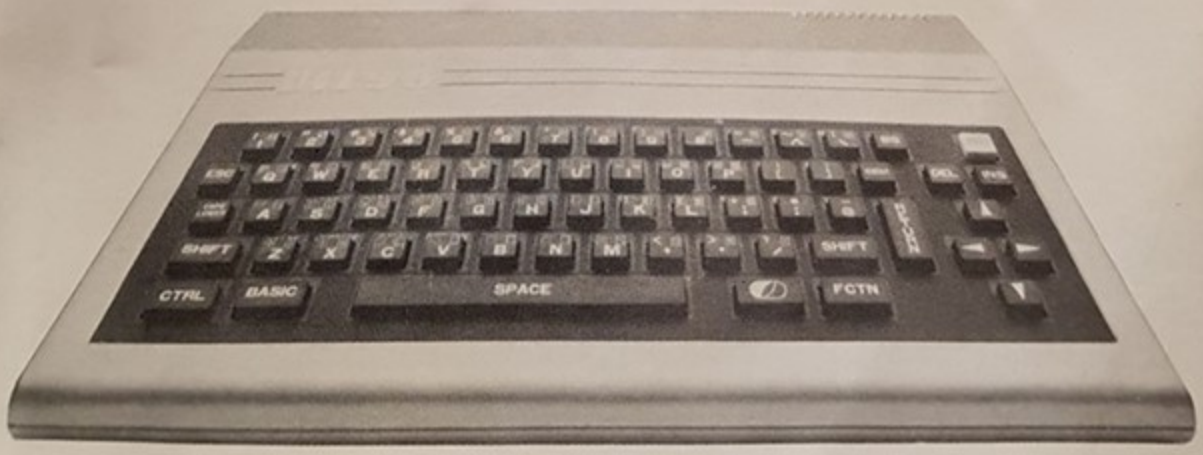
OPERATION MANUAL

Before operating this set, please read these instructions completely.

BIT90

HOME COMPUTER



BIT CORPORATION

Oct. 1. 1983

TABLE OF CONTENTS

APPENDIX

CHAPTER 0
INTRODUCTION

Congratulations ! Now you have a fantastic and powerful computer of your own. This manual will guide you to use BIT90 computer and program BASIC language. No matter you have never worked with a computer before or you have been familiar with the BASIC language, this manual would help you to make the best use of BIT90.

This manual is arranged in the sequence of following steps:

STEP 1 : How to power on your BIT90, and some points to notice when BIT90 is power on.

STEP 2 : Understanding your keyboard

Before you enter the BASIC of BIT90, you must have studied the keyboard, especially the [FCTN], [BASIC], [◐] key ..... etc.

STEP 3 : How to use one-key BASIC

Besides the BASIC function of general purpose, BIT90 also has a special function, one-key BASIC, for beginners to start with.

STEP 4 : Introduction to BIT90 commands and statements

If you are a beginner, you may start with some simple BASIC programs to enter the world of computer. If you have years of experiences in using computer, here are some useful and powerful commands, especially those for graphics and sound effect, can be applied specifically to BIT90.

STEP 5 : How to program your BASIC language

In this section, you may try to talk to your computer and enjoy it with some interactive programs. For further information about BASIC programming and others, please refer to "BIT90 BASIC MANUAL".

STEP 6 : How to use cassette recorder

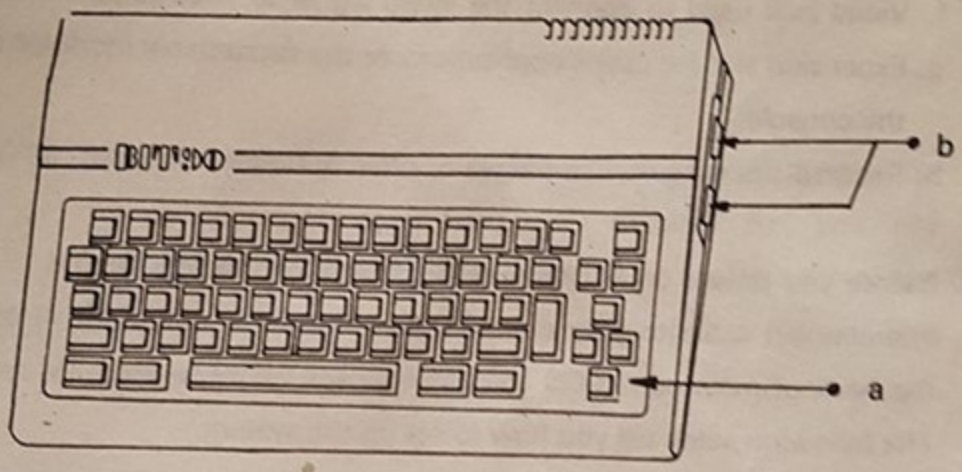
BIT90 has a high speed cassette interface for data and program which can be transfered between your recorder and computer.

This's all for the outline. Now, let's waste on more words and get starting in chapter one .....

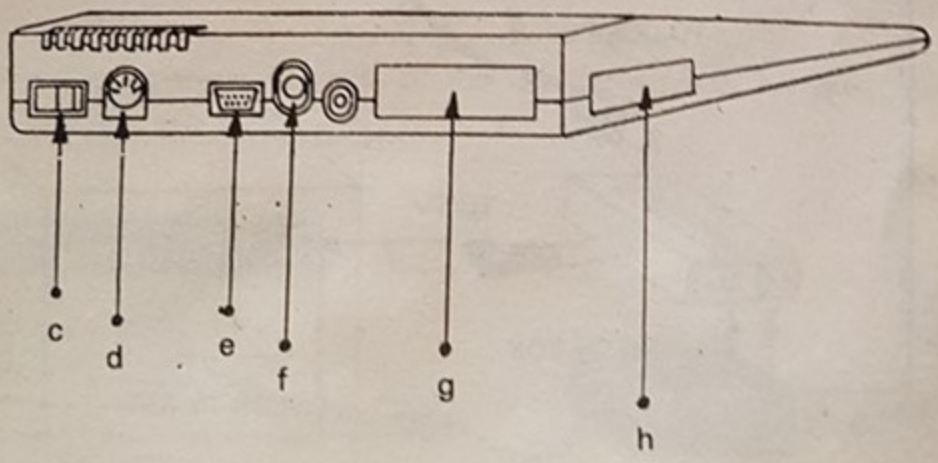
CHAPTER 1
HOW TO POWER ON YOUR BIT90 COMPUTER

1. An overview of your computer

Your computer console is the central part of your computer system so that all of the other units can be easily connected to it. No other tools are required. Let's take a look at the front and right side of your BIT90.



Following is the back and left side of the console.



a. 66-key keyboard for user to type information into the computer.

b. Joystick jacks used to play video games or other application of user's design.

c. Power on/off switch to turn on or turn off the computer.

d. DIN jack which you attach the power cord to the console.

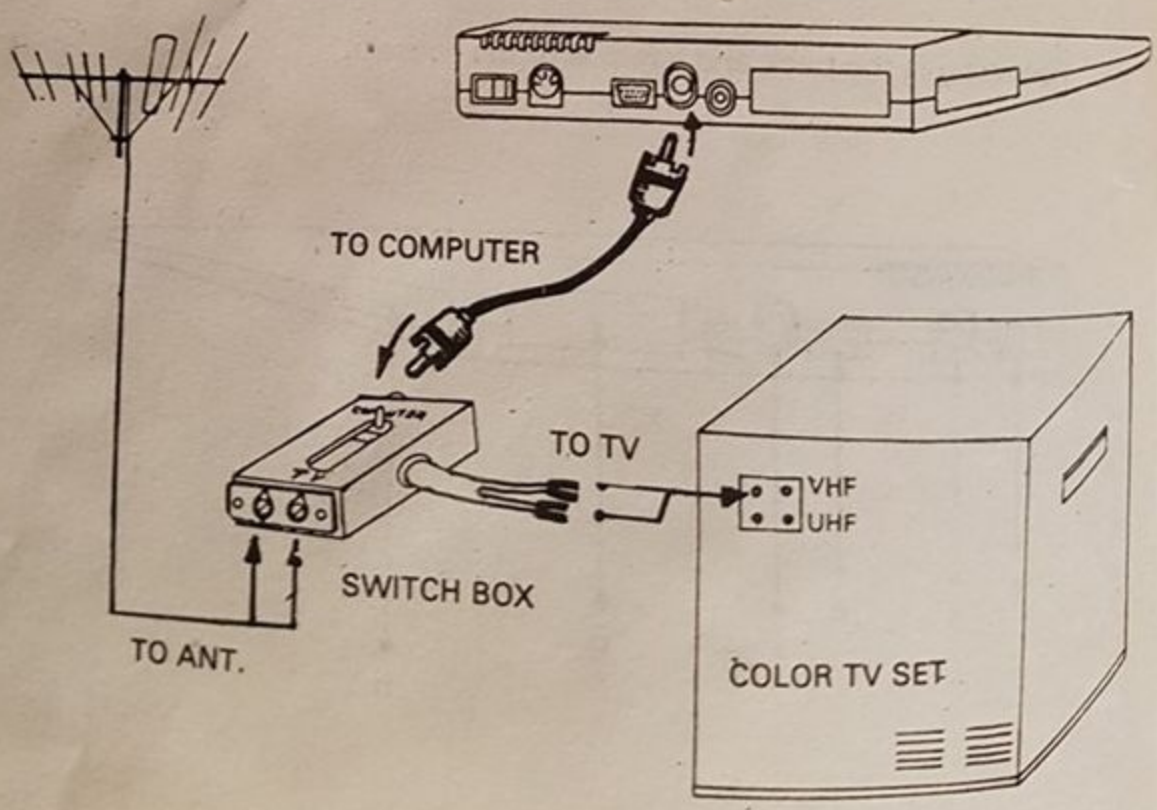
e. 9-pin jack, pin-3 and pin-5 are the tape load and tape dump respectively, pin-6 is the video output signal used to connect the RGB monitor, pin-8 is the audio sound output.

f. Video jack used to connect the video signal to your color TV set.

g. Expansion slot for user's applications or the expandable modules of the console.

h. Reserve slot for game cartridges or other software modules.

2. Before you power on BIT90, you must have connected the TV interconnect cable to your color TV set. The video outlet signal on the back of main console is used to connect BIT90 to your TV set. The following steps tell you how to set up the system.



1) Remove the VHF antenna cable from your TV set.
2) Connect the TV interconnect cable from "Video" output outlet on the back of BIT90 to the switch box. Then, connect the VHF antenna cable of the box to the VHF antenna terminals on your TV set.
3) Whenever you use your TV set and BIT90 together, set the "TV/COMPUTER" switch on switch box to "COMPUTER".
4) Make sure both the computer and your TV set are plugged into a live wall outlet.
5) Check to see that all connections are secure.

NOTE: Your BIT90 computer also supplies RGB video output signal for connecting to your RGB color monitor. So, you may connect the cable from the 9-pin socket on the rear of the BIT90 to the RGB color monitor instead of color TV set. This cable is option supported by BIT CO.

3. Connect Power Cords

Next, connect the DC power cord (with transformer) to the computer. connect the small 5-pin plug end into DIN JACK on the back of the computer as shown below: (**NOTICE**: the pins only line up one way)

Then, plug the other side of the transformer into a regular wall outlet as in the sketch below:

After connecting the TV cable and power cord to the computer, just turn on the power switch on the back of BIT90. Your TV screen will display the color title "BIT90", and just a moment, it will display "BIT90 BASIC 3.0" and prompt ">". Yet, if you have plugged a game cartridge into the slot in the left side of the console, the computer will enter the game mode for entertainments.

