TRS-80 Pocket Computer

Catalog Number 26-3501

Radio Shack®
TRS-80
MICRO
COMPUTER
SYSTEM

SOFTWARE

CUSTOM MANUFACTURED FOR RADIO SHACK A DIVISION OF TANDY CORPORATION

TABLE OF CONTENTS

Page

- INTRODUCTION 1
- TABLE OF FUNCTIONS AND STATEMENTS 3
- KEYBOARD 8
- FUNCTIONS OF KEYS 9
- SECOND FUNCTION 13
- MODE 13
- DISPLAY 14
- DISPLAY SYSTEM 16
- INPUTTING DATA 17
- COMPUTATION RANGE 17
- MANUAL CALCULATIONS 18
  1. What is a manual calculation? 18
  2. For arithmetic calculations 18
  3. Power calculations 20
  4. Calculations with parentheses 20
  5. Scientific functions 21
  6. Logic functions 24
  7. Calculations using memories 25
  8. Successive designation of expressions in manual calculation 25
  9. Recall function 26
  10. Editing expressions 26
  11. Priority of calculations 29
- PROGRAMMED CALCULATIONS 31
  1. What is a programmed calculation? 32
  2. Writing programs 34
  3. Checking stored programs 36
  4. Program correction 37
  5. Executing programs 39
  6. DEBUGing programs 41
  7. Defined programs 42
- VARIABLES 45
  1. What is a variable? 45
  2. Specifying variables 46
  3. Inputting to variables 48
  4. Recalling the contents of variables 49
- PROGRAM STATEMENTS 51
  1. LET statement 51
  2. INPUT statement 52
  3. PRINT statement 54
  4. PAUSE statement 57
  5. USING statement 57
  6. GOTO statement 59
  7. IF statement 60
  8. GOSUB statement, RETURN statement 62

9. FOR statement, NEXT statement. 64
10. STOP statement 68
11. END statement. 68
12. BEEP statement 68
13. CLEAR statement. 68
14. DEGREE, RADIAN, GRAD statements. 69
14. AREAD statement. 69
16. REM statement. 70
● COMMAND STATEMENTS. 70
1. RUN command. 70
2. DEBUG command. 71
3. CONT command 71
4. LIST command. 72
5. NEW command. 73
6. MEM command. 73
● STATEMENTS FOR OPTIONAL CASSETTE INTERFACE (Cat. No. 26-3503). 74
1. CSAVE (Cassette Save) statement. 74
2. CLOAD (Cassette Load) statement. 75
3. CLOAD? (Cassette Load?) statement 75
4. CHAIN statement 76
5. PRINT # (Print cross-hatch) statement 78
6. INPUT # (Input cross-hatch) statement. 79
● RESERVABLE KEY. 80
1. Reserve memory for reservable keys 80
2. Use of reservable keys 81
3. Checking reserve programs 83
4. Correction of reserve programs. 83
5. Deleting reserve programs 84
6. Configuration of reserve programs 84
● ERROR CODES. 85
● APPENDIX 87
● SPECIFICATIONS 88
● BATTERY REPLACEMENT 89
● CONNECTING THE CASSETTE INTERFACE 90
Replacing the Batteries. 90
Connecting the Pocket Computer to the Cassette Interface 91
Connecting the Cassette Interface to a Tape Recorder 92
● OPERATING THE CASSETTE INTERFACE AND RECORDER. 93
Recording onto Magnetic Tape. 93
Loading from a Magnetic Tape. 93
Editing Programs on Magnetic Tape 95
CLOAD 1 statement 96
Checking the Program 98
Execution of the Program. 99
● SOME SAMPLE USER'S PROGRAMS. 101
Notes for entering program listings 102
Biorhythm. 103
Guess Number Game 105
Impedance in a Series Circuit. 107
Days between Dates. 109
Random Numbers 111
Normal Distribution and Percentile. 113
● INDEX 119

INTRODUCTION

This new TRS-80 Computer is another "first" from the company which brought you the best-selling, world renowned TRS-80. A truly pocket-sized Computer (not a programmable calculator). Of course it is an ultra-powerful calculator too . . . And it "speaks" BASIC - - the most common computer language, and the easiest to learn. You'll soon be impressed by the phenomenal computing power of this hand-held TRS-80 - - ideal for mathematical, scientific, engineering and business applications.

FEATURES

- Programmable, with BASIC language.
- 24-digit alphanumeric dot matrix Liquid Crystal Display, enables easy use of BASIC language, or standard calculator function.
- Program capacity 1424 steps, 26 memories with memory safe guard.
- Reservable and definable key systems. (See page 80 and 42)
- When used with an optional Cassette Interface (26-3503), you can store or recall programs and data on a cassette tape. (See page 90)

Some Special Notes

Since the Liquid Crystal Display is made of glass material, treat the Computer with care. Do not put your Pocket Computer in your back pocket - - you may sit on it and break the LCD display.

To insure trouble-free operation:

1. The Computer should be kept in areas free from extreme temperature changes, moisture and dust.
2. Use a soft, dry cloth to clean the Computer. Do not use solvents or a wet cloth.
3. If you're not going to use the Computer for an extended period of time, remove the batteries to avoid possible damage caused by battery leakage.
4. If service is required, use only an authorized Radio Shack Service Center.
5. Keep this Manual for further reference.

Name label
Write your name on the attached name label and stick it on the back of the Computer.

For your own protection and security, we urge you to record the Serial Number of this unit in the space provided. You'll find the Serial Number on the bottom of the Computer.

Serial Number: ____________________

Where We Are Going . . .

Since this is such a radically new product and most people won't know how or where to start we thought it might be helpful to tell you where we are going with the rest of this Manual.

But first – where were coming from! This manual is written assuming at least a little familiarity with BASIC. You don't have to have hands-on experience, just be familiar with simple concepts of programming and BASIC. If you are looking for a lead-'em-by-the-hand Manual, this is not it. For that approach, stop by your Radio Shack store or Computer Center and take a look at some of our books. Two or three of them start from scratch.

Back to this Manual.

First were going to give you an over-view of the Computer
- Keyboard
- Functions
- Display

Then show you how to use the Computer
- Manual Calculations
- Programmed Calculations

And then you'll be ready for some
- Programming in BASIC

The back of the Manual has some vital information in the Appendices.

A separate Quick Reference Card has all the information you need for using your TRS-80 Pocket Computer, but in an extremely abbreviated form.

TABLE OF FUNCTIONS AND STATEMENTS

This Table will provide a quick reference for the BASIC Language functions as used by the Pocket Computer. The Page reference shows the page on which the Function/Statement is discussed.
You can use abbreviations for the Functions and statements as noted.

1. Functions

Remember to press the $\boxed{\cdot}$ key.

Functions	Abbreviations	Remarks	Ref. page
SIN	SI.	sin	21
COS		cos — Trigonometric functions	
TAN	TA.	tan	
ASN	AS.	$\sin^{-1}$	21
ACS	AC.	$\cos^{-1}$ — Inverse trigonometric functions	
ATN	AT.	$\tan^{-1}$	
LN		$\log_e X$ Natural logarithm — Logarithmic functions	22
LOG	LO.	$\log_{10} X$ Common logarithm — Logarithmic functions	
EXP	EX.	e^x Exponential function (Antilogarithm for LN)	22
$\sqrt{\ }$		Extraction of square root	22
DMS	DM.	Decimal to degree/minute/second conversion	22
DEG		Degree/minute/second to decimal conversion	22
INT		Integer	23
ABS	AB.	Absolute value	23
SGN	SG.	Signum	23

2. Statements

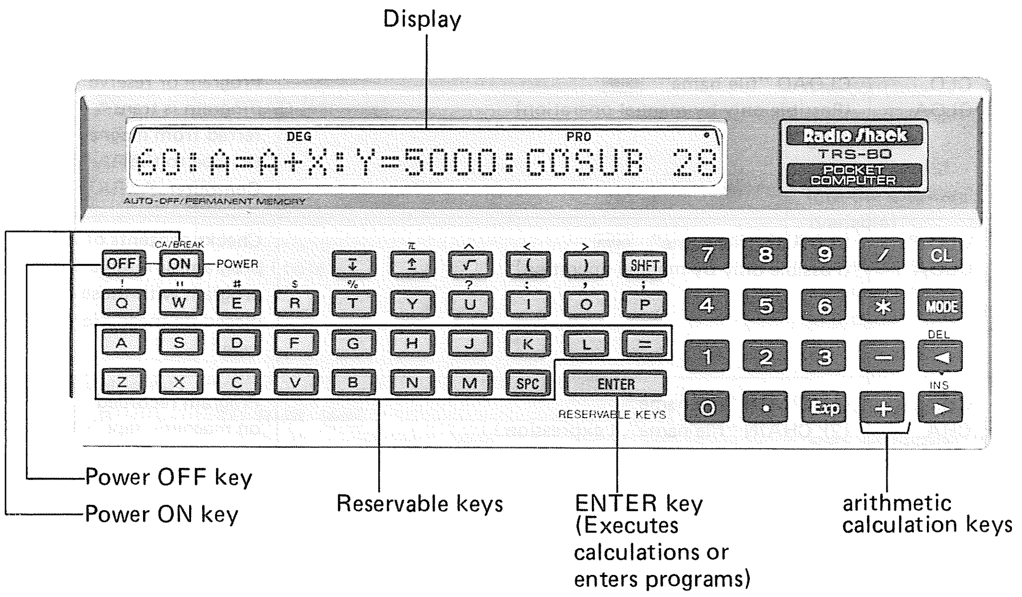
Statements	Abbreviations	General forms	Remarks	Ref. page
LET (assignment statement)	LE.	(1) LET [numerical variable] = ⟨ expression ⟩ (2) LET [Character variable] = "character" (3) LET [Character variable] = [Character variable]	LET can be omitted (except when following an IF statement).	51
INPUT	I. IN. INP. INPU.	(1) INPUT [variable], [variable], · · · (2) INPUT "character", [variable], "character", [variable], · · · (3) INPUT "character"; [variable], "character"; [variable], · · ·	Input instruction Data is input.	52
PRINT	P. PR. PRI. PRIN.	(1) PRINT ⟨ expression ⟩ (2) PRINT "character" (3) PRINT [Character variable] (4) PRINT $\left\{\begin{matrix}\text{⟨ expression ⟩}\\\text{"character"}\\\text{[Character variable]}\end{matrix}\right\}, \left\{\begin{matrix}\text{⟨ expression ⟩}\\\text{"character"}\\\text{[Character variable]}\end{matrix}\right\}$ (5) PRINT $\left\{\begin{matrix}\text{⟨ expression ⟩}\\\text{"character"}\\\text{[Character variable]}\end{matrix}\right\}; \left\{\begin{matrix}\text{"character"}\\\text{[Character variable]}\end{matrix}\right\};$ · · · $\left\{\begin{matrix}\text{"character"}\\\text{[Character variable]}\end{matrix}\right\}$	Output instruction. Specified contents are displayed.	54
PAUSE	PA. PAU. PAUS.	General forms are the same as those for PRINT statement.	Output instruction. Specified contents are programmed after being displayed for about 0.85 second.	57
USING	U. US. USI. USIN.	(1) USING "♯…♯, ♯ …♯ ∧" (2) (a) $\left\{\begin{matrix}\text{PRINT}\\\text{PAUSE}\end{matrix}\right\}$ USING "FORMAT", · · · (b) $\left\{\begin{matrix}\text{PRINT}\\\text{PAUSE}\end{matrix}\right\}$ USING; · · · (3) USING (end of statement) ENTER or : (colon)	Format designation instruction. Displaying format for numerical data is designated. Format designation is cancelled.	57
GOTO	G. GO. GOT.	(1) GOTO ⟨ expression ⟩ (2) GOTO $\left\{\begin{matrix}\text{"character"}\\\text{[Character variable]}\end{matrix}\right\}$	Jump instruction. Specified line or label is executed.	59
IF		(1) IF ⟨ expression ⟩ logic operator ⟨ expression ⟩ execution statement (2) IF ⟨ expression ⟩ excution statement (3) IF $\left\{\begin{matrix}\text{"character"}\\\text{[Character variable]}\end{matrix}\right\} = \left\{\begin{matrix}\text{"character"}\\\text{[Character variable]}\end{matrix}\right\}$ execution statement (4) IF [Character variable] execute statement	Decision instruction. Based on conditions the program branches or continues execution.	60
THEN	T. TH. THE.	This statement is defined as a execution statement in an IF statement. General form is the same as that of GOTO statement.	Jump instruction. Used only with an IF statement.	61

Statements	Abbreviations	General forms	Remarks	Ref. page
GOSUB	GOS. GOSU.	(1) GOSUB 〈 expression 〉 (2) GOSUB { "character" / [Character variable] }	Subroutine jump instruction. Execution is shifted to specified line or label, where subroutine is executed.	62
RETURN	RE. RET. RETU. RETUR.	RETURN	Return instruction. Used after execution of a GOSUB (at end of subroutine) to return execution to main program.	62
FOR STEP	F. FO. STE.	(1) FOR [numerical variable] = 〈 expression 1 〉 TO 〈 expression 2 〉 (2) FOR [numerical variable] = 〈 expression 1 〉 TO 〈 expression 2 〉 STEP 〈 expression 3 〉 〈 expression 1 〉: Initial value 〈 expression 2 〉: End value 〈 expression 3 〉: Increment	Starts FOR loop. Used in combination with NEXT statement.	64
NEXT	N. NE. NEX.	NEXT [numerical variable] This [numerical variable] must correspond to that for FOR statement.	Ends FOR loop. Used in combination with FOR statement.	64
STOP	S. ST. STO.	STOP	To stop executing program.	68
END	E. EN.	END	To indicate program end.	68
BEEP	B. BE. BEE.	BEEP 〈 expression 〉	Beep sound instruction Beep tone is generated as many times as the number of value in 〈 expression 〉.	68
CLEAR	CL. CLE. CLEA.	CLEAR (Possible to execute by manual operation) CLEAR ENTER	Data memory clear instruction	68
DEGREE	DEG. DEGR. DEGRE.	DEGREE (Possible to execute by manual operation) DEGREE ENTER	Angular mode designation. Degree (°) is designated.	69
RADIAN	RA. RAD. RADI. RADIA.	RADIAN (Possible to execute by manual operation) RADIAN ENTER	Angular mode designation. Radian ([rad]) is designated.	69

Statements	Abbreviations	General forms	Remarks	Ref. page
GRAD	GR. GRA.	GRAD (Possible to execute by manual operation) GRAD $\boxed{\text{ENTER}}$	Angular mode designation. Grad ([g]) is designated.	69
AREAD (auto read)	A. AR. ARE. AREA.	AREAD [variable]	The contents displayed at start of definable program is read into the specified [variable].	69
REM (remark)		REM ⟨ note ⟩	To designate non-execute statement in program (notes).	70
⟨ **Command statement** ⟩		Possible only to execute by manual operation.		
RUN	R. RU.	(1) RUN $\boxed{\text{ENTER}}$ (2) RUN ⟨ expression ⟩ $\boxed{\text{ENTER}}$ (3) RUN $\left\{ \begin{array}{l} \text{"character"} \\ \text{[Character variable]} \end{array} \right\}$ $\boxed{\text{ENTER}}$	Program execute start instruction. Effective only in DEF and RUN modes.	70
DEBUG	D. DE. DEB. DEBU.	The general forms are defined in the same manner as those for RUN statement.	Debugging start instruction. Effective only in DEF and RUN modes.	71
CONT	C. CO. CON.	CONT $\boxed{\text{ENTER}}$	To restart an interrupted program. Effective in DEF and RUN modes.	71
LIST	L. LI. LIS.	The general forms are defined in the same manner as those for RUN statement.	For listing programs. Effective in PRO mode.	72
NEW		NEW $\boxed{\text{ENTER}}$	In DEF, RUN and PRO modes, program memory and data memory are completely cleared. In RESERVE mode, reserve memory is cleared.	73
MEM	M. ME.	MEM $\boxed{\text{ENTER}}$	Remaining area of program memory is displayed (number of program steps and flexible memories).	73

Statements	Abbreviations	General forms	Remarks	Ref. page
〈 **Magnetic tape control statement** 〉				
CSAVE (cassette save)	CS. CSA. CSAV.	CSAVE "file name" $\boxed{\text{ENTER}}$ (Possible only by manual operation)	Program or reserve program is recorded on magnetic tape.	74
CLOAD (cassette load)	CLO. CLOA.	CLOAD "file name" $\boxed{\text{ENTER}}$ (Possible only by manual operation)	Program or reserve program is transferred from magnetic tape to the Computer.	75
CLOAD? (cassette load?)	CLO. ? CLOA. ?	CLOAD? "file name" $\boxed{\text{ENTER}}$ (Possible only by manual operation)	Checks contents of program or reserve program with those placed on magnetic tape.	75
CHAIN	CH. CHA. CHAI.	(1) CHAIN "file name" (2) CHAIN "file name", 〈 expression 〉 (3) CHAIN "file name" $\left\{\begin{array}{l}\text{"character"}\\ \text{[Character variable]}\end{array}\right\}$ (To be executed by program)	Program recorded on magnetic tape is read in and then executed.	76
PRINT #	P. # PR. # PRI. # PRIN. #	(1) PRINT # "file name" (2) PRINT # "file name"; [Label of variable] (Possible to execute both by program and manual operation)	Data memory contents are recorded on magnetic tape.	78
INPUT #	I. # IN. # INP. # INPU. #	(1) INPUT # "file name" (2) INPUT # "file name" ; [Label of variable] (Possible to execute both by program and manual operation)	Data recorded on magnetic tape is transferred into data memory of the Computer.	79

KEYBOARD


Display
60:A=A+X:Y=5000:GOSUB 28
Power OFF key
Power ON key
Reservable keys
ENTER key (Executes calculations or enters programs)
arithmetic calculation keys

FUNCTIONS OF KEYS

Here is a brief explanation of the main keys. For details, refer to the rest of this Manual.

Key	Function
CA/BREAK ON	● Use to power-on. ● Breaking (temporarily interrupting) the program being executed. ● Clearing the Computer completely. (Reset after error condition.)
OFF	● Press to power-off.
SHFT	● Secondary functions noted above the keys (such as π and $\wedge$ symbols) are activated. To obtain pi, press following sequence: SHFT $\overset{\pi}{\uparrow}$ $\rightarrow \pi$ will be displayed. Display shows a SHFT at left when a shift function is pending. ● In "DEF" mode, press before keying in the predefined function labeled as A, S, D, etc. (Definable key designation) Example, SHFT A ● In "RESERVE" mode, press before activating a key used for labeling a reserve program. (Reserve key designation) Example, SHFT B ● In "PRO" or "RUN" mode, press before activating a key used for labeling a reserve program. (Reserve key designation) Example, SHFT B
0 ~ 9	● Use to enter numbers.
•	● Enters a decimal point. ● Use to designate abbreviations when inputting instructions. ● Use to designate a display format in a USING statement instruction. (See page 57.)
Exp	● Use to input exponents. (This key function is displayed as E.)
A ~ Z	● These alphabetical keys serve to designate instructions. ● Specify variables (A to Z memory)
/	● Use for division instructions.
*	● Use for multiplication instructions.
+	● Use to input a positive sign for numbers. (Usually omitted.) ● Use for addition instructions.
−	● Use to input a negative sign for numbers. ● Use for subtraction instructions.
SHFT $\wedge$	● Use for power calculation instructions. ● Use to specify the floating decimal point system (exponent display) for numerical data in USING statement instructions.
SHFT <	● Use when inputting logical operators, such as $<, <=, <>$.

Key	Function
SHFT >	● Use when inputting logical operators, such as >, >=, <>.
=	● In assignment statements, use to assign the content (number or character) on the right for the variable specified on the left. ● Use when inputting logical operators, such as =, <=, >=.
(,)	● Use to input parentheses.
√	● Use to extract square root.
SPC	● Use to provide space when inputting programs or characters. The space is ignored in programming, executing operations, etc.
SHFT :	● Use to divide two or more statements in one line.
SHFT ;	● Use with PRINT statement instructions, to provide multi-display (two or more values/contents displayed at a time). ● Use with INPUT statement instructions, to provide pauses in comment. ● Use with PRINT # statement and INPUT # statement instructions to provide pause between the instruction and the variable.
SHFT ,	● Use to provide pause between two equations in continuous calculation sequences. ● Use with PRINT statement instructions, to provide dual display (two different values/contents are displayed at a time). ● Use with INPUT statement instructions, to provide pause between comments or variables. ● Use with CHAIN statement, to provide pause between file and expression, or between file and label when setting the opening line subsequent to execution.
SHFT #	● Use with USING statement, to provide the instruction to define the display format of numerical data. ● Use with PRINT # and INPUT # statements.
SHFT ?	● Use with CLOAD? statement.
SHFT $	● Use when assigning character variables.
SHFT "	● Use to designate and cancel characters. ● Use to specify labels.
MODE	● Use to change modes (DEF, RUN, PRO, RESERVE).
CL	● Use to clear incorrect manual input. ● Use as an instruction to clear the display contents (such as calculation results). ● Use to reset after error.
►	● Shifts the cursor to the right (press once to advance one position, hold down for automatic advance) ● Executes playback instructions. ● Recalls cursor (in case it is not displayed during program operation; recalls to right of colon)

Key	Function
◄	● Shifts cursor to the left. ● For other functions, the same as the ► key.
SHFT INS	● Inserts one space (⊏ appears) of 1-step capacity between the address (N) indicated by the cursor and the preceding address (N–1).
SHFT DEL	● Deletes the contents of the address (N) indicated by the cursor.
ENTER	● Enters a program line into the Computer. ● Use when writing in programs or reserve programs (function as above). ● Requests manual calculation or direct execution of a COMMAND statement by the Computer. ● Enters a restart instruction after inputting data required by an INPUT statement or after executing a PRINT statement.

The ↥ , ↧ and ON keys have the following functions, depending on designated modes, as well as the state of the Computer.

