format another disk, enter **N** followed by depressing the RETURN key. You will see the system prompt A: indicating that you are back to command mode.

CAUTION FORMATTING A DISK WITH INFORMATION ON IT WILL DESTROY ALL THE CONTENTS OF THE DISK. PLEASE BE CAREFUL TO CHECK THAT THE DISK IS BLANK OR THE DATA ON THE DISK IS NO LONGER NEEDED.

If you have copied the system onto your disk with the /S option, you must also copy the file COMMAND.COM. You will do this with the COPY command explained next.

HOW TO COPY YOUR FILES

COPY Command

Just as with paper files, you often need more than one copy of a disk file. You should always make back up copies of all programs in case something happens to your master. Software is costly.

The COPY command allows you to copy one or more files to another disk. You can also give the copy a different name if specify the new name in the COPY command.

The COPY command can also make copies of files on the same disk. In this case, you must supply a different filename or you will overwrite the file. You can not make a copy of a file on the same disk unless you specify a different filename for the new copy.

The format of the COPY command is:

```
COPY filespec [filespec]
```

Examples:

```
COPY A:MYFILE.TXT A:NEWNAME.TXT
```

You have duplicated your file on drive A: It now has two names: MYFILE.TXT and NEWNAME.TXT

```
COPY A:MYFILE.TXT B:MYFILE.TXT
```

Copy the file MYFILE.TXT on the disk in drive A: to a file named MYFILE.TXT on the disk in drive B:

DISKCOPY Command

You should make back up copies of all your files, especially costly software.The DISKCOPY command copies the contents of a disk onto another disk. DISKCOPY is faster than COPY because it copies the entire contents of a disk in one operation.

Single Drive Systems

For single drive systems, if you want to make a copy of a disk, type:

DISKCOPY

The system will display:

Insert source diskette into drive A:**<CR>**
Insert formatted target diskettes into drive A:**(<CR>)**
Press any key when ready

Remove the system disk. Insert the disk to be copied and depress the RETURN key. Remove the disk after the cursor drops to the second prompt . Insert a blank formatted disk and depress the RETURN key. The files will be read from memory on to the disk.

Note: If either of the disks you are using has defective tracks, DISKCOPY will not work. Use the COPY command to back up your disks in these cases. The COPY command will skip over defective tracks.

Double Drive Systems

The format for double drive systems for the DISKCOPY command is:

```
DISKCOPY [drive 1:][drive 2:]
```

Drive 1: is the disk drive that contains the disk that you want to copy; drive 2: is the disk drive that contains the blank formatted disk. For example, if you want to make a copy of your system disk which is in drive A:, type:

DISKCOPY A: B:

The system will display:

```
Insert source diskette into drive A:<CR>
Insert formatted target diskettes into drive B:(<CR>)
Press any key when ready
```

Since we want to copy the system disk, we will leave it in drive A: If we wanted to copy another disk, you would take the system disk out and insert the other disk in drive A: before depressing the RETURN (<CR>) key.

The cursor will drop to the second prompt after the disk has been written into memory. After you have a formatted blank disk in drive B: depress the RETURN key. The files will be read from memory and written to the disk. After this is done, the following prompts will be displayed:

```
Copy complete
Copy another (Y/N)?
```

Type **Y** (for yes) if you wish to copy other disks or make a second copy. If you type **N** (for no), the default drive prompt is displayed and you are back on the operating system level.

Note: If either of the disks you are using has defective tracks, DISKCOPY will not work. Use the COPY command to back up your disks in these cases. The COPY command will skip over defective tracks.

SYSTEM RESET

The operating system may be reset by two methods:

1. Power On/Off

Each time power is applied, the system will restart. This is referred to as a "cold boot". All data not saved to disk is lost upon resetting the system.

2. RESET Switch

The system may be reset by pushing the RESET switch located on the left rear edge of the keyboard. All data not saved to disk is lost upon resetting the system. The power remains on when you use the RESET switch. This is reffered to as a "cold boot".

If you encounter an error that results in making the keyboard inoperative, you must reset the system.

Note: Reset the system only when it is definitely necessary. Do not try to reset too quickly in rapid successions.

CHAPTER 2

COMPUTER VOCABULARY

Introduction

In the Sanyo software documentation for the 550 series computers many words are used which are common to users of computers. Many of these are difficult to find in dictionaries in the context of their meaning for our use. The following list is a glossary of the majority of the special words used in describing computer functions and parts.

Vocabulary	Description
A:	MS-DOS prompt, waits for your entry parameter.
ADDRESS	A particular number associated with a memory location. Usually an address is a number between 0 and 65535 (or &H0000 and &HFFFF hexadecimal).
ADDRESS BUS	A set of electronic paths which carry the binaryencoded address from a microprocessor to the computer internal architectures.
ALU	Abbreviation for Arithmetic Logic Unit.
AND	A binary function that is "true" if and only if all of its input parameters are "true".
APPLICATION SOFTWARE	The collection of programs that a user can use to perform a specific task. Examples: an inventory program, a general ledger program, or a word processor.

ARRAY	A set of data elements referred to as a group.Usually referred to with a series of subscripts. For example: A 100-element array can be expressed as A(100).
ASCII	An acronym for American Standard Code for Information Interchange. This standard code assigns a unique value (from 0 to 127) to each of 128 letters, special characters, control characters, and numbers.
ASSEMBLER	A program which converts the characters (symbols) of an assembly language into machine language, generating operation code.
ASSEMBLY LANGUAGE	A language made up of mnemonics and symbols. Special groupings do special functions.
B:	MS-DOS prompt for a second drive, waits for your entry parameter.
BASIC	An acronym for "Beginner's All-Purpose Symbolic Instruction Code". BASIC, a higher-level language,was published at Dartmouth College in the early sixties.
BAUD	Bits per second. The rate information flows between devices.
BINARY	A number system with two ciphers: "0 and 1". Each digit in a binary number represents a power of two.

BINARY FUNCTION	A logical or arithmetic operation performed by a computer which has multiple inputs and only one output. Binary signals are usually used.
BIOS	I/O system allocation for MS-DOS.
BIT	A Binary entity that is the smallest amount of information which a computer can manipulate. A bit indicates a single value: "0" or "1". Bits can be added together to form larger words called "bytes".
BOOTSTRAP ("BOOT")	To initialize a system from a "power-off" condttion. This is usually a small program that is run immediately upon power-up.
BUFFER	A register or memory which is used to hold data temporarily. Screen buffers contain information to be displayed on the video screen. Keyboard buffers hold formed input lines. Register buffers hold intermediate data.
BUS	A group of electronic paths in a computer which permit data flow from one place to another.
BYTE	A basic unit of memory in a computer's architecture.Usually comprises eight bits. Arranged in a row,bit 7 to bit 0--Bit 0 is least significant in value.
CALL	An instruction which calls a subroutine in a program, transferring execution to another location.

CHARACTER	Letters (both upper and lowercase), numbers, various symbols, and graphic descriptors for human and computer use.
CODE	A way of expressing quantities in another form.As an example, the ASCII code represents characters as binary numbers. Languages represent math relationships in terms of program operators.
COLD-START	A start-up operation from a "power-off" state.
COMPUTER	A man-made architecture that can receive and store instructions and execute them. Computers can input, process, store, and output data.
CONCATENATE	To join or merge files or data into a new composite entity.
CONNECTOR	A physical device for attachment of conductive paths. I/O to peripheral devices is through connectors.
CONSOLE	The defining label for the immediate input and output device. Here the keyboard and the CRT.
CONSTANT	A set of data which will not change. (i.e.: The values 1, 2, 3, 4, 5, or 3.14 are constants.)
CONTROL(CTRL) CHARACTER	In the ASCII character set there are characters which have no graphic symbol and are used to control or direct various functions. A BREAK character, for example, aborts execution.

CPU	Abbreviation for Central Processing Unit.
CR	Abbreviation for Carriage Return <CR>, Return or Entry Key.
CRT	The "Cathode-Ray Tube", or video screen of a computer's output function.
CURSOR	A position symbol which tells an operator where display text is to be entered.
DATA	Information to, from, or in a computer.
DB25	A type of connector with 25 pins (or conductors) used for typically RS-232C I/O communications.
DECIMAL	Usually of ten states.
DIAGNOSTIC	Usually a software program that is used to evaluat the operational performance of a machine.
DIR	MS-DOS Acronym for a directory list command.
DISASSEMBLER	A program which analyzes machine language and converts it into assembly language.
DISPLAY	An ouput device or peripheral of a computer, usually a video or CRT screen.
DUMP	To empty out the contents of, as in to DUMP memory to a printer or other device.

ENTRY POINT	A location used by software that contains the first executable command in that section.
EXCUSIVE-OR	A binary function whose result is "false" only if all of its inputs are "true".
EXECUTE	A performance of a command, instruction, or software program. A program that is "RUN" is executed.
FILE	A location or storage element into which data can be placed.
FLOPPY	The jargon for the mini-diskette (both 5 1/4" and 8") for use in the disk drive I/O.
FORMAT	The physical rule or characteristic for entry or output. Can also be a verb.
FUNCTIONS	A derived sequence of events that manipulate data, variables, and constants.
GRAPHICS	The character set for video or printer output. Can also refer to special graphic characters or modes.
HARDWARE	The touchable components of any computer.
HEXADECIMAL	A numbering system which uses 0 through 9 and the six letters A through F to represent values. All hexadecimal numbers are preceded by &H.

HIGH-LEVEL LANGUAGE	A language which makes sense to human intelligence.
HIGH-ORDER	The highest value. Used in talking of bits in a byte. The high-order bit of a byte is the highest place binary value.
I/O	Abbriviation of Input/Output.
INDEX HOLE	Floppy disk awareness (physically) of the start of each track of information. A hole is punched in the diskette. When it passes a sensor an awareness is made.
INPUT	Data entering a computer from the outside world. It can come from a console or a remote peripheral.
INPUT/OUTPUT (I/O)	Softwareroutines orhardware architectures which receive or transmit data with peripherals external to the computer.
INSTRUCTION	A command in a program that a computer can execute. In a high-level language, instructions may be comprised of many characters.
INTERFACE	A device or set of logics which make I/O possible.
INTERPRETER	A special program which interprets and executes a higher-level language. Most interpreters are written in machine language.

INTERRUPT	Interrupts are used to signal the computer that an external device has a status or some data for the computer to process.
K	In common use in a computer "K" usually represents 1024.
KILOBYTE	1,024 bytes.
LANGUAGE	Computer language express a communication tool beween programmer and computer.SANYO BASIC, PASCAL, FORTH, and FORTRAN are all languages.
LINE	A "line" is a horizontal grouping of graphic characters. Lines are used for both printers and in displays.
LOW-LEVEL LANGUAGE	Communication language more machine rather than human based.
LOW-ORDER	The least important element in a grouping. The low-order bit of a byte is the bit with the least binary value.
MACHINE LANGUAGE	Language for computer hardware use. Most machinelanguages are binary and are in hexidecimal.
MBC	A Sanyo acronym for micro Business Computer. Prefix for members of the Sanyo MBC-1100/1200/4000/550 series.

MEMORY LOCATION	The smallest element in a memory that a computer can address. Memory locations are labeled with a unique address.
MEMORY MAP	The set of all possible memory locations a microprocessor can communicate with. Also used to describe a system's memory.
MICROCOMPUTER	A computer whose architecture is reliant upon a microprocessor.
MICROPROCESSOR	A single integrated circuit computer which performs the basic I/O functions and executes machine language programs.
MNEMONIC	An easy way to remember something used in the place of something more difficult to remember. An abbreviation.
MODE	A state or set of conditions for which a certain set of rules, formats, or dates apply.
MODULO	An arithmetic function which manipulates two items.
MONITOR	1) A computer controlled television CRT or other display receiver. 2) A program which permits user access to computer architecture at the byte level.
MS-DOS	A Disk operating system for 8086/8088 micro chip computers.
MULTIPLEXER	A device or software set with many inputs and one output. Multiplexers select one of its many datainputs to be its output.

NYBBLE	Half of a byte or four bits. Can be upper 4 or lower 4.
OPCODE	A binary machine language command.
OPERATING SYSTEM	The software architecture under which a computer communicates and functions with a human operator.
OR	A binary function whose result is "true" if at least one of its inputs is "true".
OUTPUT	Data leaving the computer into the outside world. Also refers to the process of generating same.
PAGE	A full page or screen of information on a video or CRT display.
PASCAL	A French scientist after whom an interpretive language was named.
PERIHERAL	An external device attached to the combbputer peripherals are input and/or output devices which communicate through the I/O.
PINOUT	A description of the function of each pin on a connector or an integrated circuit.
PIXEL	The smallest piece or "dot" of resolution on the computer's display screen.

PRINTED CIRCUIT BOARD	The real estate for electronic circuits. Sheets CIRCUIT BOARD of fiberglass or epoxy with copper conductors "etched" onto the surface. Components mount onto the traces.
PROGRAM	Sequences of commands or instructions which define a desired effect or achievement.
PROM	Acronym for "Programmable Read-Only Memory". PROMs are ROMs that can be altered. Data in PROMs is retained even if power is turned off. Some PROMs can be erased and reprogrammed.
RANDOM-ACCESS MEMORY (RAM)	Fast read-write memory for general use by a computer. The computer stores values in RAM locations and recalls them for use at any time. Data in RAM memory is lost when the power to the computer is turned off.
READ-ONLY MEMORY (ROM)	Slower memory for constant storage of programs or data for use by the computer when the power is first turned on. Information in ROMs cannot be changed. Information in ROMs is retained when the power is turned off.
RETURN	To complete a subroutine and return back to the main program that "called" it.
RS-232C	Standard for serial communications. Serial structure for I/O.
RUN	To begin the sequence of instructions or commands of a program.

SANYO	A worldwide company that through engineering expertise has brought you this computer series.
SANYO-BASIC	A high level user friendly interactive language developed by SANYO.
SCAN LINE	The sweep of a cathode beam across the face of a cathode-ray tube or CRT, or other video display.
SCHEMATIC	An engineering diagram which depicts the interconnections between electronic devices.
SCROLL	To move text on a display upwards making room for text at the bottom of the display.
SECTOR	A portion of a track of data on a floppy diskette.
SOFTWARE	Programs, files, and data executable on the computer.
STACK	A reserved part of memory used to store temporary information.
SUBROUTINE	A much-repeated section of a program which can be accessed by "calling".
SYNTAX	The rules for instruction expression in a given language. A mistake in entering an instruction may result in a "SYNTAX ERROR".

SYSTEM DISK	A special diskette usually provided with the system that contains the operating system, languages, utilities, monitors, diagnostics, and other software for system use and initialization.
TEXT	Symbols entered for and by a program.
TRACK	One "band" of data on a floppy diskette.
VARIABLE	A location in memory where data can be placed. Also refers to the actual data itself.
VECTOR	A memory or program pointer for a "jump-to" location. Vectors are used to re-direct processor activity.
WARM-START	A restart of computer operations after an abortion of operating system control.

CHAPTER 3

SANYO-BASIC

SECTION 1
SANYO BASIC OPERATION MODE

HOW TO ENTER SAYNO BASIC

At the A: prompt of the operating system, enter:

```
BASIC [<filename>]
      [/F:< number of files to be used>]
```

Description

(1) <filename> is the Sanyo Basic program filename to be executed after you enter Sanyo Basic When this is specified, <file name> program is automatically loaded and executed. If it was omitted, the system enters key-in wait state after displaying prompt mark with Sanyo Basic mode.

(2) <number of files to be used> defines the maximum number of data files which are opened by a program at one time. If this is omitted, the system assumes 3 as its default value. The maximum number of files can be opened at one time is 15 using Sanyo Basic. (The system allocates the buffer area according to the specified number of files.)

Example

```
>BASIC
>BASIC "PRGM-1"
>BASIC "PRGM-1" /F:5
```

Prompt Mark and Cursor

When the system enters key-in waiting state after the Sanyo Basic activation or after an execution, the state is called a Sanyo Basic mode. The Sanyo Basic mode displays the following prompt mark:

```
Ready
> Cursor (blink)
```

Whenever the prompt mark is being displayed, Sanyo Basic instruction (program) can be entered.

Direct Execution Mode

If a program statement without line number and RETURN Key were entered while a prompt mark is being displayed, the entered statement is immediately executed. This is called a direct execution. The operation result of the direct execution is stored in the memory, although the instruction statement is deleted. Multi-statement can also be directly executed.

Examples

```
>A=3.14*5.8^2:PRINT A
 105.63

>FOR I=1 TO 5:PRINT I:NEXT I
 1
 2
 3
 4
 5
```

Program Edit Mode

The program edit mode means to enter, to edit, and to store a Sanyo Basic source progam into the memory.

(1) Line edit

When a line number and a statement are entered, the system stores them into the memory as a source program. Line numbers can be automatically generated using the AUTO instruction. A renumbering operation can be performed to assign numbers to an entire source program using the RENUM instruction. "A line number and a statement" can be repeated and a series of program is stored to the memory sequentially. Then this program can be executed using the RUN instruction.

(2) Screen edit

When the cursor key (↑) is entered in the direct execution mode, the control becomes the screen edit mode.

The displayed program can be partially modified. Using the screen edit mode, the cursor can be freely moved to the place where the user wants to make correction so that he can insert and delete. After modifications, the cursor returns to its original prompt mark position and the system enters Sanyo Basic mode if the BREAK key is entered. The program is modified a appeared on the screen and stored into the memory when the BREAK key is entered.

Execution Mode

The program generated and stored into the memory by the program edit mode can be executed sequentially in order of line number using the RUN,GOTO, and GOSUB instructions. The program can be temporarily terminated using the BREAK key or the STOP instruction in the program. The interrupted program can be restarted using the CONT instruction. Although it cannot be restarted if a modification was made to the source program during the temporary termination.

The system returns to the Sanyo Basic mode after executing the END statement in a program or a source program is completed (no more statements to be executed) from the execution mode.

Returning to OS

When the SYSTEM instruction is found in a program or the SYSTEM instruction is directly executed from Sanyo Basic, the control is given to the operating system (OS mode) from the Sanyo Basic interpreter. The system enters the OS mode whenever the RESET key is pressed, if the system disk is in drive A.

SYMBOLS USED FOR SYNTAX DESCRIPTION

The following symbols are used to describe program syntax in this manual:

[] Items enclosed by [] can be omitted. The default operation varies according to the instruction. See the explanation of each instruction.

{} Items enclosed by {} can be either omitted or repeated more than once.

<> Items enclosed by <> can be defined within the prescribed range but cannot be omitted.

| One of the items placed on the left and the right of vertical bar, |, must be selected. The range to be selected is underlined if it is not clear.

Characters and symbols

Characters and symbols other than above that appear in general descriptions must be written as they appeared, because they have special meanings.

Character and Symbols Used in Programs

The following characters and symbols are used in Sanyo Basic:

26 uppercase letters	ABCD. . . Z
26 lowercase letters	abcd. . . z
10 numerics	0123456789
32 symbols	

!	(Exclamation point)	;	(Semicolon)
#	(Number sign)	$	(Dollar sign)
%	(Percent)	&	(Ampersand)
(	(Left parenthesis)	*	(Asterisk)
)	(Right parenthesis)	+	(Plus sign)
-	(Minus sign)	,	(Comma)
:	(Colon)	<	(Less than)
?	(Question mark)	>	(Greater than)
@	(At sign)	[	(Left bracket)
_	(Underscore)	]	(Right bracket)
{	(Left brace)	"	(Double quotation mark)
}	(Right brace)	'	(Single quotation mark)
\	(Backslashe/integer)	/	(Division symbol)
=	(Equal sign/assignment)		(Space)

Lower case letters are converted to upper case letters for storage in memory. Other characters are available in a programmable format. If the lower case letters are regarded as strings (literal data), they are stored as they are. Usable characters correspond to the internal codes of Sanyo MBC-550 series computers and are displayed on the screen. However, the amount of useable characters that will be accepted by a printer is less. For printing out programs, we suggest determining the range of printable characters by the printer and using only those in the program.

A character is written into the program by depressing the corresponding key on the keyboard. Special

characters are created only by the multiple use of keys. In the Sanyo Basic programs, the special characters in Columns 0 and 1 of the character codes table referenced at the back of this manual cannot be keyed in through the keyboard. They are written in the programs only by the use of CHR$ functions.

Example

```
>10 PRINT CHR$(07)
    (Rings Bell)
>20 PRINT CHR$(45)
    (Prints - )
```

LINE NUMBERS

A line is the minimum unit of a program consisting of a line number, a group of statements (descriptions of instructions), and a RETURN Key (to terminate a line). The general format of a line is as follows:

```
<line number><statement>{:<statement>}
```

Example

```
>10 A=3:B=100:C=A+B:PRINT C
>RUN
 103
```

A line number is a five-digit positive integer in the range from 0 to 65535. (Two bytes are allocated for a line number in memory.) The leading zeros are ignored.

No spaces can be included in the numeric parameter to show a line number.

Examples

```
>00100 A = 3 (Same as 100,leading zeros ignored)
>1 1 0  A$ = "THREE" (Erroneous spaces)
>120 PRINT A, A$
>130 END
>RUN
Syntax error in 1
```

In Sanyo Basic the carriage return is keyed in by use of the keyboard's <CR> key(corresponds to RETURN key), The <CR> key is used to terminate a logical line, The keying-in of data during an operation is terminated by use of the RETURN key.

A line number indicates the index for program operation control and is also used for line search in program editing.

<statement> is an instruction described within the Sanyo Basic syntax rules and is also the minimum unit or element to compose a program.

Colon (:) is a delimiter placed between one statement and another in composing several statements into a single program line. A program line composed of a single statement is called "Simple Statement", and one composed of a several statements, which are delimited with colons, is called a "Multiple Statement".

<CR> indicates the end of a logical line and is entered by use of the RETURN Key.

The length of a line cannot exceed 255 characters including the line number, and spaces.

KEYWORDS AND SPACES

The commands, statements, and function names are the **key words** in Sanyo Basic and do not include spaces between letters. In addition, the keywords, variables, constants, and logical operators must be separated by spaces, enclosures, numeric operators, or other delimiters syntactically recognized. The use of spaces in other cases is at the programmer's option and is ignored by the system.

Examples

```
>10 P R I N T <CR>   (Erroneous)
>20 A = B + C     (Valid)
>30 A = B AND C  (A is assigned as a logical AND
                   operation of A and B
>40 A = BANDC     (BANDC becomes a variable name)
>RUN
Syntax error in 10
```

For keywords, refer to the list of reserved words in chapter 5.

DATA

Data is written with numbers or characters for processing in the program. Data with numbers is regarded as "numeric data" and the data with characters as "string data". Either a constant or variable can be assigned as a data item. The variables are of two types: i.e., simple variables and subscripted variables (an array). The numeric data is divided into three categories, namely integers, single-precision real numbers, and double-precision real numbers.