Having entered the BASIC mode, your TV screen displays as follow with a sound "BEEP":

BIT90 BASIC 3.0

READY

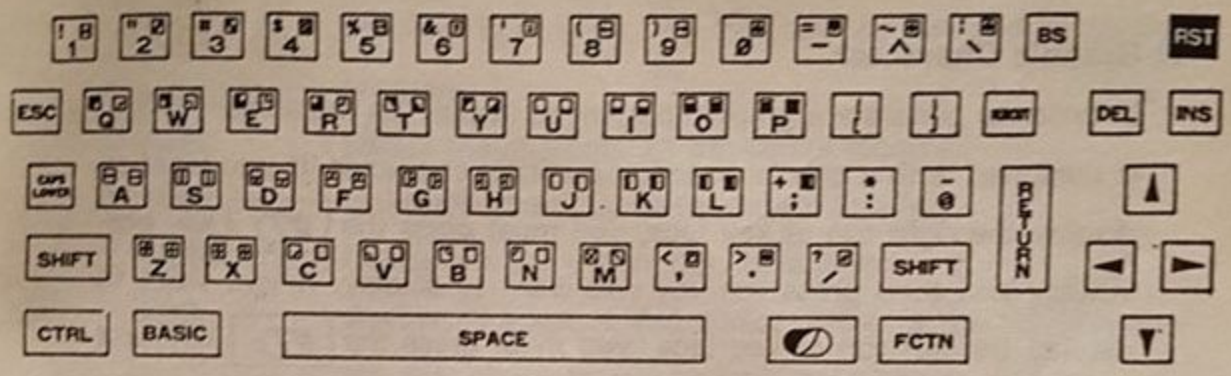
>

Memory size may be tested by "FRE" command. But if you have expanded the memory to 32 K, it will show you "32760". Until you have typed into computer a BASIC program, there will be a red square flashing on the screen. If this message doesn't appear, check connections of the devices again.

NOTICE: Do not quickly repeat the ON/OFF operation of BIT90 or connected devices. It may cause trouble or malfunction. When you erroneously turn off the power switch of BIT90, programs stored in the main memory of the BIT90 will be lost. Important programs must be recorded and stored on a tape before turning the computer off.

CHAPTER 2
UNDERSTANDING YOUR KEYBOARD

An overview of the keyboard



This keyboard of BIT90 computer is like a standard typewriter with keys of several types. Pressing any alphabetical key causes its upper-case (large capital) character to display on the screen. After having pressed [CAPS LOWER] key then pressed any alphabetical key, you would get the lower-case (small capital) character. If you have held down the [SHIFT] key the [CAPS LOWER] key then pressing any alphabetical key, on the screen there will be the upper-case (large capital) character for that key.

Except for the alphabetical keys, each key's upper-case character is printed at the top of the face, while the lower-case character is printed at the bottom. Whenever you type the [SHIFT] key and any key except for the alphabetical keys, the upper-case character will be displayed on the screen, otherwise, the low case character will be.

Some of the keys also have special functions as explained the following sections.

1) Automatic Repeat

BIT90 is designed with an automatic repeat function. If you held down any key for more than one second, that character is repeated until you release the key.

2) Graphic Character

In order to generate a graphic pattern to display on the screen, BIT90 is designed with a special function-graphics character. If you want to display the right top of key face, you must press the [◐] key previously and then press the key you want to display. If you want to display the left top of key face, you must press the [◐] key first and then press the [SHIFT] and the key you want to display simultaneously. For example, when you want to display the graphic character ◥, you must type the [◐] key first, then press the [SHIFT] and [◥T◤] at the same time. If you want to display the ◤, you just press the [◐] key then the [◥T◤]. In BIT90 computer, there are 69 graphic characters can be used.

3) One-Key BASIC

For the purpose of using BASIC commands or statements easily, BIT90 computer provides a way to either the beginner or the BASIC hobbist to program your BASIC language quickly. To enter a token of BASIC, hold down the [BASIC] key and press the appropriate command or statement key of BIT90 BASIC. For example, when you want to generate a "THEN", you only hold down the [BASIC] key and press the [◥T◤].

4) Special control keys

Several keys founction variously in BIT90 BASIC, such as RST, ESC, DEL, INS, RUBOUT, BS, ...etc.

- •RST : When power on, the system enters a cold start; press the RST key, the system enters a worm start.
- •ESC : In BIT90 BASIC EDIT mode, the ESC key is equal to RETURN. While listing the BASIC statements, press the ESC key should stop the list command. If the BASIC program is running, the ESC can be used to break the program.
- •DEL : Delete the character of the cursor locates.
- •INS : Insert the character of the cursor locates.
- •REBOUT : Delete one line buffer 128 characters.
- •BS : Backward a space and cursor shifts to the left character, that character will be replaced a space.
- • ▲ ◄ ► ▼ : These keys are used to move the cursor one space in the direction shown by arrow to correct the sentences. Correction on the screen is performed bv utilizing the DEL, INS, RUBOUT, keys.
- •RETURN : Tell the computer user has entered a command to process.

  When the RETURN key is pressed after correction on the screen, the corresponding program within the memory is corrected in the same manner as displayed on the screen. The RETURN key may be pressed at any time regardless of the cursor position on a given line.

5) Function keys

BIT90 computer has reserved 10 function for user's definition. The FCTN key is entrance point. You can hold down the FCTN key and press the number keys (0 — 9), thus, your computer will respond the

number code (224 — 233) to your for executing your special function program. In BASIC program, you may use "INPUT" command to read a code to determine which program will be executed by your computer.

6) Math or Operation keys

The math keys (or operation keys) are the keys used to instruct the computer to add, subtract, multiply, divide and raise a number to a power. The symbols for addition, subtraction and equals are the usual ones you're familiar with, but the multiplication and division symbols may be new to you.

+	ADDITION
—	SUBTRACTION
*	MULTIPLICATION
/	DIVISION
=	EQUALS

The "caret" key (∧) is also used for mathematical operations. This symbol tells the computer to perform exponentiation (raise a number to a power). Since 3^5 cannot be easily printed on your screen, the computer interprets 3∧5 to mean that five is an exponent.

7) Space Bar

The space bar is the long bar at the bottom of the keyboard. It operates just like the space bar of any common typewriter. When you press the bar, the computer leaves a blank space between words, letters or numbers. Besides, the space bar can also be used to erase character already on the screen. In general, a blank space can occur almost anywhere in a program without affecting the execution of the program.

8) [SHIFT] [? / ⊠]

When the "LIST" command is executed, the screen will display user's program line by line. To stop listing, you should hold down the [SHIFT] [? / ⊠] keys then release, and hold down the [SHIFT] [? / ⊠] keys again to go on listing.

9) PLAY GAME

Instead of using joystick with keypad, you may play the cartridge's game just use the keyboard only, these keys 1-8 number keys and "#", "*" keys on the keypad can be found on the keyboard of BIT90. The direction up, down, left, right can be controlled by [▲] [▼] [◄] [►] keys. The FIRE and ARM keys can be replaced by [CTRL] and [BASIC] respectively.

CHAPTER 3
INTRODUCTION TO BIT90 BASIC COMMANDS AND STATEMENTS

1) What's the BASIC language?

BASIC is "Beginner's All-purpose Symbolic Instruction Code", in short, a high level language used in computational problems and is especially suitable for the beginners.

Every computer language has its own grammar, we must obey the rules to communicate with the computer. This manual describes the rules of BIT90 computer.

BASIC language can be devided into statements and commands. A statement is composed of executable-statement and non-executable-statement.

2) Feature of BIT90 BASIC

1. In BIT90 BASIC mode, user can type a program with multi-statements or a very long statement but not exceed 128 characters in a line buffer.
2. Editing facilities:

   Use the "EDIT" command to delete, replace the statement input, insert a new statement or replace some characters of a statement. BIT90 computer has the ability of line-edit to move the cursor four directions (up, down, left, right) in a line buffer (128 characters) freely.
3. Immediate execution mode:

   You may type a command, the computer can execute it immediately. You need not type a complete program, even the line number is not necessary.

4. Non-executable-statement can be located anywhere without being restricted.
5. Graphic mode:

   User can define the graphic mode by himself and use the plot command to control the screen effect.
6. Hard Copy:

   BIT90 BASIC has a command "COPY" to record the characters of screen.
7. Line TAB controllable:

   In a "PRINT" statement variable is separated by ",", the print location of data can be programable.
8. Save and load program:

   User can save and load his own program between the computer and cassette tape recorder very easily.
9. The line number can be generated automatically by the command "AUTO" and adjusted by "RENUM", "GOTO" and "GOSUB" will be adjusted as well.
10. Append Program:

    Load a program from cassette recorder can be appended to the other program completely. With the command "RENUM", we can accomplish a new program or with "NEW" to clear old program.

3) BIT90 BASIC COMMANDS DESCRIPTION

1. COMMAND

   DEL [line_number] [—] [line_number 2]

   The program lines specified are erased from memory.

   e.g. > 10 PRINT "ABC"

   > 20 PRINT "DELETE EVERYTHING"

   > DEL

   SYNTAX ERROR

   > DEL 10—

```
READY
> LIST
READY
>
```

EDIT line_number

Displays a line for editing.

2. COMMAND AND STATEMENT

AUTO [initial_line] [,increment]

Automatically generates sequenced line numbers starting at 10 in increments of 10. Optionally, and initial line and/or increment may be specified. The linenumber is limited in range 1 to 9999, if the user try to type a line_number out of this range, BL ERROR will be displayed. If the line_number is generated by AUTO command, it will return to BASIC command level.

```
e.g. > AUTO 9990
     > 9990 PRINT "HEY!"
     > (no line number generated)
```

BYE

Closes open files and leaves BASIC mode.

CONT

Resumes execution after breakpoint has been encountered.
The message "CAN'T CONTINUE" will be displayed on the screen if enter a CONT command after the program has been edited.

```
>10 A = 10
>20 B= A/5
>30 STOP
>40 PRINT B/A
>RUN
 BREAK IN 30
```

```
>PRINT B
2
>CONT
.2
READY
>
```

CALL CHAR (char-code, pattern__identifier)

Redefines specified char__code using 16 character HEX coded string. The char__code is in the range 0 to 255, if the hexadecimal string less than 16 characters, computer assumes to be '0' for remains.

```
>CALL CHAR (65, "FFFFFFFFFFFFFFFF")
```

(pattern A is change to be a block)

CALL CHRCOL (character__set, foreground__color, background__color)

Specifies foreground and background colors of all characters in the specified set.

There are 32 character__sets in BIT90 computer. The color of each character__set may be defined individually. the character__set is ranged from 0 to 31, while foreground and background color codes from 0 to 15. If the range is exceeded but less than 65536, computer will round to be acceptable value. If the value specified greater than or equal to 65536, message 'BS ERROR' will be displayed on the screen.

character_set	character_code
0	0-7
1	8-15
2	16-23
3	24-31
4	32-39
5	40-47
6	48-55
7	56-63
8	64-71
9	72-79
10	80-87
11	88-95
11	96-103
13	104-111
14	112-119
15	120-127
16	128-135
17	136-143
18	144-151
19	152-159
20	160-167
21	168-175
22	176-183
23	184-191
24	192-199
25	200-207
27	216-223
28	224-231
29	232-239
30	240-247
31	248-255

color_code	color
0	transparent
1	black
2	medium greer
3	light green
4	dark blue
5	light blue
6	dark red
7	cyan
8	medium red
9	light red
10	dark yellow
11	light yellow
12	dark green
13	magenta
14	gray
15	white

e.g. >CALL CHRCOL (6, 9, 1)

(the color of character '0' to '7' are changed to be light red with black background.)