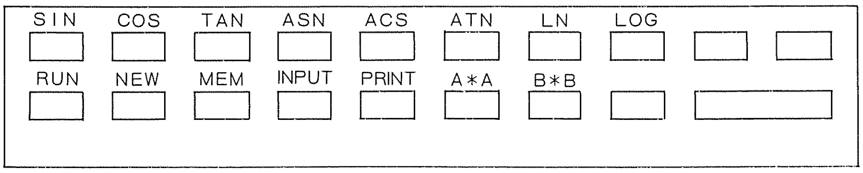
Mode	State	↥	↧	ON
	Power off			To power-on
RUN or DEF	Program being executed			BREAKs (program is temporarily interrupted)
	INPUT statement being executed	To display program line being executed or already executed, hold this key down	To execute debugging operation	
	PRINT statement just now executed			
	Under BREAK		To execute the next line	To clear completely
	Error condition during executing program	To display error-producing line, hold this key down.		
PRO (When program line is not being displayed; e.g. such as when changing to PRO mode)				
	PRINT statement just now executed	To display the interrupted line	Same as left	To clear completely
	Under BREAK			
	Error has been cleared with any key other than the ON key	To display program line in which the error occurred	Same as left	

Mode	State	$\boxed{\uparrow}$	$\boxed{\downarrow}$	$\boxed{\text{ON}}$
PRO (When program line is being displayed)				
		To display the preceeding program line	To display the next program line	To clear completely
RESERVE				

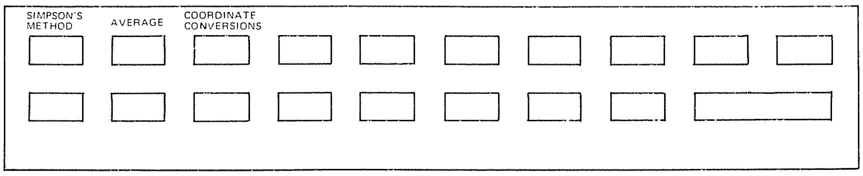
- When a letter or symbol is used in quotations (" "), it is to be considered as a character (input or displayed that way).
- When the A, S, D, F, G, H, J, K, L, =, Z, X, C, V, B, N, M, or SPC keys are pressed, following the SHFT key,
  1. In DEF mode, a program defined with the label of the same character begins execution.
  2. In RESERVE mode, a reserve program is recalled or written in.
  3. In PRO or RUN mode, the contents reserved by the key is recalled. If nothing is reserved, the symbol of the key is displayed.
- The $\boxed{\text{OFF}}$ key does not function when the Computer is executing a calculation or program.
- If no key entry is made for about 7 minutes the power is automatically turned off (unless a program operation is pending, etc.).

Two templates are supplied with your Computer. Use them to identify the functional operation assigned to the reservable keys or defined programs assigned to the definition keys.

Example: Reserved keys (For reservable key information: refer to page 80.)



Example: Program-defined keys (For program-key information defined see page 42.)



SECOND FUNCTION

You must use the yellow SHFT key to operate the functions printed above each key. When you press this key, SHFT will appear in the display. If you press this key in error, press it a second time and SHFT will disappear.

Example: $\boxed{\text{SHFT}}\ \overset{\wedge}{\boxed{\sqrt{\ }}} \rightarrow$ "∧" is entered.

> In this Manual, we'll always show the keys' second functions as follows;
> $\boxed{\text{SHFT}}\ \overset{\wedge}{\boxed{\sqrt{\ }}} \rightarrow \boxed{\text{SHFT}}\ \boxed{\wedge}$

MODE

Your TRS-80 Pocket Computer has four modes: DEFinable, RUN, PROgram and RESERVE program. Set mode by pressing the $\boxed{\text{MODE}}$ key.

Definable mode (DEF):	The defined program execution mode. Perform defined program calculations with this mode.
Run mode (RUN):	The calculation execution mode. Perform program or manual calculations with this mode.
Program mode (PRO):	The program writing mode. Enter programs when in this mode.
Reserve program mode (RESERVE):	The reserve program writing mode. Enter reserve programs when in this mode.

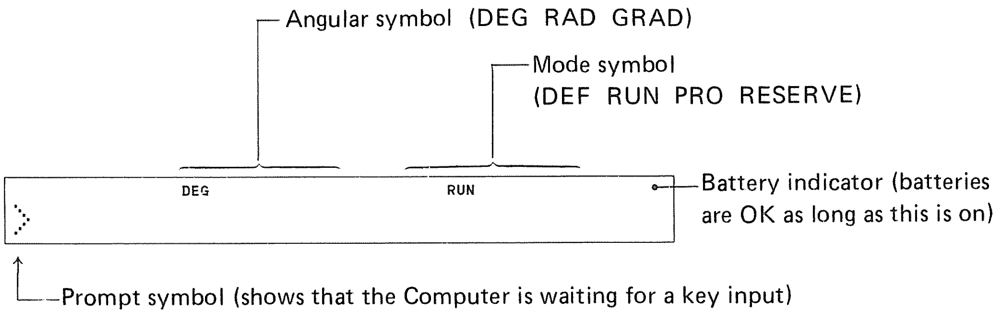
The $\boxed{\text{MODE}}$ key changes the mode in the following sequence:

$$\begin{tikzcd}
\text{DEF} \arrow[r] & \text{RUN} \arrow[r] & \text{PRO} \arrow[r] & \text{RESERVE} \arrow[lll, bend left, "\text{Press } \boxed{\text{MODE}}"]
\end{tikzcd}$$

DISPLAY

Your Pocket Computer has a 24-digit dot matrix liquid crystal display.

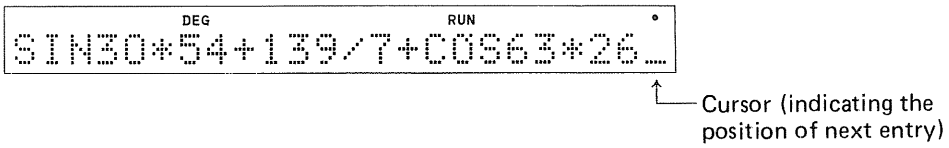
Display when the power is on or when the mode is changed:


Angular symbol (DEG RAD GRAD)
Mode symbol (DEF RUN PRO RESERVE)
DEG
RUN
>
Battery indicator (batteries are OK as long as this is on)
Prompt symbol (shows that the Computer is waiting for a key input)

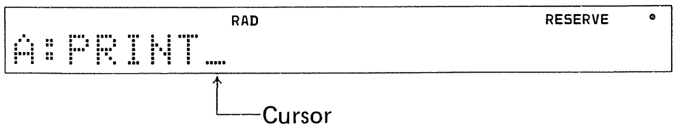
- When you turn on the power the first time after battery replacement, the prompt symbol, DEG and RUN will appear.
- When you switch it on other times, the Computer will display the prompt symbol, along with the angular symbol and mode symbol which was last displayed (just before power was turned off, either with OFF or by automatic power-off).
- When you change the Computer program mode (by pressing MODE key), the Computer will display the prompt symbol, then the existing angular symbol and the new mode symbol.
- To change the angular mode (for trigonometric functions), enter the name of the mode you want with the alpha keys. E.g. D E G R E E ENTER for DEG, R A D I A N ENTER for RAD and G R A D ENTER for GRAD mode.

Display when you input an "expression" etc. with the keys.

1. RUN mode:



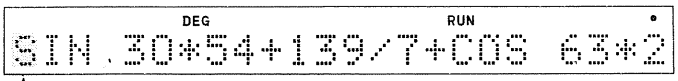
2. RESERVE mode:



- If you input more than 24 characters, the display "rolls" over to the left to provide a space to display the new input. (A maximum of 80 characters can be entered per line. The characters which disappear to the left are not "lost", just not being displayed.)

Display of recalled information

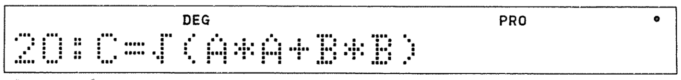
1. RUN mode:



```
DEG                RUN             •
SIN 30*54+139/7+COS 63*2
```

Cursor (If the position indicated by the cursor has a character, a block will alternately flash with the character in that position.)

2. PRO mode:

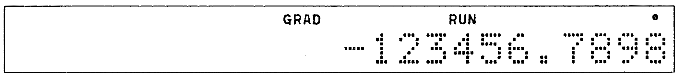


```
DEG                     PRO        •
20:C=√(A*A+B*B)
```

Line number (Displays the line number of the program. Refer to page 31)

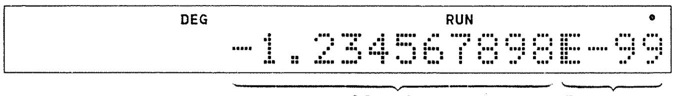
Display of calculation result

1. Normal Display



```
          GRAD       RUN           •
                   -123456.7898
```

2. Scientific notation



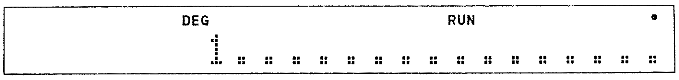
```
DEG                RUN             •
      -1.23456789E-99
```

$\underbrace{\qquad\qquad}_{\text{Mantissa}}\ \underbrace{\qquad}_{\text{Exponent}}$

- Calculation results are always displayed at the right.

Display of Error condition

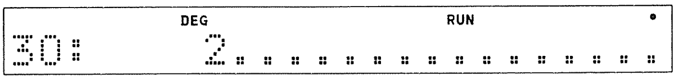
1. Manual calculation



```
DEG                RUN             •
     1...............
```

Error code

2. Programmed calculation

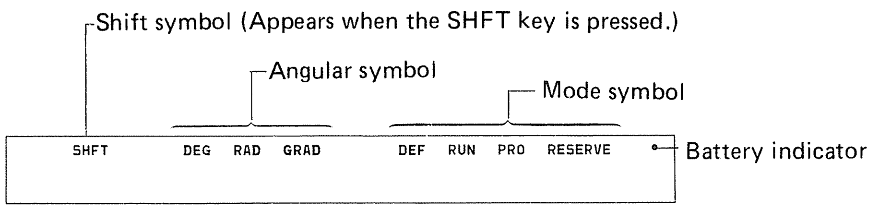


```
DEG                RUN             •
30:      2...............
```

Error code

Line number (Displays the line number in which an error is detected.)

Display of Symbols



Angular symbols

DEG:	Appears when DEGree mode is set.
RAD:	Appears when RADian mode is set.
GRAD:	Appears when GRAD mode is set.

Mode symbols

DEF:	Appears when the DEFinable mode is set.
RUN:	Appears when the RUN mode is set.
PRO:	Appears when the PROgram mode is set.
RESERVE:	Appears when the RESERVE mode is set.

● Battery indicator
The battery indicator is a dot located in the upper right corner of the display. When this dot dissappears, the batteries must be replaced. See page 89.

Number of input characters

When you enter numbers, characters and instructions into the Computer (via the keyboard) this data is stored in an input buffer. When you press the ENTER key, the Computer executes the instructions as required.
The input buffer can hold up to a maximum of 80 characters. When you have entered 80 characters into the input buffer (i.e. 80 characters on a "single line") the cursor will flash in the last display position. Further inputs will merely change this last position.
A manual calculation will not be correctly executed if it contains more than 80 characters (including ENTER key).

DISPLAY SYSTEM

This Computer displays a number in the normal manner or with scientific notation system. Numbers in programmed calculations are displayed according to the designated format, but in manual calculations, numbers within the following range are displayed in the normal manner, and other numbers are displayed in scientific notation.

Range of numbers displayed in the normal manner:

$$-9999999999 \leqq x \leqq -1 \times 10^{-9}$$
$$x = 0$$
$$1 \times 10^{-9} \leqq x \leqq 9999999999$$

● Within the range shown above, if a number can not be displayed in the normal manner the display is automatically changed over to scientific notation.

Ex. 0.000123456 ⋮ 78 → 1.2345678×10^{-4}

A calculation result is displayed in either the normal manner or scientific notation, but it is stored in the memory in the form of

$$A \times 10^{B} \quad (1 \leqq |A| \leqq 9.999999999, \ -99 \leqq B \leqq 99)$$

or as 0.

INPUTTING DATA

To input a number to the Calculator, press $\boxed{+}$ or $\boxed{-}$ key first to input a sign, and then a numeric key or the decimal point key. (The operation of the $\boxed{+}$ key can be omitted.) To input a number in the scientific notation system (A x 10^B), input the mantissa, press the $\boxed{\text{Exp}}$ key and input the exponent.

Example: $-12.345 \rightarrow \boxed{-}\ \boxed{1}\ \boxed{2}\ \boxed{\cdot}\ \boxed{3}\ \boxed{4}\ \boxed{5}$

$6.7 \times 10^8 \rightarrow \boxed{6}\ \boxed{\cdot}\ \boxed{7}\ \boxed{\text{Exp}}\ \boxed{8}$

$-9.12 \times 10^{-34} \rightarrow \boxed{-}\ \boxed{9}\ \boxed{\cdot}\ \boxed{1}\ \boxed{2}\ \boxed{\text{Exp}}\ \boxed{-}\ \boxed{3}\ \boxed{4}$

To input data with a mantissa over 10 digits, the most significant 10 digits will be displayed, but internal calculations are performed using all the data input.

Example:	1234567898765	→	displayed as $1.234567898 \times 10^{12}$
	9.87654321234	→	displayed as 9.876543212
	0.0000000002345678	→	displayed as 2.345678×10^{-10}
	0.00001234567 $\boxed{\text{Exp}}$ 24	→	displayed as 1.234567×10^{19}

For the exponent, the last 2 entries are effective.

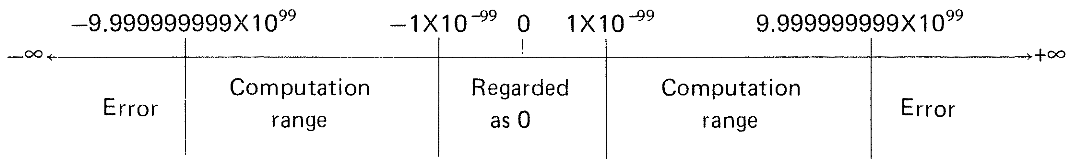
Example:	3 $\boxed{\text{Exp}}$ 123	→	displayed as 3×10^{23}
	4 $\boxed{\text{Exp}}$ $\boxed{-}$ 3210	→	displayed as 4×10^{-10}

More on inputting data when we get to programming.

COMPUTATION RANGE

The computing range is $-9.999999999 \times 10^{99}$ to -1×10^{-99}, 0 and 1×10^{-99} to $9.999999999 \times 10^{99}$.

Any calculation results outside of this range will result in an overflow error or 0. (See the illustration below.)



MANUAL CALCULATIONS

1. What is a manual calculation?

Normally you'll program the Pocket Computer in the PRO mode and execute programs in the RUN or DEF mode. For problems that don't need programming, you can Input the necessary data in the RUN (or DEF) mode and obtain immediate answers. This is called the Direct Execution mode or you might prefer to call it the manual calculation mode.

General form **(Expression)** ENTER

Example: 5 * 4 ENTER

Input an expression and press the ENTER key. The Computer will show the answer for the expression.

- **Manual calculations given in the following examples are executed in the RUN mode. Set the Computer to the RUN mode by pressing the MODE key. (The symbol "RUN" will appear on the display.)**

An Expression is composed of the following instructions:

● Constant	0 ~ 9, ·, π, Exp
● Sign	+, −
● Arithmetic operator	+, −, * (Multiplication), / (Division), Λ (Power)
● Logic operator	=, >, <, >=, <=, <>
● Functions	SIN, COS, TAN, ASN, ACS, ATN, LN, LOG, EXP, INT, ABS, SGN, √‾
● Parenthesis	(,)
● Memories	A ~ Z, A ( )

An "Expression" can be made by combining these instructions according to a mathematical formula.
A mathematical formula is defined as an "Expression", even if it is composed only of constants or memories. (Eg. 12, π, A, etc).

2. For arithmetic calculations

Addition and subtraction

Example: 7 − 9 + 14 =
−4.2 + 5 − 12.3 =

Operation	Display	Note
RUN mode		
CL 7 − 9 + 14	7−9+14_	Expression
ENTER	12.	Ans.
CL − 4.2 + 5 − 12.3	−4. 2+5−12. 3_	Expression
ENTER	−11. 5	Ans.

Multiplication and division

Example:

NOTE: In the BASIC language
* ✱ is used for Multiplication
* / is used for Division
* E is used for Exponent

Operation	Display	Note
CL 12 ✱ 24 / 5	12✱24/5_	Expression
ENTER	57. 6	Ans.
CL 27 Exp 3 ✱ 4 / 12	27E3✱4/12_	Expression
ENTER	9000.	Ans.

Mixed calculation

Example:

Operation	Display	Note
CL 54 + 24.3 ✱ 16.49	54+24. 3✱16. 49_	
/ 3.4 − 37.4	54+24. 3✱16. 49/3. 4−37. 4_	
ENTER	134. 455	

Note that multiplication and division have priority over addition and subtraction (that is, multiplication and division functions are performed before addition and subtraction). To control priority of calculation functions, use parentheses as noted later on in these examples.

When utilizing a displayed result in subsequent calculations.

In each of the above examples, the CL key is pressed first. This operation is intended to clear the preceding operations or the results of calculations.

If you want to use a calculation result for continued calculations start the next calculation without pressing the CL key.

Example ① 3 + 4 =
② −5 + 6 =

The result of (1) is incorporated into the expression (2), thus calculation 3 + 4 − 5 + 6 = is accomplished with the display of an intermediate answer.

Operation	Display	Note
CL 3 + 4 ENTER	7.	The result of ①
−	7. −_	The result of ①, (7) is incorporated into the next calculation (2).
5 + 6	7. −5+6_	
ENTER	8.	

After completing a calculation, if you press keys such as +, −, ✱, / just before inputting another (or further) expression, the preceding calculation result is incorporated as data for the next calculation.

3. Power calculations

Example: 4∧3= $(4^3=)$

3∧3.2∗4∧2.4= $(3^{3.2}\times 4^{2.4}=)$

4∧3∧2= $(4^{3^2}=)$

Operation	Display	Note
CL 4 SHFT ∧ 3	4^3 _	Expression
ENTER	64.	Ans.
3 SHFT ∧ 3.2 ∗	3^3. 2∗_	Expression
4 SHFT ∧ 2.4	3^3. 2∗4^2. 4_	
ENTER	936. 9836103	Ans.
4 SHFT ∧ 3 SHFT ∧ 2	4^3^2_	Expression
ENTER	262144.	Ans.

Note that power calculations have priority over the four arithmetic calculations.
Also note that powers of a negative number do not compute. This is due to the calculation process used by the Pocket Computer.

4. Calculations with parentheses

Calculations can be performed by using the (and) keys in the same manner as you use parentheses in mathematical formulas.

Example: (72+9)/4∗(21∗{68/(7−3)+2})=

Operation	Display	Note
CL (72 + 9) / 4 ∗	(72+9) /4∗_	
(21 ∗ (68 / (7	(72+9) /4∗ (21∗ (68/ (7_	
− 3) + 2))	+9) /4∗ (21∗ (68/ (7−3) +2)) _	
ENTER	8079. 75	Ans.

The use of parentheses results in different calculation sequences as follows. Take special care when multiplying, dividing or extracting square roots.

$A+B/C \rightarrow A+\frac{B}{C}$ $\sqrt{\;}A+B \rightarrow \sqrt{A}+B$

$(A+B)/C \rightarrow \frac{A+B}{C}$ $\sqrt{\;}(A+B) \rightarrow \sqrt{A+B}$

$A/C*D \rightarrow \frac{AD}{C}$ $\sqrt{\;}A*B \rightarrow B\sqrt{A}$

$A/(C*D) \rightarrow \frac{A}{CD}$ $\sqrt{\;}(A*B) \rightarrow \sqrt{AB}$

$A/B/C \rightarrow \frac{\frac{A}{B}}{C}=\frac{A}{BC}$ $A*B+C \rightarrow AB+C$

$A/(B/C) \rightarrow \frac{A}{\frac{B}{C}}=\frac{AC}{B}$ $A*(B+C) \rightarrow A(B+C)$

5. Scientific functions

Your TRS-80 Pocket Computer permits functions to be calculated just the same as in standard mathematic formulas.

- When performing a functional calculation of a constant or memory, use following form: SIN 30 or SIN A. In other cases use parentheses such as LN (A∗B) or SIN (π/2).

Angle mode

The angular mode is designated by the following:

Degree mode:	D E G R E E ENTER	("DEG" will appear at the top of the display.)
Radian mode:	R A D I A N ENTER	("RAD" will appear)
Grad mode:	G R A D ENTER	("GRAD" will appear)

Trigonometric functions (SIN, COS, TAN)

Examples:	S I N 30=	(sin30=)	Set the angular mode to "DEG".
	COS (π/4)=	$(\cos\frac{\pi}{4}=)$	RAD mode
	TAN 150=	(tan150=)	GRAD mode

Operation	Display	Note
DEG (D E G R E E ENTER)		"DEG"
CL S I N 30	S I N30_	
ENTER	0. 5	SIN 30°
RAD (R A D I A N ENTER)		
C O S (SHFT π	COS (π _	"RAD"
/ 4)	COS (π/4) _	
ENTER	7. 071067812E−01	$\cos\frac{\pi}{4}$(rad)
GRAD (G R A D ENTER)		"GRAD"
T A N 150	TAN150_	
ENTER	−1.	TAN 150^g

Inverse trigonometric functions (ASN, ACS, ATN)

Example:	ASN−0.5=	$(\sin^{-1}(-0.5)=)$	Set angular mode to DEG
	ACS (−0.5+0.1)=	$(\cos^{-1}(-0.5+0.1)=)$	Set angular mode to RAD
	ATN (7/3)=	$(\tan^{-1}\frac{7}{3}=)$	Set angular mode to GRAD

ASN: Arcsine
ACN: Arccosine
ATN: Arctangent

Operation	Display	Note
DEG CL A S N − .5	ASN−. 5_	Set DEG mode
ENTER	−30.	(°)
RAD A C S (− .5	ACS (−. 5_	
+ .1)	ACS (−. 5+. 1) _	Set RADIAN mode
ENTER	1. 982313173	(rad)
GRAD A T N (7	ATN (7_	
/ 3)	ATN (7/3) _	Set GRAD mode
ENTER	74. 22378832	(^{g})

Logarithmic functions (LN, LOG)

Examples: LN7.4= (ln7.4=) Note: $\ln X = \log_e X$: Natural logarithm

LOG100= (log100=) $\log X = \log_{10} X$: Common logarithm

Operation		Display		Note
CL	L N 7.4	LN7. 4_		
	ENTER		2. 00148	Ans.
	L O G 100	LOG100_		
	ENTER		2.	Ans.

