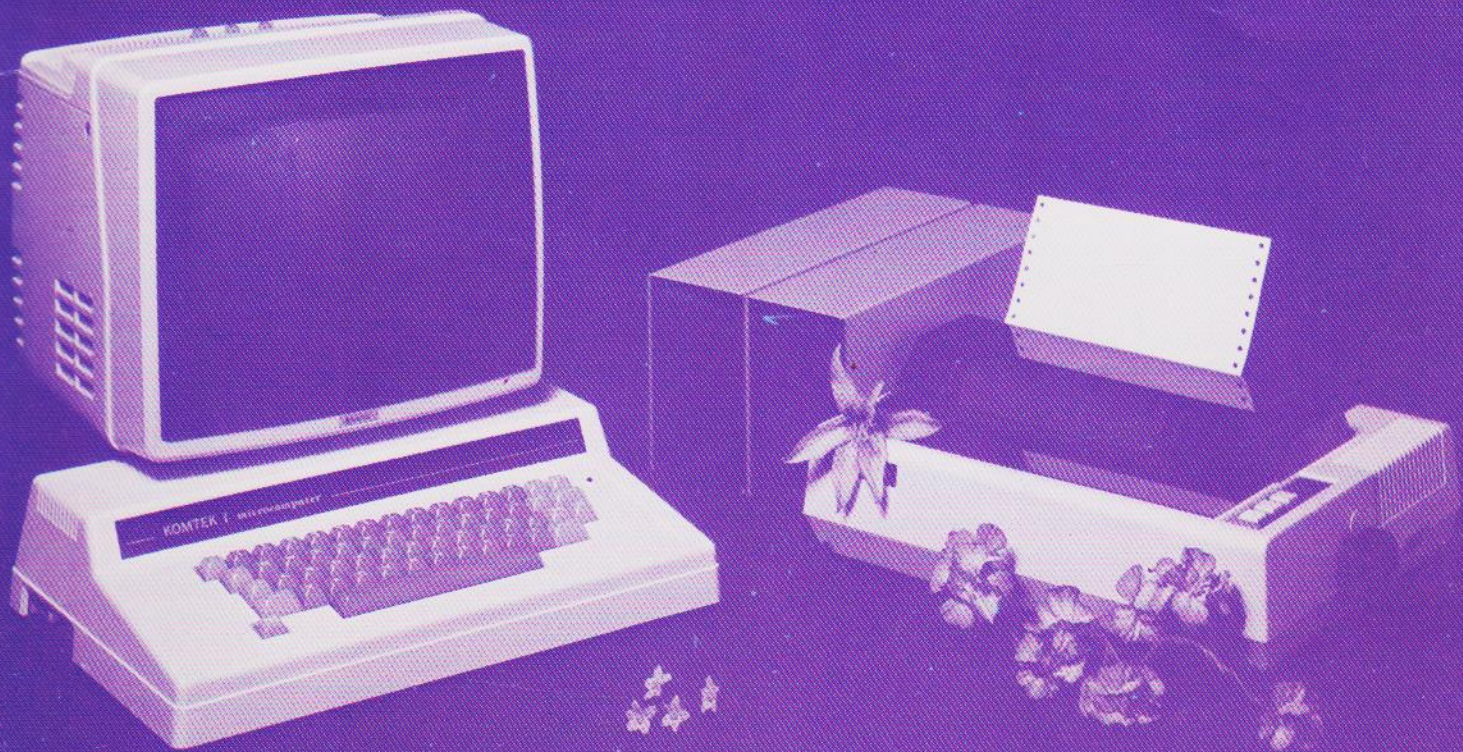
SERVICE COPY for Distributor

KOMTEK I™

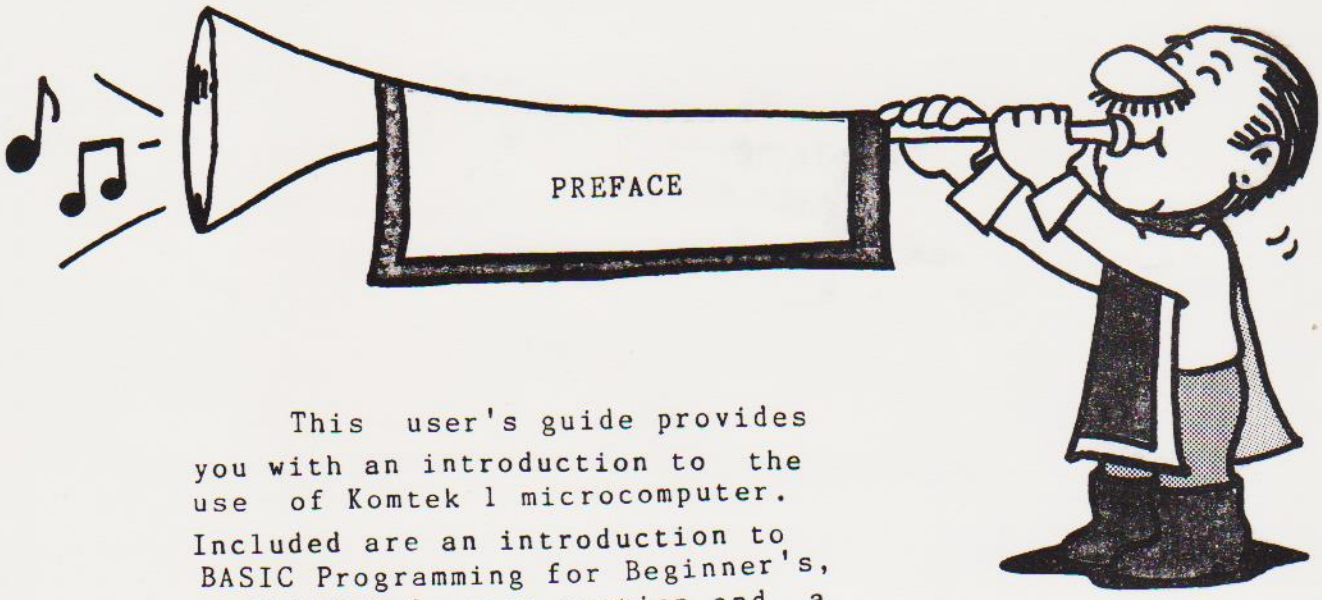
User's Guide



KOMTEK I™

User's Guide

EDUCAID



This user's guide provides you with an introduction to the use of Komtek 1 microcomputer. Included are an introduction to BASIC Programming for Beginner's, a BASIC Reference section and a System Specification section.

It is the author's belief that using microcomputers need not be difficult. This manual is therefore designed, in a very simplified and step by step way, to lead you walk through the various operation procedures. Follow the Introduction to BASIC Programming in Part II, you will find yourself programming with BASIC in no time. However experienced Programmers or these who prefer to work with disk drives should refer to other advanced manuals.

First Edition - 1st June, 1983.

EducAid Centre, Hong Kong for
Komtek Technologies Ltd.

CONTENTS

PART I

USER INSTRUCTION MANUAL
****

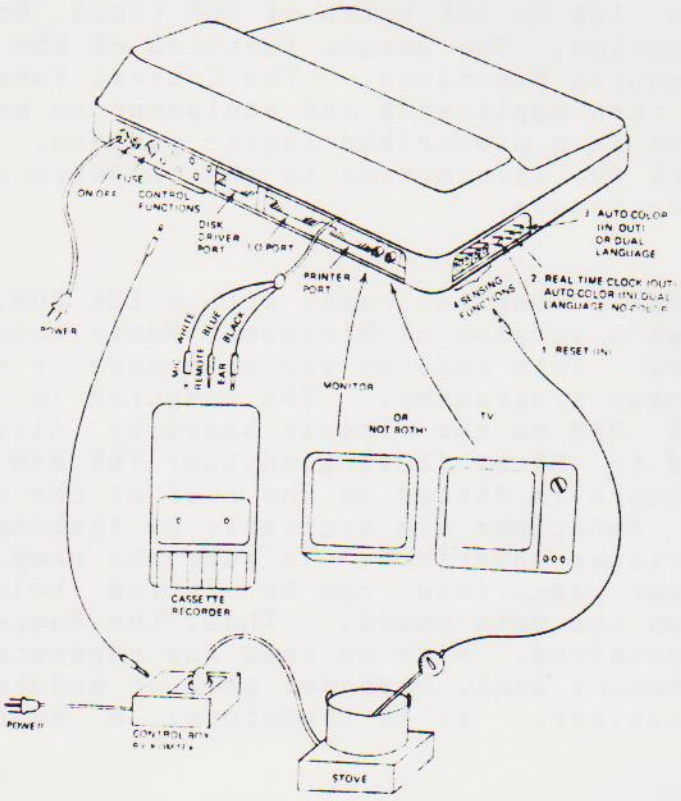
1. Introduction

The Komtek 1 Microcomputer is a general purpose computer. It is compact, durable and versatile. The basic unit comes with a Z80 CPU fitted with 16K bytes of RAM (Random Access Memory) and from 12K to 16K bytes of ROM (Read Only Memory), depending on the option. The unique features of the Komtek 1 are its Control and Sensing Functions. The Control Functions enable the computer to turn appliances and equipment on and off at a prescribed time or in a prescribed logic pattern, the Sensing Functions feed back the information to the computer to enable the computer to make decisions.