CALL HCHAR (row, column, character_code[,repetition])

Places ASCII character at specified row (0-23) and column (0-31), optionally repeats it horizontally.

e.g. >CALL HCHAR (0, 0, 65, 768)

(the screen is filled by "A" character)

CALL SCREEN (backplan_color[,screen mode])

Defines screen color or optionally screen mode. If mode 1 or 2 is selected, the screen is cleared first.

mode	display
0	Text mode (32x24 characters)
1	High resolution graphics (256x192 dots)
2	Low resolutions graphics (64x48 dots)

e.g. >CALL SCREEN (13, 0)

(defines screen mode to be normal text mode with backplan color magenta)

CALL SPRITE ([y__position],[x__position],[sprite pattern number], [color],sprite number)

Defines x__position (0-255),y__position(-31 to 207),color,and pattern__number (0-255) of specified sprite__number (0-31).

```
e.g. >10 CALL SPRPTN 1, "FFFFFFFFFFFFFFFF")
     >20 FOR I = TO 255
     >30 CALL SPRITE (100,I,1,2,1)
     >40 NEXT
```

(moves a blue block from left to right)

CALL SPRPTN (pattern__number, pattern__identifier)

Defines specified sprite pattern__number of sprite using 16 hexadecimal coded string.

```
e.g. >10 SP$ = "3C7EFFFFFFFF7E3C"
     >20 CALL SPRPTN (1,SP$)
     >30 CALL PRITE(10,10,1,3,1)
```

CALL numeric__expression

Calls Z80 assembly language subroutine with starting address evaluated by numberic__expression.

```
>NEW
>CALL A
```

(worm start)

CLEAR

Sets all numeric variables to zero and all string variables to null.

```
>DIM A(254)
OUT OF MEMORY
>CLEAR
 READY
>FRE
761
 READY
>
```

COPY [ON]

COPY OFF

Initializes and sets hard copy mode to copy the characters of sereen to printer. To use this statement, the printer interface (with CENTRONIC interface) must be connected. The number N of characters per line is controlled by memory address 29504. Command COPY [ON] is used to enable copy, while COPY OFF to disable.

```
>POKE 29504, N-1
>COPY ON
>LIST
```

(Turn on printer and set N characters per line prints).

DEF FN function_name[(parameter)] = expression

Associates user_defined numeric or string expression with FN function_name. This function_name must be a character followed by a numeric character.

DIM array_name ([integerl][,integer2][,integer3])

Dimensions the listed arrays as specified by integers.

This integer is default to be 10 for 1 dimension array if it's not specified.

[END]

Terminates program execution. This statement may be omitted if used as last statement in a program.

FOR control_variable = initial_value TO limit (STEP increment)

Repeats execution of statements between FOR and NEXT until the control_variable exceeds the limit, STEP default is one.

GOTO line_number

Unconditionally branches to specified line_number.

INPUT ["prompt",] variable_list

Suspends program execution until data is entered from the keyboard. The optional input_prompt indicates data to be entered. The input_prompt default is "?".

INPUT *print_list:variable_list

The same as above except that the print_list a PRINTed before accept data.

HOME

Clears the display screen and positions the cursor to the upper lefthand corner.

IF relational_expression|number_expression THEN statement(s)1 [ELSE statement(s)2]

Transfers control to statement(s)1 if the relational expression is true or the numeric expression is not equal to zero, control pass to the

next statement or optionally to statement(s)2 if the relational expression is false or the numeric expression is equal to zero.

[LET] numeric_variable = numeric_expression

[LET] string_variable = string_expression

Assigns the value of an expression to the specified variable.

```
e.g. >A$ = "HEY!"
     >VALUE = 3.58
     >NUMBER = "123"
     TM ERROR
```

LIST [[line_number][–][line_number2]]

Sequentially displays program statements and optionally a single line number or all lines between the specified line_numbers.

a ESC or "?" character may be pressed to stop or pause listing.

```
>LIST
   10 TEMPO 5
   20 MUSIC 0,–15, "CDEFGAB+C"
   30 PLAY
   40 FOR I=0 TO 300
   50 NEXT
  READY
>LIST 30–
   30 PLAY
   40 FOR I=0 TO 300
   50 NEXT
  READY
```

LOAD ["filename"]

Loads a BASIC program from cassette tape into the computer's

memory. Filename is a string expression with length from 1 to 15 characters. If the filename is omitted, computer loads the first file with type '.B' from current tape position.

BLOAD ["filename"]

Loads a memory image file from cassette into memory. Filename is a string expression with length from 1 to 15 characters. The BLOADed file may be binary data or machine language which is BSAVEd previously. If the filename is omitted, computer loads the first file with type '.M' from current tape position. The filenames searched by computer are displayed on the screen followed by a period (.) and a single letter. These informations are described as follows:

.B for BASIC program in compressed binary format. (created by SAVE command)

.M for memory image files (created by BSAVE command)

```
>BLOAD "BABY"
READY ? Y
* TYPE LOAD:
BABY.B
BABY.M
* END *
READY
>
```

NEXT [control_variable]

Refer to FOR statement.

MUSIC channel_number, volume(,*),string1(,string2,.....)

Prepares a score for PLAY command to start music playing. Channel may be integer 0 to 2

octave: + high

middle

—low

note: CDEFGAB# and R(rest)

length:

unit	code
1/32	0
2/32	1
3/32	2
4/32	3
6/32	4
8/32	5
12/32	6
16/32	7
24/32	8
32/32	9

volume: absolute value from 0(quietest) to 15(loudest). negative value obtains sustain sound effect. If volume = 0, then the specified channel is closed immediately, and the arguments follow volume are don't cared.

```
>10 TEMPO 3
>20 MUSIC 0,15,"C"
>30 MUSIC 1,15,"E"
>40 MUSIC 2,15,"G"
>50 PLAY
>60 FOR I=0 TO 100
>70 NEXT
>RUN
```

(computer plays C major chord)

loop play: with option*, the score is played repeatly.

```
>10 MUSIC 0,—15,*,'EE3E3E3D3C-A3-
    G3C3E3G3E3DEE3E3E3D3C-A3-G3E3D3C7"
>20 PLAY
```

```
>30 CALL SCREEN(1,1)
>40 FOR I=0 TO 255
>50 PLOT SIN(I/20)*50+100,I,15
>60 NEXT
>70 END
```

(play a music and plot sin wave)

NEW

Clears memory and prepares computer for new program.

OUT numeric_expression1,numeric_expression2

A value between 0 and 255 inclusive, provided by numeric_expression2 is output to output_port address provided by numeric_expression1. Numeric_expression1. is also a value between 0 and 255.

PLOT [Y_1,X_1,[,COLOR1][TO]Y_2, X_2,COLOR2][TO Y_3,X_3][,COLOR3] ...]

Plots a dot or draws a line for specified coordinates if color is omitted, it's determined by last PLOT command which specified color.

for screen mode 2

x=0-63

y=0-47

for screen mode 1

x=0-255

y=0-191

e.g.

```
>10 CALL SCREEN (1,1)
>20 PLOT TO RND (191),RND(255),RND(13)+2
>30 GOTO 20
```

(Draw random lines with random color)

e.g. >10 CALL SCREEN (1,1)

```
>20 FOR I=0 TO 255
>30 PLOT SIN(I/20)*50+100,I,15
>40 NEXT
```

(draws a sin curve with screen mode 1)

POKE numeric_expression1 ,numeric_expression2

A value between 0 and 255, provided by numeric_expression2 is written into memory location provided by numeric_expression1 . Numeric_expression1 is value between 0 and 65535.

PRINT [print_list]

? [print_list]

e.g. >?"HEY! HOW ARE YOU?"

```
HEY! HOW ARE YOU?
READY
>
```

RANDOMIZE

Resets random number generator to an unpredicatable sequence.

e.g. >10 PRINT RND (1)

```
>20 GOTO 10
>RUN
.648045
.794064
.248135
BREAK IN 10 (ESC key is pressed)

>10 RANDOMIZE: PRINT RND(1)
READY
>RUN
```

```
.535669
.268712
.743119
.863714
.589787
BREAK IN 20 (ESC key is pressed)
```

REM

Indicates internal program documentation. The statement delimiter ":" is ignored in this statement.

RENUM [initial_line][,increment]

Renumbers program statements starting at 10 in increments of 10. Optionally an initial line number and/or increment may be specified.

NOTE: The computer will take some time if program is large, please wait.

```
e.g. >AUTO
     >10 INPUT A,B
     >20 PRINT A+B
     >30 GOTO 10
     >RENUM 5,5
     >LIST
     >5 INPUT A,B
     >10 PRINT A+B
     >15 GOTO 5
```

RESTORE (line_number)

Resets the DATA list pointer to the beginning of the list or to the first item in the specified DATA statement.

```
>10 READ A,B
>20 DATA 1,2
```

```
>30 PRINT A,B
>40 RESTORE
>45 READ A,B
>50 PRINT A,B
>60 DATA 3,4
>RUN
1          2
1          2
READY

  EDIT 40
    40 RESTORE 30
READY
>RUN
1          2
3          4
READY
>
```

RUN [line_number]

Start program execution at the lowest numbered statement or optionally at the specified line number.

SAVE filename

Places a copy of current BASIC program on the cassette tape.

BSAVE filename,start_address,length

Saves binary contents of computer's memory on cassette tape. If any parameter is omitted, a message "SYNTAX ERROR" will display on the screen and the save is cancelled. Start_address and length are numeric expression in the range from 0 to 65535.

```
> BSAVE "BIT90",32768,1024
READY ? Y
* TAPE DUMP : BIT90.M
* END *
READY
>
```

STOP

The line of this statement is used as a breakpoint. Whenever it is performed, the message "BREAK IN line_number" is displayed on the screen. The program is continued by CONT command.

[TEMPO numeric_expression]

numeric_expression is ranged from 0 to 255. Default value is 5 if this statement omitted.

TRACE

Lists line numbers of statements before they are executed.

UNTRACE

Cancels the TRACE command.

WAIT input_port,numeric_expression1[,expression2]

Suspends program execution while monitoring the status of an input_port(0-255).

The data input from input_port is EOR'ed with numeric_expression2 and AND'ed with numeric_expression1. (default is 0 if omitted)

Program continues with the next statement when the result is nonzero.

3. STATEMENT:

DATA data_list

Stores numeric and string constant data in a program.

GOSUB line_numer

Transfers control to a subroutine at specified line_number until RETURN encountered. The subroutine(s) may be called by another subroutine(nest), there may be 10 levels of nests in BIT90 BASIC.

ON numeric_expression GOSUB line_number_list

Transfers control to the subroutine with a beginning line_numer in the position corresponding to the value of the numeric expression.

ON numeric_expression GOTO line_number_list

Unconditionally branches to line number in the position corresponding to the value of the expression.

e.g.
```
> 5 CALL SPRPTN (1, "183C73FFFF7E3C18")
> 6 CALL SPRPTN (128,96,1,9,1)
> 10 INPUT D
> 20 ON D GOTO 40,50,60,70,80,90,100,110
> 30 GOTO 10
> 40 X=X+1:GOTO 200
> 50X=X+1:Y=Y+1:GOTO 200
> 60 Y=Y+1:GOTO 200
> 70 X=X-1:Y=Y+1:GOTO 200
> 80 X=X-1:GOTO 200
> 90 X=X-1:Y=Y-1: GOTO 200
> 100 Y=Y-1:GOTO 200
> 110 X=X+1:Y=Y-1:GOTO 200
> 200 CALL SPRITE (Y,X,,,1)
> 210 GOTO 10
```

>RUN

(control a sprite for 8 directions moving)

OPTION BASE 0¦1

Sets the lowest allowable subscript of arrays to zero or one. Default is 0.

READ variable_list

Assigns number and string constants in DATA statements to variables listed.

RETURN

Transfers program control from subroutine to statement following corresponding GOSUB or ON....GOSUB statement.

4. FUNCTION:

ABS (numeric_expression)

Returns absolute value.

ASC (string_expression)

Returns the ASCII code of the first character of the string expression.

ATN (radian_expression)

Returns trigonometric Arctangent.

CHR$ (numeric_expression)

Returns the string character corresponding to the ASCII code.

COS (radian_expression)

Returns trigonometric consine.