Exponential function (EXP)

Example: EXP−13.6 = ($e^{-13.6}$) Note: EXP is anti-logarithm of LN

Operation		Display		Note
CL	E X P − 13.6	EXP −13. 6_		
	ENTER		1.24049508E−06	Ans.

Roots

Examples: √‾73= ($\sqrt{73}$=)

√‾√‾256= ($\sqrt{\sqrt{256}}=\sqrt[4]{256}$=)

√‾(3∗3+4∗4)= ($\sqrt{3^2+4^2}$=)

Operation		Display		Note
CL	√ 73	√‾73_		
	ENTER		8. 544003745	Ans.
	√ √ 256	√‾√‾256_		
	ENTER		4.	Ans.
	√ (3 ∗	√‾(3∗_		
	3 + 4 ∗ 4)	√‾(3∗3+4∗4) _		
	ENTER		5.	Ans.

Angle conversions (DMS, DEG)

DMS: Decimal degrees → Degrees/minutes/seconds
When converting decimal degrees to degrees/minutes/seconds, the answer is displayed as follows: integer portion = degrees; 1st and 2nd decimal digits = minutes; 3rd and 4th digits = seconds; and any remaining decimal digits are decimal degrees.

DEG: Degrees/minutes/seconds → Decimal degrees
To convert an angle given in degrees/minutes/seconds to its decimal equivalent, it must be entered as integer and decimal numbers as noted above.

Example: Convert 15.4125° to its degree/minute/second equivalent.
Convert 15°24′45″ to its decimal equivalent.

Operation	Display	Note
CL D M S 15.4125	DMS15. 4125_	
ENTER	15. 2445	15°24′45″
D E G 15.2445	DEG15. 2445_	
ENTER	15. 4125	15.4125°

Integer (INT)

The integer (INT) function converts numerical values to the next lowest integer value: 12.34 becomes 12.; and −2.45 becomes −3.

Examples: INT (65/3)=
INT (−0.3)=

Operation	Display	Note
I N T (65 / 3)	INT (65/3) _	
ENTER	21.	
I N T − .3	INT −. 3 _	
ENTER	−1.	

Sign function (SGN)

The SGN function takes the following values for numerical values of X. (That is, it returns a −1 for all negative values, zero for 0 and a +1 for all positive values.)

+1	if $X > 0$
0	if $X = 0$
−1	if $X < 0$

Example: SGN (5 − 9) =

Operation	Display	Note
S G N (5 − 9)	SGN (5−9) _	
ENTER	−1.	

Absolute value (ABS)

The ABS function finds the absolute value $|X|$ of a numerical value X. (In simple terms we might say it strips the sign from a number.)

Example: ABS (5−9)= (|5−9|=)

Operation	Display	Note
A B S (5 − 9)	ABS (5−9) _	
ENTER	4.	

6. Logic functions

These functions return 1 when an expression composed using logic operators (=, >, <, > =, < = and < >) is true, and 0 when false. In other words, $x \circ y$ ($\circ$ is a logic operator) gives us either a 1 or a 0 depending on the relationship of x and y.

Logic operator	
=※	1 if $x = y$. 0 if $x \neq y$.
>	1 if $x > y$ 0 if $x \leqq y$
<	1 if $x < y$ 0 if $x \geqq y$
> =	1 if $x \geqq y$ 0 if $x < y$
< =	1 if $x \leqq y$ 0 if $x > y$
< >	1 if $x \neq y$ 0 if $x = y$

Note: < > has the same meaning as ≠.

※ If you want to use a logical expression relating to contents of a memory (variable name) you must use the following form: 〈 expression 〉 = [memory/variable name]. If you use [memory/variable name] = 〈 expression 〉 the Computer will treat the statement as a normal assignment statement; e.g. A = 〈 expression 〉. (Relational equations in IF statements, are not subject to this exception; i.e. they function normally.)

Examples: (5+8)>(3∗4)=
(24/5)<=(2.4∗2)=

Operation	Display	Note
$\boxed{(}$ 5 $\boxed{+}$ 8 $\boxed{)}$ $\boxed{\text{SHFT}}$ $\boxed{>}$	(5+8) 〉 _	
$\boxed{(}$ 3 $\boxed{*}$ 4 $\boxed{)}$	(5+8) 〉 (3∗4) _	
$\boxed{\text{ENTER}}$	1.	Answer
$\boxed{(}$ 24 $\boxed{/}$ 5 $\boxed{)}$ $\boxed{\text{SHFT}}$ $\boxed{<}$ $\boxed{=}$	(24/5) 〈=_	
$\boxed{(}$ 2.4 $\boxed{*}$ 2 $\boxed{)}$	(24/5) 〈= (2. 4∗2) _	
$\boxed{\text{ENTER}}$	1.	Answer

Notes:
Logical expressions (AND) and (OR) can be executed using the following form of logical computations.

① Logical OR (logical computation) + (logical computation)

Example:	(A < 0) + (A > 8)	1 is returned if A is smaller than 0 or larger than 8.
	(B > 0) + (C > 0)	1 is returned if B or C is larger than 0 and 2 is returned if both B and C are larger than 0.

② Logical AND (logical computation) ∗ (logical computation)

Example: (B > 1) ∗ (B < 6) 1 is returned if B is larger than 1 and smaller than 6.

7. Calculations using memories

The TRS-80 Pocket Computer has two types of data memories: fixed memory (26 in all) and flexible memory. In this section we'll give examples of calculations using fixed memories. (For details, refer to page 45.)

Specifying memories ①

The fixed memories are given labels A through Z (each being specified by the $\boxed{A}$ through $\boxed{Z}$ keys).

Example: When memory A is loaded with 4 and memory B with 5.

$\boxed{A}\ \boxed{+}\ \boxed{B}\ \boxed{*}\ \boxed{B}\ \boxed{-}$ 12 $\boxed{\text{ENTER}}$ → 17. $\left(\begin{array}{l}\boxed{A}\ \boxed{=}\ 4\ \boxed{\text{ENTER}}\\ \boxed{B}\ \boxed{=}\ 5\ \boxed{\text{ENTER}}\end{array}\right)$

$\boxed{\sqrt{\ }}\ \boxed{(}\ \boxed{A}\ \boxed{+}\ \boxed{B}\ \boxed{)}\ \boxed{\text{ENTER}}$ → 3.

Specifying memories ②

The fixed memories A through Z can be used as dimension as specified in the form of A (   ). This form is used for storing data in a matrix or array.

Example: $\boxed{A}\ \boxed{(}$ 2 $\boxed{)}$ → Memory A (2), namely memory B, is specified.

$\boxed{A}\ \boxed{(}$ 2 $\boxed{+}$ 3 $\boxed{)}$ → Memory A (5), namely memory E, is specified.

Input to the memories

Numerical values and others are input to the memories in the following forms.

General form [memory] $\boxed{=}$ ⟨ expression ⟩ $\boxed{\text{ENTER}}$

Example: $\boxed{A}\ \boxed{=}$ 5 $\boxed{*}$ 6 $\boxed{\text{ENTER}}$ → Loading the answer of 5 * 6 (30) into memory A

$\boxed{Y}\ \boxed{=}\ \boxed{A}\ \boxed{+}\ \boxed{B}\ \boxed{\text{ENTER}}$ → Loading the contents of memory A plus memory B into memory Y.

$\boxed{A}\ \boxed{(}$ 26 $\boxed{)}\ \boxed{=}$ 3 $\boxed{+}$ 9 $\boxed{\text{ENTER}}$ → Loading the answer of 3 + 9 (12) into memory A (26) (memory Z).

● When the memories are loaded with new data, they are automatically cleared of their previous contents.

Recalling the memory contents

The memory contents are recalled using the following form.

General form [memory] $\boxed{\text{ENTER}}$

Example: $\boxed{A}\ \boxed{\text{ENTER}}$ → Recalling the contents of memory A

$\boxed{A}\ \boxed{(}$ 18 $\boxed{)}\ \boxed{\text{ENTER}}$ → Recalling the contents of memory A (18) (memory R)

8. Successive designation of expressions in manual calculation

In manual calculations, you can designate and solve two or more expressions in succession by separating them with a comma. However, the computer will display the result of the final execution only.

General form ⟨ Expression ⟩ $\boxed{\text{SHFT}}\ \boxed{,}$ ⟨ Expression ⟩ $\boxed{\text{SHFT}}\ \boxed{,}$ ⟨ Expression ⟩ $\boxed{\text{SHFT}}\ \boxed{,}$ .... $\boxed{\text{ENTER}}$

Example: When $A = \frac{5}{12-4}$, $B = \frac{87}{24}$, $C = \frac{12}{7+8}$, solve $A * B/C =$

Operation	Display	Note
[A][=]5[/][(]12[−]4[)]	A=5/ (12−4) _	
[SHFT][,][B][=]87[/]24[SHFT][,]	A=5/ (12−4) , B=87/24, _	
[C][=]12[/][(]7	/ (12−4) , B=87/24, C=12/ (7_	
[+]8[)][SHFT][,]	−4) , B=87/24, C=12/ (7+8) , _	
[A][*][B][/][C]	=87/24, C=12/ (7+8) , A*B/C_	
[ENTER]	2. 83203125	

9. Recall function

This function permits you to recall (into display) a portion of your original input so you can check and/or edit it. This function is activated by pressing either the [▶] or [◀] key right after the [ENTER] key in manual calculations.

Example 1: When an execution is finished without an error message:

Operation	Display	Note
[A][=] 19 [+] 54	A=19+54_	
[ENTER]	73.	
[▶] or [◀]	A=19+54	Recall

The cursor shows up at the beginning of the display.
(The complete entry is displayed.)

Example 2: When an error occurs:

Operation	Display	Note
[B][=][(][(]123456[+]	B= ((123456+_	
789012[)][*]427[/]197	(123456+789012) *427/197_	
[)][/] 0 [+] 139	+789012) *427/197) /0+139_	
[ENTER]	1	Error message
[▶] or [◀]	3456+789012) *427/197) /0+	Recall

The cursor appears where an error is detected.
(The complete entry is displayed to the location where the error has been detected.)

10. Editing expressions

Input expressions, if recalled before or immediately after their execution, can be arbitrarily edited (correction, insertion or deletion).
When you make a correction, insertion or deletion, follow the procedures given below.

Correction: Use the [▶] or [◀] key to move the cursor to the position where the correction is to be made/stored, and enter correct key operation.

Insertion: Use the [▶] or [◀] key to move the cursor to the position where you want to insert information, and press [SHFT] and [INS] keys.

The contents of that position and all after it will be shifted one step backward and an insertion mark (⊏⊐) will appear in the empty position.

Deletion: Use the $\boxed{\blacktriangleright}$ or $\boxed{\blacktriangleleft}$ key to move the cursor to the position you want to delete, and press the $\boxed{\text{SHFT}}$ and $\boxed{\text{DEL}}$ keys. The contents at that position will be deleted and all after will be shifted one step forward. The cursor still remains at the said position.

Example: You make a mistake in inputting the expression below; edit as follows.

A=5+6∗ (21/S I N 30)

Proper operation

$\boxed{A}$ $\boxed{=}$ 5 $\boxed{+}$ 6 $\boxed{*}$ $\boxed{(}$ 21 $\boxed{/}$ $\boxed{S}$ $\boxed{I}$ $\boxed{N}$ 30 $\boxed{)}$ $\boxed{\text{ENTER}}$

(1) To correct the $\boxed{+}$ key error (pressed by mistake instead of the $\boxed{*}$ key).

Operation	Display	Note
$\boxed{A}$ $\boxed{=}$ 5 $\boxed{+}$ 6 $\boxed{+}$	A=5+6+_	The $\boxed{+}$ key is added instead of the $\boxed{*}$ key.
$\boxed{\blacktriangleleft}$	A=5+6+	The cursor moves to the left.
$\boxed{*}$	A=5+6∗_	Enter the correct key.
⋮		

(2) If you forget to input the $\boxed{S}$, $\boxed{I}$ and $\boxed{N}$ keys (Insertion of SIN)

Operation	Display	Note
DEG		
$\boxed{A}$ $\boxed{=}$ 5 $\boxed{+}$ 6 $\boxed{*}$	A=5+6∗_	
$\boxed{(}$ 21 $\boxed{/}$ 30 $\boxed{)}$	A=5+6∗ (21/30) _	SIN is not entered.
$\boxed{\blacktriangleleft}$ $\boxed{\blacktriangleleft}$ $\boxed{\blacktriangleleft}$	A=5+6∗ (21/30)	The cursor moves to the left.
$\boxed{\text{SHFT}}$ $\boxed{\text{INS}}$	A=5+6∗ (21/⊏⊐30)	Spaces are added where you want to enter SIN.
$\boxed{\text{SHFT}}$ $\boxed{\text{INS}}$ $\boxed{\text{SHFT}}$ $\boxed{\text{INS}}$	A=5+6∗ (21/⊏⊐⊏⊐⊏⊐30)	
$\boxed{S}$ $\boxed{I}$ $\boxed{N}$	A=5+6∗ (21/S I N30)	SIN is inserted.
$\boxed{\text{ENTER}}$	257.	

(3) If you have input 211 by mistake instead of 21 (Deletion of 1)

Operation	Display	Note
DEG		
$\boxed{A}$ $\boxed{=}$ 5 $\boxed{+}$ 6 $\boxed{*}$ $\boxed{(}$	A=5+6∗ (_	
211 $\boxed{/}$ $\boxed{S}$ $\boxed{I}$ $\boxed{N}$ 30 $\boxed{)}$	A=5+6∗ (211/S I N30) _	
$\boxed{\blacktriangleleft}$ $\boxed{\blacktriangleleft}$ $\boxed{\blacktriangleleft}$ $\boxed{\blacktriangleleft}$	A=5+6∗ (211/S I N30)	The cursor moves left to the position where you want to make an deletion.
$\boxed{\blacktriangleleft}$ $\boxed{\blacktriangleleft}$ $\boxed{\blacktriangleleft}$ $\boxed{\blacktriangleleft}$	A=5+6∗ (211/S I N30)	
$\boxed{\text{SHFT}}$ $\boxed{\text{DEL}}$	A=5+6∗ (21/S I N30)	1 is deleted.
$\boxed{\text{ENTER}}$	257.	

(4) **If you have input 2 by mistake (instead of 6) and executed the calculation.**

Operation	Display	Note
DEG $\boxed{A}\ \boxed{=}$ 5 $\boxed{+}$ 2 $\boxed{*}\ \boxed{(}$	A=5+2* (_	Set DEG mode.
21 $\boxed{/}\ \boxed{S}\ \boxed{I}\ \boxed{N}$ 30 $\boxed{)}$	A=5+2* (21/SIN30) _	
$\boxed{\text{ENTER}}$	89.	
$\boxed{\blacktriangleleft}$	A=5+2* (21/SIN 30)	Entire expression is is recalled.
$\boxed{\blacktriangleright}\ \boxed{\blacktriangleright}\ \boxed{\blacktriangleright}\ \boxed{\blacktriangleright}$	A=5+2* (21/SIN 30)	The cursor moves left.
6	A=5+6 * (21/SIN 30)	Correction
$\boxed{\text{ENTER}}$	257.	

(5) **If you have input SIN 0 by mistake (instead of SIN 30) and executed the calculation (Error display)**

Operation	Display	Note
DEG $\boxed{A}\ \boxed{=}$ 5 $\boxed{+}$ 6 $\boxed{*}\ \boxed{(}$	A=5+6* (_	
21 $\boxed{/}\ \boxed{S}\ \boxed{I}\ \boxed{N}$ 0 $\boxed{)}$	A=5+6* (21/SINO) _	
$\boxed{\text{ENTER}}$	1	Error is detected.
$\boxed{\blacktriangleright}$	A=5+6* (21/SIN 0)	Entire expression is recalled.
$\boxed{\blacktriangleleft}\ \boxed{\text{SHFT}}\ \boxed{\text{INS}}$	A=5+6* (21/SIN ▯0)	3 is inserted.
3	A=5+6* (21/SIN 30)	
$\boxed{\text{ENTER}}$	257.	

● **Cursor positioning**

If you keep the $\boxed{\blacktriangleright}$ or $\boxed{\blacktriangleleft}$ key pressed, the cursor automatically starts moving right or left after about one second. The cursor will move about 10 steps per second. The cursor stops automatic movement as soon as you release the key. When editing long lines of information, this fast advance or return is a great aid.

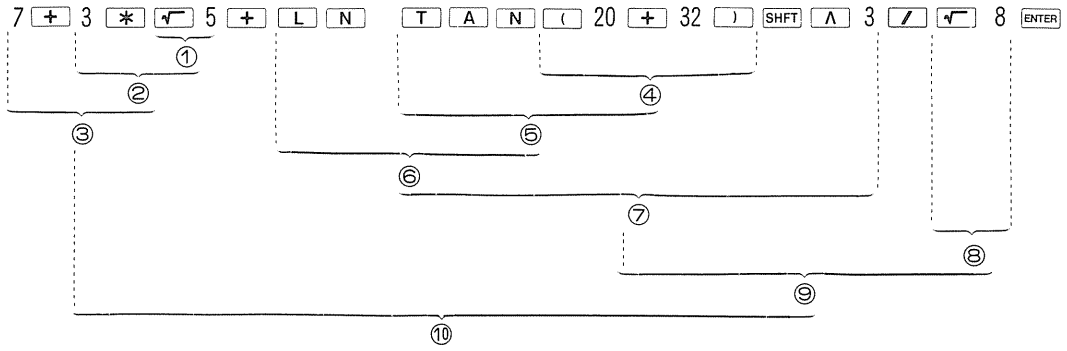
11. Priority of calculations

The computer performs calculations from the left to the right with some exceptions (for example, functions, multiplication or division has priority over addition or subtraction). The following lists the order of operations of the Computer.

1. Recalling π and fixed memories A through Z.
2. Recalling memories in the form of A (   ) (Recalling dimensioned memories)
3. Power directly preceded by multiplication (which involves memory) 2A ∧ 3, for example.
4. Multiplication where * is omitted: 2A, πB or AB (see page 30)
5. Functions (SIN, COS, TAN, ASN, ACS, ATN, LN, LOG, EXP, DMS, DEG, INT, ABS, SGN, √)
6. Power (∧) other than as defined in 3 above.
7. Sign (+, −)
8. Multiplication and division (*, /)
9. Addition and subtraction (+, −)
10. Logical computation (=, >, <, >=, <=, <>)

- Calculations in parenthesis will occur first. In multiple parenthesis, calculations in the inner-most parentheses have priority over all the others.
- Compound founctions (LN ABS A, EXP √ 8) are calculated from right to left.
- A string of powers, such as 3 4 2, is calculated from the right to the left.

Levels of pending operation



As seen from the example above, the Computer performs computations following a given mathematical formula. But this presupposes that the Computer has a place to temporarily store instructions or data (numerical values) that cannot be directly processed. Such a place is called a stack (a stack register). Your Pocket Computer has a 16-stage function stack and 8-stage data stack.

Example: Behavior of both stacks during the execution of

1.2 + A * (3.5 + SIN B) ∧ A (25) ENTER

Where A = 2.4, B = 30, A(25) = 3

Angular mode = DEG

Instruction	X register	Data stack				Function stack					
		1st stage	2nd stage	3rd stage	···	1st stage	2nd stage	3rd stage	4th stage	5th stage	···
1.2	1.2										
+	1.2	1.2				+					
A	2.4	1.2				+					
*	2.4	2.4	1.2			*	+				
(	2.4	2.4	1.2			(	*	+			
3.5	3.5	2.4	1.2			(	*	+			
+	3.5	3.5	2.4	1.2		+	(	*	+		
SIN	3.5	3.5	2.4	1.2		SIN	+	(	*	+	
B	30	3.5	2.4	1.2		SIN	+	(	*	+	
)	0.5	3.5	2.4	1.2		+	(	*	+		
	4	2.4	1.2			*	+				
^	4	4	2.4	1.2		^	*	+			
A(	4	4	2.4	1.2		A(	^	*	+		
25	25	4	2.4	1.2		A(	^	*	+		
)	3	4	2.4	1.2		^	*	+			
ENTER	64	2.4	1.2			*	+				
	153.6	1.2				+					
	154.8										

Note: X register is the Calculation register

As seen from the above, "A(" is placed in the function stack as one step. The Computer will handle up to 15 levels of parentheses unless the function stack capacity is exceeded.

NOTES

The Computer permits you to input 2 * A, 3 * π or B * A (12), for example, in the following form: 2A, 3π or BA(12) omitting the multiplication symbol * immediately in front of memory or π. Such a form of multiplication has priority over functions, but when it is directly followed by a power, the power takes precedence over it.

Example: SIN 2A → equivalent to SIN (2 * A)
2πA ∧ 3 → equivalent to 2 * π * (A ∧ 3)

However, expressions put in the Computer as mentioned above (multiplication instruction * omitted) are executed just as tho they were incorporated in the instruction.

Example: Behavior of the stacks during execution of 2ABC $\boxed{\text{ENTER}}$
If A = 3, B = 5, C = 7

Instruction	X register	Data stack			Function stack		
2	2						
A	3	2			*		
B	5	3	2		*	*	
C	7	5	3	2	*	*	*
ENTER	35	3	2		*	*	
	105	2			*		
	210						

PROGRAMMED CALCULATIONS

You program your TRS-80 using a computer language called BASIC. BASIC is generally considered the easiest computer language to understand —— since it uses simple English words. But of course BASIC is not limited to beginners —— it is a very powerful computer language used by many experienced professional programmers. You can't "talk" (or "type") to your Computer in an everyday English conversational manner. You have to use the correct words, in the correct sequence —— that is, according to certain rules.