Examples of data are shown in the following table:

	String	Integer	Numeric
CONSTANTS	"VALUES" "Your name?"	245 32	3.678 1234.567# -3.45E35 .345!
SIMPLE VARIABLES	A$ SUM$	ABC% K%	DAT H!
ARRAY VARIABLES	B$(10,4,8)	X%(N)	Z!(I,J) Y#(2,3)
MEMORY USAGE	1 byte/char.	2 bytes	4 bytes or 8 bytes
OCTAL NUMBERS		&7632	
HEXADECIMAL NUMBERS		&H3AF4	

Constants

1. Integer constants

Integers between -32768 and +32767 with no decimal point or fraction.

Examples

```
>10 A% = 365
>20 B% = -12345
>30 PRINT A% + B%
>40 END
>RUN
-11980
```

2. Single precision real constants

7-digit numbers (for display only 6 digits) between -1.7 x 10 - 1.7 x 10 that are expressed in the following three ways:

1) Numbers up to 7-digit and 6 digits are displayed. Example: 1.234567 (1.23457 is displayed)

2) Number with an exponent (E) Example: 1234567E-6 (1234567 x 10)

3) Number with a suffix ! Example: 1.234567!

3. Double precision real constants

16-digit numbers between -1.7 x 10 - 1.7 x 10 that are expressd in the following three ways:

1) Numbers of 8-digit and more Example: 12345678

2) Numbers with a exponent (D) Example: 1234567D7 (1234567 x 10)

3) Numbers with a suffix # Example: 12345#

2. Character constant

Characters enclosed in quotation marks (" ") within 255 characters.

Examples

```
>10 A$ = "SANYO "
>20 B$ = "MBC-SERIES"
>30 PRINT A$;B$
>40 END
>RUN
SANYO MBC-SERIES
```

The double quotation marks (" ") cannot be used within a string constant.

Example

```
>10 C$ = ""SANYO-MBC SERIES""
>20 PRINT C$
>RUN
Syntax error in 10
```

Quotation marks may be printed by using the CHR$(34) symbol.

Example

```
>10 C$ = "SANYO-MBC SERIES"
>20 PRINT CHR$(34); C$; CHR$(34)
>RUN
"SANYO-MBC SERIES"
```

Variables

1. Variable name

Variables are the changeable values in a program and are referred to by specifying their names. The variable names are written with an alphanumeric string of optional length beginning with an alphabetical character. The first sixteen characters identify the variable.

Example

```
>10 TR = 10
>20 TRE = 20
>30 TRANSPOSITION = 30
>40 PRINT TR, TRE, TRANSPOSITION
>RUN
 10            20            30
```

The variable names must not be spelled the same as keywords, but may include their spelling. Any variable name beginning with FN or USR is erroneous.

Example

```
>10 AND = 100     (Erroneous, AND is an operator)
>20 BAND = 0        (Acceptable)
>30 AFN10 = A + B  (Acceptable)
>40 FNABC = 1      (Erroneous, FN is an operator)
>50 BAFN = 1       (Acceptable)
```

2. Types of variables

The variables are classified into strings or numerics (integers, simple-precision real numbers, and double-precision real numbers).

The type of data is declared by the DEF statement or by attaching a suffix at the end of the variable as a type specification symbol.

Type of variable		Symbol	Example
Numerics	Integer	%	A%, INTEGER%
	Single-precision real number	!	RO! REAL!
	Double-precision real number	#	B5#, DBL#
Characters	(String) $		LITERAL$

When a variable has no type specification symbol or is not declared by the DEF statement, a single-precision real number is assumed. (The default type for a numeric variable name is single-precision.)

3. Arrays (Subscripted variables)

Arrays are declared by the DIM statement before they are used. There is no limitation on an array's dimension or number of elements. However, the whole length should fit in the total storage capacity.

An array which consists of 10 or fewer members may be used even if not declared by the DIM statement. (The declaration is assumed to have been done when the array is used for the first time.)

Expressions and Operators

An operator is a symbol to show the operational method (i.e. adding, subtracting, or comparing the date). A combination in which some number of data items are related by operators according to syntactic rules is called an expression. One expression produces one value as a result of the data operation performed, in the order that the operators specify and the priority of the operator. We call that value, in this manual, an evaluation result which naturally shall be either numeric or string data.

ARITHMETIC OPERATORS

Arithmetic operators are divided into the following eight types:

		OPERATOR	EXAMPLES	
1	^	Raising to a power (exponentiation)	A ^ B	Raise A to the Bth power
2	-	Minus sign	-A	Minus A
3	*	Multiplication	A * B	Multiply A by B
4	/	Division	A / B	Divide A by B
5	\	Integer division	A \ B	Divide A by B, and discard the decimal part.
6	MOD	Remainder	A MOD B	Remainder of A divided by B **(Both A and B are rounded off to integers before being divided.)**
7	+	Addition	A + B	Add A and B
8	-	Subtraction	A - B	Subtract B from A

ARITHMETIC EXPRESSIONS

An arithmetic expression is a combination of numeric data items and arithmetic operators, generally written as follows:

```
<data item>{<arithmetic operator><data item>}
```

The data items to be written are classified below:

1. Numeric constant
2. Numeric variable
3. Numeric function
4. (Numeric expression)

The operation of an arithmetic expression is performed in the following order according to its priority as shown below. Sequence, hence, is important.

Order of Precedence:

1. Calculations enclosed in parenthesis are always evaluated first and have the highest precedence.
2. Power calculation or exponentiation.
3. Minus sign or negation.
4. Multiplication and division.
5. Integer division.
6. Remainder.
7. Addition and subtraction.

The calculations which have the same operational priority are performed from left to right.

Example

```
>10 A = 2*(2+1)*2/3+1-2
>20 PRINT A
>RUN
 3
```

RELATIONAL OPERATORS

Relational operators are used to compare one data item to another. They are divided into the following six types:

1) =	Equal?	A=B Is A equal to B?
2) <>	Unequal?	A<>B Is A unequal to B?
3) <	Less than ?	A<B Is A less than B?
4) >	Greater than?	A>B Is A greater than B?
5) <=	Less or equal?	A<=B Is A less or equal B?
6) >=	Greater or equal?	A>=B Is A greater or equal B?

RELATIONAL EXPRESSION

A relational expression is a combination of data items and relational operators and is generally written as follows:

```
<data item>{<relational operator><data item>}
```

The data item is an arithmetic expression or character string. Data items written in relational expressions must be the same type; i.e., arithmetic expressions or character strings. (Arithmetic expressions cannot be compared with string data or vice versa.)

When the data items are related to each other with relational operators, the evaluation result will become -1 if the expression is affirmed (true) and 0 if not affirmed (false).

Comparison of the numeric values is performed in two ways-- whether or not one value is smaller/greater than the other.

The size factor determines the comparison of two numeric values, that is, the comparison of numeric values is performed by determining whether or not one value is smaller/greater than the other.

The comparison of character strings is performed by the following procedure:

1. The value of one character is determined in accordance with the table of ASCII character codes.

2. The value of strings having more than two characters are compared as follows:

The value of comparison based on Rule 1 above should be done on the first pair of characters, then on the second, and so forth.

The string in which a character with a smaller code value first appears is determined as smaller.

If either string ends but determination is not made yet, the shorter one is regarded smaller.

Example

COMPARISON	RESULT	
"A" < "B"	-1	TRUE (ASCII value 65 is less than 66)
"?" < "="	0	FALSE
"ABC" < "ABCA"	-1	TRUE
"ABA" > "AA"	-1	TRUE
"XYZ" < "ABC-1"	0	FALSE

If more than two relational operators are included in an expression, the operation is performed from left to right under priority rules.

LOGICAL OPERATORS

Six logical operators are described below in order of operational precedence.

Order of Precedence:

1.	NOT	Negation
2.	AND	Logical product
3.	OR	Logical sum
4.	XOR	Exclusive logical sum
5.	IMP	Implication
6.	EQV	Equivalent

LOGICAL EXPRESSION

A logical expression generally has the following forms:

```
[NOT]<data item>{<logical operator>[NOT]<data item>}
```

Either an arithmetic expression or a relational expression can be written as the data item. In case of an arithmetic expression, its evaluation result should be an integer between -32768 and +32767. If the evaluation result were a real number, it would be rounded off. If it falls out of the range an error occurs.

A logical expression evaluates the result with the following procedure:

1. When more than two logical operators are included:

   a. Execute the processing in accordance with the priorities.

   b. If the two different logical operators have the same priority, execute from left to right.

2. When the value of <data> is assigned with either -1 (true) or 0 (false), the evaluation of the logical expression will become -1 or 0.

3. When an integer is included in the evaluation result of <data item>, the logical operation is performed with binary notation (binary integers are allocated for each bit) regarding the result as a binary number. The evaluation result will be converted into the integer.

Examples

1) Definition of logical operators

NOT	X	NOT X
	1	0
	0	1

AND	X	Y	X AND Y
	1	1	1
	1	0	0
	0	1	0
	0	0	0

OR	X	Y	X OR Y
	1	1	1
	1	0	1
	0	1	1
	0	0	0

XOR	X	Y	X XOR Y
	1	1	0
	1	0	1
	0	1	1
	0	0	0

IMP	X	Y	X IMP Y
	1	1	1
	1	0	0
	0	1	1
	0	0	1

EQV	X	Y	X EQV Y
	1	1	1
	1	0	0
	0	1	0
	0	0	1

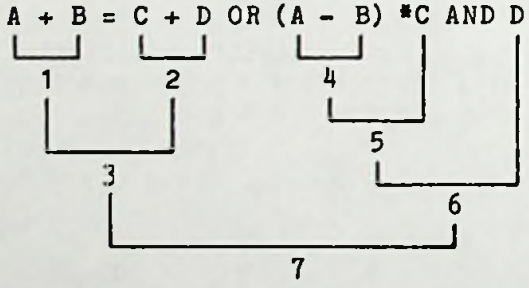
2) Logical operation including integer(s)

```
                                        Binary          Result
          15 AND 16
                 15        ->          01111
                 16        ->          10000

          15 AND 16        ->          00000     ->      0

          15 AND 10
                 15        ->          1111
                 10        ->          1010

          15 AND 10        ->          1010      ->     10

          -1 AND 16

 -1 ->    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 16 ->    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

 -1 AND 16 ->0 0 0 0 0 0 0 0 0 1 0 0 0 0 ->      16

           8 OR 16
                  8        ->          01000
                 16        ->          10000

           8 OR 16         ->          11000     ->      24

          15 OR 16
                 15        ->        1 1 1 1
                 16        ->      1 0 0 0 0

          15 OR 16         ->      1 1 1 1 1     ->      31
```

EVALUATION OF COMBINED EXPRESSIONS

A relational expression and an arithmetic expression can be written as a data item in a logical expression, and an arithmetic expression can be written as a data item in a relational expression. So, a combination of different expression is performed in the order of 1) arithmetic expression, 2) relational expression, and 3) logical expression.

When A=1, B=2, C=2 and D=1, the evaluation result ofthe following expression will become -1. With the evaluation following these steps:



Please note that "=" used in the LET statement and "=" as a relational operator are different in meaning.

Example

```
>10 LET A = 2 = 3   (0 is assigned to A)
>20 LET C = B = 1   (-1 is assigned to C when B=0
                     0 is assigned to C when B=1)
```

CONVERSION TYPES OF NUMERIC VALUES

1. Conversion to Integer

Rounds up to the decimal point.

Example

```
>10 A%=1.46:B%=1.5
>20 PRINT A%;B%

>RUN
 1 2
```

2. Conversion of Double Precision to Single Precision

A number with up to 16 significant digits is converted to a 6 digit (internally 7 digits) number.

Example

```
>10 A!=1.2345678
>20 PRINT A!

>RUN
 1.23457
```

An eight digit number is converted to a six digit number.

3. Conversion of Single Precision to Double Precision

Trailing zeros are added to the single precision number. The significant digit is seventh, but some meaningless numbers are added after the seventh digit.

Example

```
>10 A!=8.9
>20 B#=A!
>30 PRINT B#

>RUN
 8.999999618530214
```

CALCULATION RESULTS

The type of calculation results are as follows:

Calculations by +, - and * are executed to match the high order type (order if a single precision real number is higher than the order of an integer, and the order of a double precision real number is higher still), and the results of the calculations are of the corresponding type.

However, when the results of calculations of an integer are outside the integer range, the results are of the single precision real type.

For calculations by ^ and /, an integer is converted to the single precision real type, and then calculations are executed to match the high order type. The results are of the corresponding type.

Calculations by relational operators are executed to match the high order type, and the results are of the integer type.

Calculations by \,**MOD**, and logical operators are executed after conversion to the integer type, and the results are of the integer type.

STRING EXPRESSION

A STRING EXPRESSION combines one character string with another and is written in the following format:

```
<data item>{+<data item>}
```

One of the following three can be written as a data item:

1. String constant
2. String variable
3. Function which results in a string (Character functions)

The evaluation result of a string expression is also a character string with less than 255 digits. When the string length exceeds the limit an error occurs.

Examples

```
>10 A$ = "SANYO "
>20 B$ = "M"
>30 C$ = "B"
>40 D$ = "C"
>50 E$ = "-SERIES"
>60 PRINT A$+B$+C$+D$+E$
>RUN

SANYO MBC-SERIES
```

SECTION 2
GENERAL INSTRUCTION WORDS

INTRODUCTION

All Sanyo Basic instruction words can be directly executed as well as written in a user program. Therefore, there is no distinction between commands and statements, although some statements may cause the system to enter Sanyo Basic mode or OS (Operating System) mode instead of going to the next step in a program.

Some Sanyo Basic instruction words may be entered with two or three key strokes. The following table indicates the keys and corresponding instruction words:

KEYS	INSTRUCTION
CTRL +SHIFT+A	AND
CTRL + A	AUTO
CTRL + B	BEEP
CTRL + C	CLOSE
CTRL+SHIFT+C	CLS
CTRL + D	DELETE
CTRL+SHIFT+D	DIM
CTRL + E	ELSE
CTRL+SHIFT+E	END
CTRL + F	FOR
CTRL + SHIFT + F	FIX(
CTRL + G	GOTO
CTRL+SHIFT+G	GOSUB
CTRL + H	HEX$(
CTRL + I	IF
CTRL+SHIFT+I	INPUT
CTRL + K	KILL"
CTRL + SHIFT + K	KEY
CTRL + L	LOAD"
CTRL+SHIFT+L	LIST
CTRL + M	MERGE
CTRL+SHIFT+M	MID$(
CTRL + N	NEXT

CTRL + SHIFT + N	NOT
CTRL+SHIFT+O	ON
CTRL + O	OPEN"
CTRL + P	PRINT
CTRL+SHIFT+P	PUT
CTRL + R	RUN
CTRL+SHIFT+R	RETURN
CTRL + S	SAVE"
CTRL+SHIFT+S	STEP
CTRL + T	THEN
CTRL+SHIFT+T	TIME$
CTRL + U	USING
CTRL + V	VAL(
CTRL+SHIFT+V	VARPTR(
CTRL + W	WHILE
CTRL+SHIFT+W	WEND
CTRL + X	Line edit mode

Instruction words in the following section are alphabetically ordered for explanations. For information on some instruction words listed in the previous table, but not explained in the following sections, consult the Sanyo Software Basic Reference Manual, available at your authorized Sanyo computer dealer.

AUTO

Purpose

The AUTO is used to create line numbers automatically in the line edit mode at the time of program writing.

Form

```
AUTO [<line number>[,[<increment>]]]
```

Description

(1) When <line number> specification is made, the system displays the specified line number and waits for the key-in of a statement. When the statement is keyed in and <CR> is pressed, the system displays a new line number, which is the current line number plus the increment and waits for another statement to be entered. The same process is repeated.

(2) When the line number specification is omitted, the creation of line numbers starts from 10. When the specification of <increment> is omitted, 10 is assumed. (The line numbers go 10, 20, 30 and so on.)

(3) When the line number is specified as <line number>, only the previous increment (if one has been used) is designated.

(4) The AUTO mode is stopped by pressing the <BREAK> key and the control returns to the Sanyo-Basic Mode. When the line number exceeds 65535 or when a line edit error is found in the keyed-in statement, the system displays an error message and returns the control to the Sanyo-Basic Mode .

(5) .(period) can be used to specify the line number of the last line + increment as <line number.>

(6) When 0 is assigned to <line number>, an error message is displayed.

Example

```
>AUTO 100,20
100
120
140
160

>AUTO.    (The final line number of a program
           which is stored in memory will be
           displayed.)
```

BEEP

Purpose

The BEEP sounds a buzzer.

Description

(1) The buzzer is generated for 0.5 seconds.

Form

Example

```
>10 REM TEST FOR BEEP
>20 PRINT "SANYO!"
>30 BEEP
>40 END

>RUN
SANYO!
(Sounds a buzzer for about 0.5 seconds.)
```

CLS

Purpose

The CLS clears entire CRT screen.

Form

Description

(1) When the CLS statement is specified, the visual images are all erased. The image attributes are then reset. The cursor appears at the position of Line 1 and Row 1 (the upper-left side of the screen).

Example

```
>10 REM CLS STATEMENT TEST
>20 X$ = "PG. 2":Y$ = "PG. 1"
>30 PRINT Y$
>40 INPUT "CONTINUE? (Y OR N)";Z$
>50 IF Z$ = "N" THEN 50
>60 CLS
>70 PRINT X$
>RUN
```

CONT

Purpose

The CONT resumes a program which has been halted.

Form

```
CONT [N]
```

Description

(1) This instruction restarts the program which has been temporarily terminated by END, STOP instructions, or < BREAK> key.

(2) If the program was modified during interruption, CONT in invalid. Therefore, even if it is entered, the execution will not be restarted.

(3) CONT N inhibits temporary termination using the BREAK key during program execution. This CONT N mode is released by reexecuting CONT.

Example

```
>10 PRINT "'START'"
>20 PRINT "Please 'CONT'"
>30 STOP
>40  '
>50 PRINT "'END'"

>RUN
'START'
Please 'CONT'

Break in 30

Ready
>CONT
'END'
```

DELETE

Purpose

This instruction deletes a part of the program in the Program Edit Mode. After execution, control returns to the Sanyo-Basic Mode.

Form

```
DELETE [<line number 1>][-<line number 2>]
```

Description

(1) The following combinations are available for the specification of line numbers:

a. <line number 1>-<line number 2>
Deletes the part between the specified two lines

b. <line number> - . (period)
Deletes the part from the specified line number up to the end of the program.

c. -<line number>
Deletes the part from the beginning of the program up to the specified line number.

d. <line number>
Deletes the line with the specified line number only.

(2) When no program exists in the range specified for deletion an error occurs.

(3) The NEW statement is used for the entire deletion of the program.

(4) A period (.) indicates the final line of the program.

Example

```
>10 A% = 26 * 82
  '
  '
>90 PRINT A%
>100 B% = A% * 2
>110 PRINT B%
>DELETE 20-90
>DELETE 100-.
```

DIM

Purpose

The DIM instruction declares an array specification.

Form

```
DIM <variable>(<subscription maximum value>
{,<subscription maximum value>}){,<variable>(
<subscription maximumvalue>
{,<subscription maximum value>})}
```

Description

(1) <variable> indicates the name of the array.

(2) <subscription maximum value> can be described using numeric expression. If the evaluated result is a real number, it is automatically rounded. The smallest value of a subscript is 0. When DIM A(10) is declared, the number of elements will be 11(0 to 10). The smallest value can be set as 1 using OPTION BASE.

(3) The number of <subscription maximum value> in parentheses determines the array dimension size. No limits are prescribed for numbers of dimensions and elements, although one array size must be 64Kbytes or less

(4) When an array is declared and if the array is numeric, all the elements are initialized with 0s. If the array is character, it is initialized with null string.

(5) Although all the arrays are declared before using them, an array can be used without declaration if its subscription maximum value is 10 or less. Each dimension is assumed that DIM 10 was declared when it was used for the first time regardless of the number of dimensions.

(7) The array declaration can be cancelled by the ERASE instruction. (If the memory has been exhausted, the array which will not be used any more can be deleted and a new array can be declared.)

Examples

1)
```
>10 DIM A(10,20,10)
>20 FOR I = 1 TO 10
>30 FOR J = 1 TO 20
>40 FOR K = 1 TO 10
>50 INPUT A(I,J,K)
>60 NEXT K, J, I
  '
  '
```

2)
```
>10 INPUT M, N
>20 DIM A(M, N)
>30 FOR I = 1 TO M
>40 FOR J = 1 TO N
>50 INPUT A(I, J)
>60 NEXT J, I
  '
  '
>200 END
```

END

Purpose

The END instruction terminates the execution of the program, closes all the opened files and returns control to the Basic mode.

Form

Description

(1) The user can write the END statement anywhere in the program. If there is an open file at the time when the END statement is specified, it will be closed. The control stops the program operation and returns to the Basic Mode.

(2) When no END statement is found at the end of the program, the system performs the same operations automatically (closing all files and going into the Basic Mode).

Example

```
>500 END
```

FOR/NEXT

Purpose

This instructs the system to loop as many times as specified. The FOR and NEXT statements must be used together.

Form

Description

(1) <variable> is a counter controlling the repeated executions.

(2) <initial value> is the counter initial value; <final value> is the counter final value. The statement following the FOR is executed through NEXT, then the counter is incremented by <increment> value. If the counter value does not reach the final value, the execution will be repeated after returning to the next FOR statement. This procedure will be repeated until the counter value reaches the final value. When it reached the final value, control is passed to the statement following the NEXT.

(3) If STEP is omitted, <increment> is assumed as 1.

(4) If the counter initial value exceeds the final value from the beginning, the loop is not executed and control is passed to the statement following the NEXT. If several NEXTs are provided for one FOR, control is passed to the statement following the NEXT having the smallest line number.

(5) FOR/NEXT loop can be nested. Other variable must be defined as a counter in each loop if nested.

(6) If nested loops have common terminal, variables can be written in one NEXT. The counter variable of inner loop must be placed inside.

(7) If NEXT variable is omitted, the variable in the immediately before FOR is assumed. The variables in the NEXT which is used as common terminal can not be omitted.

(8) Neither array nor double-precision real number cannot be used as <variable>(loop control variable).

Examples

1)
```
>10 FOR I = 1 TO 10
>20 INPUT A
>30 NEXT I
```

2)

```
>10 FOR A = 1 TO 10 STEP 10
>20 FOR B = 1 TO 20
>30 FOR C = 1 TO -10 STEP -1
  '
  '
>200 NEXT C, B, A
  '
  '
>300 END
```

3)

```
>10 INPUT M, N
  '
  '
>100 FOR I = M TO M + N
  '
>120 NEXT I
  '
  '
>300 END
```

GOSUB/RETURN

Purpose

GOSUB / RETURN instructs a branch from the main program to the subroutine program or return from the subroutine to the main program.