EXP (numeric_expression)

Returns exponential value of the argument.

FNfunction_name

See DEF statement

HEX$ (decimal_argument)

Returns a string which represents the hexadecimal value of the decimal argument.

IN (numeric_expression)

Returns the content of a input port.

INKEY$

Returns either a one character string containing a character read from the keyboard or character with its code 255 if no character is pressed on the keyboard.

INT (numeric_expression)

Returns greatest integer less than or equal to the argument.

JOYST (key_unit)

Returns key code of correspondent key unit.

key unit	return value
1	play1 joystick
2	play2 joystick
3	play1 keypad
4	play2 keypad

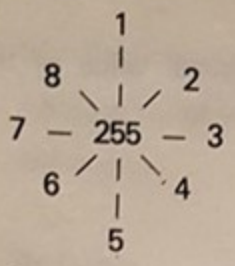
Negative value of joystick indicates fire is pressed.

Negative value of keypad indicates arm is pressed.

keypad pressed	return code

1-9	1-9
0	10
*	11
#	12
none	255

joystick return code;



LEFT$ (string_expression, length)

Returns a substring beginning from the leftmost character with the specified length.

LEN (string_expression)

Returns the number of characters in the string expression.

LN (numeric_expression)

Returns natural logarithm.

LOG (numeric_expression)

Returns logarithm.

MID$ (string_expression,position,length)

Returns a substring beginning in the specified position with specified length.

PEEK (numeric_expression)

Returns the content of memory location.

POS

Returns the current cursor position. The leftmost position is 1.

RIGHT$ (string_expression,length)

Returns a substring ending with the rightmost character with the specified length.

RND (numeric_expression)

Generates a pseudo_random numer with range in zero and specified value and less than one.

SGN (numeric_expression)

Returns 1 if argument is positive, 0 if argument equals zero, −1 if argument is negative.

SIN (radian_expression)

Returns the trigonometric sine.

SPC (numeric_expression)

Prints specified blanks on the screen.

SQR (numeric_expression)

Returns square root.

STR$ (numeric_expression)

Converts the value of the argument into a string.

TAB (numeric_expression)

Controls column position of the cursor.

TAN (radian_expression)

Returns the trigonometric tangent.

VAL (string_expression)

Converts a string representation of a number into a numeric constant.

CHAPTER 4
HOW TO PROGRAM YOUR BASIC LANGUAGE

When you use BIT90 to execute a program, at first, you must clear the memory to prevent your program from being confused. So, type a "NEW" command to start with a new program.

For example:

```
> NEW (CR)
```

If you type a program into the computer, which is perhaps not the correct one, you should display this program and modify it. The procedure named "EDIT" is used very often in entering a new program.

For example:

1. Please type the following program in sequence.

```
10 FOR I=1 TO 7
20 FOR J=1 TO I
30 PRINT "*";
40 NEXT J
50 PRINT
60 NEXT I
```

2. When the program has been input by you, list it and run it.
3. If you cancel the statement of line number 50, what is the result after execution?

4. Try to change the "I" of line number 20 to "8-I" and execute it.

Now, we follow the above procedure to text this program step by step.

1.

```
>NEW (CR)          ; CLEAR THE OLD PROGRAM.
>LIST (CR)         ; LIST VERIFIED NO OLD PROGRAM
                     EXISTED.

READY

>AUTO (CR)
>10 FOR I=1 TO 7 (CR)
>20 FOR J=1 TO I (CR)
>30 PRINT "*"; (CR)
>40 NEXT J (CR)
>50 PRINT (CR)
>60 NEXT I (CR)
>70 (CR)
>
```

2.

```
>LIST (CR)
  10  FOR I=1 TO 7
  20  FOR J=1 TO I
  30  PRINT "*";
  40  NEXT J
  50  PRINT
  60  NEXT I

READY
```

```
>RUN (CR)
 *
 **
 ***
 ****
 *****
 ******
 *******
```

3.

```
50 (CR)                ; DELETE THE LINE NUMBER 50.
LIST (CR)              ; LIST THE NEW PROGRAM HAS
                         BEEN MODIFIED.

10  FOR I=I TO 7
20  FOR J=1 TO I
30  PRING "*";
40  NEXT J
50  NEXT I

READY

>RUN (CR)
 ****************************
```

4.

```
>EDIT 20 (CR)
 20  FOR J=1 TO I      ; SHIFT THE CURSOR [►] TO THE
                         "I" LOCATES AND TYPE
                       ; "8-I" THEN PRESS "RETURN".
```

```
  20  FOR J=1 TO 8-I (CR)
>LIST (CR)
  10  FOR I=1 TO 7
  20  FOR J=1 TO 8-I
  30  PRINT "*";
  40  NEXT J
  50  PRINT
  60  NEXT I

READY

>RUN (CR)
  *******
  ******
  *****
  ****
  ***
  **
  *
```

From the above text program, you may find it's easy to learn BASIC language and edit it. Whenever you need help, just refer to the BASIC command description in chapter 3. In BIT90 computer, many useful and powerful commands can be applied especially those of graphics and music. For the purpose of learning BIT90 BASIC language very fast, we suggest you practise every command and statement following the example first. Thus, very soon, you will be expert in BIT90 BASIC language.

The following section, we will show you a few more interesting BASIC program, you may trace it and further modify with your own ideas.

NOTE: These programs are required 16K or 32K memory expansion module to add, otherwise computer may display "OUT OF MEMORY" on the screen.

[EXAMPLE 1] Compute the combination of C(N,R)

$$C(N.R) = \frac{N!}{(N-R)!\ R!}$$

$$\text{e.g. } C(5,2) = \frac{5!}{3!\ 2!} = \frac{5x4x3x2x1}{3x2x1x2x1} = 10$$

[LIST]

```
 10   READ N,R
 20   IF N=0GOTO 100
 30   C=1
 40   FOR I=NTO N-R+1STEP -1
 50   C=C*I/(N-I+1)
 60   NEXT I
 70   PRINT "C(";N;",";R;")=";C
 80   GOTO 10
 90   DATA 2,1,3,1,4,2,5,2,6,2,6,3,7,3,8,3,9,5,13,8,
100   END
```

[RUN]

```
C(2,1)=2
C(3,1)=3
C(4,2)=6
C(5,2)=10
C(6,2)=15
C(6,3)=20
C(7,3)=35
C(8,3)=56
C(9,5)=126
C(13,8)=1287
```

[EXAMPLE 2]

Please input any positve or negative integer number and print it. Calculate the sum of the digits of this number (e.g. the sum of the digits of 123 is "1+2+3 = 6", the sum of the digits of 32768 is "3+2+7+6+8=26"), and print this sum. If this sum is multiple of 3, then print the value of this sum divided by 3 and "YES", otherwise print "NO" only.

[LIST]

```
 10   PRINT B$
 20   INPUT "NUMBER=?",A$
 30   PRINT
 40   REM INPUT NUMBER STRING
 50   B$="NO"
 60   S=0
 70   FOR I=1TO LEN( A$)
 80   S=S+ASC( MID$( A$,I,1))-48
 90   NEXT I
100   IF ASC( MID$( A$,1,1))=45THEN S=S+3
110   IF S=3*INT( S/3)THEN B$="YES"
120   PRINT A$,"SUM=";S
130   IF B$="NO"THEN GOTO 10
140   PRINT B$,"NUMBER/3=";VAL( A$)/3
150   GOTO 20
```

[RUN]

```
NUMBER=?0

0          SUM=0
YES        NUMBER/3=0
NUMBER=?3

3          SUM=3
YES        NUMBER/3=1
NUMBER=?41

41         SUM=5
NO
NUMBER=?123

123        SUM=6
YES        NUMBER/3=41
NUMBER=?456789

456789     SUM=39
YES        NUMBER/3=152263
NUMBER=?765347887643

765347887643     SUM=68
NO
NUMBER=?374672884566

374672884566     SUM=66
YES        NUMBER/3=1.24890E+11
NUMBER=?-6

-6         SUM=6
YES        NUMBER/3=-2
NUMBER=?-4785

-4785      SUM=24
YES        NUMBER/3=-1595
NUMBER=?-743566543224

-743566543224     SUM=51
YES        NUMBER/3=-2.47855E+11
NUMBER=?-6434446543224

-6434446543224   SUM=51
YES        NUMBER/3=-2.14481E+12
```

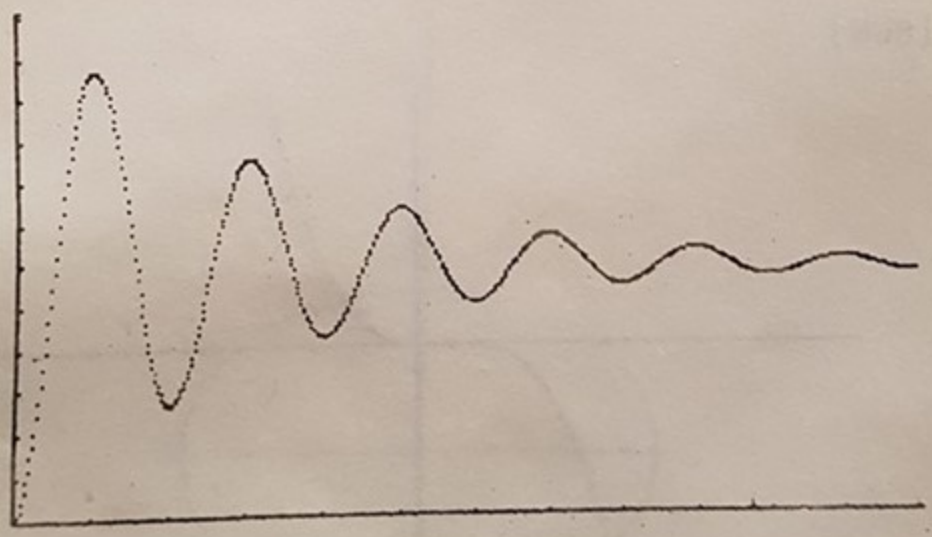
[EXAMPLE 3]

Draws a DAMPING waveform.

[LIST]

```
 10   CALL SCREEN(1,1)
 20   PLOT 0,0,1TO 191,0TO 191,255
 30   FOR I=0TO 6*PISTEP PI/2
 40   PLOT 190,I*255/6/PI
 50   PLOT 191-10*I,1
 60   NEXT I
 70   FOR I=0TO 6*PISTEP PI/90
 80   PLOT 96+100*COS( I*2)*EXP( -I/5)+0.5,I*255/6/PI
 90   NEXT I
100   END
```

[RUN]



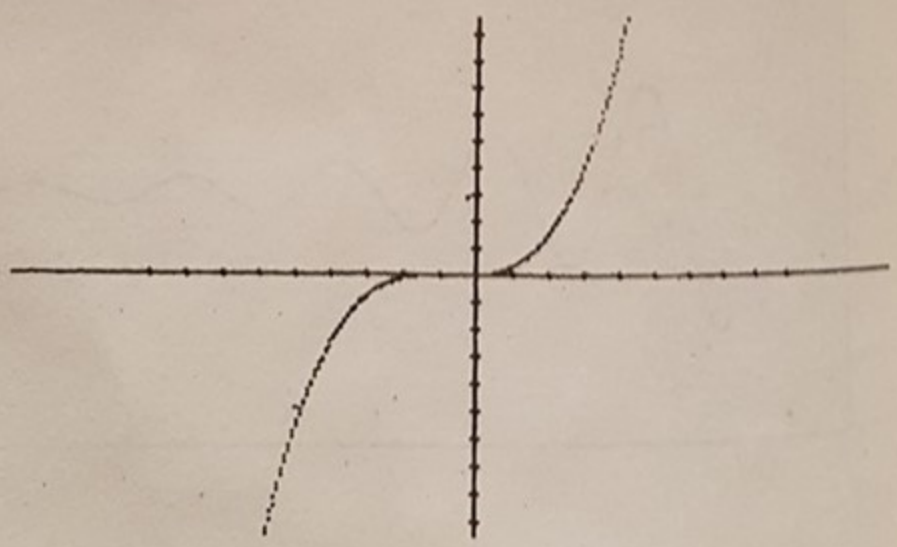
[EXAMPLE 4]

Draws the function of F(X)=X∧3−5*X∧2+6*X−3.