With a Computer, you are the master, you are the boss. The Computer can only do what you tell it. But you must give the instructions in a form the Computer can understand. That's where the BASIC language comes in. Each BASIC instruction must start with a line number (normally we start with 10 and each successive line is by 10's —— 10, 20, 30, etc.).

Since this Manual is not intended to be a simple learner's guide to BASIC Programming, if you feel we are moving along too fast, we urge you to stop by your local Radio Shack store and obtain a copy of either (or both) of the following books:

BASIC Computer Language (60-2016)
BASIC Computer Programming (60-2015)

1. What is a program calculation?

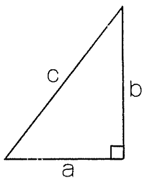
We've been discussing manual calculation for the last 8 or 10 pages –– calculations performed by manual entry of all information. Now let's find out why your TRS-80 Pocket Computer is much more than a Calculator.
With programmed calculations you enter a series of instructions to the Computer (a program). Then all you need do is enter the data for the calculation work – the Computer uses the program (stored in memory) to give you the answers. Let's give an example.
When solving problems using Pythagoras's theorem, for example, you must carry out the following operation.

Pythagora's theorem

For a rectangular triangle, its three sides a, b and c have the following relations

$$c = \sqrt{a^2 + b^2}$$

where c is the side opposite to the right angle.



Manual calculation requires the following sequence.
(when a = 3 and b = 4.)

Operation	Display	Note
Set to the RUN mode.		
A = 3 ENTER	3.	A is loaded with 3.
B = 4 ENTER	4.	B is loaded with 4.
C = √ (A * A	C=√ (A*A _	
+ B * B)	C=√ (A*A+B*B)_	
ENTER	5.	$\sqrt{A^2+B^2}$

[The above calculation of course can be accomplished by using:
√ (3 * 3 + 4 * 4) ENTER . However, we're using the key operations shown above to aid in our application example.]

Let's key in this simple BASIC program:

[PROGRAM 1]

Program	Note
10 : INPUT A , B	Input instruction
20 : C=√ (A*A+B*B)	Operation instruction
30 : PRINT C	Output instruction
40 : END	End instruction

NOTE: The letters A, B and C are called variables. You might want to refer to pages 45 to 50 for a detailed description of what variables are and how they relate to programming.
To complete a program line, you must press ENTER .

Basically, a program is complete if you have an input instruction (INPUT), output instruction (PRINT) and end instruction (END) plus the steps required to process the calculation procedures.

Input instruction:	Provides input data for memory.
Output instruction:	Provides display of calculation results (or other output).

The tables below summarize the writing and execution of Program 1.

Operation	Display	Note
[Writing] **Set to the PRO mode.**	Prompt symbol >	Press $\boxed{\text{MODE}}$ key to display PRO (i.e. program mode).
$\boxed{N}\boxed{E}\boxed{W}\boxed{\text{ENTER}}$	>	Clears program memory.
10 $\boxed{I}\boxed{N}\boxed{P}\boxed{U}\boxed{T}$	10INPUT_	Number and instruction is entered. (Input instruction)
$\boxed{A}\boxed{\text{SHFT}}\boxed{9}\boxed{B}$	10INPUTA,B_	
$\boxed{\text{ENTER}}$	10:INPUT A,B	Line 10 is "written" into Computer.
20 $\boxed{C}\boxed{=}\boxed{\sqrt{}}\boxed{(}\boxed{A}$	20C=√(A_	Number and instruction is entered. (Operation instruction)
$\boxed{*}\boxed{A}\boxed{+}\boxed{B}$	20C=√(A*A+B_	
$\boxed{*}\boxed{B}\boxed{)}$	20C=√(A*A+B*B)_	
$\boxed{\text{ENTER}}$	20:C=√(A*A+B*B)	Line 20 is written into Computer.
30 $\boxed{P}\boxed{R}\boxed{I}\boxed{N}\boxed{T}$	30PRINT_	Number and instruction is entered. (Output instruction)
$\boxed{C}$	30PRINTC_	
$\boxed{\text{ENTER}}$	30:PRINT C	Line 30 is written into Computer.
40 $\boxed{E}\boxed{N}\boxed{D}$	40END_	Number and instruction is entered. (End instruction)
$\boxed{\text{ENTER}}$	40:END	Line 40 is written into Computer.
[Execution] **Set to the RUN mode.**		
$\boxed{\text{MODE}}\boxed{\text{MODE}}\boxed{\text{MODE}}$	>	RUN will be displayed.
$\boxed{R}\boxed{U}\boxed{N}$	RUN_	RUN tells Computer program to start execution.
$\boxed{\text{ENTER}}$	?	Execution is started; the display asks you to input a variable.
3	3_	A variable, 3, is put in.
$\boxed{\text{ENTER}}$	?	The variable is written in (3 is loaded on memory A); the display asks you to input another variable.
4	4_	A variable, 4, is put in.
$\boxed{\text{ENTER}}$	5.	The variable is written in (4 is loaded on memory B); calculation result is displayed.
$\boxed{\text{ENTER}}$	>	Execution is terminated.
Repeat these operations inputting different values. Thus you can calculate many problems using Pythagoras's theorem.		

Thus, once a program is written, you can execute it simply, any number of times you want.

2. Writing programs

When you "write" programs using the keyboard, set the Computer in the PRO mode.

Preparation

When writing a new program, we suggest that you clear the program memory by using a NEW command.
However, this is not true if you want to write a program in succession with the preceding one.

[Procedures] (1) Designate the PRO mode.
(2) N E W ENTER

All contents of the program and data memories will be cleared with above (2) operation.

Writing in

Detailed below is how you would write the first program (PROGRAM 1).

Step	Operation	Display	Note
	Set to the PRO mode		
1	N E W ENTER	>	The program memory is cleared.
	A colon (:) does not need to be put in.		
2	10 I N P U T	10INPUT_	Write in line 10. (This line is placed in the input buffer.)
3	A SHFT , B	10INPUTA,B_	
4	ENTER	10:INPUT A,B (A display of colon; Space)	Line 10 is entered into program memory when you push ENTER .
5	20 C = √ (A	20C=√(A_	Write in line 20. (This line is placed in the input buffer.)
6	* A + B	20C=√(A*A+B_	
7	* B)	20C=√(A*A+B*B)_	
8	ENTER	20:C=√(A*A+B*B)	Line 20 is entered into program memory.
9	30 P R I N T	30PRINT_	Write in line 30. (This line is placed in the input buffer.)
10	C	30PRINTC_	
11	ENTER	30:PRINT C	Line 30 is entered into program memory.
12	40 E N D	40END_	Write in line 40. (This line is placed in the input buffer.)
13	ENTER	40:END	Line 40 is entered into program memory.

If you followed the Notes in the chart, you'll realize that the lines of instruction are first held in the input buffer and then loaded into program memory when you press ENTER .

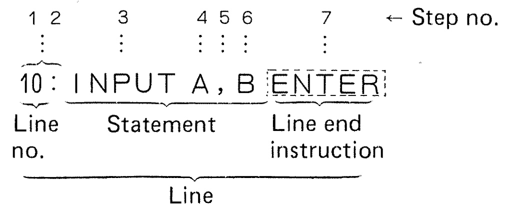
As you enter a line into the Computer you notice that a colon appears after the line number. Also the Computer adds a space after commands (PRINT, INPUT, etc.).

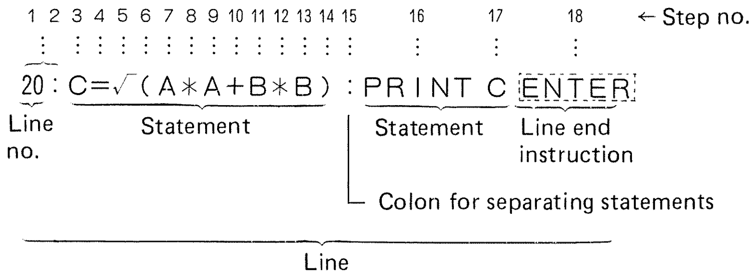
If any line contains more characters than the display can show at one time, only the beginning portion of the line of instructions will be displayed (up to 24 characters and spaces).

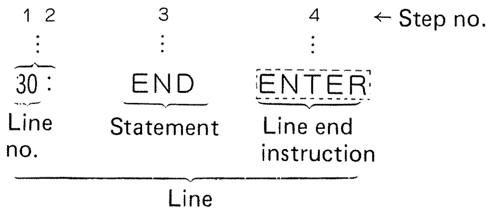
Comment

You have learned by now that the Computer's programs are made up of lines: a line is composed of a line number, label and statements.

Example: Composition of a program




1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 ← Step no.
20 : C=√(A*A+B*B) : PRINT C ENTER
Line no.
Statement
Statement
Line end instruction
Colon for separating statements
Line



Line

- Lines must be numbered with integer numbers ranging from 1 to 999.
- Ending of a line is accomplished by inputting the ENTER key. The ENTER instruction is represented by a space in the display. (Nothing appears.)

Statement

- One line consists of one or more statements (statement instructions in the BASIC language).
- Statements are divided by colons (:).

Step

One statement consists of one or more operation instructions, each of those instructions having a capacity of one step.
Instructions such as LN, SIN and INPUT are processed as one-step information written into the program memory, even though represented in the display by two to six characters.

Label

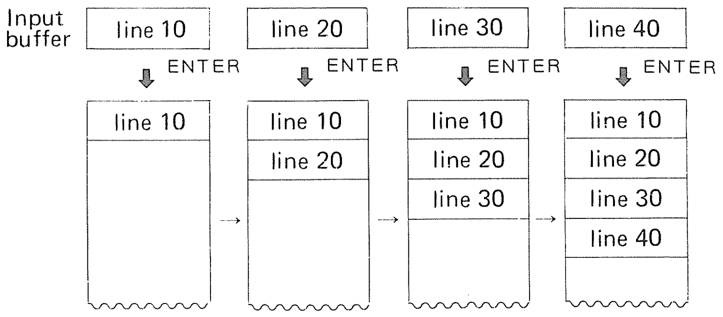
Characters (letters, numerals, symbols) are placed between quotation marks following a line number. The label serves as the sign for a program jump, etc. See page 42.

Note: Each line number (1 through 999) is held as two-step information in the program memory. Although not present in a program memory, the colon (:) that follows every line number is automatically displayed immediately after programs have been written or when they are recalled.

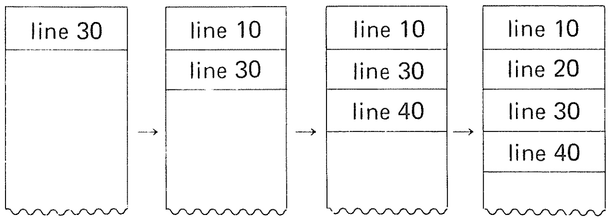
Organization of Program Memory

When loaded with programs, the program memory changes as follows:

An input is placed (on a 1-character = 1-step basis) in the input buffer, when ENTER is pressed the input is written in the program memory (after being converted into the form of 1-instruction = 1-step).



- One line of a program can hold a maximum of 80 steps (1 step = 1 character) Lines are stored in numerical order into the program memory (even if not entered in numerical order).



However, the real program memory does not have a separate storage area for every line as shown above, but stores programs step by step in a row; your Computer has a capacity of 1424 steps.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	no. 0	INPUT	A	,	B	ENTER	Line 2	no. 0	C	=	√	(	A	*	A

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	← step
+	B	*	B	)	ENTER	Line 3	no. 0	PRINT	C	ENTER	Line 4	no. 0	END	ENTER	

3. Checking stored programs

You should always check that programs are properly stored.

Whenever programs are input via the keyboard, the Computer displays them (permitting you to check each input).

After you finish writing a program, you can check it as follows:

① Select the PRO mode.

② Recall the line you want to check, by pressing the $\boxed{\downarrow}$ or $\boxed{\uparrow}$ key.
Or recall the intended line with a LIST command. (Refer to page 72.)

③ Check instructions on the display.
If a display line is more than 24-characters long, move cursor by using [►] and [◄] keys to display the remainder.

Example: Checking programs

Operation	Display	Note
Set to the PRO mode	>	
[↓]	10 : INPUT A , B	Lines are recalled in numerical order.
[↓]	20 : C= (A∗A+B∗B)	
[↓]	30 : PRINT C	
[↓]	40 : END	
[↑]	30 : PRINT C	Lines on the display are brought back to the memory.
[↑]	20 : C= (A∗A+B∗B)	
[↑]	10 : INPUT A , B	

- If you press and hold the [↓] or [↑] key for about one second, the display will automatically show the next line or the preceding line.

Note: When the program memory is loaded with nothing, pressing the [↓] or [↑] key or execution of a LIST command will only display the prompt symbol (>).

4. Program correction

When you find errors in stored programs, use the following procedures for correction.

Partial correction

[Procedures]

① Select the PRO mode.
② Display the line you want to correct by using the [↓] or [↑] key, or a LIST command.
③ Move the cursor to the step you want, by pressing the [►] or [◄] key.
④ Make a correction, insertion or deletion deletion as described on page 26.
⑤ When the correction is finished, press the [ENTER] key. This will return the corrected program to program memory.

Example: A program identical to PROGRAM 1 is set up and corrected as follows:

```
10 : INPUT A , B
20 : C=√(A∗A+B+B)
30 : PAUSE C
40 : END
```

→

```
20 : C=√(A∗A+B∗B)
30 : PRINT C
```

Step	Operation	Display	Note
	Set to the PRO mode	〉	
1	↓ ↓	20 : C=√(A∗A+B+B)	Recall the line you want to modify.
2	►	20 C=√(A∗A+B+B)	The cursor appears on the display.
3	► …… ►	20 C=√(A∗A+B+B)	The cursor moves to the position where correction is to be made.
4	∗	20 C=√(A∗A+B∗B)	Correction
5	ENTER	20 : C=√(A∗A+B∗B)	Remember to press ENTER.
6	↓	30 : PAUSE C	Recall another line you want to modify.
7	◄	30 PAUSE C	The cursor appears on the display.
8	P	30 PC	Correction
9	R	30 PR_	
10	I N T C	30 PRINTC_	
11	ENTER	30 : PRINT C	Write in corrected line.

- Pressing ► or ◄ key once after the recall of a line, recalls the cursor. (Steps 2 and 7 in the above table)
  The cursor shows up at the beginning of first statement; a colon (:) after line number disappears, leaving a space there.
- Upon reaching a 1-step imperative statement (such as INPUT or PRINT) the cursor positions only at its first character.
  If you make a change in this first character, the entire statement disappears from the display. (Step 8 above)

Inserting lines

To insert lines into written programs, follows this procedure:

(1) Select the PRO mode.
(2) Input a line. This line must be given a line number between the line numbers before and after where you want it to be positioned in the program.
If you wish to insert a new line between lines 10 and 20, you must number the new line within 11 to 19.
(3) Press the ENTER key. The new line is then written into the program memory.

Example: Insertion of PAUSE A, B between lines 10 and 20 of PROGRAM 1. (New line number is 15.)

```
10 : INPUT A,B  <————————15 : PAUSE A,B
20 : C=√(A∗A+B∗B)
30 : PRINT C
40 : END
```

Operation	Display	Note
Set to the PRO mode.		
15 P A U S E	15PAUSE_	line 15 is put in.
A SHFT , B	15PAUSEA,B_	
ENTER	15:PAUSE A,B	Writing (insertion) of line 15.
CL ↓	10:INPUT A,B	Checking
↓	15:PAUSE A,B	
↓	20:C=√(A*A+B*B)	

Deleting lines

To delete a certain line from the corresponding stored program, follow these procedures:

(1) Select the PRO mode.

(2) Input only the line number of the line you want to delete, and press the ENTER key.

Example: Deletion of line 15.

Operation	Display	Note
Set to the PRO mode.		
15	15_	Input the line number of line to be deleted.
ENTER	>	line 15 is deleted.
↓	10:INPUT A,B	Checking
↓	20:C=√(A*A+B*B)	

5. Executing programs

You must execute programs in the RUN or DEF mode.

[Procedures]

① Select the RUN mode.

② Press the R , U , N and ENTER keys. The Computer starts the execution of program.

③ When program execution stops at an INPUT instruction, the "?" symbol then appears, input data and press ENTER key.

④ When program execution stops at a PRINT instruction, Calculation result is displayed, press ENTER key without inputting data.

⑤ Program execution comes to an end at an END instruction. The prompt symbol then appears.

Example: Execution of PROGRAM 1
(1) When A = 12.3, B = 15.7
(2) When A = 36, B = 27

Operation	Display	Note
Set to the RUN mode.		
R U N	RUN_	Inputting RUN instruction.
ENTER	?	Execution starts.
Refer to ③ above { 12.3	12. 3_	Inputting data (A)
ENTER	?	
③ { 15.7	15. 7_	Inputting data (B)
ENTER	19. 94442278	Result appears.
Refer to ④ above { ENTER	>	Execution ends.
R U N ENTER	?	Execution starts.
36 ENTER	?	
27 ENTER	45.	Result appears.
Refer to ⑤ above ENTER	>	Execution ends.

6. DEBUGing programs

The DEBUG function helps you ckeck to see if prepared programs are working properly. Programs are executed a line at a time so you can check the progress.

Example: Debugging of PROGRAM 1
When A = 36, B = 27

Operation	Display	Note
Set to the RUN mode		
D E B U G ENTER	?	DEBUG command commences debugging.
36	36_	Inputting of data
↓	?	INPUT instruction for line 10 is executed.
27	27_	After execution of the above instruction, the Computer displays the line number and stops.
↓	10 :	
↑ (kept pressed)	10 : INPUT A,B_	Checking of executed instruction; the cursor indicates the executed instruction — ENTER in this example.(i.e. space)
Release the ↑	>	The prompt symbol appears after display of line 10.
↓	20 :	After the debugging of line 20, its line number is displayed.
↓	45.	Display of calculation result (execution of PRINT statement)
A ENTER	36.	Checking of memory contents
B ENTER	27.	
↓	30 :	The computer stops after debugging line 30.
↓	>	Debugging ends.

(For DEBUG command, refer to page 71.)

In debugging, as shown above, pressing the ↓ key executes the instruction on each line.

While execution is stopped, you can manually check memory data contents; (just press A ENTER or B ENTER).

The ↓ key activates the debugging operation (even if pressed after checking memory contents). Again, when execution stops, the cursor indicates the step where the Computer is waiting and you can display the corresponding program line by pressing the ↑ key.

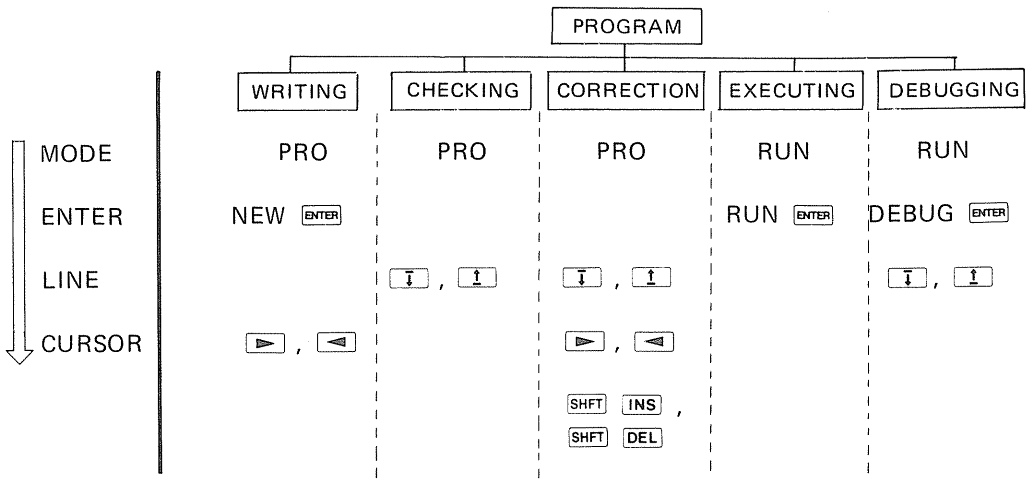
- To interrupt debugging and resume normal operation, use the CONT command.

  C O N T ENTER

[Rapid debugging]

If you hold the ↓ key down for one second or more, the Computer will stop debugging and will start to execute programs as normal.

When you release the ↓ key, the Computer will resume the debugging mode as soon as it finishes executing the present line.



7. Defined programs

When more than two programs are written in the program memory, the second, third, etc. programs then can be executed by the key operation: [R] [U] [N] [line number] [ENTER] . If keys [A] , [S] or [D] are defined by assigning programs to them, you execute those programs by using [SHFT] [A] or [SHFT] [S] in the DEF mode.

To assign programs to certain keys, you must write the labels of those keys at the beginning of the programs you want to assign: for example "A" for key [A] . (You must place the label right after the line number entry.)

● The following 18 keys are definable.

[A], [S], [D], [F], [G], [H], [J], [K], [L], [=],
[Z], [X], [C], [V], [B], [N], [M], [SPC]

Examples of defined programs written in and executed.
PROGRAM 2

Program	Note
10 : "A" : INPUT A , B 20 : C=√(A*A+B*B) 30 : PRINT C 40 : END	Label A $C=\sqrt{a^2+b^2}$: Pythagoras's theorem
50 : "S" : INPUT D 60 : E=4/3*π*D^3 70 : PRINT E 80 : END	Label S $V=\frac{4}{3}\pi r^3$: Volume of sphere
90 : " " : INPUT F , G , H 100 : I=√(F*F+G*G−2*F*G*COS H) 110 : PRINT I 120 : END	Label SPC (space) $C=\sqrt{a^2+b^2-2ab\cos\theta}$: Law of cosine

Writing

Operation	Note
Set to the PRO mode N E W ENTER A colon behind a label can be eliminated. 10 SHFT " A SHFT " SHFT : I N P U T A SHFT , B ENTER 20 C = √ (A * A + B * B) ENTER 30 P R I N T C ENTER 40 E N D ENTER	Label A
50 SHFT " S SHFT " SHFT : I N P U T D ENTER 60 E = 4 / 3 * SHFT π * D SHFT ^ 3 ENTER 70 P R I N T E ENTER 80 E N D ENTER	Label B
90 SHFT " SPC SHFT " SHFT : I N P U T F SHFT , G SHFT , H ENTER 100 I = √ (F * F + G * G − 2 * F * G * C O S H) ENTER 110 P R I N T I ENTER 120 E N D ENTER	Label SPC (Space)

Execution

Operation		Display	Note
Remember to select the DEF mode			
	SHFT A	?	Execution of the program labeled A starts.
(When A = 4)	4 ENTER	?	
(When B = 3)	3 ENTER	5.	Result
	ENTER	>	End
	SHFT S	?	Execution of the program labeled S starts.
(When D = 2)	2 ENTER	33. 51032164	Result
	ENTER	>	End
	D E G R E E ENTER	>	Set to DEGREE Angular mode.
	SHFT SPC	?	Execution of the program labeled SPC starts.
(When F = 12)	12 ENTER	?	
(When G = 14)	14 ENTER	?	
(When H = 30)	30 ENTER	7. 001104508	Result
	ENTER	>	End

To resume execution after being interrupted with an INPUT or PRINT instruction, press ENTER key as shown in the example above.