The Komtek 1 computer comes with a 12K ROM, programmed with a 12K Komtek's version of Microsoft Basic compatible with TRS Level II Basic. This enables you to create or to use a wide variety of computer programmes. The computer is adapted for expansion to 32K RAM on the circuit board by fitting 8 more memory chips, and to 48K by fitting another 16K RAM card. The disk controller board is fitted on the roof of the cabinet. In addition printer functions are available by fitting a printer control chip or printer interface. To make the computer a colour computer, a colour interface can be fitted below the disk controller board on the main board. Thus, the Komtek 1 computer is totally self contained. With no need for separate power pack, modulator pack, memory pack, expander pack or modules to be seen out side of the cabinet, it is therefore, a unique, compact computer.

2. Setting up the KOMTEK 1 Computer

The Komtek 1 is connected as shown in the following diagram.



Please note that :-

1. "Reset (IN)" Stops the real time clock function and that allows SAVing and LOADing of programmes.

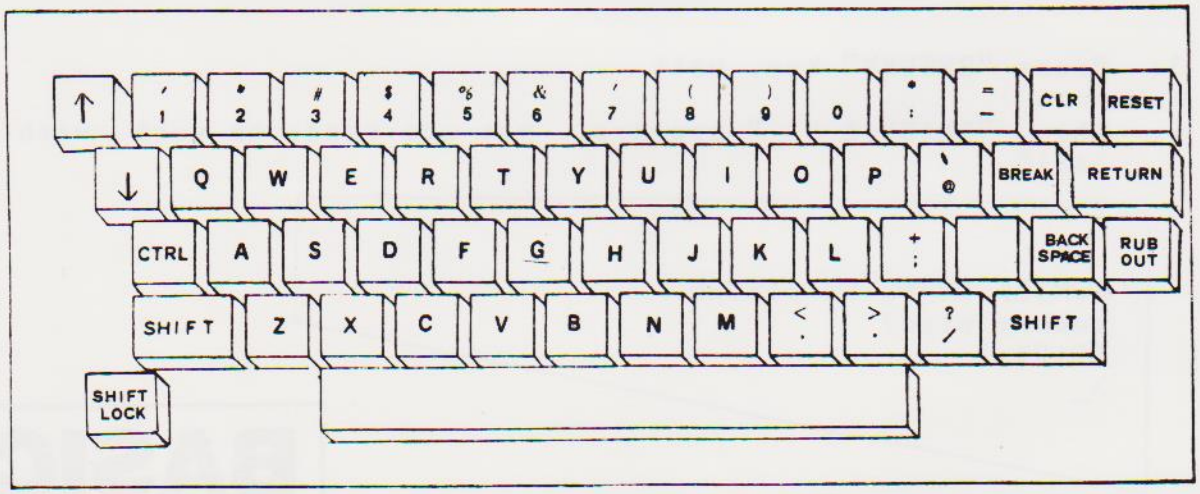
2. The "Control Box" turns power on or off upon receiving signals from the control functions.

3. The set up of the stove and the thermostat is just one example. User can develope your own.

3. Keyboard and Control keys

The Komtek keyboard is an ASCII standard, with double plastic injection keys, and featured with upper and lower case characters. The following are some special control keys for your reference.

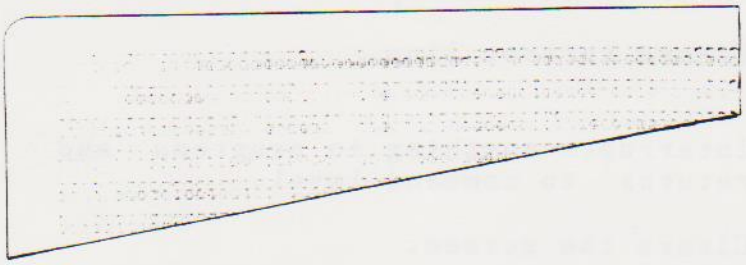
BACK SPACE = <-	Cancels last character typed; moves cursor back one space.
SHIFT & **BACK SPACE**	Erases current line.
BREAK	Interrupts anything in progress and returns to command level.
CLR	Clears the screen.
RETURN	Signifies the end of a current line.
SPACE BAR	Enters a space (blank) character and moves cursor one space forward.
RUBOUT = ->	Advances cursor to next tab position.
SHIFT & **RUBOUT**	Puts display in 32-character mode.
	Line feed and carriage return.
SHIFT & @	Causes currently-executing program to pause (press any key to continue).



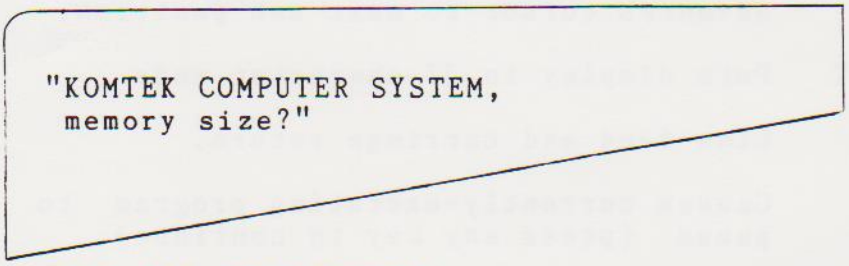
4. TURN ON YOUR COMPUTER

(A) TO RUN BASIC:

1. Turn on Komtek 1 then Monitor or TV
   -(full screen of characters or symbols should appear)



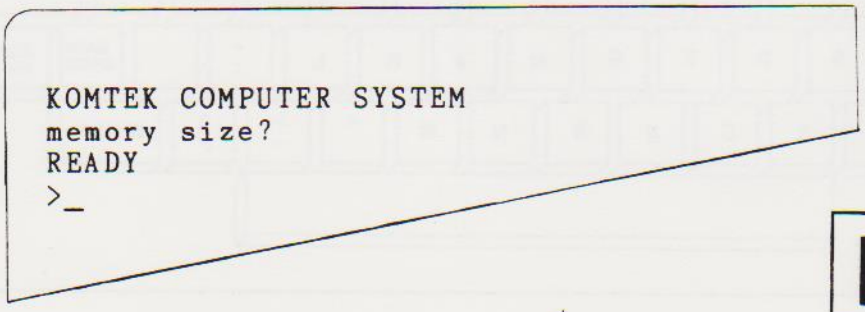
2. Press & hold on to "BREAK" key while you press & let go the "RESET" key -(the following should appears



Take off your fingers.

3. Press "RETURN" key, wait

4. When "READY & > _" appears, you are ready to work with BASIC.



BASIC

(B) TO LOAD OR TO SAVE PROGRAM WITH CASSETTES:

5. Plug in the appropriate jacks to the cassette recorder.

6. To SAVE a program.

   a. Press "RECORD" & "PLAY" buttons on cassette, recorder
   b. Type CSAVE "XXX" & press "RETURN" key.

Note: XXX = stands for your program name not exceeding 8 characters.

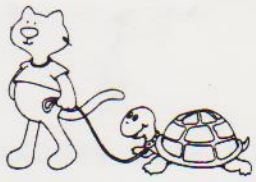
7. To LOAD a program in BASIC

   a. Type "CLOAD" & press "RETURN"*
   b. Rewind if necessary
   c. press "PLAY" button on cassette recorder

Note: * A blinking star on monitor screen indicates successful loading. Otherwise adjust "VOLUME" or check wirings.

8. To LOAD a program in Machine Codes.

   a. At READY >_, type SYSTEM and press RETURN
   b. When a "*?" appears, enter the "program name" or the initial of program name or as specified, and press RETURN
   c. When a second "*?" appears, type "/" and press RETURN

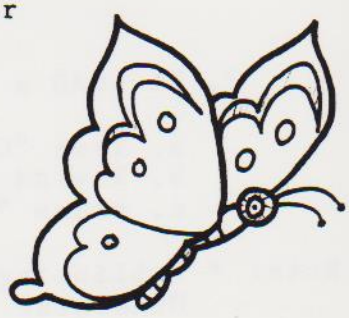


(C) FOR CONTROL FUNCTION:

a. type "POKE 13827,128"
b. type "POKE 13825,1",

where "1" signifies the number of control function signal outlet at the back of computer. Refer to our CONTROL FUNCTION Manuals for more detail.

(D) FOR COLOUR FUNCTION, to activate auto colour



(a) For black & white operation:
POKE 13579,2 Then press RETURN key

(b) For auto colour operation:
POKE 13579,12 Then press RETURN key

(c) For programmable colour operation:
POKE 13579,4 Then press RETURN key

Note that when you are in the programmable mode, colour instructions should be written within a programme. If you do not do this, every letter that you type in will have a colour patch behind it making it seemingly a mess of colours.

E) TO SET THE TIME:

To set the time on the computer you start with seconds, and then minutes and finally hour. It runs on a 24 hour basis and so there is no A.M. or P.M. Let's take an example: if the time you need is 18:45:42 proceed as follows:

(1) Reset the computer

(2) To set seconds
POKE 16449,42
Press RETURN key

(3) To set minutes
POKE 16450,45
Press RETURN key

(4) To set hours
POKE 16451,18
press RETURN key

The clock is now running at 18:45:42 provided the OUT-IN switch is in the OUT position which means the real time clock is connected. Unfortunately you still cannot see the display on the screen. To put the time display on the top right hand corner of the screen, type in the following programme:

```
10 PRINT @56, " ";:FOR A = 16451 TO 16449 STEP -1
20 A$=STR$(PEEK(A)): PRINT RIGHT$("0"+RIGHT$(A$,
   LEN(A$)-1),2);
30 IF A<>16449 THEN PRINT ":";
40 NEXT
50 GOTO 10
```

Finally type RUN and press RETURN key.