[LIST]

```
 10   REM
 20   REM SOLUTION F(X)=X^3-5*X^2+6*X-3
 30   REM
 40   CALL SCREEN(1,1)
 50   PLOT 0,127,15TO 191,127
 60   PLOT 95,0TO 95,255
 70   FOR I=-90TO 90STEP 10
 80   PLOT 95+I,126TO 95+I,128
 90   PLOT 94,127+ITO 96,127+I
100   NEXT I
110   FOR I=-8.3TO 11.6STEP 0.1
120   PLOT 95-0.1*(I*I*I-5*I*I+6*I-3),I*5+127
130   NEXT I
```

[RUN]



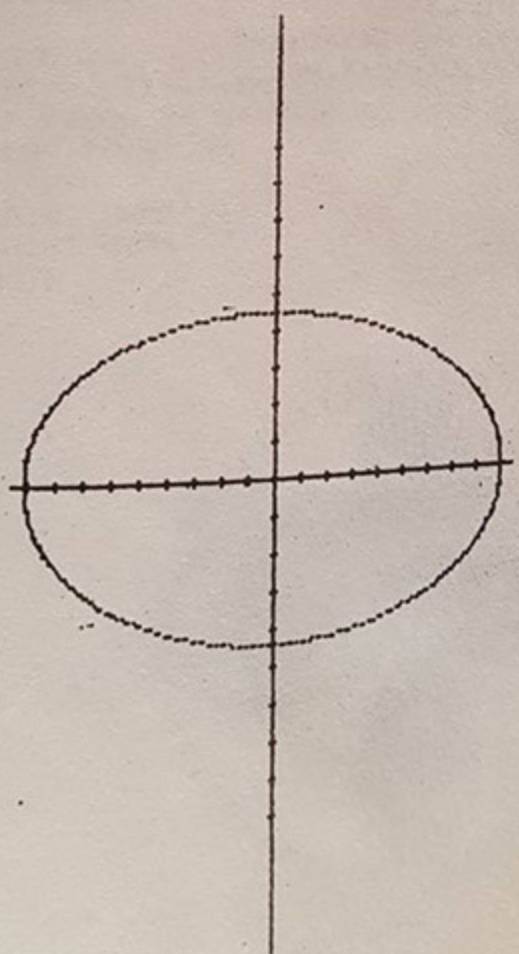
[EXAMPLE 5]

Draws a ellipse.

[LIST]

```
 10   OX=127:OY=96
 20   CO=0:DT=PI/180*1
 30   SC=.7:R=91*SC
 40   CALL SCREEN(1,1)
 50   PLOT 0,127,15TO 191,127
 60   PLOT 95,0TO 95,255
 70   FOR I=-90TO 90STEP 10
 80   PLOT 95+I,126TO 95+I,128
 90   PLOT 94,127+ITO 96,127+I
100   NEXT I
110   FOR I=0TO PI*2STEP DT
120   PLOT OY-R*SIN( I)/SC,OX+R*COS( I)*SC
130   NEXT I
```

[RUN]



[EXAMPLE 6]

Draws a POLAR HEX SIGN.

[LIST]

```
 10 REM DRAWS POLAR HEX SIGN
 20 Z8=88
 30 C3=130:C1=1
 40 C0=0
 60 CALL SCREEN(1,1):SC=.76:C=COS( .1):S=SIN( .1)
 70 CX=130:CY=96
 80 FOR TH=0 TO 6.3STEP .06
 90 R=Z8*COS( 2*TH)
100 HC=7:IF R<0THEN HC=9
110 X=CX+SC*R*COS( TH):Y=CY+R*SIN( TH):RR=(Y-CY)/(X-C
130 IF RR=0THEN GOTO 150
140 PLOT CY,CXTO Y,X
150 NEXT TH
160 HC=15
170 FOR R=90TO 95STEP .8:X=R:Y=0
180 FOR I=0TO 63
190 T=X*C-Y*S:Y=X*S+Y*C:X=T
200 PLOT CY-Y,CX+SC*X,HC
210 NEXT
220 GOTO 220
```

[RUN]

[EXAMPLE 7]

Draws a SURFACE.

[LIST]

```
 5 REM DRAWS A SURFACE Z=F(X,Y)
10 Z1=.1:Z5=.45::XR=255:YL=191:SC=.77:GOTO 100
20 GOTO 100
30 Z=COS( Z1*(X*X+Y*Y)):GOSUB 80
40 IF SX<COOR SX>XROR SY<COOR SY>YLTHEN GOTO 200
50 PLOT SY,SX
60 RETURN
70 REM
80 XE=-X*S1+Y*C1:YE=-X*C1*C2-Y*S1*C2+Z*S2:ZE=-X*S2
   *C1-Y*S2*S1-Z*C2+RHO
90 SX=(D*XE/ZE)*SC+CX:SY=CY-D*YE/ZE:RETURN
100 RHO=30:D=200:THETA=.053:PHI=1:CX=140:CY=76:S1=
   SIN( THETA):S2=SIN( PHI):C1=COS( THETA):C2=COS
   ( PHI)
110 CALL SCREEN(1,1)
120 FOR X=10 TO -10STEP -Z5
130 FOR Y=-10 TO 10STEP Z5
140 GOSUB 30
150 NEXT
160 GOTO 210
170 Z=COS( Z1*(X*X+Y*Y)):GOSUB 80
180 IF SX<CO OR SX>XROR SY<COOR SY>YLTHEN GOTO 200
190 PLOT TO SY,SX
200 RETURN
210 PRINT CHR$(2);
220 PRINT CHR$(27)
240 GOTO 240
```

[RUN]

[EXAMPLE 8]

Calculate the value of PI(π).

$$\pi = 4\times(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\frac{1}{13}-\frac{1}{15}+\ldots\ldots)$$

[LIST]

```
10   INPUT "N(N=4*INT -1)=",N
20   PRINT "N=?";N
30   P=0
40   FOR J=-NTO N-2STEP 4
50   P=P+1/J
60   NEXT J
70   PRINT "TEST PI=";4*P
80   GOTO 10
```

[RUN]

```
N(N=4*INT -1)=3
N=?3
TEST PI=2.66666
N(N=4*INT -1)=7
N=?7
TEST PI=2.89523
N(N=4*INT -1)=11
N=?11
TEST PI=2.97604
N(N=4*INT -1)=35
N=?35
TEST PI=3.08608
N(N=4*INT -1)=143
N=?143
TEST PI=3.1277
N(N=4*INT -1)=2207
N=?2207
TEST PI=3.14065
```

[EXAMPLE 9]

Program a universal calendar after 1981.

[LIST]

```
10   DEF FN A(M)=INT( ABS( M-7.5))-2*INT( INT( ABS( M-7.5))/2)
20   M$=" JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC"
30   W$="SUN MON TUE WED THU FRI SAT"
40   INPUT "WHAT YEAR(AFTER 1981)",Y
50   IF Y<1981THEN STOP
60   Z9=INT( (Y-1981)/4)+INT( (Y-2000)/400)-INT( (Y-2000)/100)
     +Y-1977
70   Z=Z9-7*INT( Z9/7)
80   Z=Z+1
90   T9=0
100  IF Y=400*INT( Y/400)THEN T9=1
110  IF Y<> 100*INT( Y/100)AND Y=4*INT( Y/4)THEN T9=1
120  PRINT SPC( 3);"     --- ";Y;" ---"
130  FOR J=1TO 12
140  T=31-FN A(J)
150  IF J=2THEN T=T+T9-2
160  PRINT
165  PRINT "+++++++++"
168  PRINT "+        +"
170  PRINT "+ ";MID$( M$,1+4*(J-1),4);"  +"
180  PRINT "+        +":PRINT "+++++++++"
190  PRINT
200  PRINT SPC( 8);"*** ";J;" ***"
210  PRINT
220  PRINT W$
230  PO=0
240  FOR T1=1TO T
250  OK=(Z-1)*4
260  PRINT SPC( OK-PO);T1;
270  PO=OK+LOG( T1+.6)
280  IF Z>6THEN PRINT :PO=0
290  Z=Z-7*INT( Z/7)+1
300  NEXT T1
310  PRINT
320  NEXT J
```

[RUN]

```
WHAT YEAR(AFTER 1981)1983
        --- 1983 ---

+++++++++
+       +
+  JAN  +
+       +
+++++++++

        *** 1 ***

SUN MON TUE WED THU FRI SAT
                        1
2   3   4   5   6   7   8
9   10  11  12  13  14  15
16  17  18  19  20  21  22
23  24  25  26  27  28  29
30  31

+++++++++
+       +
+  FEB  +
+       +
+++++++++

        *** 2 ***

SUN MON TUE WED THU FRI SAT
        1   2   3   4   5
6   7   8   9   10  11  12
13  14  15  16  17  18  19
20  21  22  23  24  25  26
27  28

+++++++++
+       +
+  MAR  +
+       +
+++++++++

        *** 3 ***

SUN MON TUE WED THU FRI SAT
        1   2   3   4   5
6   7   8   9   10  11  12
13  14  15  16  17  18  19
20  21  22  23  24  25  26
27  28  29  30  31

+++++++++
+       +
+  APR  +
+       +
+++++++++

        *** 4 ***

SUN MON TUE WED THU FRI SAT
                        1   2
3   4   5   6   7   8   9
10  11  12  13  14  15  16
17  18  19  20  21  22  23
24  25  26  27  28  29  30

+++++++++
+       +
+  MAY  +
+       +
+++++++++

        *** 5 ***

SUN MON TUE WED THU FRI SAT
1   2   3   4   5   6   7
8   9   10  11  12  13  14
15  1   17  18  19  20  21
22  2.  24  25  26  27  28
29  30  31

+++++++++
+       +
+  JUN  +
+       +
+++++++++

        *** 6 ***

SUN MON TUE WED THU FRI SAT
            1   2   3   4
5   6   7   8   9   10  11
12  13  14  15  16  17  18
19  20  21  22  23  24  25
26  27  28  29  30
```

```
+++++++++
+       +
+  JUL  +
+       +
+++++++++
```

*** 7 ***

SUN	MON	TUE	WED	THU	FRI	SAT
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

```
+++++++++
+       +
+  AUG  +
+       +
+++++++++
```

*** 8 ***

SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

```
+++++++++
+       +
+  SEP  +
+       +
+++++++++
```

*** 9 ***

SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

```
+++++++++
+       +
+  OCT  +
+       +
+++++++++
```

*** 10 ***

SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

```
+++++++++
+       +
+  NOV  +
+       +
+++++++++
```

*** 11 ***

SUN	MON	TUE	WED	THU	FRI	SAT
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

```
+++++++++
+       +
+  DEC  +
+       +
+++++++++
```

*** 12 ***

SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

[EXAMPLE 10]

Find out any date letrueen 1901 and 2000 A.D.