- When an identical label is given to two or more lines, the lowest line number is executed.
- Inputting an undefined key causes an error. (Error code: 2)

VARIABLES

1. What is a variable?

A variable is a letter (or character combination which represents a memory location in which information can be stored (this information often is called **data** and can be a number or series of characters).

In this Computer variables are divided into fixed memories (26 pieces) and a flexible memory (refer to "Calculations Using Memories" on page 15). The memories store not only numerical values, but they can also store items composed of characters (such as a person's name or item name).

Numerical variable

A data memory is called a numerical variable when it's storing numerical values, and is labeled as A, B, C, A(1) or A(28).

Character variable

A data memory is called a string or character variable when it's storing a sequence of characters, including letters, blanks, numbers, special symbols, and is labeled as A$; B$; C$ or A$(1) (the $ being called a "string" — e.g. "A-string", etc.). (One data memory can contain a maximum of seven characters.)

```
10: INPUT  A$, B$
20: PRINT  A$, B$
   :
```

Operation	Display	Note
Set to PRO mode		
$\boxed{N}\boxed{E}\boxed{W}\boxed{ENTER}$	>	Clears the memory
10 $\boxed{I}\boxed{N}\boxed{P}\boxed{U}\boxed{T}$	10 INPUT_	
$\boxed{A}\boxed{SHFT}\boxed{\$}\boxed{SHFT}\boxed{,}$	10 INPUT A$, _	
$\boxed{B}\boxed{SHFT}\boxed{\$}$	10 INPUT A$, B$_	
$\boxed{ENTER}$	10 : INPUT A$, B$	
20 $\boxed{P}\boxed{R}\boxed{I}\boxed{N}\boxed{T}$	20 PRINT_	
$\boxed{A}\boxed{SHFT}\boxed{\$}\boxed{SHFT}\boxed{,}$	20 PRINT A$,_	
$\boxed{B}\boxed{SHFT}\boxed{\$}$	20 PRINT A$, B$_	
$\boxed{ENTER}$	20: PRINT A$, B$	
Change PRO mode to RUN		
$\boxed{R}\boxed{U}\boxed{N}\boxed{ENTER}$	?	
$\boxed{S}\boxed{M}\boxed{I}\boxed{T}\boxed{H}$	SMITH_	
$\boxed{ENTER}$	?	A$ is loaded with "SMITH"
26438	2 6 4 3 8	
$\boxed{ENTER}$	SMITH 2 6 4 3 8	B$ is loaded with "26438."

Note: If a variable loaded with a numerical value is specified as a character variable, or if a character variable is specified to store a numerical value, an error (error code: 1) occurs. This error will not occur when variable are cleared, that is, when a numerical variable is loaded with 0 or a character variable is loaded with no character.

2. Specifying variables

Fixed memory

1. Specify fixed memories simply by pressing a single key such as $\boxed{\text{A}}$, or two or more key such as $\boxed{\text{B}}$ $\boxed{\text{SHFT}}$ $\boxed{\text{\$}}$.

   Example: $\boxed{\text{A}}$ → Numerical variable A is specified.
   $\boxed{\text{B}}$ $\boxed{\text{SHFT}}$ $\boxed{\text{\$}}$ → Character variable B$ is specified.

2. Fixed memories A through Z or A$ through Z$ are individually given serial numbers 1 through 26, and are specified by inputting codes such as A(1) and A(5) or A$(1) and A$(5).

   Example: A (1) → Numerical variable A is specified.
   A$ (1) → Character variable A$ is specified.
   A (48-25) → Numerical variable A(23), namely W, is specified.
   A$ (3*4) → Character variable A$(12), namely L$, is specified.

- When you use the form A ( ) or A$ ( ), only the integer part of parenthesized value is effective.
- Memories specified in the form of A ( ) or A$ ( ) are called dimension memories.
- You should note that memories A through Z, and A(1) through A(26), A$ through Z$ and A$(1) and through A$(26) actually use the same memory location. For example, A, A(1), A$ and A$(1) all use the same memory location. And E, B(5), E$ and C$(5) all use the same memory location. Only one value or piece of data can occupy any given memory location at one time.
  When B or A(2) is entered, for example, data memory B is specified as a numerical variable; when B$ or A$(2) is entered (the same data memory, namely memory B) is specified as a character variable.

Flexible memory

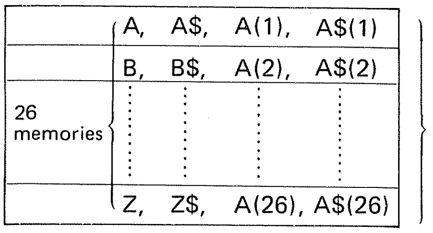
Flexible memory is specified in the form A ( ) or A$ ( ) in the same manner as in 2 above. But when the value in parentheses is smaller than 27, this flexible memory is not specified, since number of fixed memories is 26.

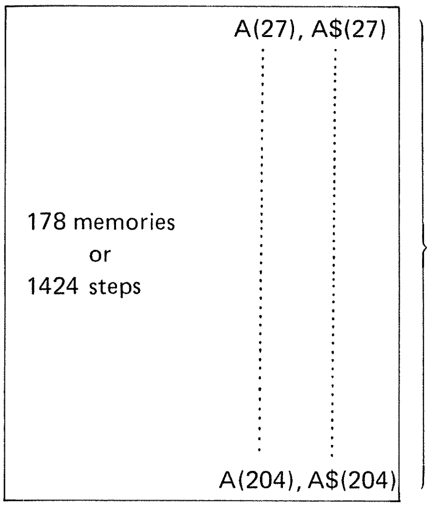
Example: A(27) → Specification of numerical variable A(27) (flexible memory)
A$(19 x 2) → Specification of character variable A$(28) (flexible memory)

Note:

$$\text{Memory}\left\{\begin{array}{l}\begin{array}{l}\text{Data memory}\\ \text{(Variable)}\end{array}\left\{\begin{array}{l}\text{Numerical memory}\\ \quad\text{(Numerical variable)}\\ \text{Character memory}\\ \quad\text{(Character variable)}\end{array}\right.\\ \\ \text{Program memory}\end{array}\right.$$

The memories of the TRS-80 Pocket Computer





Program memory
Any program memory left open (unused) after the programs are stored can also be used for data memories. When used for Data memory it is called a flexible memory. Thus, the number of available memories varies depending on how many steps are used to store programs. So, when using the flexible memory, you should always check in advance how many memory locations are available (by using the MEM command).
(For MEM command, refer to page 73.)
NOTE: 8 steps of program equals one data memory.

Indirect designation
An indirect designation of memories (variables) is a method of designating an arbitrary memory (numerical variable) corresponding to its contents.
The indirect designation is made in the form of A(B) or A$(B); that is, a numerical variable is put in a set of parentheses.
With this method you can specify all data memories (variables) according to their contents – only the integer parts of them are effective.

Example: A(A) → A numerical variable that is given a serial number corresponding to variable A.

A$(A(3)) → A character variable that is given a serial number corresponding to variable A(3), namely C (3rd fixed memory location).

Following are some examples of the advantages of indirect memory designation.

Example: Programs where data is put in variables B through Z.

```
10: INPUT  B, C, D, ……, Z
  :
```

When variables B through Z are directly specified in line 10.

```
10: FOR  A=2TO  26
20: INPUT  A (A)
30: NEXT  A
  :
```

The value of A varies from 2 to 26; programs stored in the variables corresponding to those numbers are repeatedly executed.
In response to the program, variables B through Z are executed in sequence.

The indirect method of designating memories can provide a depth of up to 15 stages by specifying dimensioned memories in sets of parentheses.

Example: When C = 2, B = 6, F = 8,

A (A(A(C))) → Variable H is specified.

$$\underbrace{A(\underbrace{A(\underbrace{A(C)}_{A(2)=B=6})}_{A(6)=F=8})}_{A(8)=H}$$

Note: In designation of dimension memories, when a specified value is below 1 or exceeds the area within which the flexible memory is specified, an error occurs. (Error code: 4)

3. Inputting to variables

You can load the memories (variables) with numerical values or chracters in the following forms:

General form (1) **[Numerical variable] = ⟨ Expression ⟩**
The value of ⟨ expression ⟩ is put in a numerical variable specified on the left side of the above equation.
Note: ⟨ expression ⟩ also covers a numerical variable.

General form (2) **[Character variable] = "Characters"**
Characters (letters, numerals, blanks, symbols, etc.) between quotation (") marks on the right side are put in the characater variable specified by the left side of the equation.
When the number of characters on the right side exceeds 7, only the first seven characters are put in.
Note: When clearing the equation, specify the right side of the equals as ".

General form (3) **[Character variable] = [Character variable]**
Characters stored in the character variable specified by the right side of the above equation are put in the character variable specified by the left side.

Examples: A = 5 * 6 ENTER → The result of 5 * 6, 30, is put in variable A.
A$(27) = "USA" ENTER → Characters "USA" are put in variable A$(27).
B$ = A$(9 * 3) ENTER → Characters stored in variable A$(27) are put in variable B$.

Example: A program that recalls names of fruit corresponding to predetermined name codes 1 through 26 put in the Computer. (Indirect designation)

Key in: code (number)
Display: character (name of fruit)

(The program assumes that you already have stored the fruit names in A$ through Z$).

```
10 : INPUT  A(27)
20 : PRINT  A$ (A(27))
```

⇨

Operation
Set to the PRO mode
$\boxed{N}$ $\boxed{E}$ $\boxed{W}$ $\boxed{\text{ENTER}}$
10 $\boxed{I}$ $\boxed{N}$ $\boxed{P}$ $\boxed{U}$ $\boxed{T}$ $\boxed{A}$ $\boxed{(}$ 27 $\boxed{)}$ $\boxed{\text{ENTER}}$
20 $\boxed{P}$ $\boxed{R}$ $\boxed{I}$ $\boxed{N}$ $\boxed{T}$
$\boxed{A}$ $\boxed{\text{SHFT}}$ $\boxed{\$}$ $\boxed{(}$ $\boxed{A}$ $\boxed{(}$ 27 $\boxed{)}$ $\boxed{)}$ $\boxed{\text{ENTER}}$

Contents of data memories

A	ORANGE	1
B	APPLE	2
C	BANANA	3
D	MELON	4

When data is input by manual operation

⇨

Operation
Set to the RUN mode
$\boxed{A}$ $\boxed{\text{SHFT}}$ $\boxed{\$}$ $\boxed{=}$ $\boxed{\text{SHFT}}$ $\boxed{''}$
$\boxed{O}$ $\boxed{R}$ $\boxed{A}$ $\boxed{N}$ $\boxed{G}$ $\boxed{E}$ $\boxed{\text{SHFT}}$ $\boxed{''}$ $\boxed{\text{ENTER}}$
$\boxed{B}$ $\boxed{\text{SHFT}}$ $\boxed{\$}$ $\boxed{=}$ $\boxed{\text{SHFT}}$ $\boxed{''}$
$\boxed{A}$ $\boxed{P}$ $\boxed{P}$ $\boxed{L}$ $\boxed{E}$ $\boxed{\text{SHFT}}$ $\boxed{''}$ $\boxed{\text{ENTER}}$

Operation	Display	Note
Set to the RUN mode		
$\boxed{R}$ $\boxed{U}$ $\boxed{N}$ $\boxed{\text{ENTER}}$	?	
2	2_	Inputting a number
$\boxed{\text{ENTER}}$	APPLE	Display of corresponding fruit's name

4. Recalling the contents of variables

To recall the contents of memories (variables), use the following form:

General form **[variable]** $\boxed{\text{ENTER}}$

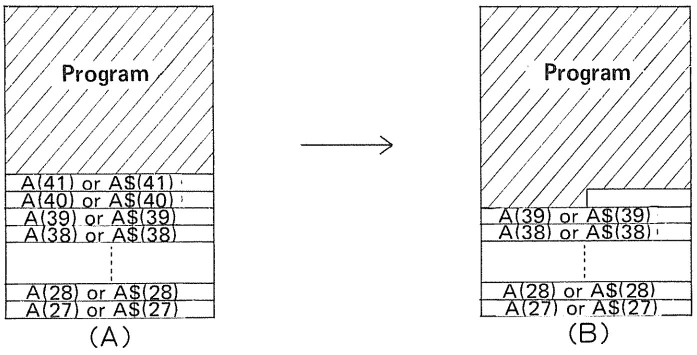
Example: When 120 and "GOOD" are stored in A and B$ respectively.

Operation	Display	Note
RUN mode $\boxed{A}$	A_	Designation of numerical variable
$\boxed{\text{ENTER}}$	120.	Display of number stored there
$\boxed{B}$ $\boxed{\text{SHFT}}$ $\boxed{\$}$	B$_	Designation of character variable
$\boxed{\text{ENTER}}$	GOOD	Display of characters stored there

Note: If you recall the contents of cleared memories (variables) using a NEW or CLEAR command, "0." is displayed when they are specified as numerical variables and a blank space appears when recalling a cleared memory character variable.
If a memory is loaded with −0 and specified as a character variable, an error occurs (Error code: 1).
Also, if you clear a variable specified as given in General Form (2) or (3) and then attempt to recall it as a numerical variable, an error also occurs (Error code: 1).

Comments:

Vacant program memory areas are available to be used as flexible memory. Programs can be written into the program memory until its capacity is exceeded. Accordingly, keep in mind that if you edit programs (insertion, deletion, correction) this changes program steps or content, **and this may affect the number of flexible memory space available.**
For example, when programs are written as shown in illustration (A) below, recalling or writing the contents of flexible memories A(42) or A$(42) results in an error (Error: 4) because no flexible memories exist corresponding to such codes.



Also, when the arrangement of progras written (added) as shown in illustration (A) is changed to be as shown in illustration (B), flexible memories corresponding to A(41) or A$(41) and A(40) or A$(40) disappear. Thus if you try to recall or write to these memories, an error occurs (Error: 4).

By contrast, a string of programs stored as shown in illustration (B) can be shortened as shown in illustration (A) (by editing), the number of memories available to be used as flexible memories increases in proportion to the degree of the change.
(One flexible memory corresponds to 8 program steps.)

Note: When you try to recall flexible memory which is loaded with nothing, unexpected displays might appear or an error might occur (Error: 1).

PROGRAM STATEMENTS

In this section we use the following form for various terms, [variable], [numerical variable], [character variable] and ⟨ expression ⟩.

[Variable]:	General name for numerical and character variables.
[Numerical variable]:	General name for fixed memories defined by A through Z and dimensioned memories defined in the form of A ( ).
[Character variable]:	General name for fixed memories defined by A$ through Z$ and dimensioned memories defined in the form of A$ ( ).
⟨ Expression ⟩:	Operational expression composed of elements of ⟨ expression ⟩ as we described on page 18, covering also [numerical variable].

1. LET statement

The variable name on the left is assigned the value of the constant or expression on the right.
The TRS-80 Pocket Computer does not require LET except when it is used with an IF statement. (For IF statement, consult page 60.)

General form (1) **LET [Numerical variable] = ⟨ Expression ⟩**

Example: LET A = 5 ∗ 3

Example:	LET A = 123	Instruction to put 123 in A (LET can be omitted as in the following example.)
	A(30) = 3 ∗ 6	Instruction to put 18 in A (30).
	A (2 ∗ B) = C + D	Instruction to put the value of C + D in A (2 * B).

General form (2) **LET [Chracter variable] = "Character"**

Example: LET Z$ = "BASIC"

Characters between quotation marks are put in the character variable specified by the left side. **When the length of a string of characters on the right side exceeds seven characters, only the first seven characters are put in (the excess is discarded).**

General form (3) **LET [Character variable] = [Character variable]**

Example: LET A$(25) = Z$

Characters stored in the character variable specified on the right side are placed in the character variable specified by the left side.

Example:	A$ = "NON"	Instruction to place "NON" in A$.
	A$(28) = "DATA ?"	Instruction to place "DATA ?" in A$(28).
	C$ = A$	Instruction to place characters stored in A$ in C$.

● General forms (1) thru (3) can be put on a single program line by dividing them with commas (,).
In this case, LET must not be used after a comma (,).

Example: 10 : LET A = 2 , B = 7 , C$ = "A = 2, B = 7"

2 is placed in A, 7 in B and "A = 2, B = 7" in C$.

2. INPUT statement

This is an instruction which requests manual input data during program execution.

General form (1) **INPUT [Variable], [Variable], · · ·**

Example: INPUT A, B, C,

The instruction causes the Computer to stop program execution and display a question mark "?". When you respond by inputting data to the Computer and pressing ENTER key, the Computer stores data in a specified variable.
The Computer carries out this process the number of times that corresponds to the number of variables specified after an INPUT statement.

General form (2) **INPUT "Character", [Variable], "Character", [Variable], · · ·**

Example: INPUT "A =", A, "B =", B, · · ·

This form of instruction causes the Computer to stop program execution and display a message instead of "?."

Example

Programming	Note
10 : INPUT A , B	General form (1)
20 : INPUT "C=" , C , "DATA D=" , D	General form (2)

Execution

Operation	Display	Note
Set to the RUN mode		
R U N ENTER	?	Display in general form (1)
3	3 _	Inputting data
ENTER	?	3 is placed in A. Display in general form (1)
4	4 _	Inputting data
ENTER	C=	4 is placed in B. Display in general form (2)
5	5 _	Inputting data.
ENTER	DATA D=	5 is placed in C. Display in general form (2)
6	6 _	Inputting data
ENTER	>	6 is placed in D.

General form (3) **INPUT "Character"; [Variable], "Character"; [Variable] · · ·**

Example: INPUT "C ="; C , "D ="; D ,

In general form (2), if data is input after the display of a message, that message disappears. In general form (3), if a semicolon (;) is entered after "Character", message does not appear and data can be displayed following that "character."

Example

```
10: INPUT "DATA E=";E
```

Operation	Display	Note
Set to the RUN mode		
R U N ENTER	DATA E=_	A display indicating that the Computer is waiting for input in general form (3).
5	DATA E=5_	Inputting data (⟨expression⟩).
* 6	DATA E=5*6_	
ENTER		30 is placed in E.

- General forms (1) to (3) can be intermixed with each other.
  Example: INPUT A, "B =", B, "WHO ?" ; C$
- There is no limit on the length of characters between quotation marks that specify a message in general forms (2) and (3).
- You can make a correction by pressing CL key in the course of inputting data.
  However, in general forms (1) and (2), pressing the CL key displays a question mark "?" alone; with general form (3) the entire message is displayed again.
  If you pressed ENTER key and an error occurs, press CL key; either "?" alone or the entire message will appear as noted above.

Note: [Variable] specified in general forms (2) and (3) must be a fixed memory, a dimensioned memory (that is specified by a code, such as A (30) or A (B), that contains a fixed memory inside the parentheses, or an integer with no sign.)
You must be aware that you cannot use a form such as A(A(30)) or A(5 * 9). You must also note that indexing the SHFT and " keys when [variable] is a character variable in general forms (1) to (3) will prevent you from using a character variable in the subsequent inputs.

[Skip operation]

If your press the ENTER key without inputting data in response to INPUT statements, the Computer will skip the remaining statements on that line and go to the next line.

Example: PROGRAM 3 ⟨ Average ⟩

Programming	Note
10: "A" :CLEAR	Definition: memory is cleared.
20: INPUT "DATA A=" ; A : B=B+A :	Inputting data: sum
C=C+1 :GOTO 20	Counting data number: Jump
30: D=B/C	Computing average
40: PRINT "AVERAGE= " ; D	(PRINT statement in general form (5))
50: END	End

The above program will find average; you must input data, the program sums the data and counts the data number in line 20, on completion of data input, the Computer will skip the statements responsible for summing and counting, and will drop down to line 30.

Execution Data 12, 24, 19, 23

Operation	Display	Note
Set to the DEF mode		
SHFT A	DATA A=_	Execution begins.
12 ENTER	DATA A=_	Inputting data
24 ENTER	DATA A=_	
19 ENTER	DATA A=_	
23 ENTER	DATA A=_	
ENTER	AVERAGE= 19. 5	A skip operation occurs by pressing the ENTER key without entry of data.
ENTER	〉	

3. PRINT statement

The PRINT statement is an output instruction to display calculation results.

This instruction commands the Computer to stop program execution after it has displayed information specified by this instruction.
To restart program execution, simply press the ENTER key without entry or have the Computer execute a CONT command. You need not input anything.

General form (1) PRINT 〈 Expression 〉

Example: PRINT 123 + 456
PRINT A

General form (2) PRINT "Character"

Example: PRINT "Character"

This form commands the Computer to display characters between two quotation marks.
Length of a string of characters is limited to 80 (capacity of one line).

General form (3) PRINT [Character variable]

Example: PRINT A$

This form causes the Computer to display the contents of [character variable].