Form

```
GOSUB <line number>
--------
RETURN [<line number>]
```

Description

(1) <line number> is the starting line number of the subroutine program. <line number> must be <integer constant>.

(2) The RETURN statement comes at the end of the subroutine program. When RETURN is executed the control returns to the next statement after the GOSUB.

(3) Calls of subroutines can be nested. (Another subroutine program can be called by a subroutine program.) However, the currently executing subprogram cannot call itself.

(4) Following a GOTO, STOP, or END statement, a subroutine program may be entered in any part of the main program. Otherwise, the main control might accidentally fall into the subroutine program.

Examples

1)

```
  '
  '
>300 GOSUB 500
  '
  '
>490 GO TO 550
>500 REM GOSUB ROUTINE #1
>510 FOR I = 1 TO 10
>520 INPUT A$
>530 NEXT I
>540 RETURN
>550 END
```

2)

```
  '
  '
>200 GOSUB 1000
  '
  '
>900 END
>1000 REM GOSUB ROUTINE #2
>1010 FOR I = 1 TO 10
>1020 INPUT A$
>1030 NEXT I
>1040 RETURN
```

GOTO

Purpose

The GOTO instruction unconditionally branches to the specified line.

Form

```
GOTO <line number>
```

Description

(1) <line number> is the line number of branch destination.

(2) When the GOTO is executed, the control is moved to <line number>.

(3) If <line number> is not found in the program, an error is indicated.

Example

```
    .
    .
>200 GOTO 500
    .
    .
    .
>500 A=A+1
    .
    .
```

IF/THEN/ELSE

Purpose

The IF instruction conditionally branches the program control.

Form

```
IF<expression>THEN<statment>{:<statement>}|<line
number>[ELSE <statement>{:<statement>} <line number>]

or

IF <expression>GOTO<line number>[ELSE
<statement>{:<statement>}|<line number>]
```

Description

(1) <expression> indicates branch condition; if <expression> is true (other than 0), THEN or GOTO statements are executed while if it is false (0), ELSE statements are executed. If ELSE statements are omitted and the <expression> is false, control is transferred to the next line. If the <expression> is true and no branch instruction is written in the THEN statement, control is transferred to the next line after executing the THEN statement.

(2) IF statement can be written in THEN or ELSE statement. (IF can be nested. Although the nesting level is not limitted, it must be 255 characters or less in total.)

(3) <statement>{:statement>} may be a multi-statement.

(4) When IF is nested, the ELSE is assumed to be paired with the IF immediately before. Therefore, the ELSE cannot be omitted in nested IF statement. Only the outermost IF statement can omit the ELSE starement.

Examples

```
>100 IF A<3 THEN A=A-1:C=D:H=H+2:GOTO 200 ELSE 300

>100 IF A=B GOTO 500 ELSE A=0:GOTO 600

>100 IF A=B THEN IF C<D THEN 200 ELSE 300
           ELSE IF C>D THEN 400 ELSE 500
```

INPUT

Purpose

The INPUT instruction is used to enter the data from the keyboard.

Form

Description

(1) When control was passed to the INPUT command, the program terminates and the system waits for the data to be entered from the keyboard. If the RETURN key (<CR>) is pressed after the necessary data has been entered, the data is stored in the variable and the program procedes to the next step. Characters which have been keyed in can be reentered after deleting the characters by the Back Space bar.

(2) <number of digits> is the number of digits of the keying data used to check and can be written using numeric expression. An integer 1 through 255 can be specified; the fraction part is automatically rounded; the number is indicated as an error if it exceeds the limit. When <number of digits> is specified, up to <number of digits> can be entered, although only the <CR> key can be entered after that and all the other input are ignored.

(3) When <message> is specified, the <message> appears on the screen while waiting for keying. This message, therefore, notifies the operator what to enter. If this

is omitted, ? is displayed. Use character expression for specifying <message>.

(4) ; before <message> is the RETURN Key code control after the keying. If ; is specified, the cursor remains at immediately after the keying data then the screen is displayed after the keying data. If ; is omitted, the cursor is carriage-returned after data keying is completed.

(5) Select either ; or , after <message>.
If ; is selected, "?" is displayed after <character constant>;it is not displayed if , is selected.

(6) The operator must enter as many <variable> data as witten in the INPUT statement. Separate the data corresponding to each <variable> using comma (,). If the entered data type differs from the <variable> type or the number of entered data differs from the number of <variables>, "Redo from start" is displayed and the system waits for keying.

(7) A comma (,) cannot be included in a character string to be entered when keying a character data. (A comma is assumed as a delimiter. Use the LINE INPUT instruction to enter the character string having commas.)

Example

```
>100 INPUT A, B, X$
>100 INPUT "DATA INPUT..", A
>100 INPUT (15) X
```

KEY

Purpose

The KEY instruction assigns the function keys (PF) to character strings. If several characters are assigned to a PF key beforehand, the assigned character string is entered just by pressing the PF key.

Form

```
KEY <PF key number>,<character expression>
```

Description

(1) <PF key number> can be written using a numeric expression whose result is 1 through 20. If the valule results in a real number, it is automatically rounded.

(2) The first 8 characters of <character expression> are effective; the rest of them are discarded.

Examples

```
>100 KEY 5, "RUN"+CHR$(&HOD) (RUN is assigned
            for PF5.)
>100 KEY 1   "TAB" (TAB is assigned for PF1.)

>100 KEY 4, A$
```

LET

Purpose

LET stores the data in the variable.

Form

```
[LET] <variable>=<expression>
```

Description

(1) The keyword LET can be omitted.

(2) An error occurs if the <variable> is numeric value and the evaluated result of the <expression> is character or vice versa.

(3) If both the <variable> and the <expression> are numeric values but different types, the <expression> is calculated first then the result is stored in the <variable> after conversion. If it cannot be converted, an error occurrs.

Examples

```
>100 A=1:B=2:C=3
>100 A=A+1
>100 A$="ABC+CHR$(&H41)"
>100 A=0.123456789D-5
>100 A%=231.5
```

LIST/LLIST

Purpose

The LIST or LLIST directs the program currently on the memory to the screen or the printer. After the execution, the system enters Sanyo Basic mode.

Form

```
LIST [[<line number 1>]-[<line number 2>]]
LLIST[[<linenumber 1>]-[<linenumber 2>]]
[,[<number of lines>]]
```

Description

(1) <line number> specifications are interpreted as follows:

1) No specification
   Directs the entire listing.

2) <line number 1>-<line number 2>
   Directs from the specified line number 1 through line number 2.

3) <line number>-
   Directs from the specified line number through the end of the program.

4) -<line number>
   Directs from the beginning of the program through the specified line number.

5) <line number>
   Directs only the specified line. . (period) can be used to indicate the last line of the

program

6) LIST.
Directs only the last line of the program.

(2) If <number of lines> is specified with the LLIST, the page is fed with 3 blank lines at the top and bottom of each page after printing <number of lines> per page.

<number of lines> must be an integer constant within the range of 1 - 255. If <number of lines> is 0, the page is fed without any blank lines. (This is same as if it is not specified.)

(3) If <number of lines> is omitted (only the comma is specified), the page is fed with 3 blank lines at the top and bottom of each page assuming one page consists of 60 lines (appropriate to 11-inch stockforms). If this is not specified, the page is fed without any blank lines.

(4) The listing execution is interrupted if the space key is pressed, and it resumes if any other key is depressed.

Examples

>LIST	Directs the entire program on the screen.
>LIST 100-500	Directs 100 - 500 on the screen.
>LIST -250	Directs from the beginning of the program through 250.
>LLIST 100-1000,	Prints 100 - 1000 with 60 lines on each page.
>LLIST 1000-, 50	Prints after 1000 with 50 lines on each page.
>LLIST	Prints the entire program without any blank lines.
>LIST.	Prints only the last line.

LOAD

Purpose

This instruction reads the Sanyo Basic program into the memory from the disk.

Form

```
LOAD <filename>
```

Description

(1) <filename> is the filename when the program is saved.

(2) If the LOAD is executed, all the currently opened files are closed and the program in the filename is stored in the memory by destroying the current program. After the execution, the system enters Sanyo Basic mode.

(3) If the program file in the <filename> is not found in the disk, the program on the memory remains without being destroyed.

(4) <file name> can be written using the character expression.

Example

```
>LOAD "PRGM-A"
>LOAD "PRGM-A",R
```

LOCATE

Purpose

The LOCATE instruction sets the cursor to the specified position on the screen.

Form

```
LOCATE <vertical position>,<horizontal position>
[,<cursor blink switch>]
```

Description

(1) <vertical position> indicates the vertical axis; it must be within the range of 1 - 25. 1 indicates the uppermost position while 25 indicates the lowermost position.

(2) <horizontal position> indicates the horizontal axis; it must be within the range of 1 - 80. 1 indicates the left most position while 80 indicates the rightmost position.

(3) <horizontal position>, <vertical position>, and <cursor blink switch> can be written using the numeric expression, although the result is automatically rounded if it includes fraction part. If the value is not in the range of the above limit, an error occurs.

(4) If 0 is specified for <cursor blink switch>, the cursor is not displayed. Otherwise the cursor is displayed. If it is omitted, the cursor status is not modified. (It is originally in blink status. The cursor is displayed whenever prompt mark is displayed in Sanyo Basic mode.)

Examples

1)

```
>X=30
>Y=10
>LOCATE X, Y
```

2)

```
>10 LOCATE 10,15
>20 PRINT "THIS SPOT BEGINS AT 10,15"
```

LPRINT

Purpose

Outputs data to the printer.

Form

```
LPRINT [<expression>{,|;<expression>}[,|;]]
```

Description

(1) The result of the <expression> is directed on the screen as a numeric value of a character.

(2) Comma or semicolon is necessary as a delimiter when several <expressions> are specified.

(3) If the comma is used as a delimiter, one line (80 characters) is divided into 5 zones each consisting of 16 characters. Therefore, the value of the <expression> following the comma is directed in the next zone following the zone of the previous <expression>.

(4) If the semicolon is used as a delimiter, the value of the <expression> following the semicolon is directed continuously to the value of the previous <expression>.

(5) If the concluding comma or semicolon is omitted, the RETURN Key code is sent to the screen at the end of the LPRINT statement and the cursor is carriage-returned (is movesd to the leftmost position of the next line).

(6) If the comma or semicolon at the end is specified, the RETURN key code is not sent to the screen at the

end of the LPRINT statement. If the comma is specified, the screen output is executed from the next zone. If the semicolon is specified, the screen output is continuously executed to the previous output.

Example

1)

```
>LPRINT 10,100,0.1,-20
 10            100           1            -20

>LPRINT 10;100;1; -20
 10  100  .1 -20

>A=10:B=20:LPRINT "A=";A, "B=";B
 A= 10          B= 20
```

2)

```
>10 LPRINT " MBC COMPUTERS"
>RUN

 MBC COMPUTERS
```

MERGE

Purpose

The MERGE instruction merges the program on the disk to the program currently on memory.

Form

```
MERGE <filename>
```

Description

(1) <filename> is the filename of the program on the disk and can be written using the character expression. This program is merged to the program on memory.

(2) If the program specified by the <file name> includes the same line number as the program currently on memory, the line of the program merged from the file replaces the one on memory.

(3) After the MERGE execution, control returns to Sanyo basic
mode.

Example

```
>MERGE"PRGM-A"
```

NEW

Purpose

The NEW instruction deletes the program currently on memory.

Form

Description

(1) When the user enters the new program, he must execute the NEW beforehand.

(2) After the NEW execution, the system enters Sanyo Basic mode.

ON GOSUB/ON GOTO

Purpose

These instructions are conditional branch instructions whose destinations are determined by the specific values of the program.

After branching, conrol returns via the RETURN instruction (in the subroutine) as in the GOSUB.

Form

```
ON <numeric expression> GOSUB <line number>{.<line
number>}

ON <numeric expression> GOTO <line number>{,<line
number>}
```

Description

(1) Control passes to one of the <line numbers> according to the <numeric expression> value. In other words, control passes to the line number which matches the value of the <numeric expression>.

(2) The <numeric expression> value must be within the range of 1 through the number of <line numbers>. If it includes the fraction part, it is automatically rounded; if it is negative or exceeds 255, an error occurs.

(3) When the <numeric expression> value is 0 or exceeds the number of <line numbers>, this instruction is not executed and control passes to the next statement.

(4) If the ON GOSUB is specified, the <line number>

must be the first line of the subroutine.

Example

```
>100 INPUT "1-3 ?", A
>110 ON A GOSUB 140,170,200
>120 GOTO 100
>130 '
>140 PRINT "A=1"
>150 RETURN
>160 '
>170 PRINT "A=2"
>180 RETURN
>190 '
>200 PRINT "A=3"
>210 RETURN

>RUN
 1-3 ? 2
 A=2
 1-3 ? 100
 1-3 ? 1
 A=1
 1-3 ?

 Break in 100
```

OPTION BASE

Purpose

The OPTION BASE instruction determines the smallest subscript value.

Form

```
OPTION BASE <n>
```

Description

(1) <n> is numeric constant and can be either 0 or 1.

(2) When an array is declared by the DIM instruction, the elements from 0 through the declared subscsription maximum value are reserved. If DIM A (20) is specified, 21 elements, A(10) to A(20), are reserved in the memory. This instruction is used to reserve only 20 elements A(1) to A(20).

(3) Only one OPTION BASE statement can be written in a program.It must be executed before any array declarations or reference. (BASEs 0 and 1 cannot be mixed within a program.)

(4) Similarly when the programs are connected using the CHAIN instruction, arrays in the CHANINed programs must all adopt the same BASE.

Example

```
>10 OPTION BASE 1
>20 DIM A(5)
>30 PRINT A(0)

>RUN

 Subscript out of range in 30
```

PRINT

Purpose

The PRINT is an output instruction for the screen.

Form

```
PRINT [<expression>{,|;<expression>}[,|;]]
```

Description

(1) The result of the <expression> is directed on the screen as a numeric value of a character.

(2) Comma or semicolon is necessary as a delimiter when several <expressions> are specified.

(3) If the comma is used as a delimiter, one line (80 characters) is divided into 5 zones each consisting of 16 characters. Therefore, the value of the <expression> following the comma is directed in the next zone following the zone of the previous <expression>.

(4) If the semicolon is used as a delimiter, the value of the <expression> following the semicolon is directed continuously to the value of the previous <expression>.

(5) If the concluding comma or semicolon is omitted, the RETURN Key code is sent to the screen at the end of the PRINTstatement and the cursor is carriage-returned (is movesd to the leftmost position of the next line).

(6) If the comma or semicolon at the end is specified, the RETURN Key code is not sent to the screen at the end of the PRINT statemen. If the comma is specified, the screen output is executed from the next zone. If

the semicolon is specified, the screen output is continuously executed to the previous output.

(7) If all the parameters are omitted, the line is simply fed.

Examples

1)

```
>PRINT 10,100,0.1,-20
 10             100             1             -20

>PRINT 10;100;1; -20
 10  100  .1 -20

>A=10:B=20:PRINT "A=";A, "B=";B
 A= 10              B= 20
```

2)

```
>10 PRINT " MBC COMPUTERS"
>RUN

 MBC COMPUTERS
```

READ/DATA/RESTORE

Purpose

The READ and DATA are used with combination; the constant defined by the DATA statement is read into the variable defined by the READ statement. The RESTORE instruction resets the pointer which indicates the data of the DATA statement.

Form

```
READ <variable>{,<variable>}
DATA <constant>{,<constant>}
```

Description

(1) <variable> can be a character of numeral and is written having one to one correspondence with the <constant> of the DATA statement. Several READ statements can be written for one DATA statement, or vice versa.

(2) The DATA instruction can be placed anywhere and any times in the program. The READ instruction reads the constants of the DATA sequentially.

(3) If the variables in the READ statement are numerals and the corresponding constants of the DATA statement are characters, an error occures.

(4) If the variables of the READ and the constants of the DATA are numerals with different types, the constants are converted to the variable type before reading. If they cannot be converted an error occurs.

(5) If , (comma) or : (colon) is included in the char-

acter constant or character constant starting with space is defined, enclose the constant using " " is not necessary for the character constant without delimiter.

Purpose

The constants in the DATA statement are used sequentially, although the order to be read can be modified using the RESTORE instruction.

Form

```
RESTORE <line number>
```

Description

(1) If the RESTORE instruction is executed, the DATA which corresponds to the next READ is the DATA statement having the <line nmber>. If the <line number> is omitted, the DATA statement having the smallest line number is processed. If the specified line number is not a DATE statement, the read pointer is set at the DATE statement which first appears after that line number.

Examples

1)
```
>100 DIM A(10)
      .
>150 FOR I=1 TO 10:READ A(I):NEXT
      .
      .
>170 DATA 1,2,5,10,3,7,15,8,3,4
```

Input the line number from 200 to 220 as follows

```
>200 FOR I=1 TO 10                    Result
>210 PRINT "A(I)=";A(I)
>220 NEXT                             A(I)=1
                                      A(I)=2
                                      A(I)=5
                                        .
                                        .
                                      A(I)=10
```

2)
```
>10 READ A,B,C,D,E,F
>20 RESTORE 60
>30 READ G,H,I
>40 DATA 10,10
>50 DATA 20,20
>60 DATA 30,30
```

Input the line number from 100 to 180 as follows:

```
                                      Result
>100 PRINT "A=";A                     A=10
>110 PRINT "B=";B                     B=10
>120 PRINT "C=";C                     C=20
>130 PRINT "D=";D                     D=20
>140 PRINT "E=";E                     E=30
>150 PRINT "F=";F                     F=30
>160 PRINT "G=";G                     G=20
>170 PRINT "H=";H                     H=20
>180 PRINT "I=";I                     I=30
```

REM

Purpose

The REM instruction inserts the comment to the Sanyo Basic source program. This has no effect in the program execution.

Form

Description

(1) <comment> can include any usable characters. The program is executed ignoring the comment lines.

(2) If the REM is used in multiple-statement, up to <CR> is assumed as the comment after the REM.

(3) An apostrophe can be used in place of the REM.

(4) Control can be passed to the REM statement, although execution starts from the statement which first appears after the REM line (other than REM)

Example

```
>10 REM ***RRGM-1
>20 ' 1990.1.1
>30 '
>40 ' by R.MILOS
>50 '
>60 DIM A$ (100)
> .
```

RENUM

Purpose

The RENUM instruction renumbers all the lines of the progam currently on memory at one time.

Form

```
RENUM [[<new line number>][,[<old line number>]
[,<increment>]]]
```

Description

(1) <new line number> is the first line number to be renumhered and <old line number> is the line number where the renumbering starts.

(2) <increment> is the interval between line numbers which is used for renumbering.

(3) If <new line number> and <incrment> are omitted, both are assumed as 10. If <old line number> is omitted, execution starts from the first line of the program on the memory.

(4) An error occurs in the following cases:
   1) <old line number> specification exceeds the end line of the program.
   2) If the new line number exceeds 65535 after renumbering.
   3) If the program order is modified by the RENUM instruction.
   4) If 0 is specified as the increment.

(5) If the characters of one line exceeds 255 by the

RENUM instruction, an error occurs.

(6) Use the integer constant for <line number> and <increment>.

Examples

Line number before RENUM	REUNM	RENUM 100	RENUM 70,52,20
10 10	100		10
30 20	110		30
50 30	120		50
52 40	130		70
56 50	140		90
70 60	150		110
90 70	160		130

RUN

Purpose

The RUN instruction executes the program.

Form

```
 (1) RUN [<line number>]

or

 (2) RUN <filename>[,R]
```

(1) (1) executes the program currently on memory. If the <line number> is specified, the execution starts from that number; if it is omitted, the execution starts from the beginning of the program.

(2) (2) loads the program from the disk onto memory before execution. If "R" options specified, the program is loaded and executed without closing the corrently opened files.

(3) Use the character expression for <filename>.

Example

```
RUN
RUN 300
RUN "PRGM"
RUN "PRGM-A", R
```

SAVE

Purpose

The SAVE instruction stores the program currently on memory in the disk file.

Form

```
SAVE <filename>[,A|P|Q]
```

Description

(1) <filename> is the name given to the program. The program is loaded into memory again if it is called using this name.

(2) If A is specified, the program is written into the disk using the ASCII code (as program characters). If P is specified, the program is converted into the intermediate code before writing into the disk. If this is omitted, P is assumed.

(3) The option Q is for the program protection. If the program was saved with this option, the listing is inhibited for the program when it is reloaded to the memory. Therefore, it cannot be modified using the editor. This program cannot be saved again. (This is used to inhibt the program to be copied because the contents are secret.)

(4) Use the character expression for <filename>.

Example

```
SAVE "PRGM"
SAVE "PRGM",A
SAVE "PRGM-A",Q
```

STOP

Purpose

The STOP instruction temporarily terminates the program execution and returns control to the Sanyo Basic mode.

Form

STOP

Description

(1) The STOP instruction can be written anywhere in the program. When it is executed;

Break in xxxx(line number)

is displayed and the system enters the Sanyo Basic mode. The program which was temporarily terminated by the STOP instruction can be restarted from the next statement using the CONT instruction.

(2) The STOP instrution does not close the file.

SYSTEM

Purpose

The SYSTEM instruction passes control to the operating system.

Form

```
SYSTEM
```

Description

(1) If the SYSTEM instruction is executed, the opened files are all closed if any, control is passed to the OS (operating system) mode.

SECTION 3
GENERAL FUNCTIONS

Functions are classified into numeric function having numerals as function value and string function having characters. Some of functions need parameters which are enclosed in ( ) after the function name. If more than one parameters are necesssary, insert commas between them. String functions have names with $

* The parameter and function value of the numeric function is integer, single precision realn umber, or double precision real number. The result of the basics function is a single precision real number. The result of the basic function is a single precision real number, Although the basic function calculates using double precision real numbers internally. Therefore, the accuracy increases if the parameter is specified using the double precision real number.

The parameters are abbreviated as follows in the explanations.

Explanation	Meaning
X, Y	<numeric expression>
I, J	<numeric expression> whose result is an integer. (If it is a real number, it is automatically rounded.)
X$, Y$	<character expression>

ABS

Purpose

Returns the absolute value of <X>.

Form

```
ABS(<X>)
```

Example

```
>10 N1=ABS(-10) -1
>20 N2=ABS(10)  -1
>30 PRINT N1, N2

>RUN
 9              9
```

COS

Purpose

The cosine of trigonometric function of <X>.

Form

```
COS(<X>)
```

Description

(1) Use radian to specify <X>.

Example

```
>10 X=3.14159
>20 PRINT COS(X)

>RUN
-1
```

EXP

Purpose

Returns the value of e<X>

Form

```
EXP(<X>)
```

Description

(1) <X> must be 145.06286 or less.

Example

```
>10 A=EXP(0)
>20 PRINT A

>RUN
 1
```

INKEY$

Purpose

Displays the keyboard character which was depressed.

Form

Description

(1) If no key is pressed, it returns the null string.