For your convenience we suggest that you load the programme on cassette so that next time you do not need to type the programme all over again.

When you turn off the computer, all time settings are lost. The real time clock only works when the computer is on.

PART II

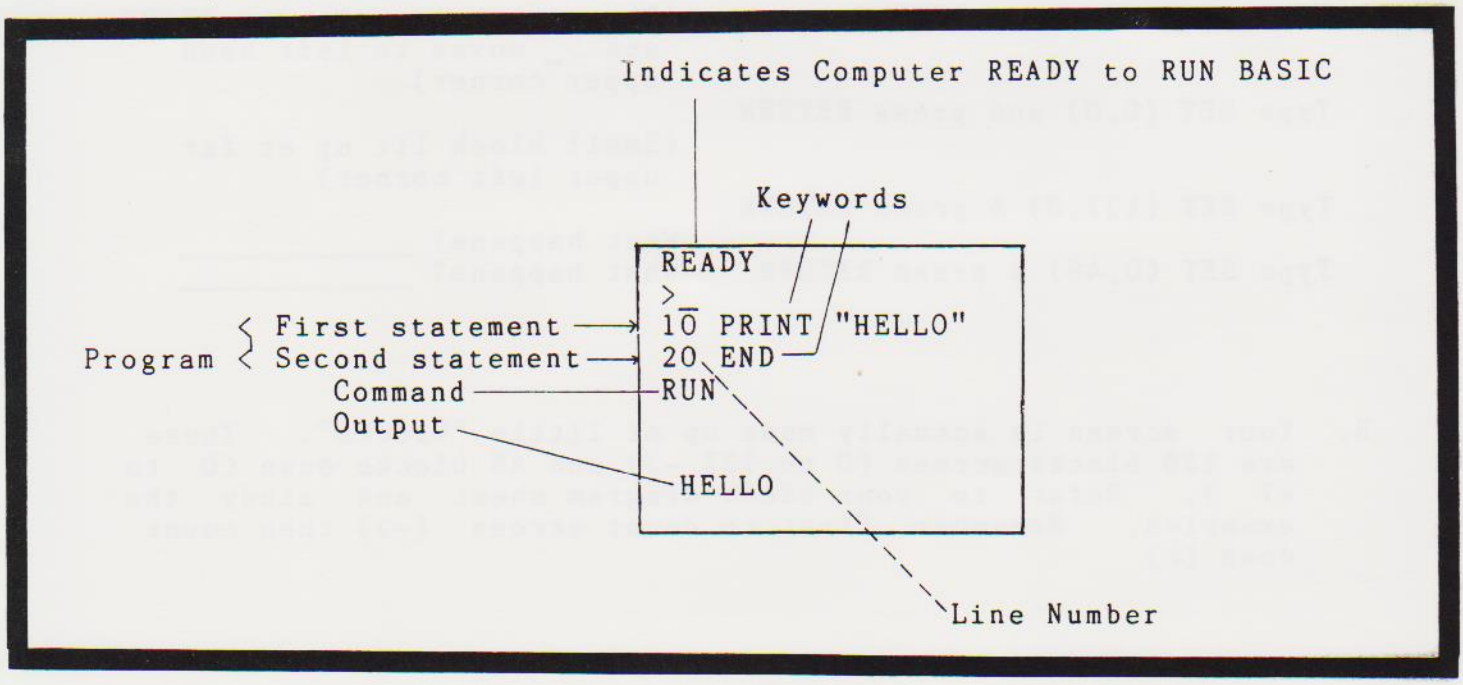
INTRODUCTION TO PROGRAMMING WITH KOMTEK BASIC

- A BEGINNERS VERSION

This programming manual is intended for 1st time computer users. It is intended primarily to get you familarized with your computer and get you started with simple programming. A complete listing of BASIC Programming reference is placed at later sections. For more complete instructional manual for BASIC programming, refer to our edition of BASIC PROGRAMMING manuals.

BASIC language stands for "Beginner's All-purpose Symbolic Instruction Code." It represents one of "computer languages" that the computer understands. We can talk to a computer by writing programs in computer language just as we converse with people from other countries in their particular languages such as English, French, Chinese or German etc..

A sample BASIC program looks like this:



Lesson One

- Use of SET statement to do graphics.

A. When the computer is ready and in BASIC mode, you shall see a prompt ">" and an cursor "_ " waiting for your entry.

```
KOMTEK Computer System
memory size?

READY
>_
```

Try the following:

When you:	The following shall occur
Type CLS and press RETURN	(The screen is clear. READY and >_ moves to left hand upper corner)
Type SET (0,0) and press RETURN	(Small block lit up at far upper left corner)
Type SET (127,0) & press RETURN	What happens? _____
Type SET (0,48) & press RETURN	What happens? _____

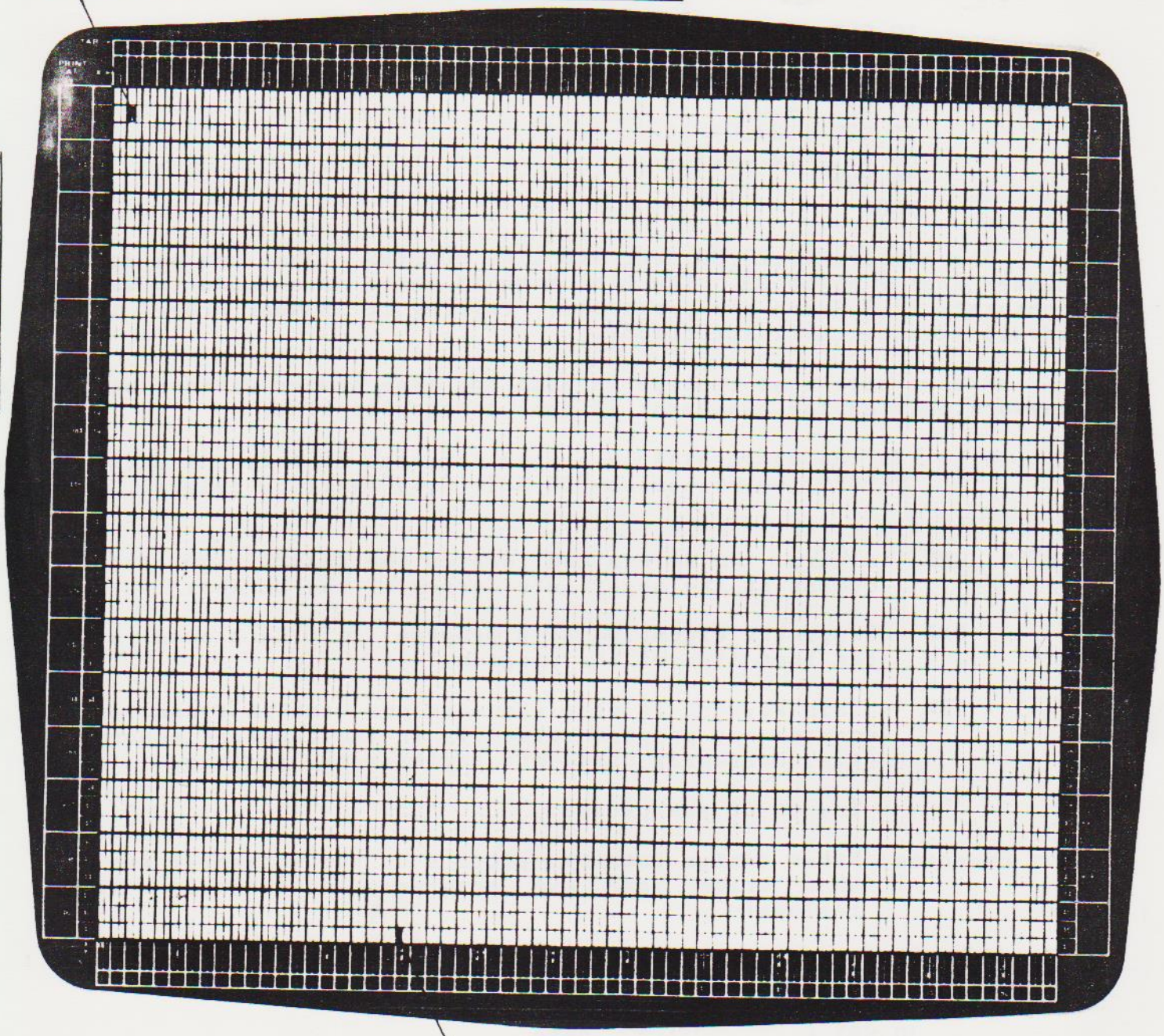
B. Your screen is actually made up of little "blocks". There are 128 blocks across (0 to 127 ->) and 48 blocks down (0 to 47). Refer to your block diagram sheet and study the examples. Remember first to count across (->) then count down (↓)

SAMPLE BLOCK DIAGRAM

SET (3, 1)

0 to 127 (128 BLOCKS) ⟶

0 to 47 (48 BLOCKS)



SET (41, 47)

The statement SET (X, Y) places a "block" on the screen at the position specified by the co-ordinates X and Y where
X = number of blocks across,
Y = number of blocks down.