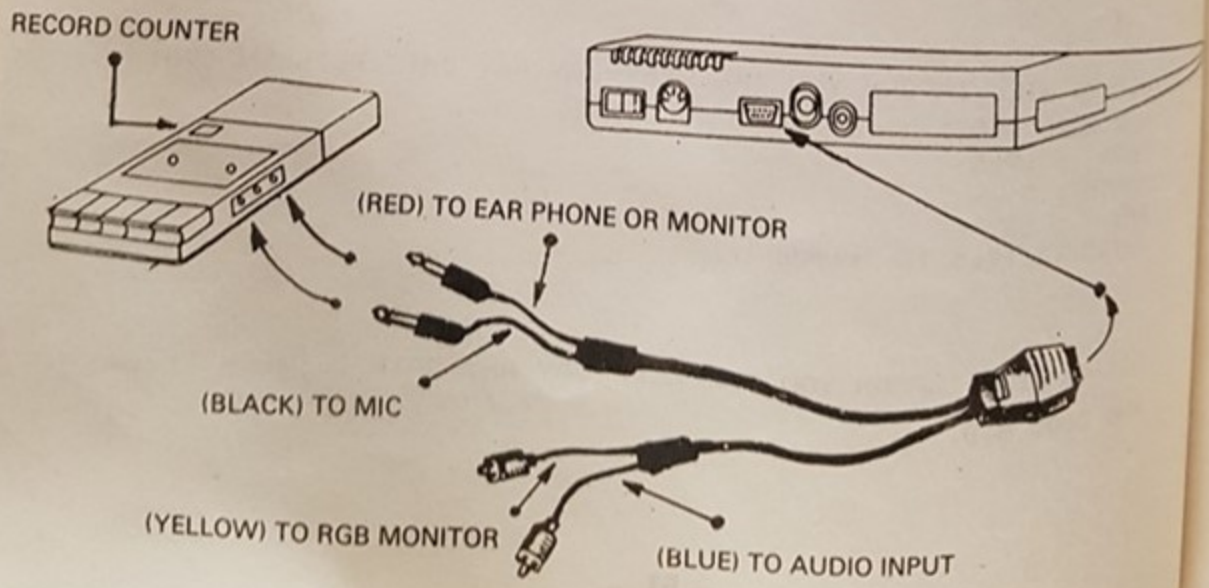
[*LIST*]

```
 10   PRINT :PRINT :PRINT
 20   PRINT "BIT-90 WILL INFORM YOU THE WEEK-DAY ";
 25   PRINT "ANY DATE BETWEEN 1901 AND ";
 27   PRINT "2000 A.D."
 30   INPUT "YEAR  : ",Y
 40   IF Y<1901OR Y>2000GOTO 20
 50   IF INT( Y/4)=(Y/4)THEN X=1
 60   W=2
 70   IF Y=1901GOTO 130
 80   FOR I=1902TO Y
 90   W=W+1
100   IF INT( (I-1)/4)=(I-1)/4THEN W=W+1
110   IF W>6THEN W=W-7
120   NEXT I
130   INPUT "MONTH : ",M
140   IF M<1OR M>12GOTO 130
150   FOR I=1TO M
160   READ D1
170   NEXT I
180   DATA 31,28,31,30,31,30,31,31,30,31,30,31
190   IF X=1AND M=2THEN D1=29
200   INPUT "DAY   : ",D
210   IF D<1OR D>D1GOTO 200
220   RESTORE :D2=0
230   IF M=1GOTO 290
240   FOR I=2TO M
250   READ D1
260   D2=D2+D1
270   NEXT I
280   IF X=1AND M>2THEN D2=D2+1
290   D2=D2+D+W
300   D2=D2-INT( D2/7)*7
310   IF D2=0THEN D2=7
320   PRINT "DATE";M;"/";D;"/";Y;" IS ";
330   ON D2GOSUB 340,360,380,400,420,440,460
331   RESTORE
332   CLEAR
335   GOTO 10
340   PRINT "SUNDAY."
350   RETURN
360   PRINT "MONDAY."
370   RETURN
380   PRINT "TUESDAY."
390   RETURN
400   PRINT "WEDNESDAY."
410   RETURN
420   PRINT "THURSDAY."
430   RETURN
440   PRINT "FRIDAY."
450   RETURN
460   PRINT "SATURDAY."
470   RETURN
```

```
[RUN]

BIT-90 WILL INFORM YOU THE WEEK-DAY ANY DATE BETWEEN 1901
AND 2000 A.D.
YEAR  : 1983
MONTH : 10
DAY   : 4
DATE10/4/1983 IS TUESDAY.


BIT-90 WILL INFORM YOU THE WEEK-DAY ANY DATE BETWEEN 1901
AND 2000 A.D.
YEAR  : 1960
MONTH : 7
DAY   : 3
DATE7/3/1960 IS SUNDAY.


BIT-90 WILL INFORM YOU THE WEEK-DAY ANY DATE BETWEEN 1901
AND 2000 A.D.
YEAR  : 1964
MONTH : 12
DAY   : 27
DATE12/27/1964 IS SUNDAY.


BIT-90 WILL INFORM YOU THE WEEK-DAY ANY DATE BETWEEN 1901
AND 2000 A.D.
YEAR  : 1962
MONTH : 10
DAY   : 14
DATE10/14/1962 IS SUNDAY.


BIT-90 WILL INFORM YOU THE WEEK-DAY ANY DATE BETWEEN 1901
AND 2000 A.D.
YEAR  : 1965
MONTH : 5
DAY   : 13
DATE5/13/1965 IS THURSDAY.


BIT-90 WILL INFORM YOU THE WEEK-DAY ANY DATE BETWEEN 1901
AND 2000 A.D.
```

CHAPTER 5
HOW TO USE THE TAPE RECORDER FOR INFORMATION ACCESS ?

BIT90 computer has a built-in cassette interface for tape recorder, with this we can store the program to tape and load the program from it. Because the computer can't hold the data of memory when the power is failure, the most popular storage device is cassette recorder that can be saved the information permanently. On the other hand, the cassette recoder can also be saved a large number of data with low cost. The data transfer speed of BIT90 is 2400BPS, so that a C-60 (30 minutes per side) tape can be saved about 1 mega bytes (or characters). Now, let's set up the useful storage device. Connect the data plugs RED & BLACK to the EAR PHONE and MIC on the cassette recorder respectively and plug the other teminal of the data cords into the 9-pin jack on the back of the console. Next, typical volume and tone control setting (highest treble positions) are indicated. Adjustments may be required for best performance, depending upon the individual recorder and tape used. Some recorders perform better because of their high-frequency response, which is necessary for good data recovery from computer.



OK ! Let's turn on the computer and type a test program. For example:

```
BIT90  BASIC 3.0                    ; POWER ON COMPUTER AND TV
                                    DISPLAY
READY
>AUTO (CR)
>10 FOR I=1 TO 10 (CR)
>20 PRINT I; I∧2 (CR)
>30 NEXT (CR)
>40 END (CR)
>50 (CR)
>
```

Now, we want to save this program into the cassette recorder, we may use the command "SAVE" and type a file name that can be any character even including reserve word, but can not exceed 15 characters. The following steps tell the computer to save a program.

```
STEP   1.>SAVE "TEST" (CR)
STEP   2.READ ? Y
STEP   3.* TAPE DUMP :
         TEST.B
STEP   4.* END *
         >
```

In step 2, the computer displays "READY" ?" means to ask whether you have set the cassette recorder ready. If so, record the cassette's counter and press the PLAY and REC keys on the cassette recorder, then, respond with "Y" to the computer.

In step 3, the computer begins to dump your program from the memory to the cassette recorder, and just for a few seconds, the computer will display "* END *" to show you the program has been dumped completely and you should press the "STOP" key to stop recording.

When the computer is turned on, you may also "read" a program from the cassette recorder. For example:

```
        BIT90 BASIC 3.0
        READY
STEP    1.>LOAD "TEST" (CR)
STEP    2.READY ? Y
STEP    3.* TAPE LOAD :
          TEST.B
STEP    4.* END *
```

In step 1, you must use the command "LOAD" of BIT90 BASIC and type the file name to be loaded, (the name may be any character even including the reserve words, but may not exceed 15 characters). After commputer responds with "READY ?" telling you to set the cassette recorder ready and rewind the tape back to the proper counter that you have recorded before, you shall type a "Y" to the computer if everything is ready.

In step 3, the computer displays "* TAPE LOAD :" means waiting the cassette recorder to transfer information, and then, you just press the "PLAY" key on the cassette recorder. When the computer finds out the file you need, it will display the name and load this program. Just a few seconds, the computer will load the program completely and display "* END *" on the screen. Perhaps the computer could not find out your file name, it will display the name found currently and search for the file name continuously. Yet, if you didn't type the file name but just type the command "LOAD" only, the computer will assume that you accept any file as it receives from cassette recorder.

If there is already a program in the computer, you may load another one into the computer, but it is important to link these two programs toge-

ther, it could produce line number or label duplicated occasionally. If there is not a command "GOTO" in the either program, the computer won't make an error message. Yet, sometimes, the program is very complex and we use "GOTO" very often, so that it is necessary to use "RENUM" command to prevent the label from being duplicated. For example:

1) There has been a program in the memory already, we can use the "LIST" command to list it.

```
>LIST (CR)
  10  FOR I=1 TO 10
  20  PRINT I,I∧2
  30  NEXT I
  40  END
  READY
>
```

If there is the other program to be linked, then, we can load this program from cassette recorder.

```
> LOAD "TEST" (CR)          LOAD A PROGRAM NAMED TEST
READY ? Y                   ; FROM CASSETTE RECORDER.
* TAPE LOAD :
TEST.B
* END *
```

Then, we list all the program in the computer:

```
>LIST (CR)
  10  FOR I=1 TO 10         ; OLD PROGRAM
  20  PRINT I,I∧2
  30  NEXT I
  40  END
  10  PRINT "LAST I=",I     ; NEW PROGRAM
  20  END
```

In above label 10, 20 are duplicated. the computer can't execute this program. We may use the "RENUM" command to rearrange the label.

```
> RENUM (CR)                ; RENUMBER PROGRAM
> LIST (CR)                 ; LIST PROGRAM AFTER RENUMBER.
  10  FOR I=1 TO 10
  20  PRINT I,I  2
  30  NEXT I
  40  END
  50  PRING "LAST I=",I  ; LABEL IS NOT DUPLICATED
  60  END
  READY
>
```

2) The following program is the other case which appears the "GOTO" command and duplicated label.

```
> LIST (CR)
  10  INPUT A,B          ; OLD PROGRAM
  20  PRINT A*B
  30  GOTO 10

  READY

> LOAD "TEST1" (CR)      ; LOAD A NEW PROGRAM FROM
                           CASSETTE RECORDER.
  READY ? Y
  * TAPE LOAD :
  TEST1.B
  * END *
> LIST (CR)              ; LIST ALL PROGRAMS.
  10  INPUT A,B          ; OLD PROGRAM
  20  PRINT A*B
  30  GOTO 10
```

```
10 INPUT C                    ; NEW PROGRAM
20 PRINT C∧2,C
30 GOTO 10                    ; LABEL IS DUPLICATED.
```

In the above program, the first label 30 and the second label 30 are the same statement: "GOTO 10". The first label 30 "GOTO 10" should jump to "10 INPUT A,B" and the second label 30 "GOTO 10" should jump to "10 INPUT C". If we use the command "RENUM" to adjust the line number directly, the computer will produce an error.

```
>RENUM (CR)
>LIST (CR)
 10 INPUT A,B
 20 PRINT A*B
 30 GOTO 10
 40 INPUT C
 50 PRINT C∧2,C
 60 GOTO 10                   ; THIS STATEMENT SHOULD JUMP
                                TO "40 INPUT C".

 READY                        ; NOT JUMP TO "10 INPUT A,B".
>
```

Now, let's go back to the old program and use the RENUM to rearrange the previous label.

```
>LIST (CR)                    ; LIST OLD PROGRAM
 10 INPUT A,B
 20 PRINT A*B
 30 GOTO 10
 READY
>RENUM 1000 (CR)              ; RENUM OLD PROGRAM TO 1000.
>LIST (CR)
```

```
  1000  INPUT A,B
  1010  PRINT A*B
  1020  GOTO 1000
  READY
>LOAD "TEST1" (CR)        ; LOAD A PROGRAM NAMED TEST1
                          ; FROM CASSETTE RECORDER
  READY ? Y
  * TAPE LOAD :
  TEST1.B
  * END *
>LIST (CR)
  1000  INPUT A,B
  1010  PRINT A*B
  1020  GOTO 1000
    10  INPUT C
    20  PRINT C∧2,C
    30  GOTO 10
  READY
>
```

Let's renumber this program once more:

```
>RENUM (CR)
>LIST (CR)
  10  INPUT A,B
  20  PRINT A*B
  30  GOTO 10
  40  INPUT C
  50  PRINT C∧2,C
  60  GOTO 40
  READY
>
```

From the above case, obviously it's better to use RENUM command to change the label of previous source program. Having loaded next program from recorder, we then use RENUM again to connect these programs completely.