Example

```
>10 PRINT "BASIC"
>20 IF INKEY$="" THEN 20
>30 PRINT "END"

>RUN
 BASIC
 END
```

This processing procedes to the next display if a key is pressed.

INT

Purpose

Returns the largest integer which does not exceed <X>.

Form

```
INT(<X>)
```

Description

(1) If the <X> is positive, the fraction part is rounded off. If it is negative, the fraction part is rounded up to obtain the result.

Examples

```
>10 PRINT INT( 10)
>20 PRINT INT( 10.9999)
>25 PRINT INT( 10.001)
>30 PRINT INT(-10.05)

>RUN
 10
 10
 10
-11
```

LEFT$

Purpose

The characters from the leftmost position through the <I>th character of the <X$>.

Form

```
LEFT$(<X$>,<I>)
```

Description

(1) <I> must be within the range of 0 - 255. If the <I> is greater then the number of characters in the <X$>, the <X$> itself is returned as the result.

Example

```
>10 A$="123abc"
>20 C$=LEFT$(A$,3)
>30 PRINT C$+"SANYO"

>RUN
>123SANYO
```

LOG

Purpose

Returns the natural logarithm of <X>.

Form

```
LOG(<X>)
```

Description

(1) <X> must be positive number.

Example

```
>10 X=2
>20 PRINT LOG(X)

>RUN
.693147
```

MID$

Purpose

<J> number characters from the <I>th character of the <X$>.

Form

```
MID$(<X$>,<I> [,<J>])
```

Description

(1) <I> must be within the range of 1-255. If the <I> is longer than the <X$> length, the null string is returned as the result.

(2) <J> must be within the range of 0-255. If the <J> is omitted, the <I>th through the last character is assumed.

Example

```
>100 OPEN "R",#1,"RND.DAT"
>120 DIM AW$(32)
>130 '
>140 GET#1
>150 J=1
>160 FOR I=1 TO 128 STEP 4
>170 AW$(J)=MID$(A$,I,4)
>180 J=J+1
>190 NEXT
```

This process divides the large field data and stores the data in the array.

RIGHT$

Purpose

The character(s) from the rightmost position of <X$> through the <I>th character.

Description

(1) <I> must be within the range of 0-255. If <I> is greater than the number of characters in <X$>, <X$> itself is returned as the result.

Example

```
>10 A$="BASIC 8088"
>20 PRINT RIGHT$(A$,4)

>RUN
>8088
```

RND

Purpose

Returns random numbers between 0 and 1.

Form

```
RND [(<X>)]
```

Description

(1) Generates random numbers between 0 and 1 (including 0). Each RUN instruction generates the same random sequence.

(2) The following random numbers are generated depending on the <x> sign. The default value is positive.

When <x> is positive: Generates the subsequent random number.

When <x> is 0: Regenerates the preceding random number.

When <x> is negative: Generates a new random sequence (always the same random sequence, independent from the <x> value).

SGN

Purpose

Returns the sign of <X>.

Form

```
SGN(<X>)
```

Description

(1) When <X> is negative = 1
(2) When <X> is 0 = 0
(3) When <X> is positive = 1

Example

```
>10 INPUT A                  This processing executes
>20 N=SGN(A)                 different subroutine
>30 ON N+2 GOSUB 60,70,80 according to the sing of
>40 GOTO 10                  the input value.
>50 '
>60 PRINT "-":RETURN
>70 PRINT "ZERO":RETURN
>80 PRINT "+":RETURN

>RUN
?-0.1
-
? 12345
+
? 0.0
ZERO
?
```

SIN

Purpose

Returns the sine of trigonometric function of <X>.

Form

```
SIN(<X>)
```

Description

(1) Use radian to specify <X>.

Example

```
>10 X=1
>20 Y=SIN(X)
>30 PRINT Y

>RUN
 .841471
```

SQR

Purpose

Returns the square root of <X>.

Form

```
SQR(<X>)
```

Description

(1) <X> must be either 0 or positive.

Example

```
>10 FOR I=1 TO 5
>20 PRINT SQR(I)
>30 NEXT

>RUN
 1
 1.41421
 1.73205
 2
 2.23607
```

TAN

Purpose

Returns the tangent of trigonometric function of <X>.

Form

```
TAN(<X>)
```

Description

(1) Use radian to specify <X>.

Example

```
>10 X=3.14159/4
>20 PRINT TAN(X)

>RUN
 0.999999
```

SECTION 4
GRAPHICS

Screen and World Coordinates

Sanyo Basic graphics are created with two types of coordinate systems. One is the physical coordinate system which corresponds on a 1 to 1 basis to the 640 x 200 dots on the screen. The other is the logical coordinate system devised for Sanyo Sanyo Basic which can be as large as 32768 x 32768. The physical coordinate system is called the screen coordinate system whereas the abstract coordinate system is called the world coordinate system.

Most of the Sanyo Basic graphic instructions are issued for the abstract world coordinate system. Which portion of the huge coordinate system should be displayed where on the screen (screen coordinate system) must be specified. This enables free programming on the abstract coordinate system using a required number of dots which are independent from the limited number of dots on the real screen, and the graphics created are displayed in the appropriate size and in the desirable position on the screen. You may create graphics without worrying about work space. If you write graphics with coordinates of less than 640 x 200 you may write directly to screen. With coordinates larger than 640 x 200, the WINDOW and VIEW instructions are needed to be able to see the finished graphic on the screen.

Viewport and Window

The WINDOW instruction specifies which portion of the world coordinate system (this portion is called a window) should be taken. The VIEW instruction specifies which portion of the screen coordinate system (screen) should be used (this portion is called a viewport). The VIEW is the definition of the drawing area.

The window and viewport combination enables outputting either part or all of the graphics belonging to the world coordinate system in various sizes and in various ratios on the screen.

The window is the area of the world coordinates that will be seen on an area of the screen defined by the VIEW instruction.

Example

```
>10 CLS
>20 WINDOW(0,0)-(599,199)
>30 VIEW(0,0)-(599,199),6,4
>40 GOSUB 110
>50 VIEW(0,0)-(399,133),5,1
>60 GOSUB 110
>70 VIEW(0,0)-(199,67),2,7
>80 GOSUB 110
>90 GOTO 90
>100 END
>110 CIRCLE(400,100),125,,,,3
>120 LINE(275,35)-(525,165),4,B
>130 RETURN
```

As the VIEW instruction only decides the screen size and does not affect the screen portions outside the

viewport, it is easy to synthesize graphics by repeating the same patterns.

Example

```
>10 CLS
>20 FOR I=190 TO 1 STEP 20
>30 VIEW(X,Y)(X+I,Y+I),,PSET
>40 CIRCLE(320,100),70
>50 X=X+30:Y=Y+15
>60 NEXT
>70 END
```

The initial vaues of the window and the viewport are (0,0)-(639,199) and (0,0)-(639,199), respectively. This window/viewport ratio makes the vertical and horizontal lines, which consist of the same number of dots in the world coordinate system, assume the same length both vertically and horizotally on the screen, as the vertical size of each dot on the screen is greater than the horizontal size.

Color Specification

In the Sanyo Basic screen, the colors of the displayed characters and the screen field can be specified. This is called <character color> and <background color>.

These can be set by the COLOR instruction using the color codes. The color codes are represented by the expressions whose vaues are within the range 0 7 (including the margins). The lower order three bits (in binary notation) correspond to red, green, and blue, respectively. When the colors indicated by the higher bits are mixed, the resulting color is the color of the code.

Color code	Lower order three bits			Color
	Red	Green	Blue	
0	0	0	0	Black
1	0	0	1	Blue
2	0	1	0	Green
3	0	1	1	Light blue
4	1	0	0	Red
5	1	0	1	Purple
6	1	1	0	Yellow
7	1	1	1	White

Color is specified for some of the graphic instructions also by the color codes. The COLOR instruction is invalid for monochrome displays because background color is always assumed as OFF and character color is assumed as ON.

INSTRUCTION WORDS

CIRCLE

Purpose

Draws a circle or a circular arc.

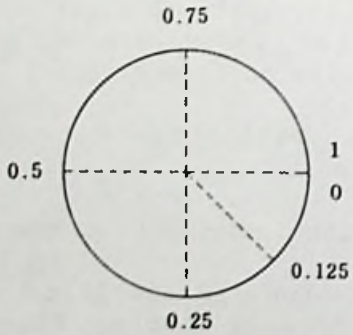
Form

```
CIRCLE(<X>,<Y>),<radius>[,[<initiatingposition>]
[,[<terminatingposition>[,[<ratio>][,<circle
color>]]]]
```

Description

(1) <X> and <Y> indicate a point on the world coordinate system. Using this point as the central point, a circle is drawn having the radius specified in <radius> (dots).

(2) For <initiating position> and <terminating position>, the value having an absolute value within the range of 0 1 and assuming the circumference is 1, is used. These values are used for creating wedges. In the illustration below, the initiating position could be 0.125 and the terminating position 0.25.



(3) When <initiating position> and <terminating position> are specified, a circular arc is drawn from the <initiating position> to the <terminating position>. The drawing direction is always clockwise.

When a negative value is specified, the absolute value is assumed as <initiating position> or <terminating position>, from which a circular arc is drawn, then a straight line is drawn from the central point of the circle to the specified position on the circumference (a sector is drawn).

When <initiating position> is omitted, the default value is 0.

When <terminating position> is omitted, a circle is drawn starting from the <initiating position>.

(4) <ratio> is specified as "(radius in the vertical direction)/(position in the horizontal direction)". When <ratio> is specified, an ellipse is drawn using the <radius> as the radius in the horizontal direction and "<radius> X <ratio>" as the radius in the vertical direction. Therefore, the following expressions can be formed:

<ratio>=1	Circle
<ratio> < 1	Ellipse (whose horizontal size is greater than the vertical size)
<ratio> > 1	Ellipse (whose vertical size is greater than the horizontal size)

When <ratio> is omitted, the default value is 1.

When <ratio> is omitted, 0.508 is assumed as <ratio>.

(5) When <circle color> is specified, the specified color is used for drawing circles. <circle color> can

be specified using the color codes. When omitted, circles are drawn using <character color>.

(6) <X>, <Y>, <radius>, <initiating position>, <terminating position>, <ratio>, and <circle color> can be written using numeric expressions.

(7) If a circle or a circular arc extends from the world coordinates, it becomes an error.

Example

```
>100 CLS
>110 WINDOW(0,0)(639,199):
     VIEW(0,0)(639,199)
>120 FOR I=1 TO 5
>130 INPUT S(I)
>140 TOTAL=TOTAL+S(I)
>150 NEXT
>160 P1=0
>170 FOR I=1 TO 5
>180 PA=S(I)/TOTAL:P2=P1+PA
>190 CIRCLE(250,90),180,P1,P2,.8
>200 P1=P2
>210 NEXT
```

COLOR

Purpose

Specifies character color and background color of the screen.

Form

```
COLOR <character color>|,<background color>|
<character color>,<background color>
```

Description

(1) <character color> and <background color> are specified using color codes. They can be written using the numeric expression. When <character color> is specified, characters to be displayed are colored in <character color>. When <background color> is specified, the screen fields to be displayed henceforth are colored in <background color>. If these are omitted, the original colors remain unchanged.

(2) <character color> and <background color> may be written together or separately. See example below.

Examples

COLOR 7: The character color is white and the background color is unchanged.

COLOR, 1: The background color is blue and the character color is unchanged.

COLOR 1, 4: The character color is blue whereas the background color is red.

GCURSOR

Purpose

GCURSOR displays a graphic cursor on the screen, and reads the coordinates when the RETURN key is depressed by variables.

Form

```
GCURSOR (X,Y),(<variable 1>,<variable 2>)
[,<color code>]
```

Description

(1) (X, Y) are the coordinates on the screen coordinate system, which indicate the point where a graphic cursor is displayed first.

(2) The srahic cursor shifts one dot each time the cursor keys are depressed. When the TAB key is pressed once, the cursor shifts eight dots each time the cursor key is pressed. When the TAB key is pressed agan, it returns to one dot interval shift status.

(3) When RETURN key is depressed, the coordinate value (on the screen coordinate system) indicated by the graphic cursor is written to <variable 1> and <variable 2>.

(4) <color code> is the color of the graphic cursor, which can be written using a numeric expression. If it is omitted, the current character color is used.

Example

Shifts the graphic pattern to the position indicated by GOURSOR.

```
>10 CLS
>20 WINDOW(0,0)(639,199)
>30 VIEW(0,0)(639,199)
>40 SYMBOL(50,50),"SYMBOL",3,3,1,0
>50 MAX=(((20030+1)+7)/8*(10040+1)*3+4)/4+1
>60 DIM A(MAX)
>70 GET(30,40)(200,100),A
>80 GCURSOR(300,200),(BX,BY),2
>90 PUT(BX,BY),A,OR
```

GET

Purpose

GET writes the screen dot information to an array.

Form

```
GET (X1,Y1)(X2,Y2),<array name>
```

Description

(1) (X1,Y1) and (X2,Y2) indicate points on the screen coordinate system. The dot information within a quadrangel which assumes the two points as opposite angles, is written to an array. (X1,Y1) and (X2,Y2) can be written using numeric expressions.

(2) <array name> is used to write the dot patterns, for which DIM declaration must be made so that the GET statement can be declared. The array must be numeric, which prohibits the use of character arrays. If no arrays having the <array name> have been declared, or the sufficient array size has not been provided, it becomes an error. The maximum subscript value for DIM declaration can be obtained as follows:

[{X2-X1+1)+7}/8*(Y2-Y1+1)*3+4]/N+1
N identifies the array type (2, 4, and 8).

(3) The points with X1=X2 or Y1=Y2 cannot be specified.

(4) 256K of memory is required for the GET command.

PUT

Purpose

PUT displays the dot information written by the GET statement from the specified coordinates.

Form

```
PUT (X,Y),<array name>[,<function>]
```

Description

(1) (X,Y) is a point on the screen coordinate system, which indicates the upper left corner of the quadrangle (which has been written by the GET statement). X and Y can be written using numeric expressions.

(2) <array name> is the name of the array which contains the dot pattern. It must be the same as that specified by the GET statement.

(3) <function> is used to display the dot pattern, which is selected from the following five types.

The default value is PSET.

PSET	Displays the array dot pattern as it appears.
PRESET	Reverses the array dot pattern before displaying it.
OR	Takes OR of the array dot pattern and the pattern on the screen, then displays the result.

AND Takes AND of the array dot pattern and the pattern on the screen, then displays the result.

XOR Takes XOR of the array dot pattern and the pattern on the screen, then displays the result.

(4) 256K of memory is required for the PUT command.

SYMBOL

Purpose

SYMBOL draws character strings (symbols) with the specified magnification and angle on the screen's optional position.

Form

```
SYMBOL (X,Y),<character string>,<horizontal
magnification>,<vertical magnification>
[,[<color code>][,<angle code>]]
```

Description

(1) (X,Y) is a point on the world coordinate system, which indicates the upper left corner of the character string to be displayed.

(2) <character string> is a character string to be displayed, and is written using, character expression.

(3)<horizontal magnification> and <vertical magnification> identify the magnification of the characters to be displayed. The characters are displayed with <horizontal magnification> X 8 dots horizontally, and with <vertical magnification> X 16 dots vertically. Because a dot here is identified with the world coordinate system, the character margin (upper, lower, left, or right margin) may be cut due to inappropriate window /viewport specification.

(4) <color code> is the color for the character string. When it is omitted, the current character color is used.

(5) <angle code> specifies the angle for turning the character string to the right, and is selected from the following types. The default vaue is 0.

0 ..... Do not turn.
1 ..... Turn 90 degrees to the right.
2 ..... Turn 180 degrees to the right.
3 ..... Turn 270 degrees to the right.

(6) Take care when specifying, as an error occurs if the magnified and turned <character string> extends from the world coordinate system.

(7) X,Y, <horizontal magnification>,<vertical magnification>, <color code>, and <angle code> can be written using numeric expressions.

Example

```
>100 WINDOW(0,0)(639,199)
>110 CLS
>120 FOR I=1 TO 4
>130 SYMBOL(200,100),"SANYO",I,I,I,I1
>140 NEXT
```

LINE

Purpose

Draws straight lines.

Form

```
LINE [(<X1>,<Y1>)]-(<X2>,<Y2>)[,<line
color>[,B|BF]]
```

Description

(1) <X1>, <X2>, <Y1>, and <Y2> indentify x and y coordinates, on the world coordinate system, and can be written using numeric expressions.

When (<X1>,<Y1>) is specified, a straight line is drawn starting from the point through (<X2>,<Y2>). When the initiating point is omitted, the terminating point of the LINE instruction executed immediately before is assumed as the initiating point. If no LINE instructions have been executed immediately before, (0,0) is assumed as the initiating point.

(2) When <line color> is specified, a line is drawn using this color. Specify using the color codes. If omitted, the <character color> at that time is used. It can be written using a numeric expression.

(3) When B is specified, a rectangle having (<X1>,<Y1>) and (<X2>,<Y2>) as opposite angles is drawn. When BF is specified, <line color> is spread inside the rectangle.

Example

```
>10 CLS
>20 FOR I=1 TO 100
>30 X=RND*639: Y=RND*199
>40 N=I MOD 8
>50 LINE(X,Y),N
>60 NEXT
```

PAINT

Purpose

Specified color is spread on the area containing the specified point.

Form

```
PAINT(<X>,<Y>)[,[<area color>][,<border color>]]
```

Description

(1) (<X>,<Y>) is a point within the window of the world coordinate system. <area color> is spread on the area (containing this point) which is surrounded by <border color>.

(2) <area color> is specified using the color codes. If it is omitted, the default value is <character color>.

(3) <border color> is specified using the color codes. If it is omitted, the default value is <area color>.

(4) <X>, <Y>, <area color>, and <border color> can be written using numeric expressions.

Example

```
>100 CLS
>110 LINE(100,70)(500,175),7,B
>120 PAINT(300,100),7,7
>130 CIRCLE(300,120),70,0,1,0.5,4
>140 PAINT(300,110),4,4
```

PRESET

Purpose

PRESET erases the specified dot.

Form

```
PRESET(<X>,<Y>)
```

Description

(1) (<X>, <Y>) are the coordinates on the world coordinate system, and can be written using a numeric expression. It spreads <background color> on the (<X>,<Y>) dots.

(2) For monochrome displays, it turns the dot OFF.

PSET

Purpose

PSET spreads color on the specified dot.

Form

```
PSET (<X>,<Y>)[,<point color>]
```

Description

(1) (<X>,<Y>) are the coordinates on the world coordinate system, and can be written using a numeric expression.

(2) When <point color> is specified, the specified color is set. It is specified using the color codes. If it is omitted, <character color> is used. It can be written using a numeric expression.

VIEW

Purpose

VIEW sets the view port.

Form

```
VIEW(<X1>,<Y1>)-(<X2>,<Y2>)[,[<area color>]
[,<border color>]]
```

Description

(1) <X1>, <X2>, <Y1>, and <Y2> are the respective x and y coordinates on the screen coordinate system, and can be written using numeric expressions. After this instruction has been issued, the area surrounded by a rectangle with points (<X1>, <Y1>) and (<X2>, <Y2>) as opposite angles is set as the viewport on the screen to which the window in the world coordinate system is displayed.

(2) When <area color> is specified, the inside of the viewport is cleared using <area color>.

When <border color> is specified, the viewport frame is drawn using <border color>. Both <area color> and <border color> are specified using color codes, which can be written using numeric expressions.

(3) The VIEW instruction only sets the display area on the screen. It does not affect the area outside the viewport.

(4) The initial value of the viewport is (0,0)-(639,199), and is set throughout the entire screen.

(5) If the specified X coordinate does not exist within the range of 0-639, or the Y coordinate does not exist within the range of 0199, it becomes an error. Even if these coordinates are within the specified range, points with <X1>=<X2> or <Y1>=<Y2> cannot be specified.

Example

```
>10 CLS
>20 WINDOW(0,0)-(599,199)
>30 VIEW(0,0)-(599,199),6,4
>40 GOSUB 110
>50 VIEW(0,0)-(399,133),5,1
>60 GOSUB 110
>70 VIEW(0,0)-(199,67),2,7
>80 GOSUB 110
>90 GOTO 90
>100 END
>110 CIRCLE(400,100),125,,,,3
>120 LINE(275,35)-(525,165),4,B
>130 RETURN
```

WINDOW

Form

```
WINDOW(<X1>,<Y1>)-(<X2>,<Y2>)
```

Descrption

(1) <X1>, <X2>, <Y1>, and <Y2> are x and y coordinates on the world coordinate system, and can be written using numeric expressions. After this instruction has been issued, they are surrounded by a rectangle which has points (<X1>, <Y1>) and (<X2>, <Y2>) as opposite angles is displayed on the viewport (on the screen) as the window.

If a graphic on the world coordinate system is not within the window, it is not displayed on the screen.

(2) The WINDOW instruction only sets the area on the world coordinate system. It does not affect the actual screen.

(3) The initial vaue of the window is set within the range of (0,0)-(639,199).

(4) When the specified coordinates are not within the range of 0 32767, it becomes an error.

Even if they are within the specified area, points with <X1>=<X2> or <Y1>=<Y2> cannot be specified.

CHAPTER 4
MS-DOS INTRODUCTION

WHAT IS MS-DOS ?

Microsoft (R) MS(tm)-DOS is the disk operating system for the MBC 550 series computer. Through MS-DOS, you communicate with the computer, disk drives and a printer.

An operating system provides the interface between the computer and both you and the software you are using. It can be compared to the electricity in a house -- you need it for the television, but you are not always aware that it is there.

This section will briefly describe MS-DOS and present you will the important commands you may need to use. For further information please consult the Sanyo Software MS-DOS Reference Manual available at your authorized Sanyo computer dealer.

MS-DOS RULES

Words called commands and statements are typed into the computer to operate MS-DOS. The Sanyo computer understands these commands, but just as any language needs rules and grammar, the command and statement words of MS-DOS have rules called syntax. These rules help the computer understand what you want to do.

The following rules (syntax notation) are used throughout this section in the descriptions of command and statements:

[ ] Square brackets indicate that the enclosed entry is optional.

< > Angle brackets indicate data you must enter. When the angle brackets enclose lower case text, you must type an entry defined by the text; for example, <filename>. When the angle brackets enclose upper case text, you must press the key named by the text; for example, <RETURN>.

{ } Braces indicate that you have a choice between two or more entries. At least one of the entries enclosed in braces must be chosen unless the entries are enclosed in square brakets.

... Ellipses indicate that an entry may be repeated as many times as needed or desired.

| A bar indicates an OR statement in a command.

CAPS Capital letters indicate portions of statements or commands that must be entered, exactly as shown.

All other punctuation, such as commas, colons, slash marks and equal signs, must be entered exactly as shown.

MS-DOS FILES

What Is A File?