C. Type the following, and press RETURN after each entry.

```
10 CLS
20 SET (0,0)
30 SET (0,4)
40 SET (4,0)
50 SET (127,0)
60 SET (127,47)
70 END
>RUN
```

The numbers 10,20,30, etc are LINE NUMBERS. This can be any number from 1 to 65529. The LINE NUMBERS serve as a guide to the computer in RUNning the programme, telling it in what order it should execute the instructions.

Please note that to type in just the line number without any keyword deletes the statement in that particular line.

e.g. In the above example, when you type 30 only and press RETURN the statement "30 SET (0,4)" will be deleted.

NOTE* CLS - clears the screen and locates the READY sign and cursor at the upper left corner of the screen.

END - Stops the execution of a program. It is placed at the last line of the program.

NEW - This keyword is used whenever you want to erase or scratch any program currently in the computer memory (i.e. currently used) and begins a new program.

Try it if you please.

D. Exercise: First design your own picture on the worksheet and then write your own programme accordingly as follows:-

```
NEW
10 CLS
20 SET (___, ___)      Block addresses
30 SET (___, ___)      here.
40 SET (___, ___)
 ← - - - - - - - - - - - -add more SET lines here if you
999 END                need them.
RUN
```

E. The LIST Command

Type the following

```
40 SET (120,32)
10 CLS
20 SET (0,5)
60 SET (0,47)
30 SET (120,30)
50 SET (120,31)
```

What do you notice about the order of line numbers? _____

_____

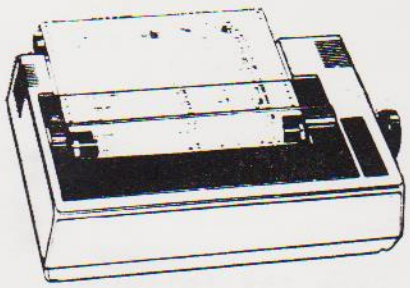
LIST is a command to list the statements in a program according to the ascending order of the line numbers.

Now Type LIST and press RETURN.

What happens when compared with the original program?____

_____

Your Komtek 1 is equipped with a BASIC printer interface to list the program on a printer. Hook up the printer properly and type LLIST and press RETURN.



Lesson 2

- Use of PRINT statement

A. The use of PRINT is to:-

a. print an item or list of items in the display at current cursor position.

b. print values of numerical expressions when using numbers.

B.

1. Type: PRINT "KOMTEK" and press RETURN.
   What do you see on the screen? _____

2. Now type: PRINT KOMTEK and press RETURN
   What happens ? _____.

You should see a "0" instead.

To print "statement words or characters", normally called "strings" on the screen, you must enclosed the expression with open and closed double quotations like this:

"statement, words or characters".

C. Type PRINT "2 + 3" & press return

What happens ? _____
Inside the quotations, "2, +, & 3" are treated as strings of characters.

Now Type PRINT 2 + 3 & pres return.

What happens ? _____
Without the quotation, PRINT actually compute calculations. The symbols used by computer to do arithmetic are + (add), - (subtract), * (multiply),/ (divided by).

D. Type and RUN the following programs.

```
(a) 10 PRINT "     *     "
    20   "    "    ***    "
    30   "    "   *****   "
    40   "    "  *******  "
    50   "    "     *     "
    60   "    "     *     "
    RUN
```

```
(b) 10 PRINT TAB(4)"*"
    20   "     " (3)"***"
    30   "     " (2)"*****"
    40   "     " (1)"*******"
    50   "     " (4)"*"
    60   "     " (4)"*"
    RUN
```

What happens ? _____

The new Statement TAB(4) specified the position of the "*" on the screen at 4 spaces from the left margin, so as the others.

E. Type and RUN this:

```
PRINT  2 + 3, 4 - 3, 2*4, 9/3,
```

What happens ? _____

Your screen should look like this:

```
5 ---------------1 --------------- 8 --------------- 3
```

As you can see The comma "," separates each print out by 15 spaces.

Type PRINT 2; 2 + 3; 4 - 3; 2*4; 9/3

What happens ?_____

The semicolon ";" separates each answer by 1 space only.

LESSON 3

- LET STATEMENT

A. The LET Statement in BASIC.

1. gives a label to the memory location.
2. stores a number in this memory location.

For example :

```
10 LET Y = 365
```

1. It gives the label Y to a location in the computer memory.
2. Stores the number 365 in the memory location having the label Y.

Type and RUN the following:-

```
10 LET S = 60
20 LET M = 60
30 LET H = 24
40 LET Y = 365
50 LET T = S * M * H * Y
60 PRINT "TOTAL NO. OF SECONDS IN A YEAR =  ";T
RUN
```

Now Add :
```
70 LET Q = S*M*H
80 PRINT "TOTAL NO. OF SECOND IN A DAY = ",Q
RUN
```

B. Relation of LET Statememt in a BASIC programme and the memory.

BASIC Program

```
READY

>    10 LET A = 8
     20 LET B = 3
     30 LET C = A+B
     40 LET D = 20*C
     50 LET D = D+1000
     60 PRINT A;B;C;D;

8 3 11 1220
```

MEMORY

A	B	C	D
8			
	3		
		11	
			220
			1220

This is printed for the value of D instead of 220, because the computer always displays the final answer in the memory location.

C. Exercise:

Complete the following program.

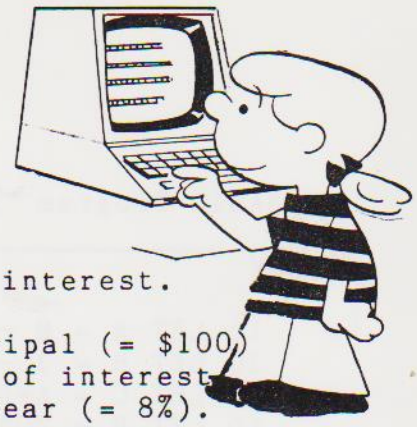
```
>- READY
   10 LET A = 17
   20 LET B = 13
   30 LET C1 = A + B
   40 LET C2 = A - B
   50 LET C3 = A * B
   60 LET C4 = A / B
   70 PRINT A;B;C1;C2;C3;C4
   80 LET A = A * 100
   90 LET B = A + B
   100 LET Q = A + B
   110 PRINT Q
   120 END
>RUN
```

A	B	C1	C2	C3	C4	Q
17						
	13					

Before you actually RUN the program, try to guess the actual answers _____.

Now RUN and compare your answers.

Lesson 4



-INPUT

A. Here is a program to calculate annual simple interest.

```
10 LET P = 100
20 LET R = 0.08
30 PRINT I = P*R
40 END
>RUN
```

P = Principal (= $100)
R = Rate of interest per year (= 8%).
I = annual Interest

Before you actually RUN the program, try mentally compute the answer _____.

Now RUN the program and compare the answers.

B. What about creating a program that allows you to change the value of the variables namely the principal (P) and the Interest Rate (R) by INPUTting other values.

(1) Change the previous program to:-

```
10 INPUT P
20 INPUT R
30 PRINT I = P*R
40 END
>RUN
```

What happens ? _____

You'll find a "?" appear awaiting your INPUTting of your data for P in line number 10.

(2) Type 100 & RETURN

You'll find a second "?" appear awaiting your input for R in line number 20.

(3) Type 0.08 & RETURN

What happens ? _____

(4) Now RUN your program again.
This time enter P = 100,000 and Rate = 0.14.

What is your answer for I? _____

C. Exercise.- Write a program which calculates how many hours you have lived by INPUT your age. (Hint: Hours lived = 24 hr x 365 days x years lived).

D. - Define your input:

(1) Type INPUT "What is the Principal"; P

What happens ? _____

Other than a "?", a comprehensive question for the specified input is awaiting you.

(2) Change line number 10 & 20 in section B according to what you have just learned.

RUN the new program. Feel better?

Lesson 5

- GOTO Statement

Here is your chance to tell the computer where to go.

A. Remember in Lesson 4 Section B, you have to type RUN in order to re-RUN a program. A GOTO Statement when properly used can eliminate these procedures.

Change line number 40 as follows and RUN the program

```
10 INPUT "What is the principal";P
20   "   " "   "   "  Rate     ";R
30 PRINT "INTEREST IS  ";I=P*R
40 GOTO 10
50 END
```

Computer executes statement in the order governed by the line numbers.

The GOTO or GO TO statement actually creates a "loop" by instructing the computer to go to a specific statement anywhere in the program.

An infinite looping/execution of a program can be halted abruptly by pressing the BREAK key.

B. Exercise:

Try the GOTO statement on the program in Lesson 2 section D.b.

Lesson 6

- IF... THEN & STOP

A IF...THEN Statement sends the program execution to a given Line number if the condition is true.

Here a program to determine one's eligibility to obtaining a driver's license.

```
10 INPUT "Type your age  "; A
20 IF A>=18 THEN 50
30 PRINT "COME BACK WHEN YOU GROW UP, KID"
40 STOP
50 PRINT "YOU ARE ELIGIBLE TO DRIVE,  HAVE YOUR
   MONEY READY?"
60 END
```

The "IF" sets the conditions and "THEN" sends the program execution to a given line.

The "STOP" stops the execution of a program at a specified place

(e.g. at line number 40 this time).