Besides save (or load) BASIC program, BIT90 can also save (or load) binary data or machine language codes to (or from) cassette recorder for the purpose of playing cassette's game or special usage. User may refer to chapter 3 about these commands' description.

CHAPTER 6
CARE OF YOUR BIT90 COMPUTER

1. Place the main unit of BIT90 away from places of high or low temperature, direct sunshine (the main unit may be discolored), or dusty places. Locate it away from a place where the change of temperature is abrupt.
2. The unit is a precision piece, made up of electronic parts. Any modification of, or tampering with, inner parts may cause trouble or accidents.
3. Do not connect a home TV antenna, etc. to the RF converter output terminal of the BIT90.
4. Do not use any equipment other than optional units, to expand your system.
5. Do not drop or throw the main unit. Strong physical shocks must be avoided.
6. Locate the main unit away from a receiver such as a radio receiver since noise may be produced in the radio speaker when BIT90 is operated nearby.
7. Do not spill coffee, juice, tea, etc. over the main unit.
8. Do not disassemble the main unit. When trouble is found, contact the nearest service center immediately.
9. Do not connect the RF converter output terminal of the BIT90 to a home TV master antenna terminal. Use antenna switch box.
10. Do not use volatile solvents such as paint thinner or benzene for cleaning. Wipe the surface of the main unit with a dry cloth.

APPENDIX 1

CONTROL KEY CODES

ASCII CODE	MNEMONIC CODE	PRES	COMMENTS
1	SOH	CONTROL A	Start of heading
2	STX	CONTROL B	Start of text
3	ETX	CONTROL C	End of text
4	EOT	CONTROL D	End of transmission
5	ENQ	CONTROL E	Enquiry
6	ACK	CONTROL F	Acknowledge
7	BEL	CONTROL G	Bell
8	BS	CONTROL H	Backspace
9	HT	CONTROL I	Horizontal tabulation
10	LF	CONTROL J	Line feed
11	VT	CONTROL K	Vertical tabulation
12	FF	CONTROL L	Form feed
13	CR	CONTROL M	Carriage return
14	SO	CONTROL N	Shift out
15	SI	CONTROL O	Shift in
16	DLE	CONTROL P	Data link escape
17	DC1	CONTROL Q	Device control 1 (X-ON)
18	DC2	CONTROL R	Device control 2
19	DC3	CONTROL S	Device control 3 (X-OFF)
20	DC4	CONTROL T	Device control 4
21	NAK	CONTROL U	Negative acknowledge
22	SYN	CONTROL V	Synchronous idle
23	ETB	CONTROL W	End of transmission block
24	CAN	CONTROL X	Cancel

25	EM	CONTROL Y	End of medium
26	SUB	CONTROL Z	Substitute
27	ESC	CONTROL [	Escape
28	FS	CONTROL	File separator
29	GS	CONTROL]	Group separator
30	RS	CONTROL Λ	Record separator
31	US	CONTROL_	Unit separator

APPENDIX 2

(1) ASCII CHARACTER CODES

The defined character on the BIT90 computer are the standard ASCII character for codes 32 through 127. The following chart lists these character and their codes.

ASCII CODE	CHARACTER
32	(space)
33	!(exclamation point)
34	"(quote)
35	# (number or pound sign)
36	$(dollar)
37	%(percent)
38	&(ampersand)
39	'(apostrophe)
40	(open parenthesis)
41	)(close parenthesis)
42	*(asterisk)
43	+ (plus)
44	(comma)
45	—(minus)
46	.(period)
47	/(slant)
48	0
49	1
50	2
51	3

ASCII CODE	CHARACTER
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[(open bracket)
92	(reverse slant)
93	](close bracket)
94	∧ (exponentiation)
95	_(line)
96	‘(grave)
97	a
98	b
99	c

52	4
53	5
54	6
55	7
56	8
57	9
58	:(colon)
59	;(semicolon)
60	< (less than)
61	= (equals)
62	> (greater than)
63	?(question mark)
64	@(at sign)
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{ (left brace)
124	|
125	} (right brace)
126	~ (tilde)

(2) TOKEN CODES:

128	NEW	160	PRINT	192	ABS(
129	SAVE	161	RESTORE	193	ATN(
130	LOAD	162	READ	194	COS(
131	EDIT	163	STEP	195	EXP(
132	DELETE	164	THEN	196	INT(
133	RUN	165	PLOT	197	LOG(
134	STOP	166	RETURN	198	LN(
135	CONT	167	TO	199	SGN(
136	MUSIC	168	UNTRACE	200	SIN(
137	TEMPO	169	IF	201	SQR(
138	POKE	170	TRACE	202	TAN(
139	PEEK(	171	COPY	203	STR$(
140	DEF	172	RENUM	204	CHR$(
141	OPEN	173	PLAY	205	IN(
142	DIM	174	RESERVE	206	JOYST(
143	RND(	175	FRE	207	EOF(
144	TAB(	176	BYE	208	SPC(
145	AUTO	177	END	209	RIGHT$(
146	GOSUB	178	OPTION BASE	210	ASC(
147	CALL	179	LET	211	VAL(
148	DATA	180	OUT	212	LEFT$(
149	ELSE	181	RESERVE	213	MID$(
150	FOR	182	RESERVE	214	LEN(
151	GOTO	183	WAIT	215	INKEY$
152	HOME	184	REC	216	POS
153	INPUT	185	>=	217	BLOAD
154	RANDOMIZE	186	<=	218	FN
155	CLEAR	187	<>	219	BSAVE
156	LIST	188	AND	220	RESERVE

157	REM	189	OR	221	DEL
158	NEXT	190	NOT	222	RESERVE
159	ON	191	HEX$(	223	RESERVE

(3) FUNTION CODES:

CODE	PRESS	CODE	PRESS
224	FNTN 0	229	FNTN 5
225	FNTN 1	230	FNTN 6
226	FNTN 2	231	FNTN 7
227	FNTN 3	232	FNTN 8
228	FNTN 4	233	FNTN 9

(4) MISCELLANEOUS:

CODE	COMMENTS
240	UP ARROW
241	DOWN ARROW
242	LEFT ARROW
243	RIGHT ARROW
244	DEL(delete character)
245	INS (insert character)
247	GRAPHIC SWITCH

APPENDIX 3

OPERATORS

SYMBOL	EXAMPLE	PURPOSE
∧	30 PRINT A∧2	Exponentiation, A must be greater than 0
=	10 LET A=B	Assigns a value equal to a variable
/	20 C=A/B	Division
*	30 C=A*B	Multiplication
+	20 C=A+B	Addition
=	10 IF A=B GOTO 1000	A equals to B
<>	2 IF A<>B GOSUB 10	A does not equal to B
<	3 IF A<B THEN END	A is less than B
>	5 IF A<B THEN PRINT A	A is greater than B
<=	1 IF A<=B GOTO 100	A is less than or equal to B
>=	2 IF A>=B THEN STOP	A is greater than or equal to B
AND	10 IF A=0 AND B=0 GOTO 111	Logical operation of AND
OR	20 IF C=0 OR B=0 GOSUB 100	Logical operation of OR

APPENDIX 4

SPECIAL SYMBOLS

COMMANDS,
STATEMENTS,

SYMBOL	EXAMPLE	PURPOSE
$	10 A$="ABCDE"	STRING IDENTIFIER
"	10 A$="ABCDE"	STRING ENCLOSURES
:	10 A=1:B=0:C=6	ALLOWS MULTIPLE STATEMENTS ON A LINE
;	22 PRINT A;B 23 PRINT A$;B$ 24 PRINT A;B$	ALLOWS SAME LINE PRINTING, BUT CONCATENATED IN CASE OF STRING
,	12 PRINT A,B 13 PRINT A$,B	ALLOWS SAME LINE PRINTING, ELEMENTS ARE SEPARATED BY TABBED PRINTING POSITION
PI	PI	VALUE OF PI IS 3.14159
RUBOUT		THIS IS USED TO DELETE THE CURRENT INPUT LINE BEFORE THE RETURN KEY WAS DEPRESSED.

APPENDIX 5

BUILD-IN FUNCTIONS

FUNCTION	EXAMPLE	PURPOSE
ABS	ABS(X)	Absolute value of X
ASC	ASC(A$)	ASCII code for the first character of string expression
ATN	ATN(X)	Arctangent of X; result is in radians
CHR$	CHR$(X)	Generates a one-character ASCII string; X is in the range 0-255
COS	COS(X)	Cosine of X: X must be expressed in radians
EXP	EXP(X)	Return constant of naperian (=2.71827) raised to power of the argument X.
INT	INT(X)	Largest integer less than or equal to X
LEFT$	LEFT$(A$,N)	Returns leftmost N characters from string A$
LEN	LEN(A$)	Returns the length of the string A$
LN	LN(A)	Natural logrithm of A; A must be greater zero.
LOG	LOG(N)	Logrithm of N; the base of logrithm is ten, N must be larger than zero.
MID$	MID$(A$,A,B)	Returns B characters from string A$, starting with the Ath character.
STR$	STR$(A)	Let the number A as a string
RIGHT$	RIGHT$(B$,C)	Returns rightmost C characters from string B$.
RND	RND(A)	Pseudo-random number between 0 and specified value.

SGN	SGN(X)	Sign function; equal 1 for X>0, 0 for x=0 and −1 for x<0.
SIN	SIN(X)	Sine X; X must be expressed in radians.
SQR	SQR(A)	Square root of A; A must be > =0.
TAN	TAN(X)	Tangent X; X must be expressed in radians.
VAL	VAL(A$)	Returns numeric representation of string. If string not numeric, return zero.
HEX$	HEX$(A)	Return hexadecimal value of A.
FN	FN A(X) FN A$(X$)=X$&""	User defines function
JOYST	JOYST(A)	Return value of joystick unit A.

APPENDIX 6

BIT90 BASIC RESERVE WORDS

ABS(
ASC(
ATN(
AUTO
BYE
CALL
CHR$(
CLOSE
CLEAR
CONT
COPY
DATA
DEL
DELETE
DEF
DIM
EDIT
ELSE
END
EOF(
EXP(
FOR
FRE
BSAVE

FN
GOSUB
GOTO
HEX$
HOME
IN(
IF
INT(
INKEY$
JOYST(
LEN(
LET
LIST
LOAD
LN(
LOG(
NEXT
NEW
MUSIC
MID$
ON
ONERR GOTO
OPEN
BLOAD

OPTIONBASE
OUT
PEEK(
PLAY
PLOT
POKE
POS
PRINT
RANDOMIZE
READ
REM
RESUME
REC
RIGHT$
RENUM
RESTORE
RND(
RUN
SAVE
SIN(
SGN(
SQR(
SPC(

STEP
STOP
STR$
TAB(
TAN(
TEMPO
THEN
TO
TRACE
VAL(
WAIT
>=
<=
<>
AND
OR
NOT

APPENDIX 7

ERROR MESSAGES

ERROR CODE	#	ERROR MESSAGE
	0	BREAK
	1	SYNTAX ERROR
	2	OUT OF MEMORY
	3	CAN'T CONTINUE
	4	LINE BUFFER OVERFLOW
FC	5	ILLEGAL FUNCTION CALL
OV	6	OVERFLOW
UN	7	UNDERFLOW
ST	8	ERROR IN EXPRESSION STACK
EC	9	EXPRESSION TOO COMPLEX
UF	10	UNDEFINED FUNCTION
RD	11	REDIMENSIONED ARRAY
/0	12	DIVIDED BY ZERO
TM	13	TYPE MISMATCH
NR	14	NO RESUME
RE	15	RESUME WITHOUT ERROR
NF	16	NEXT WITHOUT FOR
FN	17	FOR WITHOUT NEXT
RG	18	RETURN WITHOUT GOSUB
OD	19	OUT OF DATA
BL	20	BAD LINE NUMBER
BS	21	BAD SUBSCRIPT
IS	22	INCORRECT STATEMENT
ID	23	ILLEGAL DIRECT
TP	24	TAPE I/O ERROR

1. SYNTAX ERROR:

Line contains incorrect sequence, misspelled characters or incorrect punctuation.