A file is a collection of related information. A file on your disk can be compared to a file folder in a desk drawer. For example, one file folder might contain the names and addresses of the employees who work for a company. This file could be named the Employee Master File. A file on your disk could also contain the names and addresses of employees in a company and could also be named Employee Master File.

All programs, text, and data on your disk reside in files and each file has a unique name to distinquish it form other files.

The names of files are kept in directories on a disk. These directories also contain information on the size of the files, their location (where they are stored) on the disk, and the dates they where created and updated. The directory you are working in is called your current or working directory.

An additional system area is called the File Allocation Table. It keeps track of the location of your files on the disk. It also allocates the free space on your disks so that you can create new files.

These two system areas, the directories and the File Allocation Table, enable MS-DOS to recognize and organize the files on your disks.

The DIR (Show Directory) Command

If you want to know what files are on your disk, you can use the DIR command.This command tells the operating system to display all the files in the current directory on the disk that is named. For example, if your disk is in drive A: and you want to see the listing for the current directory on that disk, type **DIR** and then depress the RETURN key.

A listing of the files on that disk with their size (expressed in bytes), the date each was created or changed and the time of the creation or changed will appear.

If you have two drives, you may obtain a directory on drive B: by typing in **DIR B:** and then depress the RETURN key.

You can also get information about any file on your disk by typing **DIR** and a filename. For example, if you have created a file named MYFILE.TXT, the command

DIR MYFILE.TXT

will give you a display of all the directory information (name of file, size of file, and date of creation or modification) for the file MYFILE.TXT.

The CHKDSK (Check Disk)Command

The CHKDSK command is used to check your disks for consistency and errors, much like a secretary proofreads a letter. CHKDSK analyzes the directories and the File Allocation Table on the disk that you specify. It then produces a status report of any inconsistencies, such as files which have no data in them.

To check the disk in drive A:, type:

CHKDSK A:

A status report will be displayed on the screen and any errors that it has found.

How to Name Your Files

The name of a typical system file (a file created by MS-DOS) looks like this:

NEWFILE.TXT

The name of a file consists of two parts: the filename is NEWFILE and the

filename extension is .TXT

A filename can be from one to eight characters long. The filename extension can be one to three characters long or not at all. Extensions are another way of distinquishing types of files. When naming your files, never use the following names for filenames or extensions as they are reserved for internal use:

AUX CON LST PRN NUL

Characters may be in upper or lower case; the operating system will translate into all upper case.

In addition to filename and extensions, the name of your files may include a drive designation. A drive designation tells the operating system to look on the disk in the drive designated to find the filename typed. For example, to find directory information about the file NEWFILE.TXT which is located on the disk in drive B:, type the following command:

```
DIR B:NEWFILE.TXT
```

Directory information about the file NEWFILE.TXT will be displayed on the screen.

The following list contains the characters you can use in creating filenames and extensions.

```
A-Z  0-9  # $ % & ' ( ) {} []  <> !
```

Do not use * or ? This will be explained in the following section.

Wild Cards

Two special characters called wild cards can be used in filenames and extensions: the asterisk * and the question mark?. These special characters give you greater flexibility when using filenames in system commands.

The ? Wild Card

A question mark ? in a filename or extension indicates that any character can occupy that postion. For example, the command

DIR TEST?RUN.TXT

will list all directory entries on the default drive (when no drive is designated, the operating system always assumes drive A:) that have 8 characters, begin with TEST and have any next character followed by RUN.TXT

Here are some examples of files that might be found with the above command:

TEST1RUN.TXT
TEST2RUN.TXT
TEST6RUN.TXT
TEST$RUN.TXT

The * Wild Card

An asterisk * in a filename or extension indicates that any character can occupy that position or any of the remaining positions in a filename or extension. For example:

DIR TEST*.TXT

will list all directory entries on the default drive with filenames that begin with the characters TEST and have are extension of .TXT Here are some examples of files that might be found with the above command:

TEST1RUN.TXT

```
TEST2RUN.TXT
TEST6RUN.TXT
TEST$RUN.TXT
TESTALL.TXT
TEST$&'.TXT
```

The wild card designation *.* refers to all files on the disk. Note hat this can be a very powerful and destructive tool when used in system commands. For example, the command DEL *.* deletes all files on the default drive, regardless of filename or extension.

Examples

To list the directory entries for all files named NEWFILE on drive A: (regardless of their filename extension), simply type:

```
DIR A:NEWFILE.*
```

To list the directory entries for all files with filename extension of .TST regardless of their filenames) on the disk in drive B:, type:

```
DIR B:????????.TST
```

This command is useful if, for example, you have given all your test programs an extension of .TST By using the DIR cmmand with the wild characters, you can obtain a listing of all your test files even if you do not remember all their names.

Batch Processing

Often you may find yourself typing the same sequence of commands over and over to perform some commonly used task. With MS-DOS, you can put the command sequence into a special file called a batch file. "Batches" of your commands in such a file are procesed as if they were typed at the terminal. Each batch file must be named with the .BAT extension, and is executed by typing the filename without its extension.

You create a batch file by using the EDLIN command.

EDLIN and Batch Files

The form of EDLIN is EDLIN <filename.BAT>

Enter the command EDLIN at the system prompt A:
You will see the following message:

```
EDLIN Version X.XX
New File
*
```

After the asterisk *, enter I for insert. The following will be displayed:

```
1:*
```

Enter a command such as DIR, and depress the RETURN key. The number 2 with an asterisk will appear. Enter the commands you want in the batch file, depressing the RETURN key after each entry.

When you have entered all the commands you need, depress the CTRL and Z keys together, and then the RETURN key. An Asterisk * will appear in the far right. Enter E to end and depress the RETURN key. You are at the system level. If you enter a DIR you will see the file you have created. You may execute the file by entering the filename only, do not include the extension .BAT

```
10:*^Z      (This is the CTRL and Z keys
            entry followed by RETURN)
 *E
 A:
```

Auto Loading Batch Files

If you want to run a specific program automatically each time you start your Sanyo computer, you may do so with the EDLIN command. Enter the EDLIN command followed by this file name: AUTOEXEC.BAT and create the file you wish following the instruction given previously.

When you start (boot up) the system, the command processor (COMMAND.COM) searches the disk for a file named AUTOEXEC.BAT. If the system finds this file, the Cile is immediately executed by the command processor and the date and time prompts are bypassed. If the system does not find an AUTOEXEC.BAT file when you first load the system disk, then the date and time prompts will be displayed.

You may have only one AUTOEXEC.BAT file per disk.

More Information about EDLIN

For more information on the EDLIN command, consult the Sanyo Software MS-DOS Reference manual available at your authorized Sanyo computer dealer.

HOW TO COPY YOUR FILES

Copy Command

Just as with paper files, you often need more than one copy of a disk files. You should always make back up copies of all programs in case something happens to your master. Software is costly.

The COPY command allows you to copy one or more files to another disk. You can also give the copy a different name if you specify the new name in the COPY command.

The COPY command can also make copies of files on the same disk. In this case, you must supply a different filename or you will overwrite the file. You can not make a copy of a file on the same disk unless you specify a different filename for the new copy.

The format of the COPY command is:

```
COPY filespec [filespec]
```

Examples:

```
COPY A:MYFILE.TXT A:NEWNAME.TXT
```

You have duplicated your file on drive A: It now has two names: MYFILE.TXT and NEWNAME.TXT

```
COPY A:MYFILE.TXT B:MYFILE.TXT
```

Copy the file MYFILE.TXT on the disk in drive A: to a file named MYFILE.TXT on the disk in drive B:

DISKCOPY Command

You should make back up copies of all your files, especially costly software.The DISKCOPY command copies the contents of a disk onto another disk. DISKCOPY is faster than COPY because it copies the entire contents of a disk in one operation.

The format of the DISKCOPY command is:

```
DISKCOPY [drive 1:][drive 2:]
```

Drive 1: is the disk drive that contains the disk that you want to copy; drive 2: is the disk drive that contains the blank formatted disk.

SINGLE DRIVE if you want to make a copy of a disk, type:

DISKCOPY

The system will display:

Insert source diskette into drive A:**<CR>**
Insert formatted target diskettes into drive A:**(<CR>)**
Press any key when ready

Remove the system disk. Insert the disk to be copied and depress the RETURN key. Remove the disk after the cursor drops to the second prompt . Insert a blank formatted disk and depress the RETURN key. The files will be read from memory on to the disk. Double Drives If you have two drives, you would enter:

DISKCOPY A: B:

Remove the system disk from drive A:, insert the disk

to be copied and a blank formatted disk in drive B: Continue as in single drive. After the disk is copied, The following prompts will be displayed.

```
Copy complete
Copy another (Y/N)?
```

Type **Y** (for yes) if you wish to copy other disks or make a second copy. If you type **N** (for no), the default drive prompt is displayed and you are back on the operating system level.

Note: If either of the disks you are using has defective tracks, DISKCOPY will not work. Use the COPY command to back up your disks in these cases. The COPY command will skip over defective tracks.

COMMANDS

Commands are a way of communicating with your Sanyo computer. By entering commands, you can ask the operating system to perform useful tasks.

Single Disk Drive Users

For single disk drive users, system commands are exactly the syntax as for two drive users. The difference lies in your perception of the "arrangement" of the drives.

You must think of the system as having two disk drives-drive A: and drive B:.However, instead of A: and B: meaning physical disk drive mechanisms, the A: and B: designate diskettes. Therefore, when drive B: is specified, disks should be swapped.

Information Common To All MS-DOS Commands

1. Commands are usually followed by one or more options.

2. Commands and options may be entered in upper case or lower case or combination.

3. Commands and options must be separated by delimiters. Because they are the easiest, you will usually use the space and comma as delimiters.

   For example:

   ```
   DEL MYFILE.OLD NEWFILE.TXT

   RENAME,THISFILE,THATFILE
   ```

   You can also use the semicolon ; the equal sign = or the tab key as delimiters when entering commands.

4. Do not separate a file specification with delimiters, since the colon and the period already serve as delimiters.

5. When instructions say "Press any key" you can press any alpha (A-Z) or numeric (0-9) key.

6. You must include the filename extension when referring to a file that already has a filename extension.

7. You can abort commands when they are running by depressing the CTRL key and the C key at the same time.

8. Commands take effect only after you have depressed the RETURN key.

9. Wild cards (global filename characters) and device names (AUX, CON, LST, PRN, or NUL) are not allowed in the names of any commands.

10. When commands produce a large amount of output to the screen, the display will automatically scroll to the next screen. You can depress the CTRL key and the S key at the same time to suspend the display. Press any key to resume the display on the screen.

11. The prompt from the command processor is the default drive designation (drive A) plus a colon A:

12. Disk drives will be referred to as source drives and destination drives A source drive is the drive you will be transferring information from. A destination drive is the drive you will be tansferring information to.

Control Character Functions

While commands are being entered, MS-DOS recognizes seven control character functions.

CONTROL CHARACTER	FUNCTION
CTRL N	Cancel echoing of output to the printer.
CTRL C	Abort current command.
CTRL H	Remove last character from the command line and erase the character from the screen.
CTRL J	Insert a physical end-of-line, but do not empty the command line. Use linefeed to extend the current logical line beyond the physical limits of one screen line.
CTRL P	Echo screen output to the printer.
CTRL S	Suspend display of output to the screen. Press any key to continue.
CTRL X	Cancel the curent line, empty the command line, and then output a back slash, RETURN key, and linefeed.

COPY

Purpose

Copies one or more files to another disk. If you prefer, you can give the copies different names. This command can also copy files on the same disk.

Form

```
COPY <filespec>[filespec]
```

Description

1. IF the second filespec option is not given, the copy will be on the default drive (drive A:) and will have the same name as the original file (the first filespec option). If the first filespec is on the default drive and the second filespec is not specified, the COPY will be aborted. Copying files to themselves is not allowed. The system will display the error message:

```
File cannot be copied onto itself
0 File(s) copied
```

The second option may take three forms:

a) If the second option is a drive designation (d:) only, the original file is copied with the original filename to the designated drive.

b) If the second option is a filename only, the original file is copied to a file on the default drive with the filename

specified.

c) If the second option is a full filespec, the original file is copied to a file on the default drive with the filename specified.

2. ASCII and binary files may be arbitrarily combined by using **/B** on binary files and **/A** on ASCII files. A switch (/B or /A) takes effect on the file it is placed after and applies to all subsequnet files until another switch is found.

A **/A** or **/B** switch on the destination file determines whether or not CTRL Z is placed at the end of the file. Source files read while /A is in effect have CTRL Z stripped off. If /A is in effect when the file is written, a single CTRL Z will be put back.

3. The COPY command also allows concatentation (joining) while copying. Concatentation is accomplished by simply listing any number of files as options to COPY, separated by **+**. See example 1.

4. You can also combine several files using wild cards, into one file. See example 2,3 and 4.

Examples

1) COPY A.XYZ + B.COM + B:C.TXT BIGFILE.CRP

This command concatenates files named A.XYZ, B.COM and (on B drive) C.TXT and places them in the file called BIGFILE.CRP (on the default drive, A since a drive was not specified after the command COPY).

2) COPY *.LST COMBIN.PRN

This command would take all files with an extension of

3) COPY *.LST + *.REF *.PRN

This command combines all files with the extensions filename. All TEST.LST and TEST.REF would be combined in TEST.PRN, all SAMPLE.LST and SAMPLE.REF would be combined in SAMPLE.PRN and so forth.

4) COPY MBC550.* + TEST?.* ONEFILE.*

This command combines all MBC550 files and TEST files with any fifth character that have the same extension into the file ONEFILE with the same extension.

DATE

Purpose

Allows you to change the date known to the system.

Form

```
DATE [<mm>-<dd>-<yy>]
```

Description

1. If you type DATE and depress the RETURN key the following message will appear:

```
Current date is mm-dd-yy
Enter new date:
```

Depress the RETURN key if you do not want to change the date shown.

If you want to change the date enter the number of the month followed by a hyphen, the number of day followed by a hyphen and all four digits of the year or only the last two.

Hyphens may be substituted for slashes (/).

2. You can also type a particular date after entering DATE and before depressing the RETURN key. See example.

3. If the options or separators are not valid, the following message will be displayed:

```
Invalid Date
Enter new date:
```

You must enter the date or depress the RETURN key to terminate the DATE command.

Example

```
DATE 5/6/84
```

DEL

Purpose

Deletes all files with the designated filespec.

Form

```
DEL [filespec]
```

Description

1. If the filespec is *.*, the prompt Are you sure? appears. If a **Y** or **y** is entered as a response, then all files are deleted as requested.

2. You can also type **ERASE** for the DEL command.

CAUTION: DEL AND ERASE ARE POWERFUL COMMANDS. PLEASE USE WITH CARE. MAKE SURE YOU AHVE BACK UP FILES AND DISKS BEFORE USING THESE COMMANDS.

Examples

1. DEL MBC550.TXT

This command erases file MBC550.TXT

2. ERASE MBC550.TXT

This command also erases file MBC550.TXT

3. DEL MBC550.*

This command will erase all MBC550 files with any extension.

DIR

Purpose

This command lists the files in a directory

Format

```
DIR [filespec][/P][/W]
```

Description

1. All directories on drive A will be listed if the DIR command is entered alone.

2. If the drive is specified (see example 1) all files on that drive will be listed.

3. If only a filename is specified with no extension (see example 2), then all files with that filename on drive A will be listed.

4. You may use wild cards in filename specification. See example 3.

5. Files are listed with their size in bytes and with the time and date of their creation or modification.

6. The /P switch selects Page Mode. With /P, display of the directory pauses after the screen is filled. To resume display of output, press any key.

7. The /W switch selects Wide Display. With /W, only the filenames are displayed withour other file information. Files are displayed five per line.

Examples

1. DIR B:

2. DIR MBC550.*

3. DIR ??DE. TXT

EDLIN

Purpose

This command creates, modifies and displays files.

Form

```
EDLIN <filespec>
```

Description

1. <filespec> is the name of the files you want to create or modify.

2 After entering the command EDLIN and a new filename, depress the RETURN key. The following prompt will be displayed:

```
EDLIN Version x.xx
New file
*
```

Insert a command listed below. Command I begins creating a new file:

```
*I
   1:*
```

Write the command(s) you want for the files followed by depressing the RETURN key for each command. To end , depress both the CTRL and Z key at the same time, then the RETURN Key. You will be back to the system level.

3. To edit an existing file, enter the EDLIN command followed by the complete filename and depress the

RETURN key. The following prompt will appear:

```
EDLIN Version x.xx
End of input file
*
```

This indicates that the file has been found and read. Enter the EDLIN command you want. To terminate editing a file, enter E after the asterisk * prompt followed by depressing the RETURN key. You will be returned to the system level, A:.

4. EDLIN commands:

Command	Explanation
<line>	Edits line no.
A	Appends lines
D	Deletes lines
E	Ends editing
I	Inserts lines
L	Lists text
Q	Quits editing
R	Replaces lines
S	Searches text
W	Writes lines

For detailed information on EDLIN, consult the Sanyo Software MS-DOS Reference manual, available at your authorized Sanyo computer dealer.

EXE2BIN

Purpose

Converts .EXE (executable) files to binary format. This results in a saving of disk space and faster program loading.

Form

```
EXE2BIN <filespec>[d:][<filename>[<.ext>]]
```

Description

1. This command is useful only if want to convert .EXE files to binary format.

2. The file named by filespec is the file that will be changed to binary (input file). If no extension is specified, it defaults to .EXE

The input file is converted to .COM file format (memory image of the program) and placed in the output file (converted file).

3. For double drive systems, if you do not specify a drive, the drive of the input file will be used.

4. If you do not specify an output filename, the input filename will be used.

5. If you do not specify a filename extension in the output filename, the new file will be given an extension of .BIN

6. IF EXE2BIN finds an error, one or more of the following error messages will be displayed:

File not found
The file is not on the disk specified.

File creation error
EXE2BIN can not create the output files.

Insufficient disk space
There is not enough disk space to create a new file.

Fixups needed - base segment (hex):
The source (.EXE) file contained information indicating that load segment is required for the file. Specify the absolute segment address at which the finished module is to be located.

File can not be converted
The input file is not in the correct format.

WARNING - Read error on .EXE file.
Amount read less than size in header
This is a warning message only.

Example

```
EXE2BIN MBC550.TST MBC555
```

The file, MBC550.TST will be converted to a binary file named MBC555.BIN

FORMAT

Purpose

Formats a disk to accept system files.

Form

```
FORMAT [d:][/S]
```

Description

1. This command initializes the directory and file allocation tables.

2. If no drive is specified, d:, it is assumed to be drive A.

3. /S is an option that allows you to copy the operating system onto the disk as you format it.

4. The following message will appear if no drive is designated:

```
Remove disk now in drive A
Insert a blank disk in drive A and depress
any key
```

Remove the system disk. Place a blank disk in drive a and depress any key.Formatting will begin. The following prompt will appear as long as the disk is being formatted:

```
Formatting...
```

Formatting is complete when this prompt is displayed:

Formatting completed
Format another disk (Y/N)?

If you wish to format another blank disk, remove the formatted disk, insert a blank disk in the drive and enter **Y** and depress the RETURN Key. IF you do not wish to format another disk, enter **N** and depress the RETURN key. You will see the system prompt A: indicating you are back to command mode.

5. The following message will appear if you designate drive B: as the drive to be formatted:

Insert a blank disk in drive B: and depress any key

CAUTION FORMATTING A DISK WITH INFORMATION ON IT WILL DESTROY ALL THE CONTENTS OF THE DISK. PLEASE BE CAREFUL TO CHECK THAT THE DISK IS BLANK OR THE DATA ON THE DISK IS NO LONGER NEEDED.

PAUSE

Purpose

Suspends execution of a batch file.

Form

```
PAUSE [comment]
```

Description

1. During the execution of a batch file, you may need to change disks or perform some other action. PAUSE suspends execution until you press any key, except the CTRL and C keys depressed together.

2. When the PAUSE command is encountered by the system, this prompt will be displayed:

Strike a key when ready ...

3. If you depress the CTRL and C keys together at this point, another prompt is displayed:

Abort batch job (Y/N)?

If you enter Y in response, executoin of the reminder of the batch command file is ended ands control is returned to the operating sysem.

4. An option comment or message, [comment], may be entered on the same line as pause. You may want to prompt the user of the batch file with some meaningful message when the batch file has paused. The comment is displayed before the "Strike a key" message.

REM

Purpose

Displays remarks entered on the same line as REM when encountered during execution of a batch file.

Form

```
REM [comment]
```

Description

1. The REM command has no other effect. The only delimiters for the comment are any one of the three legal delimiters to start the comment (blank space, tab, or comma).

Example

```
1:* REM  PLEASE CHECK THE BYTES LEFT ON
   TEST.ONE
2:* DIR TEST.ONE
```

REN

Purpose

Changes names of files

Form

```
REN <filespec><filename>
```

Description

1. The first option (filespec) must be given a drive designation if the disk is in any other drive than A. Any drive designation for the second option (filename) is ignored. The file will remain on the disk where it resides.

2. The wild card characters may be used in either option. All files matching the first filespec are renamed. If wild card characters appear in the second filename, corresponding character positions will not be changed.See example 2.

3. An attempt to rename a filespec to a name already present in the directory will result in the error message "File not found".

Examples

1. REN TEST.ONE TEST.TWO

2. REN *.ONE *.TWO

TIME

Purpose

Displays and sets the time.

Form

```
TIME [<hh>[:<mm>]]
```

Description

1. If the TIME command is entered without any arguments, the following message is displayed:

```
Current time is hh:mm:ss.cc
Enter new time:
```

Press the REURN key of you do not want to change the time shown. A new time may be given as an option to the time command as in: TIME 8:20

The new time must be entered using numerals only; letters are not allowed. The allowed options are:

```
hh = 00-24
mm = 00-59
```

2. The hour and minute entries must be separated by colons. You do not have to type the seconds <ss> or hundredths of seconds <cc> options.

3. The system uses the time entered as the new time if the options and separators are valid. If the options or separators are not valid, the system displays the message:

```
Invalid time
Enter new time:
```

At this point you will enter a valid time or depress the RETURN key to go back to system level, A:

TYPE

Purpose

Displays the contents of the file on the screen.

Form

```
TYPE <filespec>
```

Description

1. The TYPE command is used to examine a file without modifying it.

2. A display of binary files causes control characters (such as CTRL-Z) to be sent to the screen, including buzzer and escape sequences.

Example

```
TYPE SANYO.BAS
```

CHAPTER 5

TECHNICAL REFERENCE

SPECIFICATIONS

Dimensions/Weight

Mainframe : 15 (width) X 14 1/5" (depth) X
4 2/5" (height) / 19.9 lbs.

Keyboard: 17 2/5" (width) X 6 4/5" (depth) X
1 3/5" (height) / 4.4 lbs.