Here's another example.

```
>READY
10 LET N = 1
20 IF N > 10 THEN 60
30 PRINT N*N;
40 LET N=N+1
50 GOTO 20
60 END
```

The "loop" executes N*N with N's value from 1 to 10. Line number 40 "N=N+1" actually increase N by value of 1 each time.

Lesson 7

- FOR and NEXT

A. The FOR and NEXT Statements are used to simplify the writing of programs the usually repeats execution.

Looping with IF....THEN		Looping with FOR and NEXT
10 LET N = 1		
20 IF N>12 THEN 60	<-->	10 FOR N=1 TO 12
30 PRINT N*N;	<-->	20 PRINT N*N
40 LET N=N+1	<-->	30 NEXT N
50 GOTO 20		
60 END		40 END

Type and RUN both programs

These two programs do the same thing:

- They both start N out equal to 1.
- They both PRINT N*N, and then increase N by 1.
- They both continue to run over and over until finally N reaches 12.
- Then they both stop.

B. Use of keyword STEP X , where X = any number.

For example FOR N = 1 TO 12 STEP 2
means counting from 1 to 12 by 2's

Type and RUN

```
10 FOR N = 1 TO 5 STEP 1
20 PRINT N
30 NEXT N
40 NEW
>RUN
```

Screen shows

```
 1
 2
 3
 4
 5
```

Change STEP in line number 10 to STEP2, What happens ? _____

Change 10 to 10 FOR N = 5 TO 1 STEP-1, What happens ? _____

C. Using variables in FOR - NEXT STATEMENTS

(1) Type and RUN the following.

```
10 FOR N = 1 TO 16
20 PRINT "HONG KONG IS NO. 1"
30 NEXT N
40 END
>RUN
```

(2) Let's add and change to 5 INPUT V

```
10 FOR N = 1 TO V
```

(3) RUN the program.

When "?" appears, input the number (1 to 16) for the number of rows you want to print on the monitor screen.

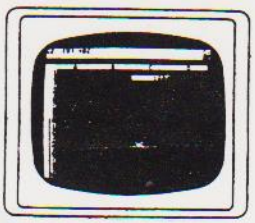
D. Use of FOR...NEXT Statement in Graphics

(1) Type the following program.

```
READY
>10 FOR X = 0 TO 127
 20 FOR Y = 0 TO 47
 30 SET (X,Y)
 40 NEXT Y
 50 NEXT X
 60 END
```

(2) What happens if you actually RUN this program ?

(3) Now RUN and verify your answer.



Lesson 8

- READ and DATA Statements

(1) So far you have learned two ways of entering data into a program: the LET statement and the INPUT statement.

Type and RUN the following:

```
 5 FOR I = 1 TO 5
10 READ I
20 PRINT I
30 NEXT I
40 DATA  1,2,3,4,5
```

The READ Statement names the variables (in this case "I") in which the values are to be stored.

The DATA statement contains the value (here 1,2,3,4,5) which will be stored in the variables.

(2) Examples of how these statements are used.

```
READ   A  B
READ      C  D  E
DATA   1  3  5  7  9  11  13
```

They are never used.

(3) An OD error will occur (OD = Out of Data) with the following condition since there's no "DATA" for C to "READ".

```
READ   A  B
READ      C
DATA   5  7
```

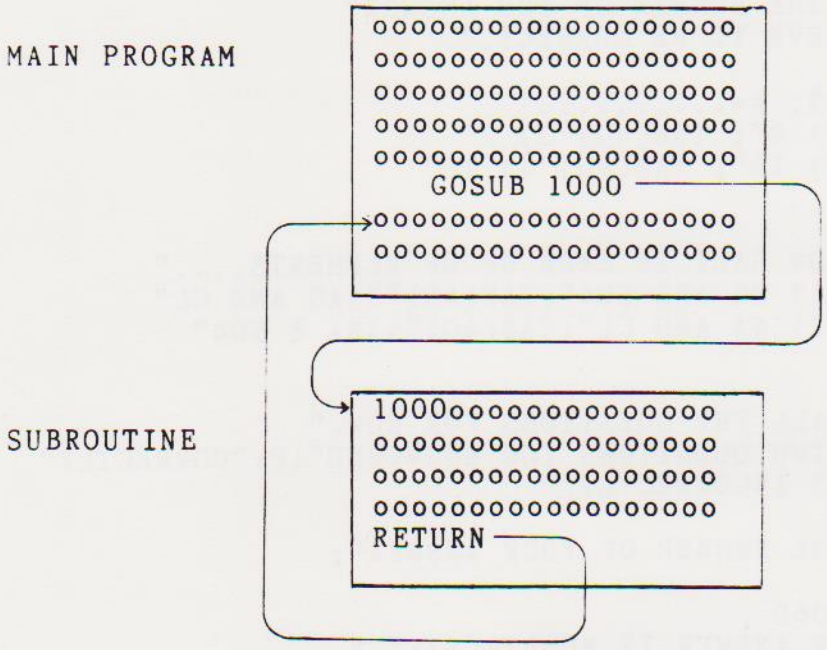
(4) Exercise: modify the program in Lesson 3, section A into one using READ _ DATA statements.
(Hint: Let $,M,H,Y, be read as datas)

Lesson 9

- GOSUB.....RETURN

A. A GOSUB statement sends the program execution to a subroutine in accordance with the line number specified.

Example:



The RETURN Statement is used to tell the computer that the subroutine is finished and the program should now resume execution where it left the main program.

B. Study the following program. It is a program to create multiple-choice exercises for learning.

Type and RUN the followings.

READY

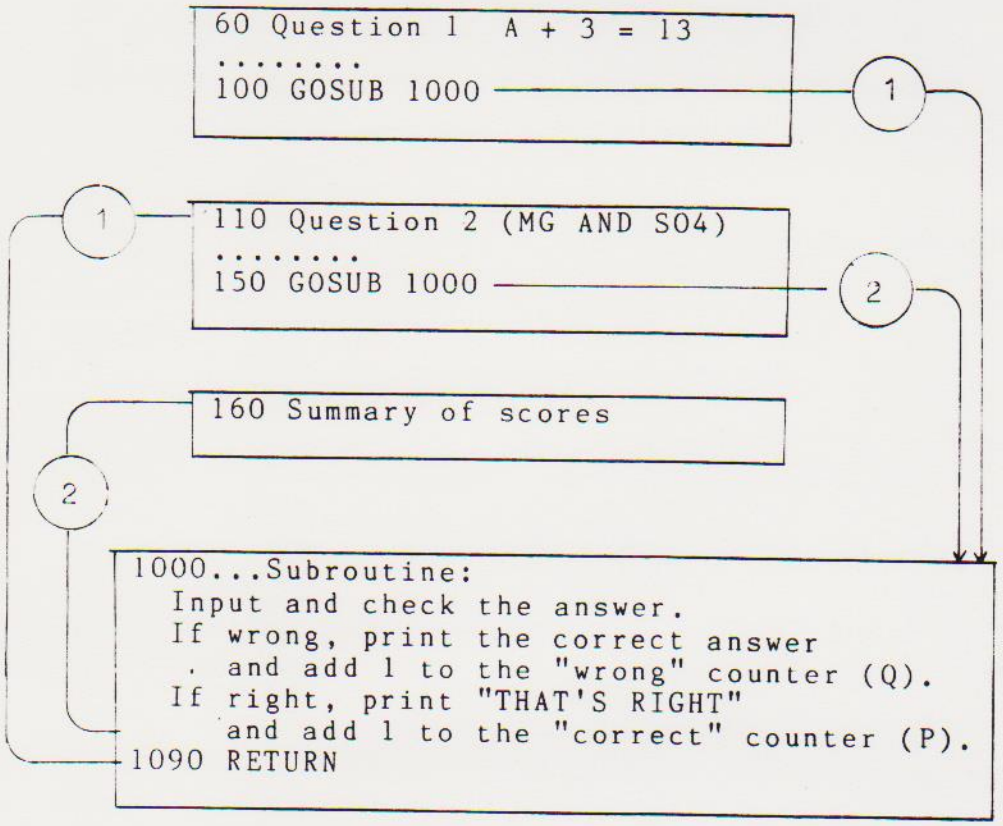
main program

```
10 PRINT "ANSWER THE TWO QUESTIONS IN THIS PROGRAM"
20 PRINT
30 PRINT "TYPE IN THE NUMBER OF THE ANSWER"
40 PRINT "YOU BELIEVE TO BE CORRECT."
50 PRINT
60 PRINT "1. A+3=13, A=........."
70 PRINT TAB(10)"1) 9"; TAB(40)"3) 7"
80 PRINT TAB(10)"2) 16"; TAB(40)"4) 10"
90 LET A = 4
100 GOSUB 1000
110 PRINT "2. COMMON SALT IS MADE UP OF ELEMENTS...."
120 PRINT TAB(10)"1) MG AND SO4";TAB(40)"3)AG AND CL"
130 PRINT TAB(10)"2) NA AND CL";TAB(40)"4)NA & SO4"
140 LET A=2
150 GOSUB 1000
160 PRINT "THAT'S ALL THE QUESTIONS FOR NOW."
170 PRINT "OUT OF TWO QUESTIONS YOU ANSWERED";P;"CORRECTLY"
180 PRINT "AND";Q;" INCORRECTLY."
190 GOTO 1900
```

Subroutine

```
1000 PRINT "TYPE THE NUMBER OF YOUR ANSWER";
1010 INPUT R
1020 IF A=R THEN 1060
1030 PRINT "NO, THE ANSWER IS NUMBER";A;"."
1040 LET Q=Q+1
1050 GOTO 1080
1060 PRINT "GREAT!"
1070 LET P=P+1
1080 PRINT
1090 RETURN
1900 END
```

Here's a sketch of how this multiple-choice program works:



** Composers and Programmers belong to the same breed of people, some are born to be and some try harder - Author **

END OF PART II

PART III

BASIC Reference Section

(1) BASIC Command And Statements

AUTO

Automatically generates line.

e.g. AUTO

CHR$(x)

Returns the (ASC II) equivalent of the decimal within the parenthesis.

e.g. CHR$(23)

CLEARn

Reserves n bytes of string storage space; initializes all variables.

e.g. CLEAR

CLOAD

Load program from cassette.
e.g. CLOAD

CLOAD?