1. IF A=0 A=9
2. LIST 10,200
3. DEL 5,100
4. READ 10,20
5. a variable name is invalid.

   DEF FN 3A(X)=X+3

   DEF FN A(A=A+3

   FOR —K=1 TO 10

   NEXT —I
6. undimensioned array or DIM with more than 3 dimensions.

   DIM A(

   DIM A(10,10,10,10)

   DIM A(A$)
7. OPTION BASE not followed by 0 or 1.

   OPTION BASE 3
8. A$="123
9. LIST 10.5-20
10. LET 1=A
11. INPUT A,

    INPUT ,A

    INPUT A$.
12. SIN(12.A)
13. LEFT$(A,3)
14. ASC("")
15. subscript or assembly language address > 65535 in CALL statement.

    CALL 65536

    CALL SCREEN (100000)

***2. OUT OF MEMORY**

1. Not enough memory to allocate a program.
2. Array size too large.
3. FOR .... NEXT exceeds 10 nests.
4. GOSUB exceeds 10 nests.

***3. CAN'T CONTINUE**

An attempt is made to continue a program that:

1. Has halted due to an error.
2. Has been modified during a break in execution.
3. Program not exists.
4. Continue entered with no previous break point.

***4. LINE BUFFER OVERFLOW**

An attempt is made to input a line that has more than 127 characters.

***5. ILLEGAL FUNCTION CALL (FC ERROR)**

1. A function call such as sin, cos, etc. with parameter out of range.
2. A negative or 0 argument with LOG, LN,
3. A negative argument in SQR.

***6. OVERFLOW (OV ERROR)**

Value greater than +5.5E+18 or less than −5.5E+18

```
>PRINT 5.6E18
OV ERROR
```

*7. UNDERFLOW (UN ERROR)

Value with exponent less than −20.

```
>PRINT −5.4E−20
UN ERROR
>PRINT 5.5E−20
5.49999E−20
>PRINT 5.4E−20
UN ERROR
```

*8. ERROR IN EXPRESSION STACK (ST ERROR)

1. invalid operands in expression.

```
FOR I=2 TO
DIM A( )
VAL (,$)
PRINT PEEK(30208
PRINT (A+B)*C)
PLOT 20,30,
PRINT TAB10)
PRINT INT(,89)
PRINT ASC("0"))
CALL
```

*9. EXPRESSION TOO COMPLEX (EC ERROR)

An expression is too complex.

```
>PRINT LEFT$("ABCDEF",12+1345*12−1890+(98/5+100)/
 56−90*90−70+5−67−67*40/100+(90−10+1000/60−89)*
 10/100−567/21+1890−1000
EC ERROR
```

***10.UNDEFINED FUNCTION (UF ERROR)**

An user defined function is called before the function definition (DEF statement) given.

***11.REDIMENSIONED ARRAY (RD ERROR)**

```
10 DIM A(5)
•
•
•
50 DIM A(10)
```

***12.DIVIDED BY ZERO (/0 ERROR)**

```
10 A=5
20 PRINT A/B
```

***13.TYPE MISMATCH (TM ERROR)**

1. A string variable name is assigned a numeric value or vice versa.

```
10 READ A,B$,C
20 DATA 1,2,3

10 READ A,B,C
20 DATA 1,2,K
```

2. A function that expects a numeric argument is given a string argument or vice versa.

```
PRINT LEFT$(A,3)
PRINT 3 OR 4
CALL SCREEN(6&1)
```

***14.NO RESUME (NR ERROR)**

An error trapping routine is entered but contains no RESUME statement.

```
> 10 ONERRGOTO 200
> 20 B=10
> 30 INPUT A
> 40 PRINT B/A
> 50 END
> 200 PRINT A
> RUN
?0
NR ERROR IN 200
```

***15.RESUME WITHOUT ERROR (RE ERROR)**

A RESUME statement is encountered before an error trapping routine is entered.

```
> 10 ONERRGOTO 200
> 20 INTPUT A
> 30 IF A=0 GOTO 100
> 40 END
> 100 PRINT A
> 200 RESUME
> RUN
?0
RE ERROR IN 200
```

***16.NEXT WITHOUT FOR (NF ERROR)**

A FOR was encountered without a matching NEXT.

```
> 10 FOR I=0 TO 100
> RUN
FN ERROR IN 10
```

***17.FOR WITHOUT NEXT (FN ERROR)**

```
> 10 FOR I=1 TO 10
> 20 J=J+I
> 30 PRINT J
> 40
  READY
> RUN
1
FN ERROR IN 30
```

***18.RETURN WITHOUT GOSUB (RG ERROR)**

```
> 10 RETURN
> RUN
RG ERROR IN 10
```

***19.OUT OF DATA (OD ERROR)**

A READ statement is execute when there are no DATA statement with unread data remaining the program.

```
> 10 READ A,B,C
> 20 PRINT A,B,C
> 30 DATA 1,2
```

***20.BAD LINE NUMBER (BL ERROR)**

1. A line number specified in EDIT, LIST, DEL, RUN... is not found in program.
2. Line number equals 0 ro greater than 9999 or invalid line number.

```
> 10 ONERRGOTO 300
> 20 INPUT A
> 30 PRINT A
> 40 GOTO 20
> RUN
BL ERROR IN10
```

*21.BAD SUBSCRIPT (BS ERROR)

1. subscript in DIM <=0 or greater than 254.

```
>DIM A(0)
>DIM B (256)
```

2. subscript in CALL less than −65535 or greater than 65535.

```
>CALL HCHAR (0,0,65,100000)
>CALL 65537
```

*22.INCORRECT STATEMENT (IS ERROR)

1. A command is used as a statement.

```
>10 A=10
>20 DEL
>RUN
>IS ERROR IN 20

>10 A=2
>20 DATA "$100", "$50"
>30 EDIT 20
>40 RED A$, B$
>50 PRINT A,B
>RUN
IS ERROR IN 30
```

2. a value follow STEP is 0.

```
>10 FOR I=1 TO 10 STEP A
>20 PRINT I
>30 NEXT
>RUN
IS ERROR IN 10
```

3. any error not covered by another message.

```
> 10 DIM A(5,5)
> 20 FOR I=1 TO 11
> 30 A(I,I)=I TO 11
> 40 PRINT A(I,I),
> 50 NEXT
> RUN
1       2       3       4
5
IS ERROR IN 20
```

***23.TAPE I/O ERROR (TP ERROR)**

Data transfer error during tape loading.

APPENDIX 8

PATTERN-IDENTIFIER CONVERSION TABLE

BLOCKS	BINARY CODE (0 = off:1 = on)	HEXADECIMAL CODE
	0000	0
	0001	1
	0010	2
	0011	3
	0100	4
	0101	5
	0110	6
	0111	7
	1000	8
	1001	9
	1010	A
	1011	B
	1100	C
	1101	D
	1110	E
	1111	F

COLOR CODES

COLOR	CODE #	COLOR	CODE #
Transparent	0	Medium Red	8
Black	1	Light Red	9
Midium Green	2	Dark Yellow	10
Light Green	3	Light Yellow	11
Dark Blue	4	Dark Green	12
Light Blue	5	Magenta	13
Dark Red	6	Grey	14
Cyan	7	White	15

APPENDIX 9

MUSIC (CODE OF NOTE LENGTH)

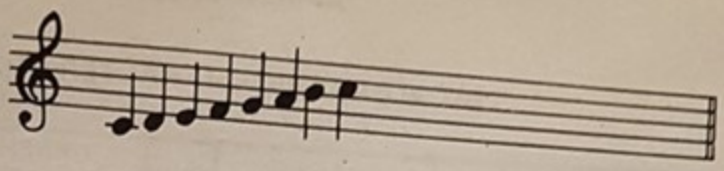
CODE	LENGTH OF NOTE			
0	DEMISEMIQUAVER		DEMISEMIQUAVEREST	
1	SEMI QUAVER		SEMI QUAVEREST	
2	SEMI QUAVER DOT		SEMI QUAVEREST DOT	
3	QUAVER		QUAVEREST	
4	QUAVER DOT		QUAVEREST DOT	
5	CROTCHET		CROTCHET	
6	CROTCHET DOT		CROTCHET DOT	
7	MINIM		MINIM REST	
8	MINIM DOT		MINIM REST DOT	
9	SEMIBRAVE		SEMIBRAVE REST	

Now, let's have a practice, the following transformation is performed according to the above rules.

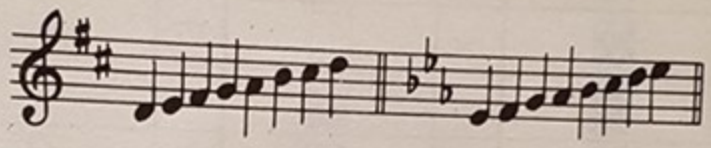
Here are the scales of C MAJOR, D MAJOR and E FLAT MAJOR:

C MAJOR	"CDEFGAB + C"
D MAJOR	"DE # FGAB + # CD"
E FLAT MAJOR	" # DFG # G # A + C + D + # D"

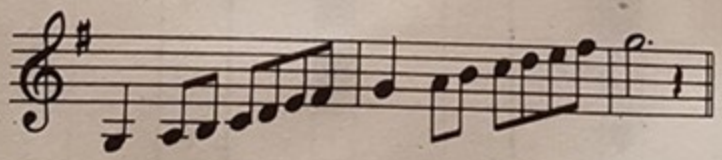
C MAJOR



D MAJOR



E FLAT MAJOR



APPENDIX 10

BIT 90 GRAPHIC CHARACTER CODES

CODE	GRAPHIC CHARACTERS	CODE	GRAPHIC CHARACTERS	CODE	GRAPHIC CHARACTERS	CODE	GRAPHIC CHARACTERS
128	♠	148		168		188	
129		149		169		189	
130		150		170		190	
131		151		171		191	
132		152		172	♣	192	
133		153		173		193	
134	↑	154		174	♦	194	
135	↓	155		175		195	
136	←	156		176		196	
137	→	157		177		197	
138		158		178		198	
139		159		179		199	
140	○	160		180		200	
141	♥	161		181		201	
142	●	162		182		202	
143	π	163		183			
144		164		184			
145		165		185			
146		166		186			
147		167		187			

APPENDIX 11
HARDWARE SPECIFICATION

1. VIDEO

(1) OUTPUT LOAD : 75Ω

(2) OUTPUT LEVEL : 0 ~ 1 VOLT (WHITE)

2. RF

(1) OUTPUT LOAD : 75Ω

(2) CHANNEL : VHF CH4 (STANDARD)

3. AUDIO

(1) SOUND OUTPUT : 1.5 Vp-p

(2) TAPE LOAD : 500m Vp-p

TAPE DUMP : 90m Vp-p

4. POWER DISSIPATION (MAIN BOARD)

(1) +5V : 600 mA

(2) +12V : 200 mA

(3) −5V : 20 mA

5. POWER ADAPTOR

(1) +5V : 900 mA

(2) +12V : 300 mA

(3) −5V : 100 mA