Power Supply

Voltage: Local Voltage ± 10%

Frequency: 50/60Hz

Power Consumption: 50W

Environmental Requirements

Ambient Temperature:

Operating	50 to 95 degrees(F)
Not operating	5 to 149 degrees(F)

Relative humidity:

Operating	30% to 80% without dewing
Not operating	20% to 80% without dewing

Options

Sanyo color CRT display monitor(RGB only, CRT-70)
Sanyo monochromatic CRT display monitor - CRT-36
Sanyo Printer - PR-5000/5500 and the complete
Sanyo line of printers
Second Sanyo drive - FDD-1655
Extension RAM - MBC-64K
RS232C Interface board - MBC-232C
Apple compatible joy stick

CHARACTERS CODES

Explanation

The following chart depicts the character codes for the SANYO computer series. You will note immediately there are 256 possible codes. The character set is a combination of both the standard 7-bit ASCII set and the graphic characters that are on the second half of the set (with the 8 bit at a high state). The configuration for the digits in the X dimension repesent bits:

B7 B6 B5 B4

and the bits in the Y (vertical) dimension represent the bits:

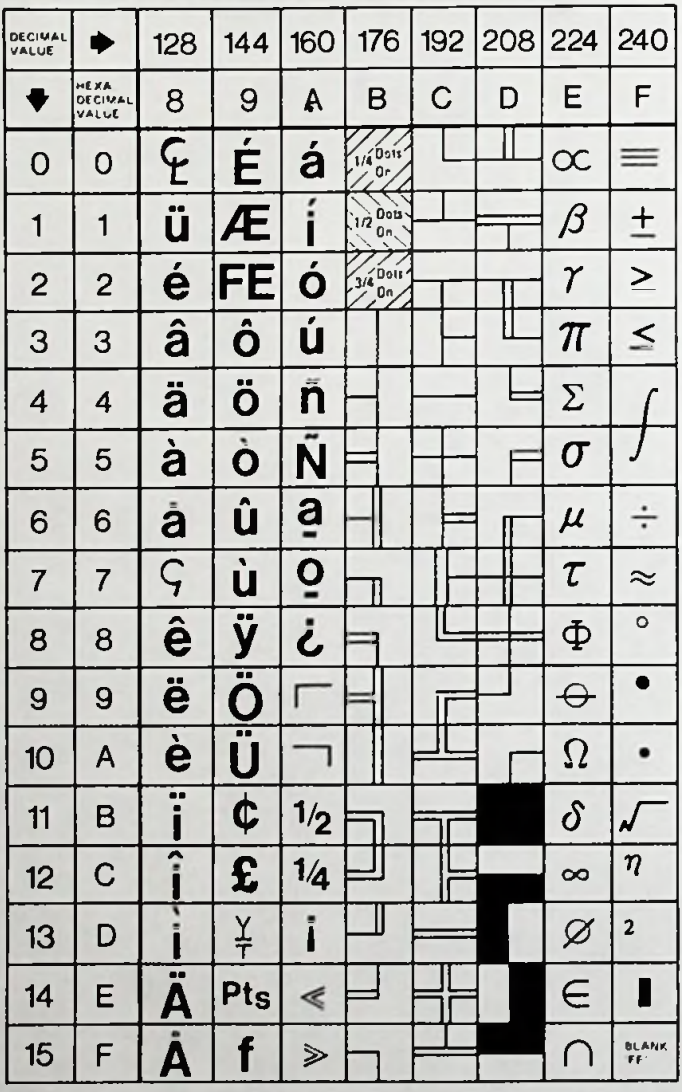
B3 B2 B1 B0

The combination of these two "nybbles" represent the character code byte.

Character Set (00-7F)

DECIMAL VALUE ➡	0	16	32	48	64	80	96	112
⬇ HEXA DECIMAL VALUE	0	1	2	3	4	5	6	7
0 0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1 1	☺	◄	!	1	A	Q	a	q
2 2	☻	↕	"	2	B	R	b	r
3 3	♥	!!	#	3	C	S	c	s
4 4	♦	¶	$	4	D	T	d	t
5 5	♣	§	%	5	E	U	e	u
6 6	♠	▬	&	6	F	V	f	v
7 7	•	↨	’	7	G	W	g	w
8 8	◘	↑	(	8	H	X	h	x
9 9	○	↓	)	9	I	Y	i	y
10 A	◙	→	*	:	J	Z	j	z
11 B	♂	←	+	;	K	[	k	{
12 C	♀	∟	,	<	L	\	l	¦
13 D	♪	↔	-	=	M	]	m	}
14 E	♫	▲	.	>	N	^	n	~
15 F	☼	▼	/	?	O	_	o	△

Character Set (80-FF)



LIST OF RESERVED WORDS

The following are reserved words which have some meaning as functions or statements. These words should not be used as variables in your programs or as labels in any form. In fact, your variables cannot have these words as pieces of your text.

ABS	CVS	GCURSOR	LOAD
ALL		GET	LOC
AND	DATA	GO	LOCATE
AS	DATE$	GOSUB	LOF
ASC	DEF	GOTO	LOG
ATN	DEFDBL		LPOS
AUTO	DEFINT	HEX$	LPRINT
BASE	DEFSNG		LSET
BEEP	DEFSTR	IF	
	DELETE	IMP	MERGE
CALL	DIM	INIT	MID$
CDBL		INKEY$	MKD$
CHAIN	ELSE	INP	MKI$
CHR$	END	INPUT	MKS$
CINT	EOF	INPUT$	MOD
CIRCLE	EQV	INSTR	
CLEAR	ERASE	INT	NAME
CLOSE	ERL		NEW
CLS	ERR	KEY	NEXT
COLOR	ERROR	KILL	NOT
COM	EXP		
COMMON		LEFT$	OCT$
CONT	FIELD	LEN	OFF
COS	FILES	LET	ON
CSNG	FIX	LFILES	OPEN
CSRLIN	FN	LINE	OPTION
CVD	FOR	LIST	OR
CVI	FRE	LLIST	OUT
PACK$	SYMBOL		

```
PAINT           SYSTEM
PEEK
POINT           TAB
POKE            TAN
POS             THEN
PRESET          TIME$
PSET            TINPUT
PRINT           TO
PUT             TROFF
                TRON
RANDOMIZE
READ            UNPACK$
REM             USING
RENUM           USR
RESET
RESTORE         VAL
RESUME          VARPTR
RETURN          VIEW
RIGHT$
RND             WAIT
RSET            WEND
RUN             WHILE
                WIDTH
SAVE            WINDOW
SCREEN          WRITE
SEG
SET             XOR
SGN
SIN
SPACE$
SPC
SQR
STEP
STOP
STR$
STRING$
SUB
SWAP
```

BASIC Error Message List

NO.	CODE	MESSAGE	EXPLANATION
1	01	NEXT without FOR	FOR was not found in the FOR/NEXT loop.
2	02	Syntax error	Syntax error.
3	03	RETURN without GOSUB	RETURN was found although the GOSUB was not executed.
4	04	Out of DATA	Number of constants in the DATA state ment is less than the number of va riables in the the READ statement.
5	05	Illegal function call	Illegal statement or function call was found.
6	06	Overflow	Numeric value overflow. Calculation result too large.
7	07	Out of memory	Memory was exhausted. Also too large an array.

8	08	Undefined line number	Specified line number was not found.
9	09	Subscript out of	Array subscript was out of the range.
10	0A	Duplicate definition	The definition was duplicated. Two DIMENSION statements for the same variable.
01	0B	Division by zero	Number divided by 0.
12	0C	Illegal direct	It cannot be directly executed A statement illegal in the direct mode.
13	0D	Type mismatch	Variable type does not match the data type.
14	0E	Out of string space	The character string area was exhausted.
15	0F	String too long	The character length exceeded 255 characters.
16	10	Printer not ready	The printer is not ready to print

17	11	Can't continue	Cannot be continued (CONT).
18	12	Undefined user	The user function function was not defined.
19	13	No RESUME	RESUME was not found in the error processing routine.
20	14	RESUME without error	RESUME was executed although no error has occurred.
21	15	Unprintable error	Undefined error.
23	17	Line buffer overflow	The entered line was too long.
24	18	Position not on	The position is screen out of the screen.
25	19	Print field overflow	The data cannot be stored in the specified print field.
26	1A	FOR without NEXT	NEXT was not found in the FOR/NEXT loop.
28	1C	Out of array space	Out of array space.
29	1D	WHILE without WEND	WEND was not

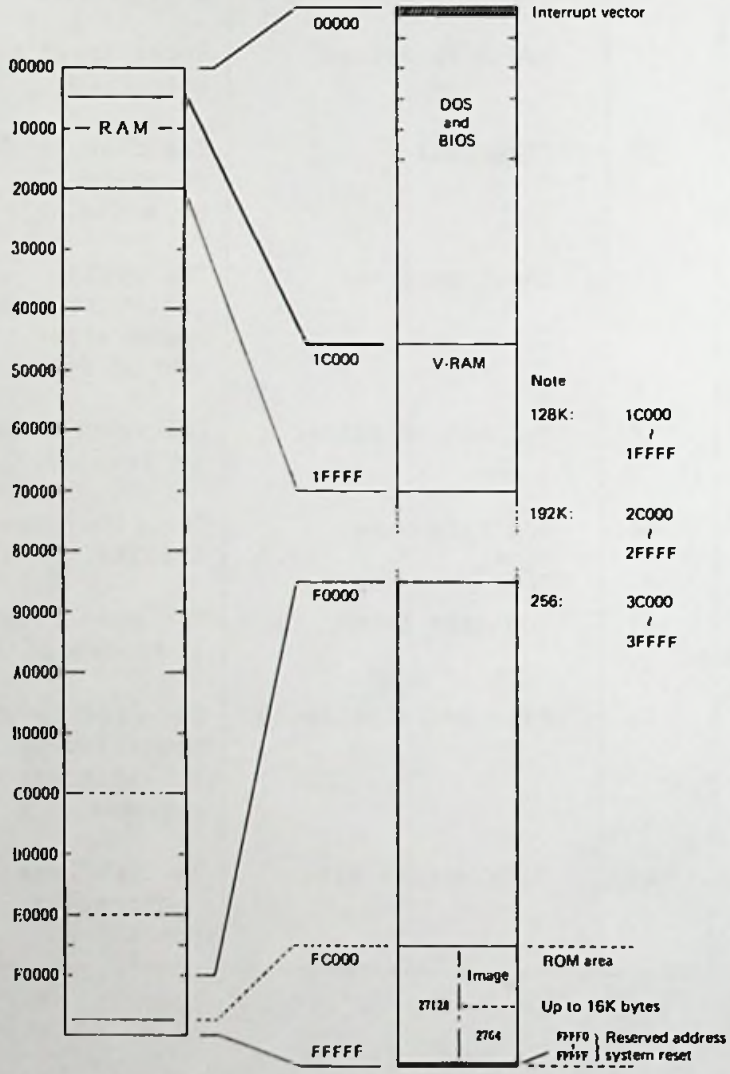
			found in the WHILE/WEND loop.
30	1E	WEND without WHILE	WHILE was not found in the WHILE/WEND loop.
51	33	Port I/O Error	RS232 not set. Baud rate mismatched.

DISK ERROR MESSAGES

50	32	Field overflow	The FIELD length exceeded the specified record length. More than 255 characters.
52	34	Bad file number	The file number is invalid.
53	35	File not found	The specified file was not found.
54	36	Bad file mode	The file open mode was not found.
55	37	File already open	The file was already opened.
56	38	Bad file data	The file data is invalid.
57	39	Disk I/O error	The disk is invalid.
58	3A	File already exists	The specified file already exists.

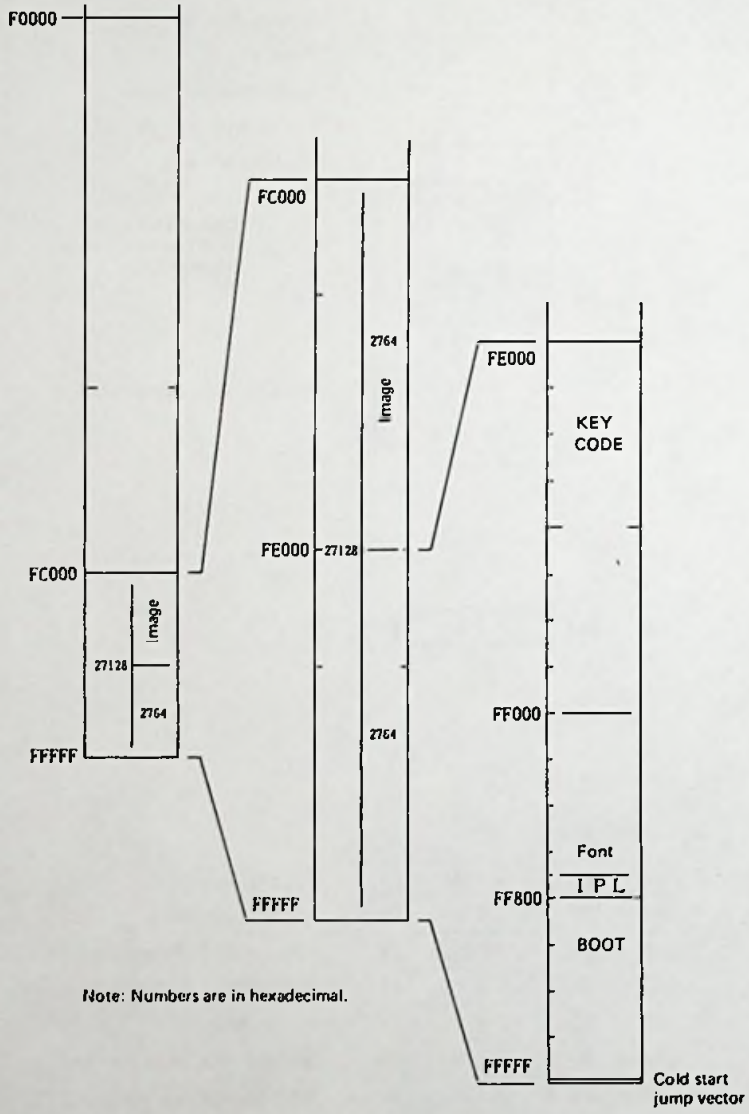
59	3B	Disk not ready	The disk device is not ready to operate.
60	3C	Bad disk select	Nonexistent drive specified.
61	3D	Disk full	The disk is full; no space to write.
62	3E	Input past end	The write instruction was found after the end of file.
63	3F	Bad record number	The record number is invalid.
64	40	Bad file name	The file name is invalid.
67	43	Too many files	Too many files are opened.
68	44	File write protected	The file is write protected or the diskette has been replaced.
69	45	NAME across disk	The NAME was incorrectly specified.

MEMORY MAP



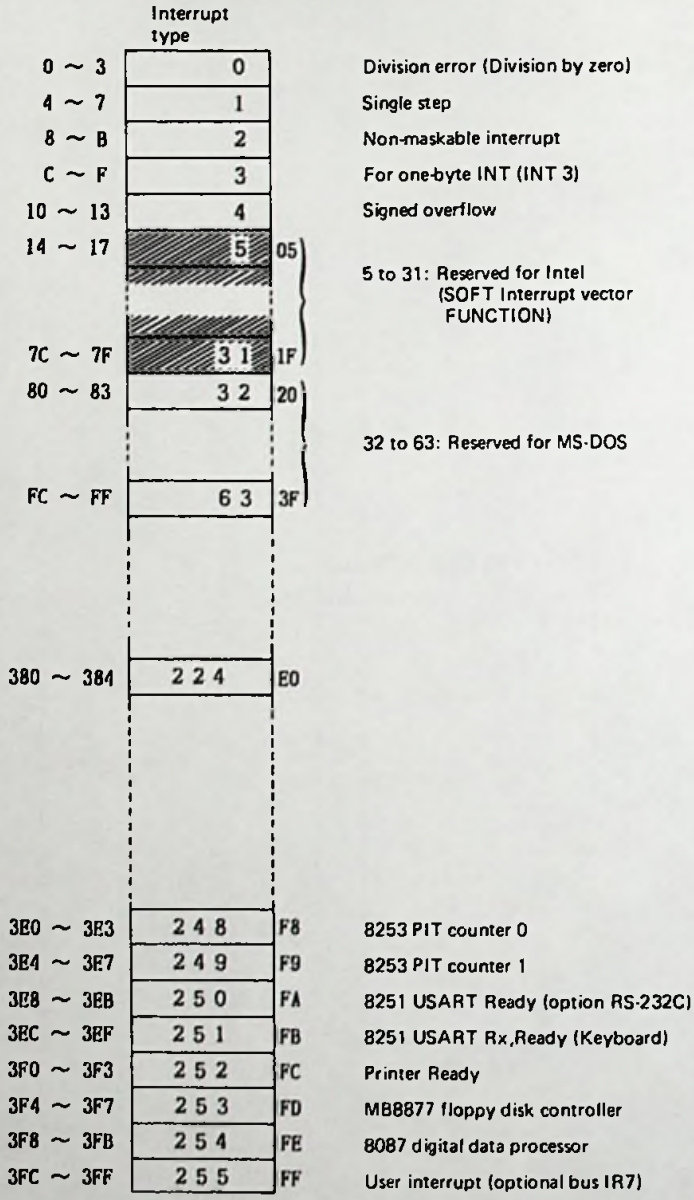
Note: Numbers are in hexadecimal.

ROM-MAP



Note: Numbers are in hexadecimal.

INTERRUPT VECTOR



I/O Interrupt Controller 8259A

8259A control interrupts. 8259A has eight levels of interrupt request pins. Interrupt requests arriving at the respective pins are detected when the signal goes high or low.

Pin	Vector	
IR0	248(F8)	Interval timer (established to 10 msec in period)
IR1	249(F9)	Interval timer 1 (clock counter)
IR2	250(FA)	USART Ready (RS232C)
IR3	251(FB)	USART Rx Ready (keyboard)
IR4	252(FC)	Printer Ready
IR5	253(FD)	Floppy disk controller
IR6	254(FE)	8087 digital data processor
IR7	255(FF)	User interrupt (external bus IR7) (*1) *1 Negative logic

The priorities of pins are fixed to the order of IR0 (high) to IR7 (low).

Operation Command Word (OCW) 1 changes the priorities. Of IR0 to IR7, only IR1 to IR3 are associated with BIOS.

Pins other than IR1 to IR3 are masked by the interrupt mask register (MR) in 8259. The IMR can be read/written by performing Read/Write command on I/O address 02.

If IR2 or IR3 is masked, response may be delayed because fetching results in polling by software. Within BIOS, only the I/O routine related to the floppy disk is disabled. No interrupts are accepted while this routine being executed.

Of IR0 to IR7, interrupts other than IR1 k to IR3 and IR5 can be used by a transient program.

To use interrupts, write the segment and offset the interrupt servicing routine and reset the corresponding IMR bits. If the program has terminated, restore the rewritten vector, set the reset IMR bits, then return control to DOS. IF the vector and the IMR are not restored, an overrun results.

NOTE: to use interrupt, master the operations of 8088 and 8259A beforehand.

Interrupt 13
Diskette I/O

This interface provides access to the 5 1/4" diskette drives.

Input

(AH) = 0 Reset diskette system
Hard reset to nec, prepare command, recal reqd on all drives
(AH) = 1 Read the status of the system into (AL)
Diskette status from last op'n is used
Registers for READ/WRITE/VERIFY/FORMAT
(DL) - Drive number (0-3 allowed, value checked)
(DH) - Head number (0-1 allowed, not value checked)
(CH) - Track number (0-39, not value checked)
(CL) - Sector number (1-8, not value checked)
(AL) - Number of sectors (max = 8, not value checked)

(ES:BX) - Address of buffer (not required for verify)

(AH) = 2 Read the desired sectors into memory
(AH) = 3 Write the desired sectors from memory
(AH) = 4 Verify the desired sectors

Data variable -- Disk pointer
Double word pointer to the current set of diskette parameters

Output

AH = Status of operation
Status bits are defined in the equates for

diskett status variable in the data segment of this module

CY = 0 Successful operation (AH=0 on return)

CY = 1 Failed operation (AH has error reason)

For READ/WRITE/VERIFY

DS,BX,DX,CH,CL preserved

NOTE: (1) If an error is reported by the diskette code, the appropriate action is to reset the diskette, then retry the operation. On read acceses, no motor start delay is taken so that three retries are required on reads to ensure that the problem is not due to motor start-up.

(2) Inside the ( ) shows the Registor and data is in hexadecimal.

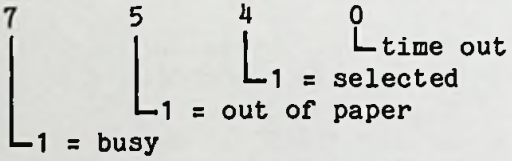
Interrupt 17
Printer I/O

This routine provides communication with the printer.

(AH) = 0 Print the character in (AL) on return, AH = 1 If character could not be printed (time out)
Other bits set as n normal status call

(AH) = 1 Initialize the printer port
Returns with (AH) set with printer satatus

(AH) = 2 Read the printer status into (AH)



(DX) = printer to be used (0,1,2) corresponding to actual values in printer base area

Data area printer base contains the base address of the printer card(s) available (located at beginning of data segment, 408H absolute, 3 words)

Registers AH is modified
All other unchanged

Interrupt 10
Video I/O

These routines provide the CRT interface.
The following functions are provided:

(AH) = 0 Set mode (AL) contains mode value
(AL) = 2 80X25 BW
(AL) = 3 80X25 color

(AH) = 2 Set cursor position
(DH,DL) = Row, column (0,0) is upper left
(BH) = Page number (must be 0 for graphics modes)

(AH) = 3 Read cursor position
(BH) = Page number (must be 0 for graphics modes)
On exit (DH,DL) = Row, column of current cursor
(CH,CL) = Cursor mode currently set

(AH) = 5 Select active display page (valid only for alpha modes)
(AL) = New page value (0-7 for modes 0&1, 0-3 for modes 2&3)

(AH) = 6 Scroll aotive page up
(AL) = Number of lines, input lines blanked at bottom of window
AL = 0 means blank entire window
(CH,CL) = Row, column of upper left corner of scroll
(DH,DL) = Row, column of lower right corner of scroll
(BH) = Attribute to be used on blank line

(AH) = 7 Scroll active page down
(AL) = Number of lines, Input lines blanked at top of window
AL = 0 means blank entire window
(CH,CL) = Row, column of upper left corner of scroll
(DH,DL) = Row, column of lower right corner of scroll
(BH) = Attribute to be used on blank line

CHARACTER HANDLING ROUTINES

(AH) = 8 Read attribute/character at current cursor position
(BH) = Display page (valid for alpha modes only)
On exit:
(AL) = Characters read
(AH) = Attribute of character read (alpha modes only)

(AH) = 9 Write attribute/character at current cursor position
(BH) = Display page (valid for alpha modes only)
(CX) = Count of characters to write
(AL) = Characters to write
(BL) = Attribute of character (alpha)/color of char (graphics) see note on write dot for bit 7 of BL =I.