Compares program on tape byte-for-byte with resident program.

e.g. CLOAD?

CLS

Clears the display screen and returns the cursor to the upper left corner of the screen.

e.g. CLS

CONT

Continues execution of program after BREAK or STOP.

e.g. CONT

CSAVE

Save program onto cassette.

e.g. CSAVE"MAILING"

DATA

Holds the data(values) for READ statements.

e.g. 200 DATA 2,3,6

DEFDBL

Defines variables as double-precision.

e.g. DEFDBL V,X-Z

DEFINT

Defines variables as integer type.

e.g. DEFINT A, I-N

DEFSNG

Defines variables as single-precision.

e.g. DEFSNG I, W-Z

DEFSTR

Defines variables as string type.

e.g. DEFSTR C, L-Z

DELETE

Delete line number specified.

e.g. DELETE 1-300 = DELETE -300

DIM

Declares maximum sizes of arrays.

e.g. 150 M(20), N(15,20)

EDIT

Puts computer into edit mode for specified line.

e.g. EDIT 100

END

Last line of program.

e.g. 999 END

ERROR(n)

Simulates the specified error,n=1-23.

e.g. ERROR(1)

FOR...TO... (STEP)

Sets up and runs the body of a loop a stated number of times.

e.g. 40 FOR I=1 TO 9 STEP 2 (Body of the loop)

GOSUB

Sends the program execution to a subroutine.

e.g. GOSUB 750

GOTO

Sends the program execution to another line.

e.g. 60 GOTO 205

GOTO...OF (ON...GOTO)

Sends the program execution to one of several lines depending on the value of the variable.

e.g. 310 GOTO Y OF 35, 90,125 (310 ON Y GOTO 35, 90,125)

IF...THEN

Sends the program execution to the given line if the condition is true.

e.g. 90 IF W8<=4 THEN 260

INKEY$

Causes the program to halt and wait for a string variable X to be input. It is not necessary for the RETURN key to be pressed for the input to be accepted.

e.g. 20 IF INKEY & = " " THEN GOTO 20

INPUT

Requests data for certain variables from the terminal (during a RUN).

e.g. 380 INPUT A,B

INPUT#b

Inputs data from specified cassette unit.

e.g. INPUT#-1,A

LET

Calculates an expression and assigns the value to a given location.

e.g. 50 LET Y = 7, 60 LET X = 2*B+X

LIST

Prints out the current program.

e.g. LIST

LLIST

Prints out the current program through the printer.

e.g. LLIST

LPRINT

Print out the messages through the printer.

e.g. 10 LPRINT "Hello"

LPRINT TAB

Moves printer carriage to sepcified position.

e.g. LPRINT TAB(25) "NAME"

LPRINT USING

Prints formatted numbers and strings on the printer.

e.g. LPRINT USING "####,";1234

NEW

Erases the current program.

e.g. NEW

NEXT

Closes the loop.

e.g. 80 NEXT I

ON ERROR GOTO

Sets up an error-handling routine.

e.g. ON ERROR GOTO 2100

ON ERROR GOTO

Disables an error-handling routine.

e.g. ON ERROR GOTO0

ON...GOSUB

Multi-way branch to specified subroutines.

e.g. ON Y GOSUB 50,100,150,200

OUTp,v

Sends value to specified port.p and v =0-255.

e.g. OUT 255,0

PEEK

Returns the decimal equivalent of the contents of the memory address specified by (X).

e.g. A = PEEK (X)

POKE

Stores the machine language equivalent of the Variable Y at memory address X.

e.g. POKE 13579,4 to activate auto colour

PRINT

Types out messages or values of numerical expressions or both.

e.g. PRINT X! + Y!

PRINTn

Prints beginning at n,n=0-1023.

e.g. PRINT @477,"CENTER"

PRINT#-b

Writes data to specified cassette deck.

e.g. PRINT#-1,A

PRINT TAB

Moves cursor right to specified tab position.

e.g. PRINT TAB(20) "NAME"

PRINT USING

Formats strings and numbers: # Formats numbers.

e.g. PRINT USING "#####";66.2

RANDOM

Reseeds random number generator.

e.g. RANDOM

READ

Assigns values from DATA statements to given variables.

e.g. 150 READ A(J), B(J),C

REM

Permits comments.

e.g. 105 REM CALCULATES AREA

RESET(x,y)

Turns off graphics block at specified location. X(horizontal)=0-127;Y(vertical)=0-47.

e.g. RESET(21,40)

RESTORE

Allows data to be used again.

e.g. 238 RESTORE

RESUME

Ends an error-handling routine by specifying where normal execution is to resume.

e.g. RESUME

RETURN

Sends the program execution back to the line after GOSUB.

e.g. 320 RETURN

RND

Returns a value between o to X.

e.g. RND (X)

RUN

Begins execution of the program.

e.g. RUN

SET

Turns on a graphic block specified by the variables X and Y or any expression used to produce a number within the limits of the screen.

e.g. SET (14,13)

STOP

Halts RUN of program (may be anywhere within the program).

e.g. 65 STOP

SYSTEM

Puts computer in monitor mode, allows loading of object files.

e.g. SYSTEM

TROFF

Turns off the trace.

e.g. TROFF

TRON

Turns on the trace.

e.g. TRON

(2) BASIC Functions

Argument ranges are indicated below by special symbols:

```
x:   (-1x10 38, -1x10 -38), (1x10 -38,1x10 38)
c:   (0,255)
n:   (-32768,32767)
b:   (0,15)
str: string argument
var: variable name
```

ABS(x)

Computes absolute value.

e.g. Y=ABS(X)

ASC(str)

Returns ASCII code of first character of string.

e.g. A=ASC(T$)

ATN(x)

Computes arctangent; value returned in radians.

e.g. Y=ATN(x/3)

CDBL(x)

converts to double-precision.

e.g. X#=CDBL(N*3)

CHR$(c)

Returns character for ASCII, control, or graphics code.

e.g. P$=CHR$(T)

CINT(n)

Returns largest integer not greater than n.

e.g. PRINT CINT(15.0075)

COS(x)

Computes cosine; angle must be in radians.

e.g. Y=COS(X)

CSNG(x)

Converts to single-precision.

e.g. FC=CSNG(TM#)

ERL

Returns the line number in which an error has occurred.

e.g. PRINT ERL

ERR

If an error occurs, returns a value related to the error code: value returned = (error code -1)*2.

e.g. IF ERR = 12 THEN 650 ELSE 800

EXP(x)

Computes natural antilog.

e.g. Y=EXP(X)

FLX(x)

Truncates all digits to right of decimal point.

e.g. Y=FIX(X)

FRE(numeric)

Finds amount of free memory.

e.g. F=FRE(X)

FRE(str)

Returns amount of unused string space. str is any string constant or string variable.

e.g. FRE("C")

INKEY$

Gets keyboard character if available.

e.g. A$=INKEY$

INP(p)

Gets value from specified prot. p=0-255.

e.g. V=INP(255)

INT(x)

Returns largest whole number not greater than x.

e.g. Y=INT(X)

LEFT$(str,c)

Returns left portion of string.

e.g. P$=LEFT$(M$,7)

LEN(str)

Returns the number of characters in a string.

e.g. X=LEN(SEN$)

LOG(x)

Computes natural logarithm.

e.g. Y=LOG(X)

MEM

Finds amount of free memory.

e.g. PRINT MEM

MID$(string, pos, len)

returns a substring of another string. If length option is omitted, the entire string right of pos is returned.

e.g. PRINT MID$(A$3,2)

PEEK(l)

Gets value in location l (l=0 to end of memory).

e.g. V=PEEK(18520)

POINT(x,y)

Tests whether specified graphics block is on or off. x(horizontal) =0-127;y(vertical) =0-47.

e.g. IF POINT (13,35) THEN PRINT "ON" ELSE PRINT "OFF"

POS(x)

Returns column position of cursor (0-63). x is a dummy argument.

e.g. PRINT TAB(40)POS(0)

RIGHT$(str,c)

Returns right portion of string.

e.g. ZIP$=RIGHTS(AD$,5)

RND(n)

Generates a pseudorandom number between 1 and n if n>1, or between 0 and 1 if n=0.

e.g. Y=RND(100)

SGN(x)

Returns sign component: -1,0,1, if x is negative, zero, positive.

e.g. X=SGN(A*B)

SIN(x)

Computes sine; angle must be in radians.

e.g. Y=SIN(X)

SQR(x)

Computes square root.

e.g. Y=SQR(A+B)

STR$(x)

Converts a numeric expression to a string.

e.g. S$=STR$(X)

STRING$(l,c)

Returns string of characters of length l. Character c can be specified as an ASCII code or as a string.

e.g. B$=STRING$(125,"?")