Example:

Programming	Note
10: INPUT A$	Input instruction
20: PRINT 5*6	General form (1)
30: PRINT "PROGRAM A"	General form (2)
40: PRINT A$	General form (3)

Operation	Display	Note
Set to the RUN mode		
R U N ENTER	?	Execution of input instruction
W O R L D ENTER	30.	Inputting "WORLD" Display in general form (1)
ENTER	PROGRAM A	Display in general form (2)
ENTER	WORLD	Display in general form (3)

General form (4) **PRINT** $\left\{\begin{array}{l}\langle \textbf{Expression} \rangle \\ \textbf{"Character"} \\ \textbf{[Character variable]}\end{array}\right\}$, $\left\{\begin{array}{l}\langle \textbf{Expression} \rangle \\ \textbf{"Character"} \\ \textbf{[Character variable]}\end{array}\right\}$

Example:
```
PRINT A, B
PRINT "A=", A
```

The 24-digit display is divided into two sections: right and left 12-digit sections. The right section displays the first set of (expression), "character" and [character variable] ahead of comma (,), and the left section another set.
The value of ⟨ expression ⟩ is displayed in 12 digits (right or left sections).
When the whole numerical data cannot be displayed in 12 digits, the least significant decimal-digits are cut off, and when the length of a string of characters exceeds 12 digits, only the first 12 digits are displayed.

Example: Polar coordinates → rectangular coordinates program
This program converts polar coordinates (r, θ) into rectangular coordinates (x, y).

```
10: INPUT R, C
20: X=R*COS C: Y=R*SIN C
30: PRINT X, Y
40: END
```

Operation	Display	Note
Set to the RUN mode DEG		
R U N ENTER	?	
(When r = 12) 12 ENTER	?	
(When θ = 30°) 30 ENTER	$\underbrace{10.\ 39230485}_{\text{Value of } x}$ $\underbrace{6.}_{\text{Value of } y}$	Two numerical values are displayed at the same time.

General form (5) **PRINT** $\left\{\begin{array}{l}\langle\textbf{Expression}\rangle\\ \textbf{"Character"}\\ [\textbf{Character variable}]\end{array}\right\}$; $\left\{\begin{array}{l}\textbf{"Character"}\\ [\textbf{Variable}]\end{array}\right\}$; · · · $\left\{\begin{array}{l}\textbf{"Character"}\\ [\textbf{Variable}]\end{array}\right\}$

Example: PRINT "A ="; A ; "B =" ; B ; · · ·
PRINT A$; B ; C$; C ; · · ·

The form provides a concurrent display of multiple information; information items are separated with a semicolon (;).

Example:

Programming	Note
10: "A" :CLEAR	
20: INPUT "DATA=" ; A	Input instruction
30: B=B+A : C=C+1	Sum: count of data number
40: PRINT " TOTAL = " ; B ; " QTY = "; C	Total of sums: display of data number
50: GOTO 20	Jump to line 20

Operation	Display	Note
Set to the DEF mode		
SHFT A	DATA=_	
456 ENTER	TOTAL=456. QTY=1.	Display in general form (5)
ENTER	DATA=_	
789 ENTER	TOTAL=1245. QTY=2.	Display in general form (5)

● When the number of characters to be displayed exceeds 24, only the first 24 characters are displayed.

Note: The 2nd and subsequent display items can be only "character" and "variable"; the "variable" must be a fixed memory, or a dimensioned memory that has a fixed memory in parentheses or that is specified by an integer (up to 204) with no sign [such a manner as A (30) or A$ (C)].

4. PAUSE statement

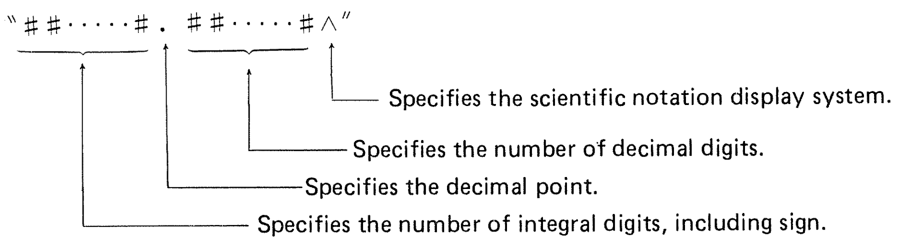
The PAUSE statement is an output instruction like a PRINT statement. But, these statements are different in function. The PRINT statement causes the machine to temporarily stop program execution after it has executed (displayed) a given instruction, while the PAUSE statement forces it to display a specified item for about 0.85 sec., and **restart program execution automatically.** The definition form (general form) of PAUSE statement is the same as for the PRINT statement.

5. USING statement

The USING statement is an instruction to specify a PRINT or PAUSE display format for numerical data.

General form (1) **USING** "##...#. ##···#∧"

This form of USING statement specifies a format depending on the number of "#" between quotation marks, • and ∧.


"##·····#. ##·····#∧"
Specifies the scientific notation display system.
Specifies the number of decimal digits.
Specifies the decimal point.
Specifies the number of integral digits, including sign.

General form (2) **USING (Statement end)**

"Statement end" would be an ENTER or a colon (:).

This form commands the Computer to cancel the format specification. That is, a PRINT or PAUSE statement used after this form of instruction will display numerical information in the same format as a manual calculation.

Example: Display 123.4567891 with PRINT or PAUSE statement.

```
10 : A=123.4567891
20 : USING "Specified format"
30 : PRINT A
```

Specified format	Display
Cancel	123. 4567891
#	
##	Error (Error code: 6)
###	
####	123
#####	123
####.	123.
######. #	123. 4
#######. ###	123. 456
####. #######	123. 4567891
####. ##########	123. 456789100
##. ###^	1. 234E 02
####. ######^^^	1. 234567E 02
###. ##########^	1. 234567891E 02
. ########: ^	1. 234567891E 02

When the scientific notation display is specified, a 2-digit display is always assured regardless of how many integral digits you specify. Besides, the number of specified " ∧ " does not affect the display at all.

● When numerical data cannot be displayed in a specified format, an error (Error code: 6) results.

General form (3) **(a)** $\left\{\begin{matrix}\text{PRINT}\\\text{PAUSE}\end{matrix}\right\}$ **USING "Format" ;·····**

(b) $\left\{\begin{matrix}\text{PRINT}\\\text{PAUSE}\end{matrix}\right\}$ **USING ;·····**

Example: PRINT USING "###. ##"; A

Formats can be specified (a) or canceled (b) by USING even in a PRINT or PAUSE statement. In this case the part after the USING statement is defined with a semicolon (;).

Example:

```
10:A=-123. 456
20:PAUSE USING "####" ;A
30:PAUSE USING "####. #" ; "A=" ,A
40:PAUSE  "A=" ,USING"####. #" ;A
50:PAUSE A,USING  "####. ##" ;A
60:PAUSE A;USING  "####" ;A
70:PAUSE USING ;A
```

Operation	Display	Note
Set to the RUN mode		
R U N ENTER	- 123	
	A= -123. 4	
	A= -123. 4	
	-123. 45 -123. 45	
	-123. 45-123	
	-123. 456	

Note: If USING is put in a PRINT or PAUSE statement of general form (4), a display of numerical data occurs according to the format in which the latter half of the numerical value of those statements is displayed. Thus, two numerical data are displayed in the format specified for the latter data, even though their display format is individually specified. (This is true of line 50 in the above example.)

6. GOTO statement

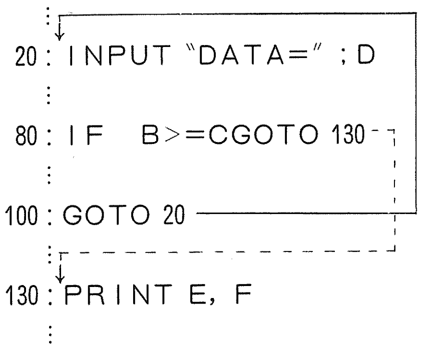
The GOTO statement is an instruction to make program execution jump to a specified line.

General form (1) GOTO 〈 Expression 〉

Example:
```
GOTO 10
GOTO 5*9
GOTO A
```

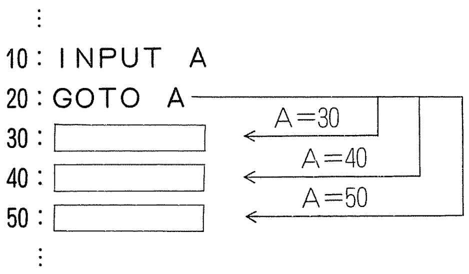
This form makes program execution jump to a line that corresponds to the value of 〈expression〉. The value of "expression" is effective in its integer part only (limited to 1 through 999). Other values cause an error (Error code: 2).

Example:



```
   :
 20: INPUT "DATA=" ; D
   :
 80: IF  B>=CGOTO 130
   :
100: GOTO 20
   :
130: PRINT E, F
   :
```

Example: Indirect jump



```
   :
10: INPUT  A
20: GOTO  A
30:
40:
50:
   :
```

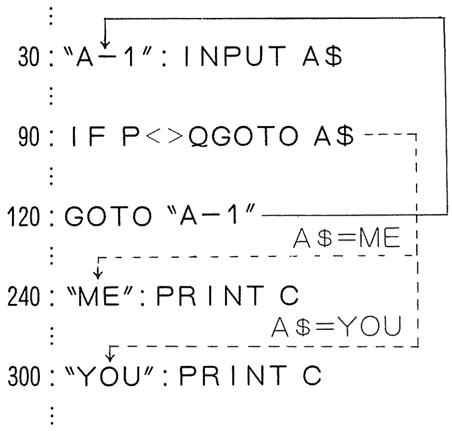
Jumping occurs in accordance with the contents of A.

General form (2) **GOTO** $\left\{ \begin{array}{l} \text{"Character"} \\ \text{[Character variable]} \end{array} \right\}$

Example: GOTO "AB"
GOTO A$

This form causes a jump to a label whose contents are the same as those of "character" or [character variable]. The length of "character" and label is limited to seven characters. Thus, if you use a label of more than 7 characters, the Computer ignores all after the first 7.

Example:



```
 :
 30 : "A-1": INPUT A$
 :
 90 : IF P<>QGOTO A$
 :
120 : GOTO "A-1"
 :
240 : "ME": PRINT C
 :
300 : "YOU": PRINT C
 :
```

In the execution of "GOTO A$" in line 90, when the contents of A$ are "ME", a jump to label "ME" occurs, and if "YOU", a jump to label "YOU" occurs.

Note: No statement can follow a GOTO statement; a line with a GOTO statement must have the GOTO statement at the end of the line.

7. IF statement

IF is a statement which uses conditions to determine action (condition can be "larger/smaller" decision, "equal" decision, "not-equal" decision, etc.).

General form (1) **IF 〈 Expression 〉 Logic operation 〈 Expression 〉 Execution statement**
Logic operator: <, <=, =, >, >=, <>

If the relational expression that follows IF is affected [if the logical operation results in 1 (true)] the next statement (instruction) is executed, and if not [if the logical operation results in 0 (false)] the program skips the next execution statement, going to the following line.

Example:

```
   :
40 : IF  A*B>=C  PAUSE  A*B : GOTO  90
50 : A=A+1
   :
90 : A=A+B
   :
```

If A ✱ B >= C, the program begins by executing the next statement "PAUSE A ✱ B".
If A ✱ B < C, the program begins by executing line 50, "A = A + 1", skipping "PAUSE A ✱B : GOTO 90".

Note: 1. If you want to execute a LET statement directly after executing an IF statement, be sure to add a LET.

Example: IF B > C LET B = B + 1

2. When a GOTO statement follows an IF statement, the former can be defined with THEN statement (in this case the THEN statement has the same function as the GOTO statement).

Example: IF B >= C THEN 50 ⟷ IF B >= C GOTO 50

General form (2) IF ⟨ Expression ⟩ Execution statement

If its value is larger than 0, the ⟨ expression ⟩ is judged to be true, and the next statement will be executed.
If its value is 0 or smaller than 0, the ⟨ expression ⟩ is judged to be false, and the program execution goes to the next line.

Example:

```
   :
30: IF  AGOTO  80
40: A=B*C
   :
```

If A > 0, "GOTO 80" is executed.
If A ≦ 0, "A = B ✱ C" on line 40 is executed.

General form (3) IF $\left\{\begin{array}{l}\textbf{"Character"}\\ \textbf{[Character variable]}\end{array}\right\} = \left\{\begin{array}{l}\textbf{"Character"}\\ \textbf{[Character variable]}\end{array}\right\}$ **Execution statement**

Example: IF A$ = "ABC"
IF A$ = B$

The Computer compares the contents of "character" or [character variable] on both sides of =; if they are equal, it executes the next statement; if they are not equal, it moves directly to the next line.

Example:

```
   :
30: IF A$= "GUARD" GOTO  100
40: INPUT  A$
   :
```

If the contents of A$ are "GUARD," "GOTO 100" is executed.
If not, "INPUT A$" in line 40 is executed.

● When the length of "character" is more than 7, only the first 7 characters are used for comparison, and the excess is ignored.

General form (4) IF [Character variable] Execution statement

If any character is stored in the [character variable], the next statement is executed. If nothing is there, the program moves to the next line.

8. GOSUB statement, RETURN statement

When you use a certain processing procedure a number of times, you will have a more efficient program if you handle these procedures as a sub-routine.
When a program comes to a GOSUB statement, the Computer moves to a specified line or label and executes the programs stored there.
At the end of a sub-routine you must add a RETURN statement, which returns program execution to the statement immediately after the GOSUB statement.
A subroutine can contain other subroutines, up to a maximum of 4 levels.

(1) GOSUB statement

General form (1) **GOSUB ⟨ Expression ⟩**

```
Example:   GOSUB 10
           GOSUB  A
```

General form (2) **GOSUB** $\left\{ \begin{array}{l} \text{"Character"} \\ \text{[Character variable]} \end{array} \right\}$

```
Example:   GOSUB "ABC"
           GOSUB A$
```

The RETURN statement must be at the end of a line (it can not be followed by any other another statement on the same line.

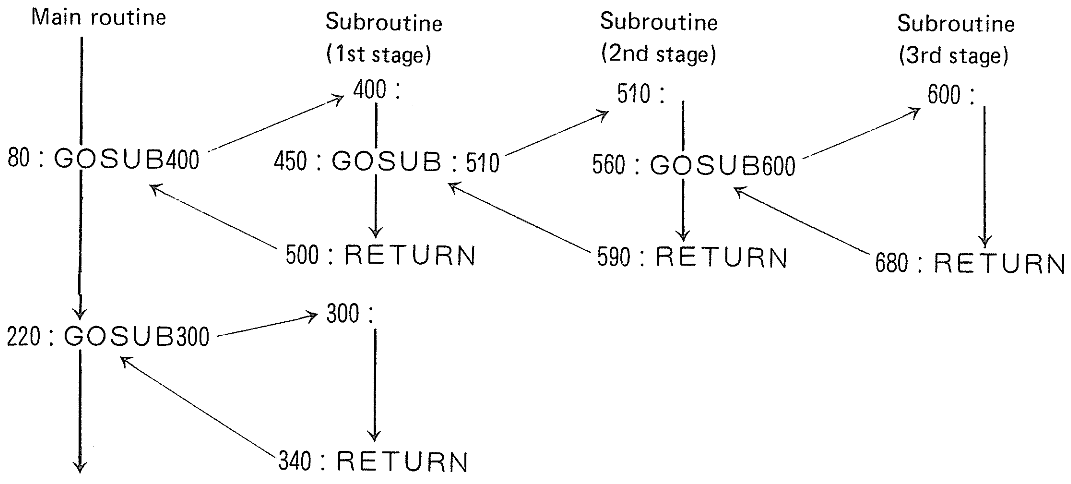
(2) RETURN statement

General form **RETURN**

The RETURN statement must be at the end of a line (it can not be followed by any other statement on that line).

Note: An error (error code: 3) occurs, if a RETURN is used without a GOSUB earlier on.
Also be sure to put END statement on end of main routine, else computer will go into subroutine and error 3 will occur.

Example 1



Example 2: PROGRAM 4 ⟨ Approximate definite integral by Simpson's method ⟩

Formula

Compute a definite integral by using Simpson's rule.

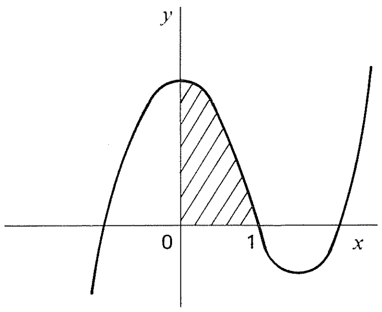
$$S = \int_{x_0}^{x_{2p}} f(x)dx = \frac{h}{3}[(y_0 + y_{2p}) + 4(y_1 + y_3 + \cdots\cdots + y_{2p-1}) + 2(y_2 + y_4 + \cdots\cdots + y_{2p-2})]$$

$$h = \frac{(x_{2p} - x_0)}{2p} \qquad \boxed{\text{p : number of divisions}}$$

[Example]

$$y = x^3 - 2x^2 - x + 2 = ((x-2)x-1)x+2$$

$$\int_0^1 y\,dx = \frac{13}{12}$$



Write in the function, as a subroutine, after line 500.

Set to PRO mode (by pressing MODE key).

500 Y = ((x − 2) ∗ x − 1) ∗x + 2 ENTER

510 RETURN ENTER This ends writing.

Next, change to DEF mode, and execute.

Programming	Note
10: "A" : INPUT "X0= "; D,"X2P= "; E, "P= "; F 20: B = (E − D) / 2 / F 30: A = 0: X = D: GOSUB 500 40: A = Y + A: X = X + B: GOSUB 500 50: A = Y ∗ 4 + A: X = X + B: GOSUB 500 60: A = Y + A: F = F − 1 70: IF F <> 0 GOTO 40 80: C = A ∗B / 3 90: BEEP 3: PRINT "ANS.", C 100: END 500: Y = ((X − 2) ∗ X − 1) ∗ X + 2 510: RETURN	

Some Notes to help you understand this program:
Line 30 sets memory "A" to accumulate 'Y' values. Memory "X" is to be 'X'. Since the initial value of 'X' is in memory "D", first input value of "D" to "X". The GOSUBs to calculate first 'Y' value.
Line 40 adds calculated 'Y' to memory "A". 'X' is incremented by 'H'. Again GOSOBs to calculate next 'Y' value. Now, the order of 'X' and 'Y' is of odd number.
According to the Program we have to multiply by 4. So line 50 multiplies by 4.
'X' is incremented and GOSUBs.

Line 60 adds the 'Y' to "A". Now we are in even number order. Why don't we multiply by two? See second portion of line 60 and line 70. Memory "F" (in which the number of division is stored) is decremented by one. Line 70 compared result with zero. If not zero, we've not yet reached the last 'X' and 'Y' and returns to line 40 where 'Y' is added again I.E. multiplied by two. If F=O, then we reached the last, and according to the program the last one should not be multiplied by two, so program goes to next line and does the last calculation (multiply by 3/H).

Operation	Display	Note
Set to the DEF mode		
SHFT A	X0 =	
0 ENTER	X2P =	(x_0)
1 ENTER	P =	(x_{2p})
20 ENTER	ANS. 1.083333333	(P)

When the ENTER is pressed at this step, the Pocket Computer starts the calculation while displaying the "RUN" symbol and about 40 seconds later, an answer will be displayed with 3 beep sounds.

Debugging the program will help you understand its execution process. Try it.

9. FOR statement, NEXT statement

When you need to execute identical programs repeatedly, or when you need to solve a calculation equation repeatedly by replacing only the values of variables contained in it, the FOR-NEXT statement will provide an efficient method. (Of course you can combine it with the decision function capability of GOTO and IF statements.)

General form (1) **FOR [Numerical variable] = ⟨ Expression 1 ⟩ TO ⟨ Expression 2 ⟩**
NEXT [Numerical variable]

Example:
```
FOR A = 1 TO 26
  :
NEXT A
```

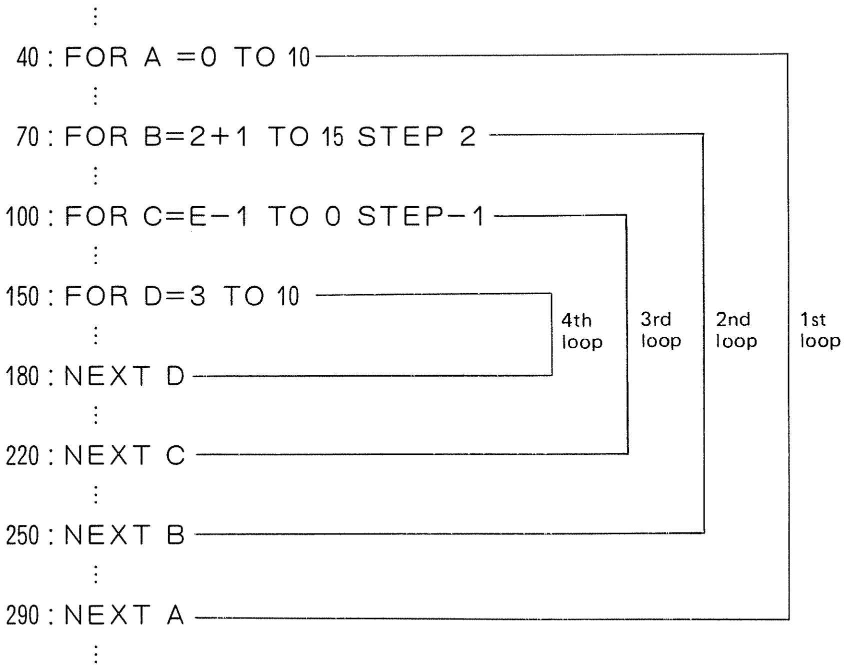
The FOR and NEXT statements (plus numerical variables) are used as a pair, and instructions between these statements are repeatedly executed the specified number of times.
In the first execution, the value of ⟨ expression 1 ⟩ is stored as the initial value in a specified numerical variable.
When the Computer comes to the NEXT statement, it adds an increment of 1 to the numerical variable and executes the instructions between the FOR and NEXT statements and loops back to the FOR over and over until the numerical variable is equal to (or more than) the value of ⟨ expression 2 ⟩.