(AH) = 10 Write character only at current cursor position
(BH) = Display page (valid for alpha modes only)
(CX) = Count of characters to write
(AL) = Character to write

ASCII TELETYPE ROUTINE FOR OUTPUT

(AH) = 14 Write teletype
(AL) = Character to write
(BL) = Foreground color in graphics mode
(BH) = Display page in alpha mode
NOTE -- Screen width is controlled by previous mode set

(AH) = 15 Current video state
Returns the current video state
(AL) = Mode currently set (see AH = 0)
(AH) = Number of character columns on screen
(BH) = Current active display page

CS,SS,DS,ES,BX,CX,DX preserved during call. All others destroyed.

Interrupt 12
Memory Size Determine

This routine determines the amount of memory in the system as represented by the switches on the planar. Note that the system may not be able to use I/O memory unless there is a full complement of 64K bytes on the planar.

Output

(AX) = Number of contiguous 1K blocks of memory

Interrupt 11
Equipment Determination

This routine attempts to determine what optional devices are attached to the system.

Input

No registers
The equip flag variable is set during the power on diagnostics using the following hardware assumptions:
Port 60 = Low order byte of equipment
Port 3FA = Interrupt ID register of 8250
Bits 7-3 are always 0
Port 378 = Output port of printer -- 8255 port that can be read as well as written

Output

(AX) is set, bit significant, to indicate attached I/O
Bit 15,14 = Number of printers attached
Bit 13 Not used
Bit 12 = Game I/O attached
Bit 11,10,9 = Number of RS232 cards attached
Bit 8 Unused
Bit 7,6 = Number of diskette drives
00 = 1, 01 = 2, 10 = 3, 11 = 4 only if Bit 0 = 1
Bit 5,4 = INITIAL VIDEO MODE
00 - Unused
10 - 80X25 BW using color card
Bit 3,2 = Planar RAM size (00=16K,01=32K,10=48K,11=64K)
Bit 1 Not used
Bit 0 = IPL from diskette -- this bit indicates that there are diskette drives on the system
No other registers affected

Interrupt 16
  Keyboard I/O Interrupt Routine

These routine provide keyboard support.

INPUT

(AH)=0 Read the next ASCII character struck from the keyboard
Return the result in (AL), scan code in (AH)

(AH)=1 Set the Z flag to indicate if a code is available to be read
(ZF)=1 -- No code available
(ZF)=0 -- Code is available
IF ZF = 0 , the next character in the buffer to be read is in AX, and the entry remains in the buffer.

KEYBOARD TRANSLATION TABLE

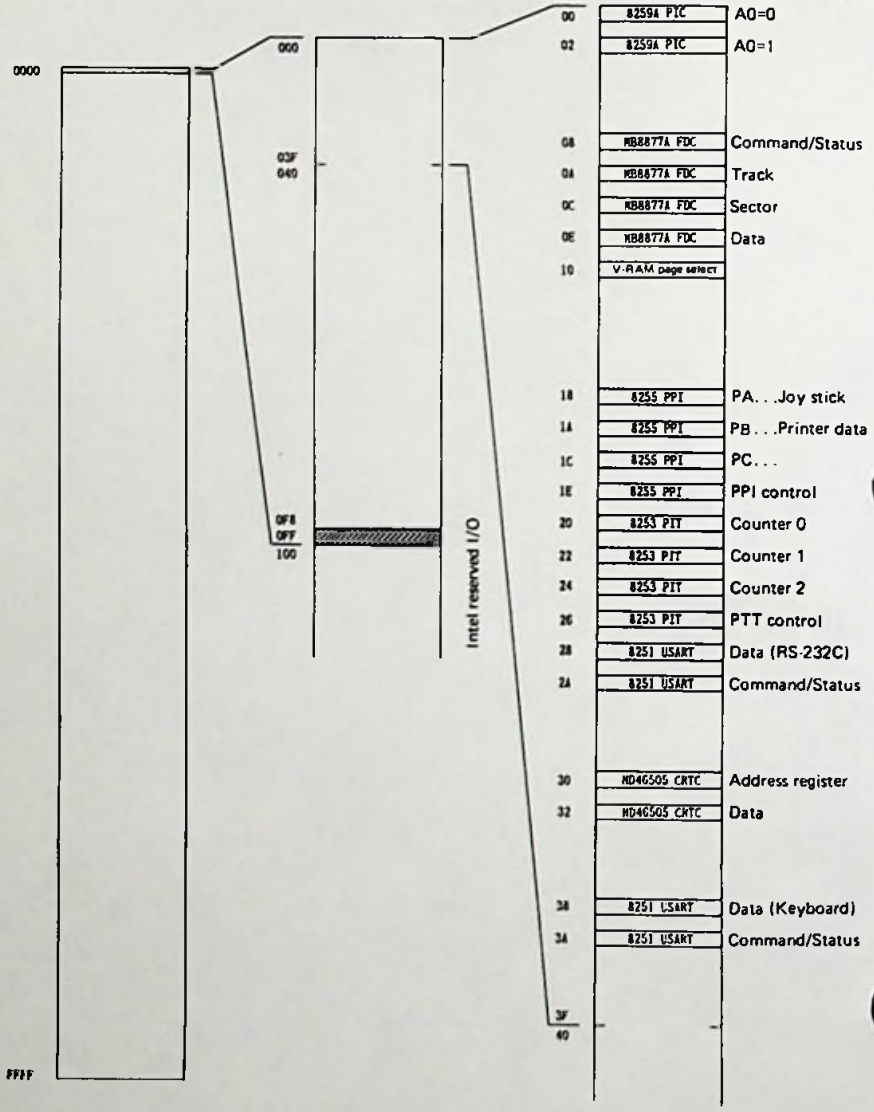
Sanyo keyboard and IBM keyboard differs a little bit a
the following table shows difference of Sanyo keyboa
from IBM keyboard.

IBM	Sanyo
ALT	CTRL + SHIFT
CTL + PF1	CTRL + =
CTL + PF2	CTRL + SHIFT + {
CTL + PF3	CTRL + SHIFT + }
CTL + PF4	CTRL + SHIFT + :
CTL + PF5	CTRL + SHIFT + "
CTL + PF6	CTRL + ;
CTL + PF7	CTRL + '
CTL + PF8	CTRL + ,
CTL + PF9	CTRL + .
CTL + PF10	CTRL + /
ALT + PF1	CTRL + PF1
ALT + PF2	CTRL + PF2
ALT + PF3	CTRL + PF3
ALT + PF4	CTRL + PF4
ALT + PF5	CTRL + PF5
ALT + PF6	CTRL + SHIFT + PF1
ALT + PF7	CTRL + SHIFT + PF2
ALT + PF8	CTRL + SHIFT + PF3
ALT + PF9	CTRL + SHIFT + PF4
ALT + PF10	CTRL + SHIFT + PF5
SHIFT + PF1	CTRL + 1
SHIFT + PF2	CTRL + 2
SHIFT + PF3	CTRL + 3
SHIFT + PF4	CTRL + 4
SHIFT + PF5	CTRL + 5
SHIFT + PF6	CTRL + 6
SHIFT + PF7	CTRL + 7
SHIFT + PF8	CTRL + 8

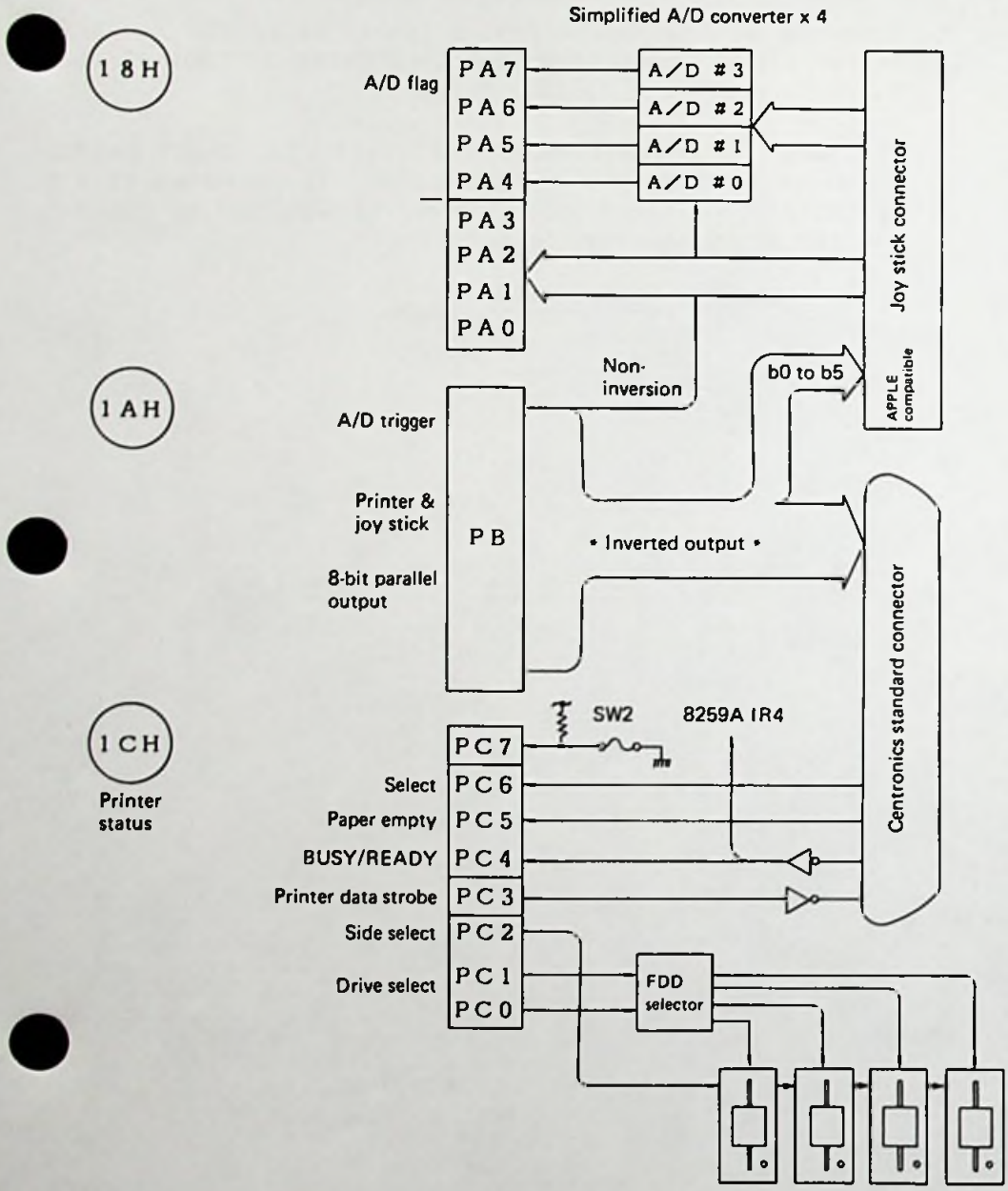
SHIFT + PF9	CTRL + 9
SHIFT + PF10	CTRL + 0
CTL + 2	CTRL + `
CTL + 6	CTRL + SHIFT + ~
↓	5(Numeric pad)
END	2(Numeric pad)

I/O MAP

I/O Addresses



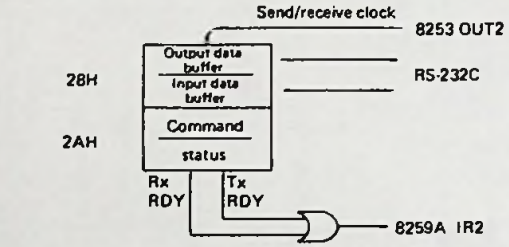
8255A PPI I/O Map

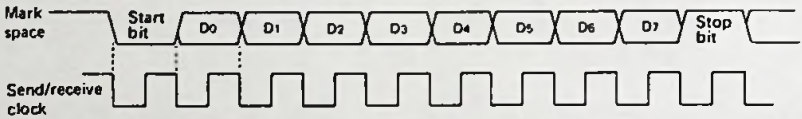


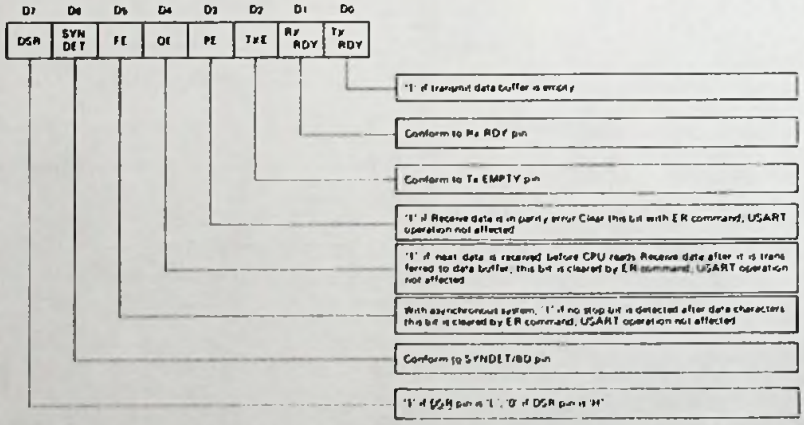
USART 8251A

8251A is a combination device operating in the double buffer mode. Input to output form the RS-232C interface board is performed by this USART.

The USART is intialized to one start bit, eight data bits, no parity, and one stop bit. It operates at a transmission rate of 1200 bits/sec (determined by timer counter 2) in the asynchronous mode.

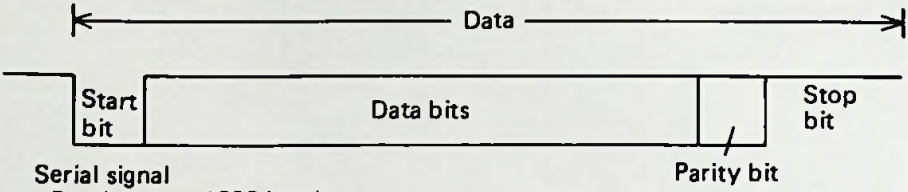






Keyboard Signals

The output signals from the keyboard have the following serial data train:

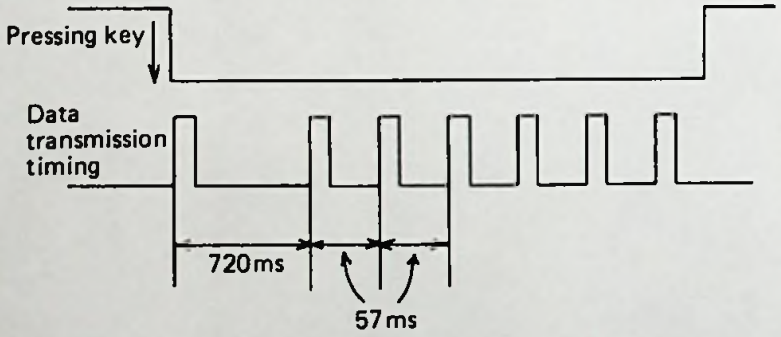


Serial signal
Baud rate: 1200 baud
Data bit: 8 bits
Parity Enable
Even parity
Stop bit: 2 bits

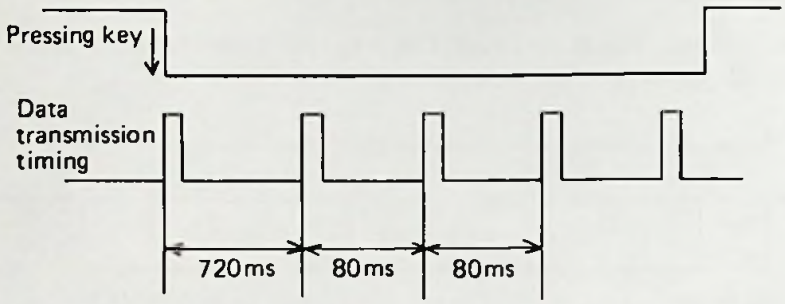
Key Operation

General Key

Depressing these keys causes data signals to be generated. Depressing the keys continuously for 720 msec or more results in auto repeat operation.

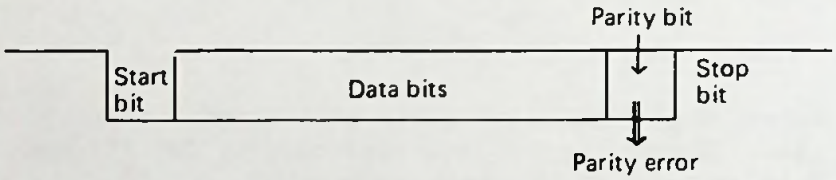


Cursor Key



CTRL Key(Key no. 31)

Depressing a general key together with this key causes transmission data parity bit to be sent as a parity error.



SHIFT Key

Depressing a general key together with this key causes the shift code for the mode (alphanumeric mode) to be sent.

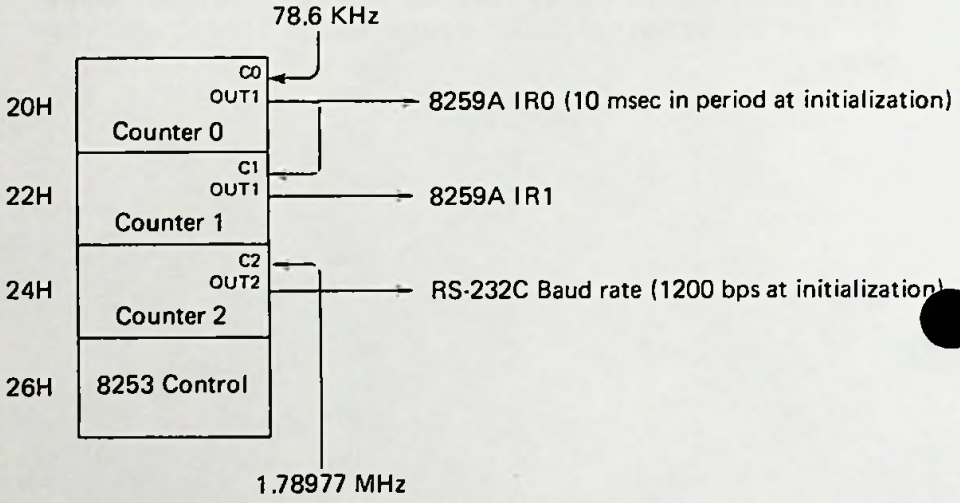
LOCK Key

This key is used to switch to lower case code mode. The led illuminates when operated in the upper case mode.

Timer 8253

8253 has three channels for timer counters. This equipment uses three channels as the interval timer, tone oscillator, and baud rate generator respectively. Its allocation is as follows:

I/O Address



Baud Bate Establishing

Baud rate can be changed by writing a new point into counter 2.

Example of baud rate changing program
(4800 baud)

```
BPS EQU 4800: Baud rate       ¦10 BPS = 4800:REM BAUD RATE
RSBR EQU 24H: COUNTER 2       ¦20 RSBR=&H24:REM COUNTER 2
RATE EQU 59658/((BPS/30)*16)¦30 RATE=59658/((BPS/30)*16)
                              ¦40 AL = RATE AND 255
     MOV AX, RATE             ¦50 AH = (RATE/256) AND 255
     OUT RSBR, AL :LSB        ¦60 OUT RSBR, AL:REM LSB
     MOV AL,AH                ¦70 AL =AH
     OUT RSBR AL  :MSB        ¦80 OUT RSBR,AL:REM MSB
       Assembler              ¦      Basic
```

NOTE: Sanyo Basic is able to handle baud rates up to 1200. Rates higher than 1200 must be accessed through assembler.

Default upon cold boot is 1200.

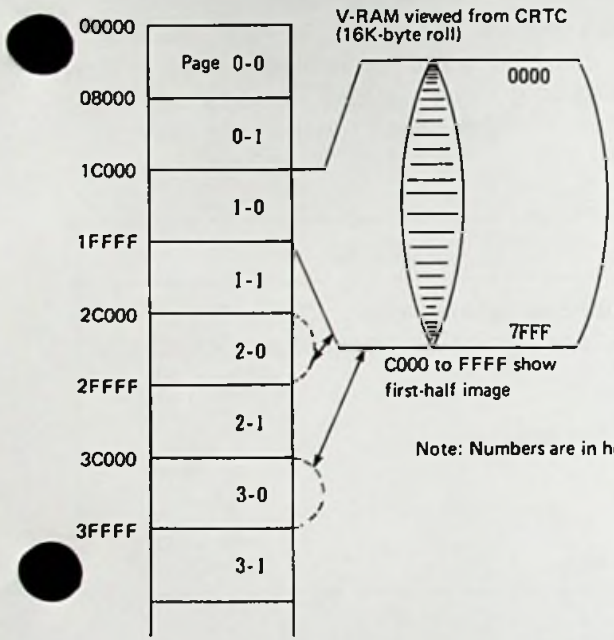
Video RAM and HD46505 CRTC

The MBC 550 series computer allows any section of the 256K byte area to be converted into Video RAM. Allocation is effected by the following method:

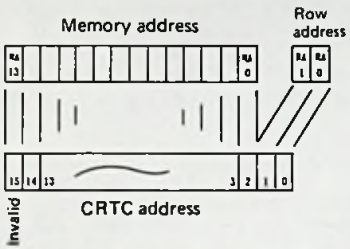
(1) Select a 16K byte page (usually page 1) using the page selector (at I/O address X'10').

(2) Determine a display address by CRTC.

Relations between CPU addresses and pages:



CRTC address composition



Note: The start address has the meaning of a 4-byte step on the V-RAM because the two low-order bits refer to the row address.

Note: Numbers are in hexadecimal.

DTS-4 DIP SWITCH

The DTS-4 Dip switch, located at the rear of the main memory board, is factory shipped with switches 2 and 3 on. This is the setting for most software applications. Please refer to the chart below for the functions of each switch.

Switch #	Function
1	Monochrome mixed mode. Three levels of green for graphic programs. (used for test)
2	Half intensity.
3	Blink.
4	Not used.

Normal setting is as follows:

Switch #	Setting
1	OFF
2	ON
3	ON
4	OFF

DOS EDITING KEYS

The DOS editing keys are employed to make corrections of commands and input lines when LDLINE(line editor) program is used. The following are outline of DOS editing keys.

PF1	Copies one character from the template and displays it.
PF2	Copies characters up to a designated character(the next character you want to be copied) from the template to the screen.
PF3	Copies all remaining characters from the template to the screen.
PF4	Skips over all characters to a designated character.
PF5	Initialize the template.
PF6	Kills the new template(no change in old template).
PF7	Enters the template insertion mode.
PF8	Resets the template insertion mode.
	Skips over one character in the template.
	Backspace and deletion.

CHAPTER 6
OPTIONAL PERIPHERAL INSTALLATION

INSTALLATION INSTRUCTIONS

The installation instructions contained in this guide allow you to install Sanyo peripherals on your MBC 550 series computer. Sanyo recommends, however, that you let a qualified technician do the installation. Please consult your authorized Sanyo computer dealer.