TAN(x)

Computes tangent; angle must be in radians.

e.g. Y=TAN(X)

USR(x)

Calls a machine-language subroutine.

e.g. PRINT USR(-1)

VAL(str)

Evaluates a string as a number.

e.g. V%=VAL("100 DOLLARS")

VARPTR(var)

Gets address where variable contents are stored.

e.g. Y=USR1(VARPTR (X))

(3) BASIC OPERATORS

Each operator or group of operators is precedent over the group below it.

^	Exponentiation (returns single-precision)
-, +	Unary negative, positive
*, /	Multiplication, division
+, -	Addition and concatenation, subtraction
<,>,=,<=,>=,<>	Relational tests
NOT	
AND	
OR	

(4) BASIC SPECIAL CHARACTERS

%	Makes variable integer-precision.
!	Makes variable single-precision.
#	Makes variable double-precision.
$	Makes variable string type.
:	Separates statements on the same line.
?	Same as PRINT (but L? can't be substituted for LPRINT).

(5) BASIC EDIT COMMANDS

A	Cancels changes and starts over.
nC	Changes n characters.
nD	Deletes n characters.
E	Ends editing and saves all changes.
H	Hacks line and inserts at end.
I	Inserts characters.
nKc	Kills all characters up to nth occurrence of c.

L	Lists the line.
Q	Quits edit mode and cancels all changes.
nSc	Searches for nth occurrence of c.
X	Extends line (inserts at end).
SHIFT	Causes escape from command.
ENTER	Records all changes and exits edit mode.
n SPACE BAR	Moves cursor n spaces to the right.
n <-	Moves cursor n spaces to the left.

(6) VIDEO CONTROL CODES

Dec	Hex	PRINT CHR$ (code)
8	08	Backspaces and erases current character.
10	0A	Line feed with carriage return.
13	0D	Line feed with carriage return.
14	0E	Turns on cursor.
15	0F	Turns off cursor.
23	17	Shifts to 32-character mode.
24	18	Backspaces cursor.
25	19	Advances cursor.
26	1A	Downward line feed.
27	1B	Upward line feed.
28	1C	Homes cursor.
29	1D	Moves cursor to beginning of line.
30	1E	Erases to end of line.
31	1F	Clears to end of screen.

(7) BASIC ERROR MESSAGES

Code	Abbreviation	Explanation
1	NF	NEXT without FOR
2	SN	Syntax error
3	RG	RETURN without GOSUB
4	OD	Out of data
5	FC	Illegal function call
6	OV	Overflow
7	OM	Out of memory
8	UL	Undefined line
9	BS	Subscript out of range
10	DD	Redimensioned array
11	/0	Division by zero
12	ID	Illegal direct
13	TM	Type mismatch
14	OS	Out of string space
15	LS	String too long
16	ST	String formula too complex
17	CN	Can't continue
18	NR	No RESUME
19	RW	RESUME without error
20	UE	Underfined error
21	MO	Missing operand
22	FD	Bad file data
23	L3	Disk BASIC feature

(8) COLOUR CODES

When one put colour in a certain location on the screen by programmbale colour the colour codes are as follows (Please note that colour are approximate it will depend on the cour setting of your colour TV):

1) Light Green
2) Red
3) Dark Green
4) Blue
5) Greenish Blue
6) Rose Red
7) Dusty Blue
8) Greenish Yellow
9) Light Yellow (Greenish)
10) Golden Red
11) Grey
12) Redish Green
13) Pale Green
14) Orange
15) Ogre Green
16) Off White

PART IV

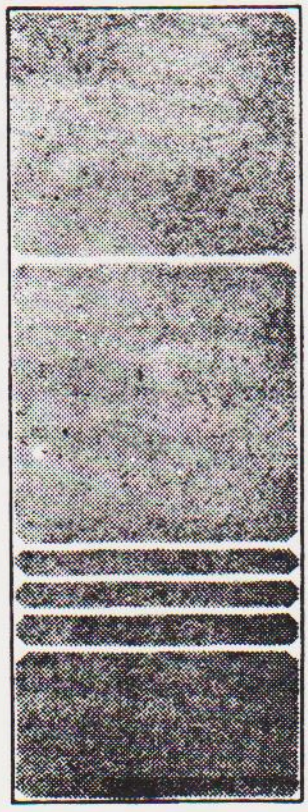
APPENDICES
****

1. MEMORY SIZE AND PROTECTION

When you first switch on the computer and press RESET you will notice that there appear on the screen the words: MEMORY SIZE? Normally when you are doing small programmes, you simply ignore the question by pressing RETURN and proceed to operate.

However when you are doing two types of programmes, for example, a Basic Langauge Programme and a Machine language Programme you would not want one programme which occupies certain memory space (RAM) to overflow into another memory space alloted to the other type of programme. The way to do it is to allot certain amount of memory space for the Basic Programme and the remaining memory space for Machine Language Programme. Before you do this, you need to know how the Komtek Memory is mapped:

MEMORY MAP



To familiarize yourself with this memory map switch on your Komtek 1. If it is fitted with 16K RAM the following will happen:

Press RESET
Press RETURN the following will appear READY
Now type? MEM and press RETURN >
the screen will show 15572 which is the memory size available for usage. This is 15.5K less 300 Bytes, remember that in a computer 1K is actually 1024 Bytes.

Let us suppose that you wish to reserve 4K out of the 15572 for your Machine Language or other reserve programmes then you should enter a total size including the 16K ROM space i.e.
(15572 + 16K) less (Reserve Space)

This turn out to be 28672. Now press RETURN and the screen will show again READY >

Now type 28672.

Then ask about the memory space : Type ?MEM you will get the answer 11475. This is less than the calculated 11476 by 1.

The reason for this is that location 0 in the memory is also a space. so counting from 0 to 11475 you get 11476 spaces.

At this stage you may need to know: what happens if you exceed a certain memory space in doing your programme? The computer will give you an overflow indication, to see how this works, try the following:

(A) Type POKE 32768,0

The above statement means we put in the value of 0 into memory location 32768 Bytes but the location (or address) or the last Byte is actually 32767 because 0 is also a location. So we are one Byte beyond the memory. The computer will simply throw back the problem at you and show:

```
?OV Error
READY
>
```

which means overflow.

(B) Now try typing POKE 32767,0

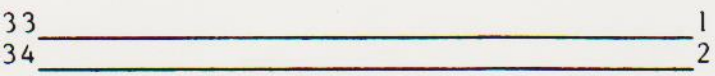
The computer accepts it and show:

```
READY
>
```

Because even though you are at the last memory location, you are still within the memory capacity.

2. CARD-EDGE PIN ASSIGNMENT FOR PARALLEL PRINTER INTERFACE

Pin	Signal	Pin	Signal
1	DATA STROBE	2	GND
3	D0	4	GND
5	D1	6	GND
7	D2	8	GND
9	D3	10	GND
11	D4	12	GND
13	D5	14	GND
15	D6	16	GND
17	D7	18	GND
19	NC	20	GND
21	BUSY	22	GND
23	OUT OF PAPER	24	GND
25	UNIT SELECT	26	NC
27	NC	28	NC
29	NC	30	NC
31	NC	32	NC
33	NC	34	NC



3. CARD-EDGE PIN ASSIGNMENT FOR FLOPPY DISK INTERFACE

(Refer to the component layout diagram)

Pin	Signal	Description
odd pins	GND	odd pins of 1 through 33 all GND
2	NC	
4	NC	
6	SIDE SELECT	To double-sided drives
8	INDEX/SECTOR	Input to FDC, active LOW
10	DS1	(Output from FDC,
12	DS2	(active LOW
14	DS3	(Drive Select 1, 2 and 3
16	MOTOR ON	Output from FDC, active LOW
18	DIR SEL	Output from FDC, step out when HIGH, step in when LOW
20	STEP	Output from FDC, active LOW-to-HIGH
22	WRITE DATA	Output from FDC, low going pulses.
24	WRITE GATE	Output from FDC, write data when LOW, read data when HIGH
26	TRACK 00	Input to FDC, active LOW
28	WRITE PROTECT	Input to FDC, active LOW
30	READ DATA	Input to FDC, low going pulses.
32	DS4	Output from FDC, Drive Select 4, active LOW
34	NC	

33 ——— 1
34 ——— 2

(C) Available Software

- Available in education, business, games & entertainment for as low as US$5.00 per cassette.

- For listing or order of software contact EducAid Centre P.O. Box 98269 T.S.T. Kowloon, Hong Kong. Tlx : 35414 OEMIN HX or your local dealers.