General form (2) **FOR [Numerical variable] = ⟨ Expression 1 ⟩ TO ⟨ Expression 2 ⟩ STEP ⟨ Expression 3 ⟩**
NEXT [Numerical variable]

Example:
```
FOR A = 1 TO 26 STEP 2
  :
NEXT A
```

This form differs only in that the STEP increment is a value other than 1 ⟨ expression 3 ⟩.
In program execution, a set increment is added to the [numerical variable] every execution. If the value of ⟨ expression 3 ⟩ is negative, program execution occurs repeatedly until the value of the [numerical variable] is equal to or smaller than the value of ⟨ expression 2 ⟩.

- Only the integer values of ⟨ expression 2 ⟩ and ⟨ expression 3 ⟩ are effective and are limited to less than three digits. When the value of ⟨ expression 3 ⟩ is 0, an error (Error 1) occurs.
- A FOR-NEXT statement can have up to 4 levels (or loops as to they are often called).

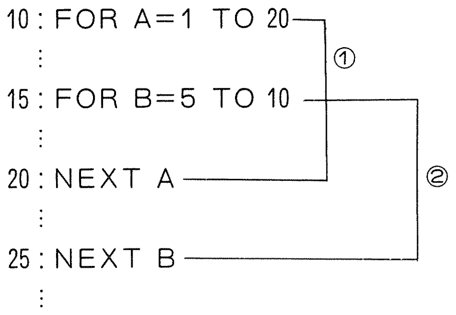


```
      :
 40 : FOR A =0 TO 10
      :
 70 : FOR B=2+1 TO 15 STEP 2
      :
100 : FOR C=E-1 TO 0 STEP-1
      :
150 : FOR D=3 TO 10
      :
180 : NEXT D
      :
220 : NEXT C
      :
250 : NEXT B
      :
290 : NEXT A
      :
```

Note: When using memory locations (variables) with FOR-NEXT statements, use 23 (W) and after for fastest execution.

In the following cases, an error will occur during execution.

⟨ **Crossing** ⟩

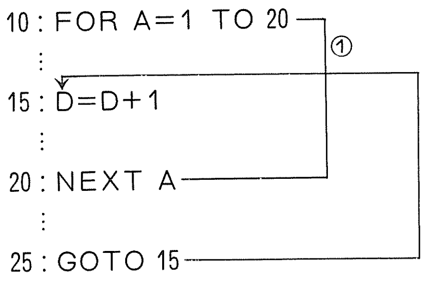


```
10 : FOR A=1 TO 20
     :
15 : FOR B=5 TO 10
     :
20 : NEXT A
     :
25 : NEXT B
     :
```

Crossing of (1) and (2)

In this case, the FOR-NEXT statement of ① will be executed, an error (error 4) will occur in the execution of line 25 because "FOR B = 5 TO 10" on line 15 can not be incremented (NEXT B can never be executed).

Joining in mid course



```
10 : FOR A=1 TO 20
   :
15 : D=D+1
   :
20 : NEXT A
   :
25 : GOTO 15
```

Line 25 joins line 15 with the ① loop

The result is that program execution moves from line 25 to 15, and generates an error (error 4) code as soon as it reaches line 20.

Example: **Program 5 〈 Decision of order 〉**

Programming	Note
10: "A" CLEAR :A=5	Label A
20: INPUT "DATA=";D	
30: IF D=E99GOTO 50	Input program
40: A (A) =D: A=A+1 : GOTO 20	
50: FOR B=5TO A-1	
60: FOR C=B+1TO A-1	
70: IF A (B) 〉=A (C) GOTO 110	
80: D=A (B)	
90: A (B) =A (C)	② ①
100: A (C) =D	
110: NEXT C	
120: NEXT B	
130: "B" FOR B=5TO A-1	Label B
140: BEEP 2: PRINT B-4 , A (B)	③ Output program
150: NEXT B	
160: BEEP 5: END	

①, ② and ③ indicate the loops of FOR·NEXT statements.

Program debugging will help you understand the process of program execution caused by a FOR-NEXT statement.

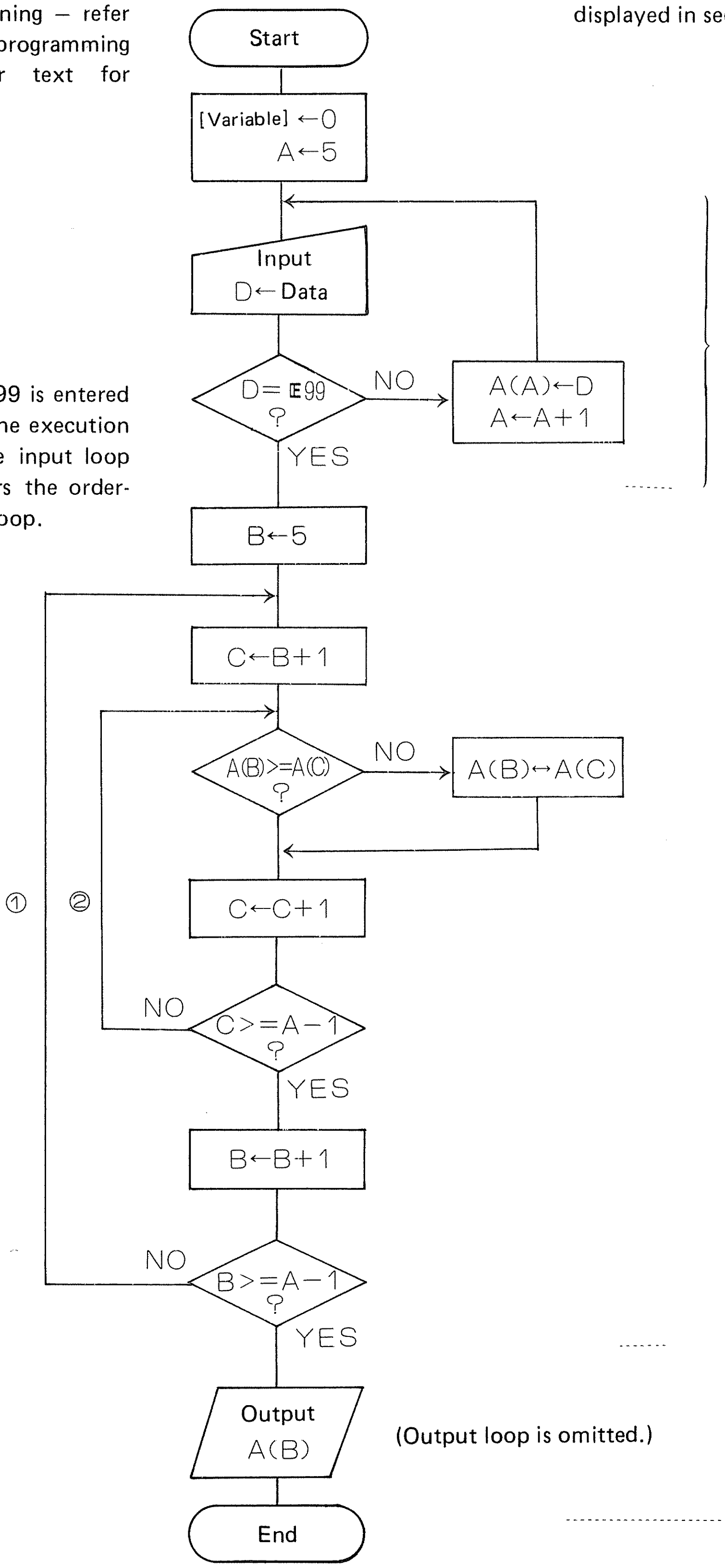
Flow chart

(often flow charts are used to aid in understanding the flow of program execution; each symbol shape has its own meaning – refer to a programming guide or text for details)

Decision of order

Input numerical data are arranged in order of their magnitudes and displayed in sequence.

Note:
When IE 99 is entered as data, the execution leaves the input loop and enters the order-decision loop.



Preparation

Input loop

(Output loop is omitted.)

10. STOP statement

This statement is an instruction to temporarily stop program execution.

General form **STOP**

When the Computer executes the STOP statement, it displays a BREAK message together with line number, and stops program execution.
Manual operation is then possible. To restart program execution, use a CONT command.

Example:

```
100 : C=3*7
200 : STOP
300 : PRINT "C=" ; C
       :
```

Example:
When the STOP statement is executed on line 200, the Computer displays BREAK as shown below.

Operation	Display	Note
	BREAK AT 200	Display of BREAK message
C	C_	When program execution stops due to STOP statement, Manual calculations are possible.
ENTER	21.	
4 * 8 ENTER	32.	
C O N T	CONT_	CONT command restarts program execution.
ENTER	C=21.	

11. END statement

The END statement is an instruction to end program execution.

General form **END**

The execution of this instruction causes the Computer to end program execution and display the prompt symbol. When you have multiple programs stored in the Computer, use END at the end of each program to prevent one program from flowing into the next one in sequence.

12. BEEP statement

The BEEP statement is an instruction to force the Computer to make a beep sound.

General form **BEEP 〈 Expression 〉**

Example: BEEP 10
BEEP A

The Computer beeps the number of times defined by the value of 〈 expression 〉.
(This value is effective only in its positive integral part.)

13. CLEAR statement

The CLEAR statement is an instruction to clear all data memories: fixed and flexible memories.

General form **CLEAR**

● NOTE: **Program and reserve memories are protected.**

14. DEGREE, RADIAN, GRAD statements

These statements are instructions to specify the unit of angle for the input of trigonometric functions (SIN, COS, TAN) and for the output of inverse trigonometric functions (ASN, ACS, ATN).

(1) DEGREE

The statement sets the Computer's unit of angle calculation to the degree mode.

General form DEGREE

(2) RADIAN

The statement sets the radian mode.

General form RADIAN

(3) GRAD

The statement sets the grad mode.

General form GRAD

$90^\circ = \frac{\pi}{2}$ [rad] $= 100^g$

15. AREAD statement

The AREAD statement is an instruction to automatically store numerical value, the value of ⟨ expression ⟩ or "character" in a specified variable that has been displayed before the start of program execution. This instruction is operational only in the DEF mode.

General form AREAD [Variable]

Example: AREAD A
AREAD A$

This instruction is skipped, if not present at the head of a definable program.

Example: Compound-interest computation program.

Writing

Programming	Note
`10: "A" :AREAD I`	Inputting of interest rate (%)
`20: I=I/100`	
`30: END`	
`40: "S" :AREAD N`	Inputting of term
`50: END`	
`60: "D" :AREAD P`	Inputting of principal
`70: END`	
`80: "F" :F=P* (1+I) ^N`	Computation of principal plus interest
`90: PRINT F`	Display of principal plus interest
`100: END`	

Execution

Operation	Display	Note
Set to the DEF mode		
6.8 SHFT A	>	Interest rate 6.8%
4 SHFT S	>	Term 4 years
5000 SHFT D	>	Principal DLS 5,000
SHFT F	6505. 115547	Principal plus interest
5 SHFT S	>	Term changed to 5 years
SHFT F	6947. 463404	Principal plus interest

16. REM statement

The REM statement is not an execution statement, it is a handy way to add notes in a program for reference. The program execution skips the notes, or REMarks that follows this instruction, going onto the following line.

General form REM ⟨ Note ⟩

Use this statement when you want to insert lines or space between programs for dividing them clearly (and yet have no effect on their execution).

Example:

```
    :
80 : REM * OUTPUT PROGRAM *
    :
```

COMMAND STATEMENTS

In addition to program execution statements, the TRS-80 Pocket Computer can process instructions capable of starting program execution or displaying program contents. These instructions are called commands. These commands are not executed until you press the ENTER key.

1. RUN command

The RUN command functions only in the RUN or DEF mode. Use it to start program execution.

General form (1) RUN ENTER

This form starts program execution at the first line of a program.

General form (2) RUN ⟨ Exepression ⟩ ENTER

Example: RUN 30 ENTER

This form starts program execution at the line specified, or by the value of ⟨ expression ⟩. That value is effective in its integral part, limited to 1 through 999.

Example: R U N 10 ENTER   Program execution begins at line 10.

General form (3) RUN $\left\{ \begin{array}{l} \text{"Character"} \\ \text{[Character variable]} \end{array} \right\}$ ENTER

Example: RUN "ABC"

This form starts the program execution at the line given a label such as is stored in "character" or [character variable].
When the length of "character" and label exceeds seven characters, only the first seven characters are used and excess is ignored.

Example: R U N SHFT " P R O − 1 SHFT " ENTER

Program execution starts at the line labeled "PRO−1".

- If you use this command without a program, the Computer will merely return a prompt (>) display.
- When the specified line does not exist in general form (2) or when the specified label does not exist in general form (3), an error (Error 2) occurs.

2. DEBUG command

The DEBUG command functions only in the RUN or DEF mode. It can be used in the same form as RUN. However, when you use this command to execute a program, the Computer displays the line number as it executes each line and then goes into a break condition (temporary stop).
Then, when you press the ↧ key, the Computer will advance to the next execution line and display its line number, going into break again. (Such a line-by-line execution is called debugging and gives you an opportunity to debug [correct errors] and/or study the program line-by-line.)

Note: For program debugging, consult page 41.

- The general form of this command can be defined in the same manner as the RUN statement.

3. CONT command

The CONT command functions in the RUN or DEF mode. It clears a temporary stop of program execution and CONTinues program execution.

General form CONT ENTER

Various forms of "temporary stop" of program execution are described below:

(1) Break condition
- ① Temporary stop due to STOP statement during program execution.
- ② Temporary stop caused by pressing ON (BREAK) key during program execution.
- ③ Temporary stop accompanied by the display of a line number during debugging.

(2) After the execution of a PRINT statement (while specified contents are displayed)

Example:

```
10:A=0
20:FOR B=1TO 3
30:A=A+B:PAUSE B, A
40:NEXT B
50:END
```

Operation	Display	Note
Set to the RUN mode		
D E B U G	DEBUG_	
ENTER	10:	
↧	20:	
↧	1. 1.	
	30:	
↧	40:	
C O N T	CONT_	Inputting a CONT command
ENTER	2. 3.	Execution of the CONT command (continuing after temporary stop)
	3. 6.	

4. LIST command

This command lists a program; functions only in the PRO mode.

General form (1) **LIST** ENTER

This form displays the first line of a program.

General form (2) **LIST 〈 Expression 〉** ENTER

Example: LIST 10

This form displays the program line specified by the value of 〈 expression 〉.
(The value is operative only in its positive integer part, limited to 1 through 999.)

General form (3) **LIST** $\left\{ \begin{array}{l} \text{"Character"} \\ \text{[Character variable]} \end{array} \right\}$ ENTER

Example: LIST "ABC"

This form displays the program line with the label identification stored in "character" or [character variable].
When the length of "character" and label exceeds seven, only the first seven characters are used, any excess is ignored.

Example:

```
10:A=0
20:FOR B=1TO 3
30:A=A+B:PAUSE B, A
40:NEXT B
50:END
```

Operation	Display	Note
Set to the PRO mode		
L I S T	LIST_	
ENTER	10 : A=0	Display of the first line
L I S T 30	LIST30_	
ENTER	30 : A=A+B : PAUSE B , A	Display of line 30

- Recall an arbitrary line by LIST and then you can display preceding and following lines using the ↑ or ↓ key.
- When programs use more than 24 characters on one line, they are displayed beginning with the first number/character on the left. To display the remainder of the line, use ▶ key.
- When the specified line does not exist an error (Error code: 2) occurs.

5. NEW command

The NEW command clears all programs, reserve programs and data (but not reserve memories).

General form NEW ENTER

(1) DEF, RUN and PRO modes

The execution of a NEW command in these modes clears all program and data memories. (but not reserve memories)

(2) RESERVE mode

The execution of a NEW command in this mode clears all reserve memories (but not program and data memories).

- The prompt symbol appears after execution of this command.

6. MEM command

This commmand displays the number of program steps and flexible memories, so you can tell how much memory you have used up and how much you have available.

General form MEM ENTER

This command functions in all modes.

Example: The execution of a MEM command with all program memories cleared.

Operation	Display	Note
PRO mode		
N E W ENTER	>	
M E M ENTER	1424STEPS 178MEMORIES	※

※ The displays shows that you have space for 1424 steps or that a maximum of 178 memories can be used as flexible memories.

One flexible memory corresponds to 8 steps of program.

Thus, if you load a program with 8 steps or less, the number of flexible memories will be reduced by one (if a program uses 9 through 16 steps, that takes up two flexible memories, etc.).

- The displayed number of memories does not include the 26 fixed memories.

STATEMENTS FOR OPTIONAL CASSETTE INTERFACE (Cat. No. 26-3503)

You can use your TRS-80 Pocket Computer along with a cassette tape recorder as an external memory storage unit by using the optional Cassette Interface 26-3503. This will enable you to record programs, reserve programs and the contents of data memories stored in the Computer on to a magnetic cassette tape. Also, you can read them out of tape, as well as compare the contents recorded on the tape with the contents loaded in the Computer.
Giving file names to all recordings (programs, reserve programs and data) the Computer provides an automatic search when reading them out.
For instructions on operating the tape recorder, refer to page 93.

1. CSAVE (Cassette save) statement

The CSAVE statement is an instruction to record programs or reserve programs on a magnetic tape. **This statement can only be executed manually.**

General form: CSAVE "File name" ENTER

If the length of file name is more than 7 characters, the excess is ignored.
This is true of all magnetic tape control instructions.

In DEF, RUN and PRO modes

The CSAVE instruction commands the Computer to first record the specified file names on a magnetic tape and then the entire stored program identified by that file name.
However, if the memory has no programs, the prompt symbol appears on the display.

Example: PRO mode CSAVE "PROG.–1" ENTER	"PROG.–1" is recorded on tape as the file name, plus all the identified program lines.

In RESERVE mode

The CSAVE instruction forces the tape recorder to record specified file names and then all the reserved programs.
However, if the memory has no reserve programs, the display shows the prompt symbol.

Example: RESERVE model CSAVE "RESRV–1" ENTER	"RESRV–1" is recorded as the file name along with all reserve programs.

[By executing a CLOAD? statement (mentioned later on) after executing a CSAVE statement you can check that programs have been accurately recorded.]

Note: You must avoid recording programs given the same file name – but different in contents – on the same side of the same tape, otherwise the reading (transferring) of wrong contents may occur in the execution of a CLOAD or CHAIN statement. Do not record programs with the same file-names on the same side of the tape. (Or you may end up with the incorrect data or program.)

Also do not allow program recordings to overlap (even slightly); either or both programs may be full of errors.

2. CLOAD (Cassette Load) statement

This statement is an instruction to transfer (load in) programs or reserve programs from a magnetic tape into the Computer. **This instruction can only be executed manually.**
When transferring programs, rewind the tape to the portion of tape where they are recorded, then execute this instruction.

General form **CLOAD "File name"** ENTER

In DEF, RUN and PRO modes

This instruction makes an automatic reference of specified file names and transfers the corresponding programs from the magnetic tape to the Computer.

Example: PRO mode CLOAD "PROG.–1" ENTER	A program on the magnetic tape whose file name is "PROG.–1" is found and transferred to the program memory.

In RESERVE mode

This instruction makes an automatic reference of specified file names and transfers the corresponding reserve programs from the magnetic tape to the Computer.

Example: RESERVE mode CLOAD "RESRV–1" ENTER	A reserve program on the tape whose file name is "RESRV–1" is found and transferred to the reserve memory.

Note 1. The Computer cannot decide whether a certain file name refers to a program or reserve program. Therefore, an improper mode selection leads to an improper transfer: reserve programs to the program memory or programs to the reserve memory.

2. If file names you specify are not on the magnetic tape, **the Computer continues to search for the absent file names even after the tape has come to an end.**
(In this case, cancel the instruction by pressing the $\boxed{\text{ON}}$ key.)
This is also true of CLOAD?, CHAIN, and INPUT # statements described later.

3. If an error is encountered during the transfer of programs, only the program memory is cleared. This is also true of a CHAIN statement (discussed later).

3. CLOAD? (Cassette Load?) statement

The CLOAD? statement is an instruction to check the contents of the program or reserve memory inside the Computer with the recording on the magnetic tape that shares the specified file name. **This instruction can only be executed manually.**
If the above check is not good, an error (error code: 5) occurs.

(When making a CLOAD? check, rewind the tape to the portion of the tape that is to be checked, then execute this instruction.)

General form **CLOAD? "File name"** ENTER

In DEF, RUN and PRO modes

This instruction checks the contents of program memory against the recorded programs on the magnetic tape (specified file name only).

In RESERVE mode

This instruction checks the contents of reserve memory against the recorded programs on the magnetic tape (specified file names only).

- When the program memory is loaded with nothing in the DEF, RUN or PRO mode or when the reserve memory is loaded with nothing in the RESERVE mode, the execution of a CLOAD? statement will display only the prompt symbol.