Sanyo Business Systems assumes no responsibility for all improper installations and their consequences due to misinterpretations of the installation directions contained in this chapter.

These installation instructions apply only to Sanyo supplied peripherals or recommended compatible components mentioned in this documentation.

CABINET COVER REMOVAL

In order to install Sanyo peripherals you must remove the cabinet cover and in most cases the back panel. The instructions below will guide you in accomplishing this. Please read them carefully and refer to them for all peripheral installations.

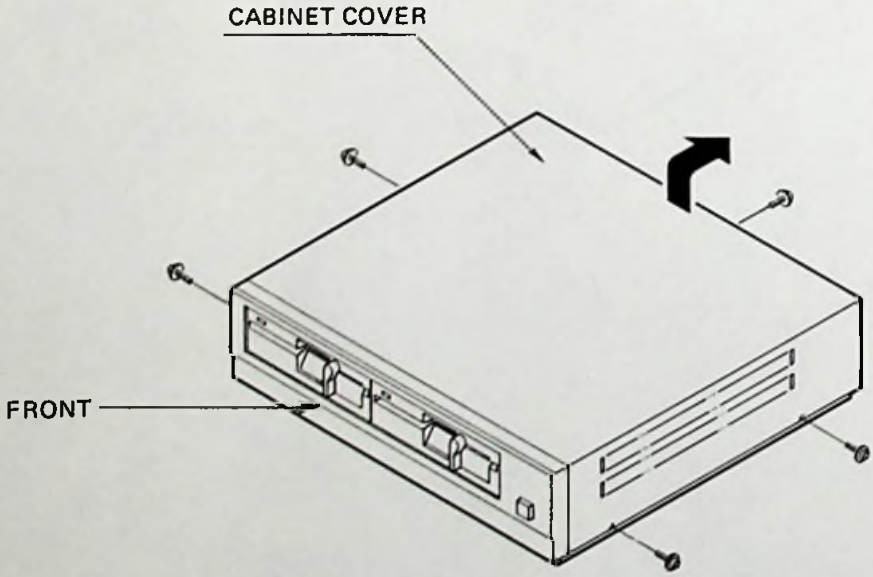
1) **Turn the power off and unplug the AC cord from the electrical outlet.**

2) Unscrew (counter clockwise) the case screws on the sides of the computer (two on each side).

3) Unscrew the top center screw in the rear of the computer (one screw).

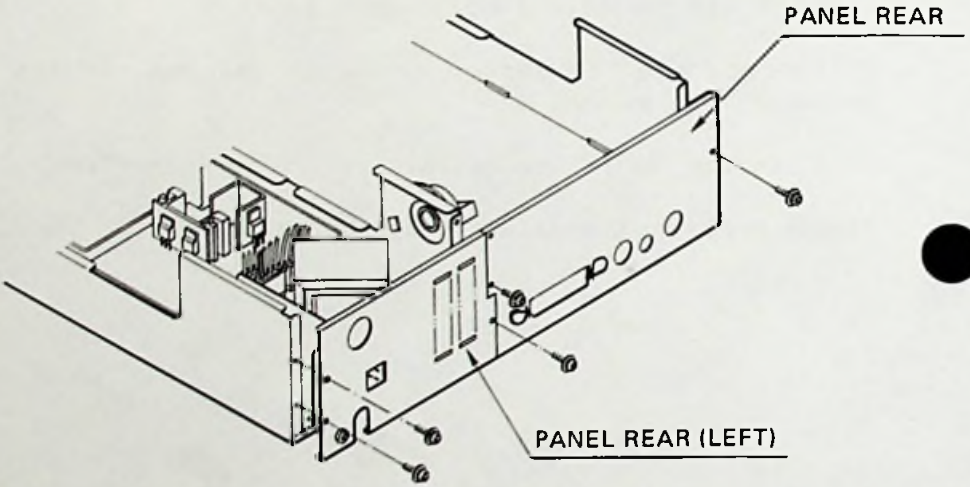
4) Slide the top of the cabinet away from the drives.

Please refer to the illustration below.

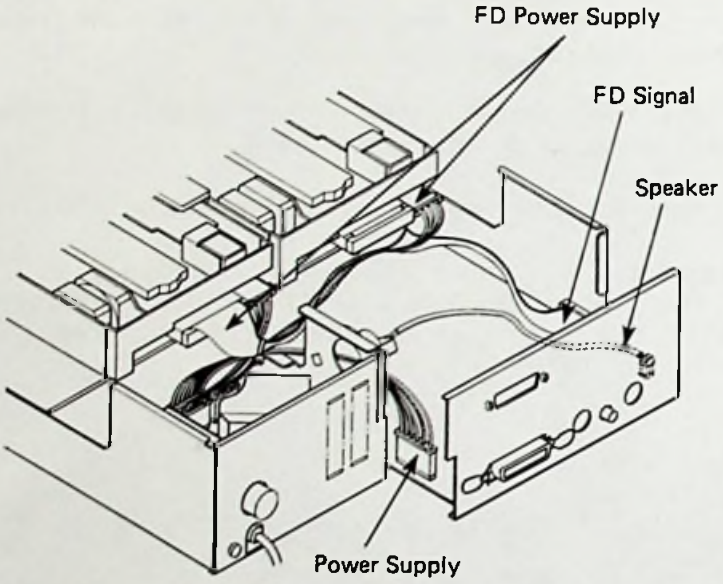


REAR PANEL REMOVAL

1) Unscrew the screws holding the left panel (2 screws). Please refer to the illustration below.



2) Disconnect the two plugs and speaker plug on the main board. Please refer to the illustration below.



SECOND DRIVE FDD 1655

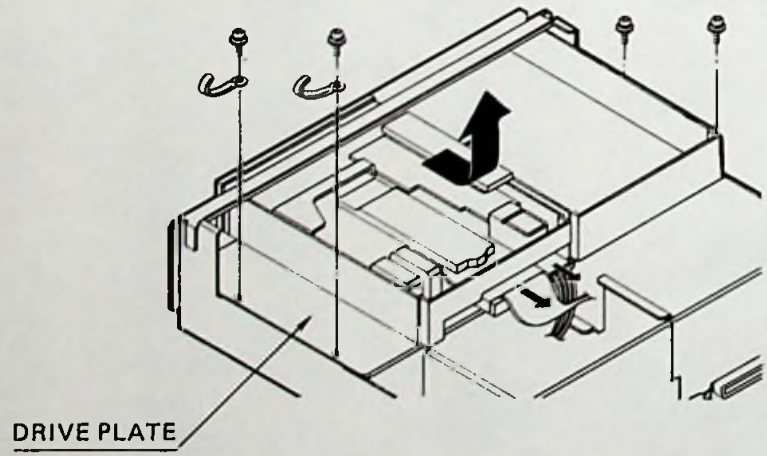
To install any configured second drive, follow the instructions below.

1) **Turn the power off andy unplug the AC cord from the electrical outlet.**

2) Remove the cabinet cover. Refer to the instructions at the beginning of this chapter.

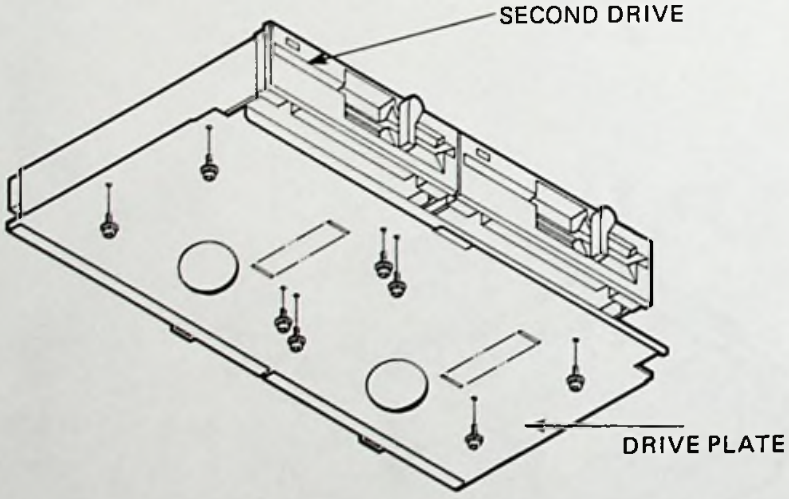
3) Disconnect the two plugs in the back of the drive.

4) Remove the screws holding the drive plate, and slide the plate,drive and diskette holder away from the front of the computer and up. Please refer to the illustration below.



5) Place the second drive next to the existing drive. Compare the two; they both should look the same. If they do not, turn the second drive over. The bottom of the second drive has a circular drive motor; it looks like a turn table on a phonograph. This side should be facing down.

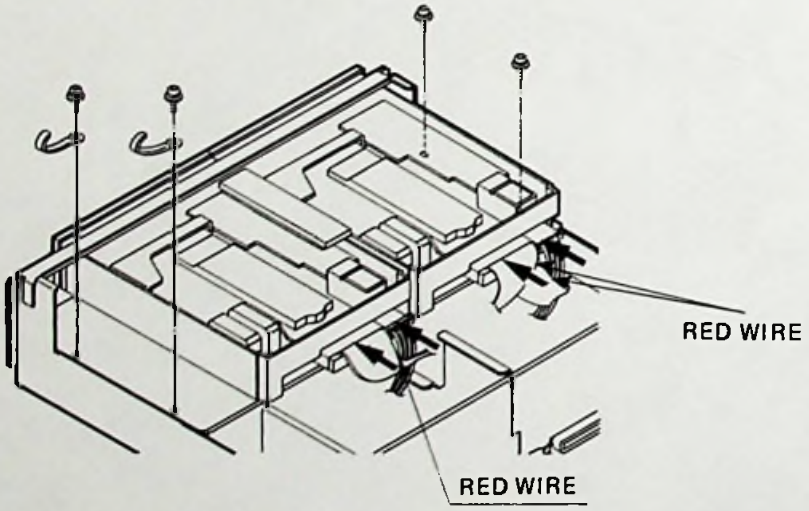
6) Carefully turn the drive plate, with both drives on it, over. Align the second drive to the screw holes, and screw the plate onto the drive.



7) Replace the drive plate and the two drives back into the cabinet.

8) Align the screw holes and tighten.

9) Replace the plugs for the initial drive and insert plugs for the second drive. The signal line plug (large black plug) has one red wire. This should be at the right side of the plug as you insert it. Please refer to the illustration. The power plugs (4 wires only - 1 yellow, 2 black, and 1 red) can only be inserted the correct way. If you have trouble pushing the plug in, turn it over and try again. Again, the red wire is to the right.



10) Check to make sure you did not loosen anything while you were installing the drive.

11) Replace the cabinet cover.

12) Restore power to your unit.

MEMORY EXPANSION

MBC 64K

First 64K Upgrade = 192K Total

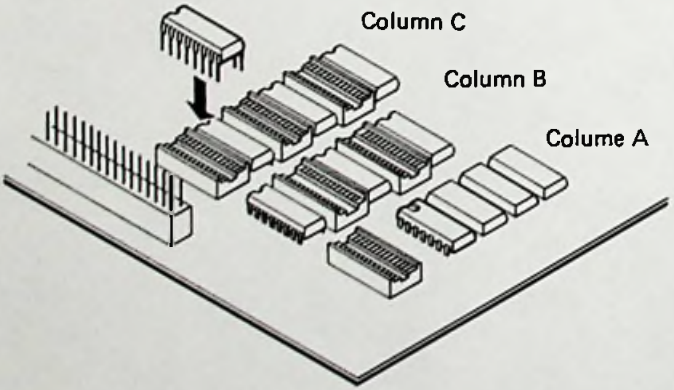
1) **Turn the power off and unplug the AC cord from the electrical outlet.**

2) Remove the case top. Refer to the instructions at the beginning of this chapter.

3) Remove the back panel. Refer to the instructions at the beginning of this chapter.

4) Slide the board back, away from the drives by pulling gently on the panel.

5) Locate column C toward the end of the board. Please refer to the illustration.



6) The eight memory chips will fit into the sockets in column C. The order that you place the chips does not matter. How you insert the chips does. Look at a chip. One end is flat, one end has a half circle cut in it. This half circle should be facing toward the panel. All lettering on these chips and the ones in column C should be reading in the same direction.

7) Place all the pins of the chip on one side of the socket row and gently push in as far as possible. Push the other side down till you hear a click. Be **careful not to break the pins. If you have trouble, please consult an authorized Sanyo computer dealer.**

8) After you have placed all eight chips in column C, replace the board and connect the plugs.

9) Check to make sure everything is secure before you replace the cabinet cover.

10) Restore power to your unit.

Second 64K Upgrade = 256K Total

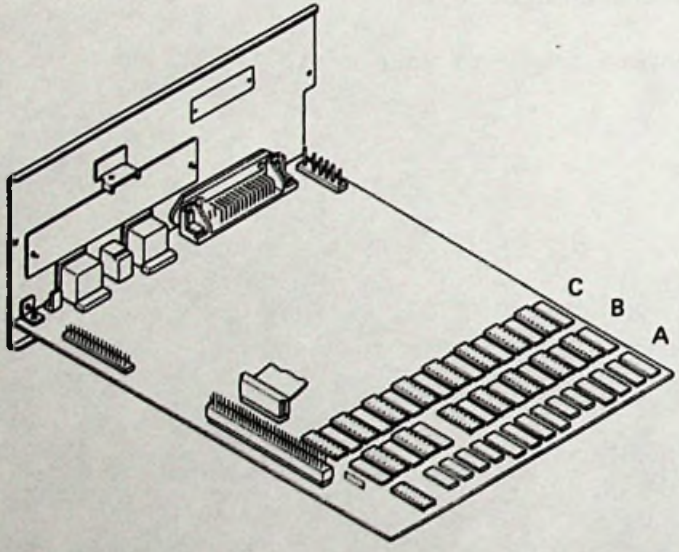
1) Turn the power off and unplug the AC cord from the electrical outlet.

2) Remove the case top. Refer to the instructions at the beginning of this chapter.

3) Remove the back panel. Refer to the instructions at the beginning of this chapter.

4) Slide the board back, away from the drives by pulling gently on the panel.

5) Locate column **B** and **A** toward the end of the board. Please refer to the illustration.



6) Seven memory chips will fit into the sockets in column **B**. One memory chip must be place in the last socket in column **A**. The order that you place the chips does not matter. How you insert the chips does. Look at a chip. One end is flat, one end has a half circle cut in it. This half circle should be facing toward the panel. All lettering on these chips and the ones in column B and A should be reading in the same direction.

7) Place all the pins of the chip on one side of the socket and gently push in as far as possible. Push the other side down till you hear a click. **Be careful not to break the pins. If you have trouble, please consult an authorized Sanyo computer dealer.**

8) After you have placed seven chip in column B and one chip in the last socket of column A, replace the board and connect the plugs.

9) Check to make sure everything is secure before you replace the cabinet cover.

10) Restore power to your unit.

SERIAL PORT-RS232C

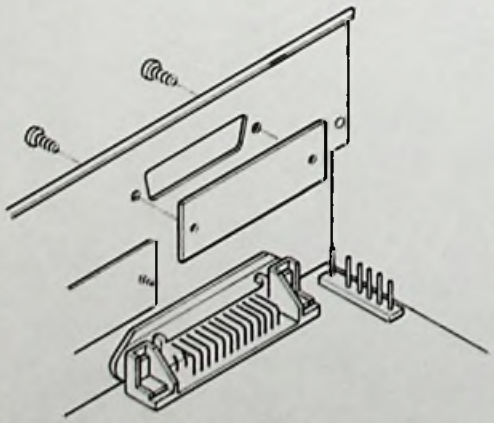
MBC 232C

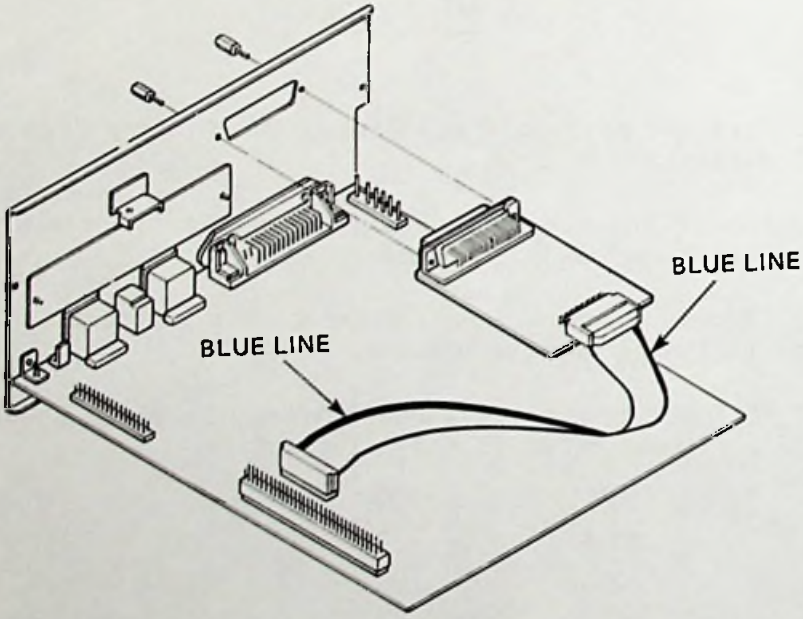
1) **Turn the power off and unplug the AC cord from the electrical outlet.**

2) Remove the case top. Refer to the instructions at the beginning of this chapter.

3) Remove the back panel. Refer to the instructions at the beginning of this chapter.

4) Disconnect the two screw holding the plate labeled LINE. Please refer to the illustration.




BLUE LINE
BLUE LINE

5) Holding the board with **SANYO** on the left, insert the board into the cutout, from behind the panel. Screw board onto the panel.

6) Attach any end of the cable to the end of the RS 232 board (pins are labeled CN-1). The number 35 should be visible and the blue edge of the cable will be on the side of the board labeled SANYO.

7) Attach the other end of the cable to the blue pins labeled CN-6 on the main board. These pins are located at the end of column F. The blue edge of the cable will be facing the rear panel and located on the side of the pins labeled 0 and 1.

8) Insert the side of the plug with the blue line first. Gently press down as far as possible. Press the other side onto the pins.

9) Check to make sure everything is secure before you replace the back panel and cabinet cover.

10) Restore power to your unit.

Jumper

There is a green plastic jumper on the RS 232 board. This jumper governs receive and transmit signals.

Please leave the jumper the way it was shipped to you.When the jumper is placed over pins 1 and 2 as it is now, you may send and receive. If it is placed over pins 1 and 4, you may receive only. Any other pin configuration will result in your RS 232 port not operating.

Setting the Baud Rate

Baud rate can be changed by writing a new point into counter 2.

Example of Baud Rate Changing Program
(4800 baud)

```
BPS EQU 4800: Baud rate     |10 BPS = 4800:REM BAUD
                             RATE
RSBR EQU 24H: COUNTER 2     |20 RSBR=&H24:REM COUNTER 2
RATE EQU 59658/((BPS/30)*16)|30 RATE=
                             59658/((BPS/30)*16)
                            |40 AL = RATE AND 255
     MOV AX, RATE           |50 AH = (RATE/256) AND 255
     OUT RSBR, AL :LSB      |60 OUT RSBR, AL:REM LSB
     MOV AL,AH              |70 AL =AH
     OUT RSBR AL  :MSB      |80 OUT RSBR,AL:REM MSB
       Assembler            |      Basic
```

To set any other baud rate, subsitute the rate in place of 4800 in the examples above.

The default rate upon cold boot is 1200. Other baud rates must be set upon start up.

Sanyo Basic may handle baud rates up to 1200. For baud rates over 1200, you must set them through assembler.

JOY STICK

The MBC Series computer are Apple(r) Joy stick compatible.

1) **Turn the power off and unplug the AC cord from the electrical outlet.**

2) Remove the case top. Refer to the instructions at the beginning of this chapter.

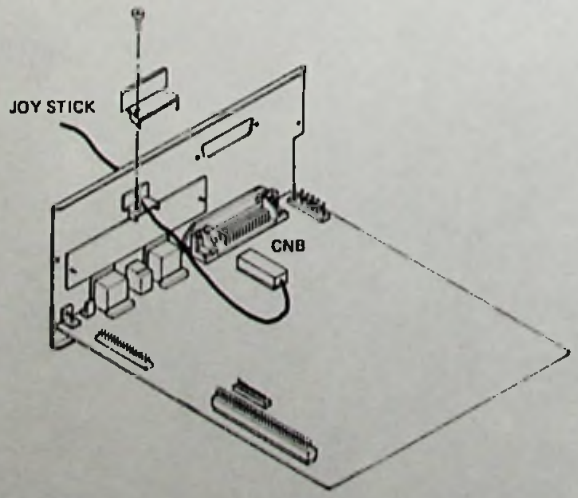
3) Remove the plate labeled JOY STICK/PADDLE

by removing the single screw inside the rear panel.

4) Thread the joy stick or paddle cable through the hole and attach to the socket labeled **CN8.** This is located about the middle of column H.

5) You will notice that one edge of the joy stick/paddle plug is rounded. This edge will be inserted at the corner of the socket labeled CN8.

6) Press the plug into the socket gently, but firmly.



7) Check to make sure everything is secure before you replace the cabinet cover.

8) Restore power to your unit.

MONITOR CRT-36

The Sanyo CRT-36 monitor may be placed on the top of the computer cabinet or in a convenient location nearby.

1) Connect the video cord to the monitor in the rear socket labeled VIDEO OUTPUT.

2) Connect the other end to the rear of the Sanyo 550 series computer in the socket labeled MONO CHROME.

3) Connect the power cord to the power outlet of the computer located in the rear of the cabinet. The power switch is located below the screen.

NOTE: FOR OTHER MONITORS - SOME MONITORS PROVIDE POOR EMI PERFORMANCE. TO ELIMINBATE THE NOISE PROBLEM, CONNECT THE MONITOR POWER CORD TO THE MBC 550 SERIES COMPUTER POWER OUTLET LOCATED IN THE REAR OF THE COMPUTER.

ADDITIONAL SOFTWARE AND MANUALS

Your authorized Sanyo Computer dealer has a wide range of software packages available for your MBC 550 series computer.

In addition, these reference manuals are also available from your Sanyo computer dealer for detailed information you might require:

Sanyo Software Reference Manuals

MS-DOS Reference Manual by Microsoft

Sanyo Basic Reference Manual

WordStar Reference Manual by MicroPro

SpellStar and MailMerge Reference Manual by MicroPro

ReportStar Reference Manual by MicroPro

DataStar Reference Manual by MicroPro

INDEX

System commands and Sanyo Basic commands and functions are listed first in alphabetical order for each section.

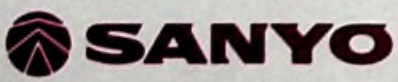
D

E

F

G

S



SANYO ELECTRIC CO., LTD
Printed In Japan Nov. 1983 Rev. 1.0
9376411914800