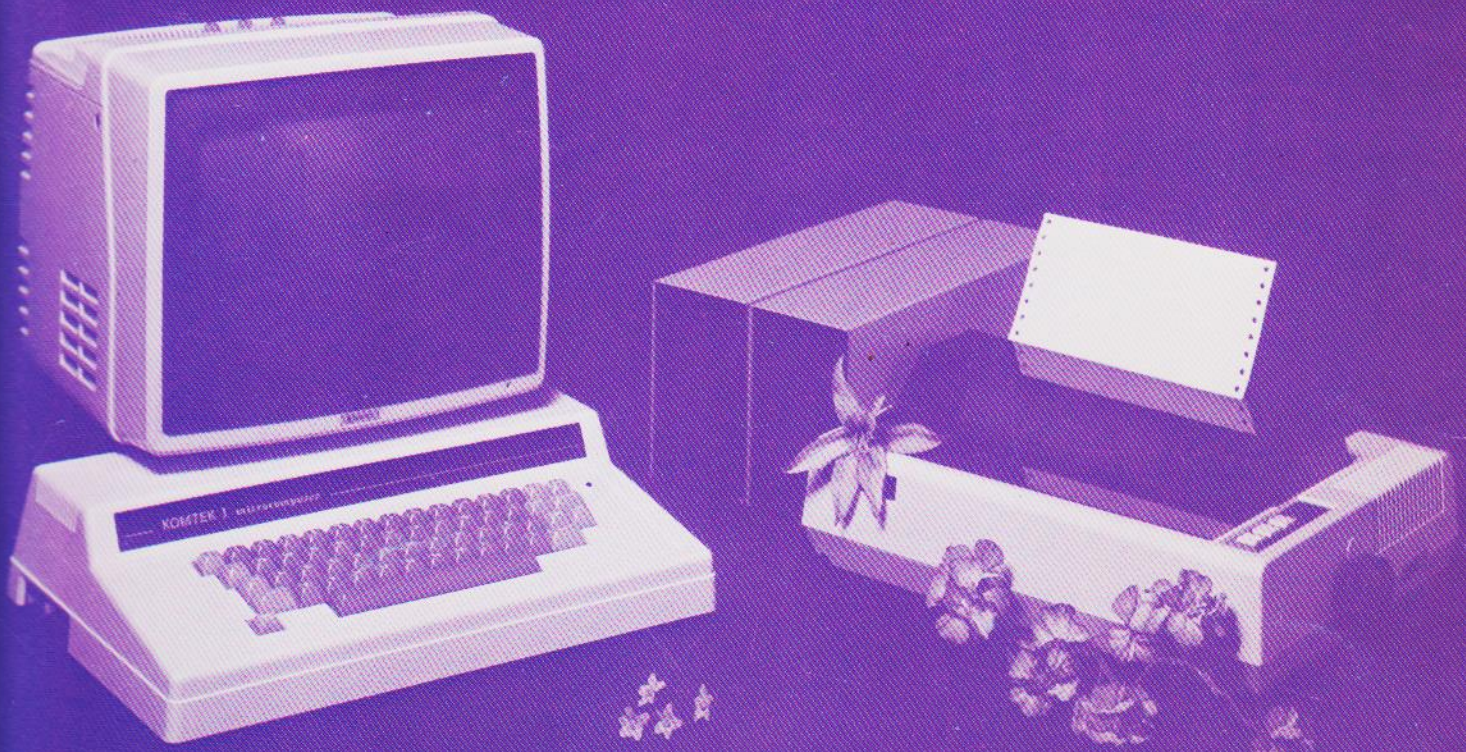
6. SAMPLE PROGRAMS

(1)

```
 1 REM * SPEED READING PROGRAM
 4 GOTO 8
 5 FOR I=1 TO B:NEXTI:CLS:RETURN
 6 REM AUDIO PROMPT GOES BEFORE RETURN IN ABOVE LINE
 8 CLS:PRINT"    SPEED READING"
 10 INPUT"HOW MANY WORDS PER MINUTE DO YOU READ" ;W
 20 B=(12*60/W) * 344
 30 REM 344=FOR/NEXT LO*PS IN ONE SECOND
 40 CLS
 100 PRINT@448,"   HE  WATCHED HIS RIGHT HAND  PICK  UP  THE
     KNIFE." :GOSUB5
 102 PRINT@448,"THEN  HIS LEFT ALSO GRASPED  THE  HILT,  THE
     BLADE STEADY":GOSUB5
 105 PRINT@448,"AND POINTING AT HIS HEART. NOW THERE WAS ONLY
     THE":GOSUB5
 107 PRINT@448,"SOUND  OF HIS LIFE,  BUILDING AND  BUILDING,
     SOARING":GOSUB5
 110 PRINT@448,"LOUDER AND LOUDER UNTIL HE COULD LISTEN NO
     MORE.":GOSUB5
 120 PRINT@448,"HIS SOUL CRIED OUT FOR ETERNAL SILENCE.  THE
     CRY":GOSUB5
 130 PRINT@448,"TRIGGERED HIS REFLEXES.  HIS HANDS DROVE THE
     KNIFE":GOSUB5
 140 PRINT@448,"UNERRINGLY TOWARDS ITS TARGET.  OMI HAS  BEEN
     READY":GOSUB5
 150 PRINT@448,"TO  STOP HIM BUT HE WAS UNPREPARED  FOR  THE
     SUDDENNESS":GOSUB5
 160 PRINT@448,"AND FEROCITY OF BLACKTHORNE'S THRUST,  AND AS
     OMI'S":GOSUB5
 170 PRINT@448,"LEFT HAND CAUGHT THE BLADE AND HIS RIGHT THE
     HAFT,":GOSUB5
 180 PRINT@448,"PAIN BIT INTO HIM AND BLOOD SPILLED FROM  HIS
     LEFT HAND.":GOSUB5
```

(2)
```
10 PRINT "METRIC CONVERSION PROGRAM"
20 PRINT
30 REM - ESTABLISH VARIABLES FOR 17 CONVERSION FACTORS
40 DIM C(17)
50 REM - LOOP TO ASSIGN CONVERSION FACTORS INTO C( )
60 FOR N=1 TO 17
70 READ C(N)
80 NEXT N
90 REM - DATA TABLE OF SEVENTEEN CONVERSION FACTORS
100 DATA 0.3937,3.281E - 2,3.281,1.094,0.6214,.01136,.
    003785,.23366,2.113
110 DATA   1.057,0.2642,0.02838,2.104,0.3527,2.205,9.842E-
    4,0.6214,0
120 REM - GET NUMBER OF CONVERSION FROM PROGRAM DESCRIPTION
130 PRINT "(TO END PROGRAM ENTER 0)"
140 PRINT "WHICH CONVERSION DO YOU NEED (1 TO 17)"
150 PRINT  "1.    INCHES  TO  CENTIMETERS",  "2.   FEET  TO
    CENTIMETERS"
160 PRINT "3.  FEET TO  METERS", "4.  YARDS TO METERS"
170 PRINT  "5.   MILES TO  KILOMETERS",  "6.   TEASPOON  TO
    MILLILITERS"
180 PRINT "7.    TABLESOPPON TO MILLILITERS",  "8.   CUPS TO
    LITERS"
190 PRINT "9.  PINTS TO LITERS", "10.  QUARTS TO LITERS"
200 PRINT "11.  GALLONS TO LITERS", "12.  BUSHELS TO LITERS"
210 PRINT "13.  PECKS TO LITERS", "14.  OUNCES TO GRAMS"
220 PRINT  "15.    POUNDS  TO  KILOGRAMS",  "16.   TONS  TO
    KILOGRAMS"
230 PRINT "17.  DEGREES FAHRENHEIT TO CELSIUS"
240 INPUT N
250 REM - END PROGRAM?
260 IF N=0 THEN 730
270 REM - CONVERSION AVAILABLE?
280 IF N>17 THEN 140
290 PRINT "VALUE TO BE CONVERTED";
300 INPUT I
310 REM - PERFORM CONVERSION USING PROPER CONVERSION FACTOR
320 R=I/C(N)
330 REM - DIRECT PROGRAM TO PROPER CONVERSION  UNITS,  PRINT
    RESULTS
340 ON N GOTO 350,370,390,410,430,450,470,490,510,530,550,
    570,590,610,630,650,680
350 PRINT I;"INCHES =";R;"CENTIMENTERS"
360 GOTO 700
370 PRINT I;"FEET =";R;"CENTIMETERS"
380 GOTO 700
390 PRINT I;"FEET =";R;"METERS"
400 GOTO 700
410 PRINT I;"YARDS =";R;"METERS"
420 GOTO 700
430 PRINT I;"MILES =";R;"KILOMETERS"
440 GOTO 700
```

```
450 PRINT I;"TSP. =";R;"CUBIC CENTIMETERS"
460 GOTO 700
470 PRINT I;"TBSP. =";R;"CUBIC CENTIMETERS"
480 GOTO 700
490 PRINT I;"CUPS =";R;"LITERS"
500 GOTO 700
510 PRINT I;"PINTS =";R;"LITERS"
520 GOTO 700
530 PRINT I;"QUARTS =";R;"LITERS"
540 GOTO 700
550 PRINT I;"GALLONS =";R;"LITERS"
560 GOTO 700
570 PRINT I;"BUSHELS =";R;"LITERS"
580 GOTO 700
590 PRINT I;"PECKS =";R;"LITERS"
600 GOTO 700
610 PRINT I;"OUNCES =";R;"GRAMS"
620 GOTO 700
630 PRINT I;"POUNDS =";R;"KILOGRAMS"
640 GOTO 700
650 PRINT I;"TONS =";R;"KILOGRAM"
660 GOTO 700
670 REM - CONVERT FROM DEGREES FAHRENHEIT TO CELSIUS
680 R=(I-32)*5/9
690 PRINT I;"DEGRESS FAHRENHEIT =";R;"CELSIUS"
700 PRINT
710 REM - RESTART PROGRAM
720 GOTO 140
730 END
```


КОМПЕК 1