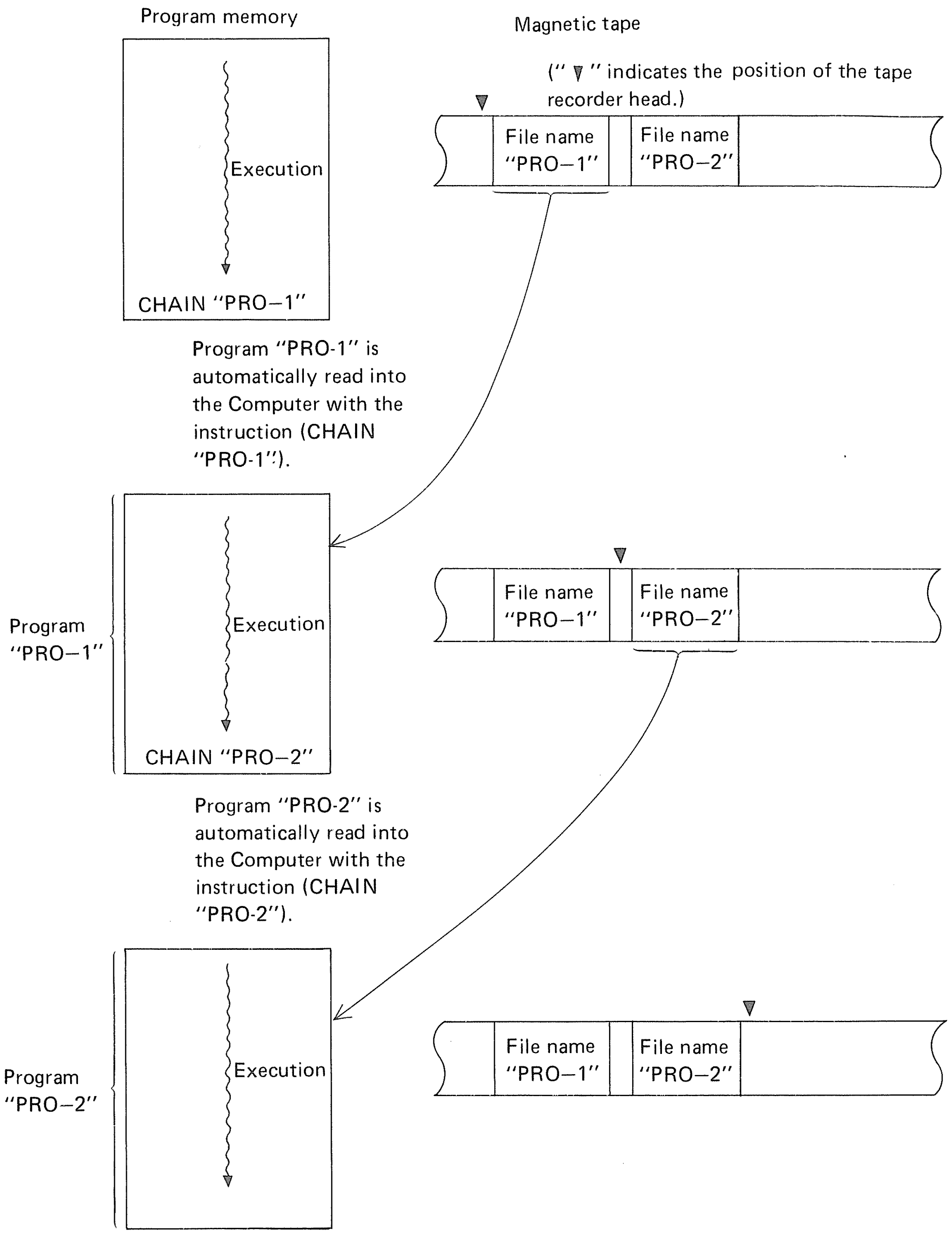
4. CHAIN statement

The CHAIN statement is a program execution instruction. When the Computer comes to this instruction during program execution, it automatically searches programs recorded on a magnetic tape for the specified file names and transfers those programs to its program memory.
With the CHAIN statement it also starts execution of the tranferred programs at the line specified.
In other words, if you use this instruction, even programs long enough to exceed the capacity of the Computer's program memory can be read out to the Computer in sequence to be executed.
These programs of course must be divided in some manner so they can be stored individually in the Computer's memory (while execution takes place).
(Rewind the tape and then execute specified programs.)

General form (1) **CHAIN "File name"**

Example: CHAIN "PRO–1"

The instruction tells the Computer to execute a transferred program from the beginning.


Program memory
Magnetic tape
(" ▼ " indicates the position of the tape recorder head.)
File name "PRO–1"
File name "PRO–2"
Execution
CHAIN "PRO–1"
Program "PRO-1" is automatically read into the Computer with the instruction (CHAIN "PRO-1").
Program "PRO–1"
Execution
CHAIN "PRO–2"
File name "PRO–1"
File name "PRO–2"
Program "PRO-2" is automatically read into the Computer with the instruction (CHAIN "PRO-2").
Program "PRO–2"
Execution
File name "PRO–1"
File name "PRO–2"

If each program is arranged to end with a CHAIN statement as shown above, a new program can be automatically transferred from the tape and executed in succession every time the preceding program is completely executed.

General form (2) **CHAIN "File name", ⟨ Expression ⟩**

Example: CHAIN "PRO–1", 30

This form starts the execution at the line specified by the value of ⟨ expression ⟩ contained in a transferred program.
The value of ⟨ expression ⟩ is effective only in its integer part, limited to positive numbers from 1 through 999.

General form (3) **CHAIN "File name", { "Character" / [Character variable] }**

Example: CHAIN "PRO–1", "A"

This form starts the execution at the line given the same label as the contents of "character" or [character variable] contained in a transferred program.
The length of "character" or label is effective up to seven characters (excess is ignored).

Example:	⋮ 100 : CHAIN "ABC"	A program given file name "ABC" is transferred from the magnetic tape to the program memory, and is executed from its beginning.
Example:	⋮ 200 : CHAIN "XYZ", 10	A program given file name "XYZ" is transferred from the magnetic tape to the program memory and executed from line 10.

5. PRINT # (Print cross-hatch) statement

The statement PRINT # is an instruction to record the contents of data memories on a magnetic tape. **This statement can be executed both by program and manual operation.** (Executable in the DEF and RUN modes)

General form (1) **PRINT # "File name"**

Example: PRINT # "DATA 1"

This form tells the Computer to record "file name" and then record all the contents of data memories in sequence starting with fixed memory A (or A$).

General form (2) **PRINT # "File name"; [Label of variable]**

Example: PRINT # "DATA 1"; A(5)

This form commands to first record file names on a magnetic tape and then record the contents of specified data memory.
The [label of variable] is specified by characters A through Z or in the form of A ( ). In the latter case, however, material in parentheses is limited to positive integers from 1 to 204, or to fixed memories. (If program memories loaded with program are specified as flexible memories, an error occurs.)
This method of specification also applies to the INPUT # statement described below.
A PRINT # statement must always end with $\boxed{\text{ENTER}}$, when executed manually.

Example:	Manual execution PRINT # "DATA–1" $\boxed{\text{ENTER}}$	"DATA–1" is recorded as file name, and the contents of data memory no. 1 (memory A or A$) and all subsequent that can be specified as flexible memories are all recorded in sequence.
Example:	Program execution ⋮ 150 : PRINT # "DATA–1"; A(26) ⋮	"DATA–1" is recorded as file name, and the contents of data memory no. 26 (memory Z or Z$) and all following are recorded in sequence.

6. INPUT # (Input cross-hatch) statement

The INPUT # statement is an instruction to transfer data recorded on a tape to the data memory of the Computer. **This statement can be executed both by program and manual operation.** (Executable in the DEF or RUN mode)
(Rewind the tape before executing this instruction.)

General form (1)   INPUT # "File name"

Example: INPUT # "DATA 1"

This form commands the Computer to automatically search for specified file names and loads data memory no. 1 (A or A$) and all following, in sequence with the corresponding recorded data.

General form (2)   INPUT # "File name"; [Label of variable]

Example: INPUT # "DATA 1"; A(5)

The form compels the Computer to search for specified file names automatically and loads the data memory specified by [label of variable] and the following, in sequence, with recorded data corresponding to the specified file names.

Example: Execution through manual operation

```
    INPUT # "DATA–1" ENTER
```

Recorded data whose file name is "DATA–1" is put in data memory no. 1 and all after in sequence.

Example: Execution through program

```
          :
   50 : INPUT # "DATA–1"; A$(26)
          :
```

Recorded data whose file name is "DATA–1" is put in date memory no. 26 and all after in sequence.

● The Computer can distinguish and transfer file names recorded as programs and as data even if they are identical.

Note: If the number of recorded data is smaller than that of data memories to be loaded, the execution activated by the INPUT # statement ends as soon as all the data are transferred.
Also, all data will be transferred until all the data memories are loaded, and then execution will end even if there is more data.

RESERVABLE KEY

Up till now we've only discussed how to program one key at a time. But the Pocket Computer has a very powerful feature which allows you to use just one key for an entire program or function which you use frequently. The reserved key function can be used for manual calculations as well as in programs.
For example, you assign [P] [R] [I] [N] [T] to the [A] key for reserve, all you have to do is press [SHFT] [A] to have the Computer print the required data.
The following keys are available for reserve:

[A]、 [S]、 [D]、 [F]、 [G]、 [H]、 [J]、 [K]、 [L]、 [=]、 [Z]、 [X]、
[C]、 [V]、 [B]、 [N]、 [M]、 [SPC]

1. Reserve memory for reservable keys

The total numbers of steps allocated for reservable keys are 48. If the contents of one or more reservable keys exceed 48 steps, error 4 (insufficient memory) occurs.

To write reserve programs, select the RESERVE mode and follow the same procedures as in manual calculations.

(1) Preparation

Before writing a new reserve program clear the reserve memory using a NEW command.
However, if you are writing reserve programs to follow in sequence one or more already written, there is no need to use NEW.

[Precedures]

(1) Select the RESERVE mode.

(2) [N] [E] [W] [ENTER]

NEW clears the entire reserve memory.

(2) Writing

Example: Reserve the following key operation:
"COS" to [A]
"A ∗ A + B ∗ B" to [S]
"RUN 130" to [Z]

Writing

Operation	Display	Note
Set to the RESERVE mode		
N E W ENTER	> Colon is displayed automatically.	Reserve memory is cleared.
SHFT A	A : _	Key is specified.
C O S	A : COS_	Input
ENTER	A : COS	Writing (the cursor disappears)
SHFT S	S : _	Key is specified.
A * A + B * B	S : A*A+B*B_	Input
ENTER	S : A*A+B*B	Writing
SHFT Z	Z : _	Key is specified
R U N 1 3 0	Z : RUN130_	Input
ENTER	Z : RUN 130	Writing

● When the reserve memory (48 steps) is full, pressing the ENTER key causes an error. (Error code: 4)

2. Use of reservable keys

The reservable keys are used in the PRO or RUN mode.

Example 1: Manual calculation by using the reservable keys to which reserve programs have been assigned programs as written above.

Operation	Display	Note
Set to the RUN mode.		
Set the angular mode to DEG		Degree is specified.
SHFT A	COS _	
60	COS 60_	
ENTER	0. 5	
A = 32 SHFT ,	A=32,_	
B = 53 ENTER	53.	
SHFT S	A*A+B*B_	
ENTER	3833.	$32^2+53^2=$

Example 2: Writing and execution using the reservable keys as assigned above.

Operation	Display	Note
Writing		Law of cosine
Set to the PRO mode		$C=\sqrt{a^2+b^2-2ab\cos\theta}$
130 $\boxed{I}\boxed{N}\boxed{P}\boxed{U}\boxed{T}$	130 INPUT_	
$\boxed{A}\boxed{SHFT}\boxed{,}\boxed{B}$	130 INPUTA,B_	
$\boxed{SHFT}\boxed{,}\boxed{D}\boxed{ENTER}$	130: INPUT A,B,D	
140 $\boxed{C}\boxed{=}\boxed{\sqrt{\ }}\boxed{(}$	140C=√(_	
$\boxed{SHFT}\boxed{S}$	140C=√(A*A+B*B_	Recall of reserve
$\boxed{-}\boxed{2}\boxed{*}\boxed{A}\boxed{*}$	140C=√(A*A+B*B−2*A*_	
$\boxed{B}\boxed{*}\boxed{SHFT}\boxed{A}$	0C=√(A*A+B*B−2*A*B*COS _	Recall of reserve
$\boxed{D}\boxed{)}\boxed{ENTER}$	140: C=√(A*A+B*B−2*A*B*	
150 $\boxed{P}\boxed{R}\boxed{I}\boxed{N}\boxed{T}$	150PRINT_	
$\boxed{C}\boxed{ENTER}$	150: PRINT C	
160 $\boxed{E}\boxed{N}\boxed{D}\boxed{ENTER}$	160: END	
Execution		
Set to the RUN mode		
Set the angular mode to DEG		
$\boxed{SHFT}\boxed{Z}$	RUN130_	Program execution starts.
$\boxed{ENTER}$	?	
(When A = 10) 10 $\boxed{ENTER}$	?	
(When B = 12) 12 $\boxed{ENTER}$	?	
(When D = 60) 60 $\boxed{ENTER}$	11. 13552873	

Example 3: When using reservable keys in the course of manual calculation or programming.

Operation	Display	Note
Set to the RUN mode		
5 $\boxed{*}$ 6 $\boxed{*}$ 60	5*6*60_	
$\boxed{\blacktriangleleft}\boxed{\blacktriangleleft}\boxed{SHFT}\boxed{INS}$	5*6*□60	Adding one space
$\boxed{SHFT}\boxed{A}$	5*6*COS 60	COS is inserted as 1-step instruction.
$\boxed{\blacktriangleleft}\boxed{SHFT}\boxed{INS}\boxed{SHFT}\boxed{INS}$	5*6*□□COS 60	Two-step space
$\boxed{SHFT}\boxed{S}$	5*6*A*A+B*BCOS 60	When the number of reservable key steps to be inserted is larger than the codes, the necessary number of steps are secured to insert the remaining code.
$\boxed{*}$	5*6*A*A+B*B*60	
$\boxed{SHFT}\boxed{INS}$		
$\boxed{SHFT}\boxed{INS}\boxed{SHFT}\boxed{INS}$	5*6*A*A+B*B*□□□60	Three-step space
$\boxed{SHFT}\boxed{A}$	5*6*A*A+B*B*COS □□60	When the number of reservable key steps to be inserted is smaller than the insertion codes, the empty insertion codes are filled with nothing.

3. Checking reserve programs

To check what information is assigned to reservable keys, specify those keys in the RESERVE mode.

Example: Checking reserve programs written in [A], [S], [D] and [Z] as above.

Operation	Display	Note
Set to the RESERVE mode		
[SHFT] [A]	A : COS	Display of reserve.
[SHFT] [S]	S : A∗A+B∗B	Display of reserve.
[SHFT] [D]	D : _	When nothing is reserved
[SHFT] [Z]	Z : RUN 130	Display of reserve

4. Correction of reserve programs

If you need to correct written reserve programs or use different key operations:

Example: "A : COS", "S : A ∗A + B ∗ B" and "Z : RUN 130" as reserved above are to be changed to "A : SIN", S : LOG A" and "Z : RUN 50."

Operation	Display	Note
Set to the RESERVE mode		
[SHFT] [A]	A : COS	Recall of reserve
[◄]	A : COS	Recall of cursor
[S] [I] [N]	A : SIN_	Inputting through keys
[ENTER]	A : SIN	Writing
[SHFT] [S]	S : A∗A+B∗B	Recall of reserve
[►]	S : A∗A+B∗B	Recall of cursor
[L] [O] [G] [A]	S : LOGAB∗B	Inputting through keys
[SPC] [SPC] [SPC]	S : LOGA _	Unnecessary instructions are cleared by spacing.
[ENTER]	S : LOG A	None of spaces are written in the reserve memory.
[SHFT] [Z]	Z : RUN 130	Recall of reserve
[►] [►]	Z : RUN 130	The cursor moves
[5] [0] [SPC]	Z : RUN 50 _	Inputting
[ENTER]	Z : RUN 50	Writing

5. Deleting reserve programs

(As you know, $\boxed{N}$ $\boxed{E}$ $\boxed{W}$ $\boxed{\text{ENTER}}$ key clears all reserve memories, but here is how you delete just one.)

Example: To delete "RUN 50" reserved for $\boxed{Z}$.

Operation	Display	Note
Set to the RESERVE mode		
$\boxed{\text{SHFT}}$ $\boxed{Z}$	Z : RUN 50	Recall of reserve
$\boxed{\triangleright}$	Z : RUN 50	Recall of cursor
$\boxed{\text{SPC}}$	Z : 50	All instructions are replaced with spaces.
$\boxed{\text{SPC}}$ $\boxed{\text{SPC}}$	Z : _	
$\boxed{\text{ENTER}}$	〉	Reserve is deleted and prompt symbol appears.

6. Configuration of reserve programs

Example: Operation $\boxed{S}$ $\boxed{I}$ $\boxed{N}$ $\boxed{A}$ $\boxed{+}$ $\boxed{C}$ $\boxed{O}$ $\boxed{S}$ $\boxed{B}$ is reserved for $\boxed{A}$ key.

```
   1     2     3 4    5     6    ← Step no. of reserve memory
   :     :     : :    :     :
   A : S I N   A+COS  B
   ↑
   └─ Reservable key label
```

Operation $\boxed{I}$ $\boxed{N}$ $\boxed{P}$ $\boxed{U}$ $\boxed{T}$ is reserved for $\boxed{B}$ key.

```
 1      2     ← Step no.
 :      :
 B : INPUT
 ↑
 └─ Reservable key label
```

The labels of keys are also placed in reserve memory (such as $\boxed{A}$ and $\boxed{B}$), each of which takes 1 step of space. (Colons after key labels are not placed in the reserve memory.)

ERROR CODES

If you attempt to execute statements or programs which the Computer cannot process (such as words not defined by the Computer or incorrect operations) an error will occur. The Computer displays an error code and stops execution.

Example 1

Operation	Display
RUN mode	
5 $\boxed{+}$ $\boxed{*}$ 3	5+*3_
$\boxed{\text{ENTER}}$	1. . . . .
$\boxed{\blacktriangleright}$	5+*3

In this example, * results in a syntax error (improper computer "grammar") which produces the error symbol.
If you press the $\boxed{\blacktriangleright}$ or $\boxed{\blacktriangleleft}$ key, the input contents will be displayed with the cursor positioned at * (where the error occurred).

Example 2

```
      :
30 : A=5+*3
      :
```

Operation	Display
RUN mode	Line number Error code 30 : 1. . . . .
$\boxed{\uparrow}$	30 : A=5+*3 > Error position

In this case, the Computer displays the line number where the error occured, along with an error code.
If you press the $\boxed{\uparrow}$ key, the position where the error occurred is marked by the cursor; the display will remain only while you hold down $\boxed{\uparrow}$ key. (When you release $\boxed{\uparrow}$ the prompt symbol (>) will appear.)
To correct this error, press the $\boxed{\text{MODE}}$ key to select the PRO mode and press the $\boxed{\uparrow}$ or $\boxed{\downarrow}$ key to display the program for correction.
To clear an error in manual operation, use the $\boxed{\text{ON}}$ or $\boxed{\text{CL}}$ key. For other errors (except for exceeding program memory capacity and error code 5) you can also use the $\boxed{\blacktriangleright}$ or $\boxed{\blacktriangleleft}$ keys. If you encounter an error during program execution, clear using the $\boxed{\text{ON}}$, $\boxed{\text{CL}}$ or $\boxed{\uparrow}$ keys. (An error encountered during the execution of a CHAIN statement cannot be cleared using the $\boxed{\uparrow}$ key.)

Table of Error Codes

Error code	Nature	Description
1	● Grammatical (syntax) error ● Operational error ● Error in memory specification	● Occurs when the absolute value of a calculation result exceeds 1 x 10^{100}, or when the divisor is 0. ● Occurs when memories to which numerical values are assigned are specified as character variables or vice versa.
2	Line error	● Occurs when lines and labels specified by GOTO, GOSUB, RUN, DEBUG or LIST statements do not exist.
3	Level Error	● Occurs when the level exceeds 4 stages in a GOSUB statement or FOR-NEXT statement. ● Occurs when you try to execute a RETURN statement without a preceding GOSUB statement. ● Occurs when you try to execute a NEXT statement without a mating FOR statement.
4	Insufficient memory	● Occurs when you try to write programs with more steps than the remaining available program memories. ● Occurs when you try to write more reserve programs steps than can fit the available remaining reserve memories. ● Occurs when existing dimension memories are specified and there is insufficient memory left.
5	Control error of magnetic tape	Occurs when an error occurs during the execution of magnetic tape control instruction. (Verify error, check-sum error, etc.)
6	Error in format	Occurs when a display of numerical data is not in specified format when used with PRINT or PAUSE.

APPENDIX

The remainder of this Manual includes reference information for your TRS-80 Pocket Computer.

SPECIFICATIONS

Model:	TRS-80 Pocket Computer	
Number of calculation digits:	10 digits (mantissa) + 2 digits (exponent)	
Calculation system:	According to mathematical formula (with priority judging function)	
Program system:	Stored system	
Program language:	BASIC	
Capacity:	Program memory;	1424 steps maximum
	Data memory;	26 Fixed memories 178 Flexible memories, maximum (shared with program memory)
	Reserve memory;	48 steps maximum (up to 18 different reserve programs)
	Input buffer;	80 characters
Stack:	For data;	8 steps
	For function;	16 steps (in parentheses, 15 levels)
	For subroutine;	4 levels
	For FOR-NEXT statement;	4 loop-levels
Calculations:	Four arithmetic calculations, power calculation, trigonometric and inverse trigonometric functions, logarithmic and exponential functions, angular conversion, extraction of square root, sign function, absolutes, integers, and logical calculations.	
Editing function:	Cursor shifting (► ◄) Insertion (INS) Deletion (DEL) Line up and down (↑ , ↓)	
External memory function:	By using the optionally available TRS-80 Cassette Interface 〈 26-3503 〉, program, reserve program, and data memory can be recorded or read out to or from magnetic tape (tape recorder).	
Memory protection:	CMOS battery back-up (program, data and reserve memories are protected)	
Display:	24-digit alphanumeric dot matrix liquid crystal display	
Component:	CMOS LSI, etc.	
Power supply:	5.4V, DC: 4 mercury batteries (Type 675, Radio Shack Cat. No. 23-1521)	
Power consumption:	5.4V, DC: 0.011W 5.4V, DC: 0.013W (with Cassette Inteface)	
Operating time:	Approx. 300 hours on mercury batteries (Type 675) ambient temperature: 20°C (68°F). The operating time changes slightly depending on the type of battery and usage.	
Operating temperature:	0°C ~ 40°C (32°F ~ 104°F)	
Dimensions:	175(W) x 70(D) x 15(H) mm 6-7/8"(W) x 2-3/4"(D) x 19/32"(H)	
Weight:	Approx. 170g (0.37 lbs.)	
Accessories:	Carrying case, four mercury batteries (built-in), two keyboard templates.	

BATTERY REPLACEMENT

When the battery indicator is not lit (upper right hand corner of display), replace the mercury batteries (type 675).

1. Turn off the Computer.
2. Remove the screws from the back cover with a small screw driver (Fig. 1). (Note that two types of screws are used.)
3. Replace the batteries as shown in Figure 2.

- Use a dry cloth to wipe off the surface of the new batteries before installing.
- Always replace all 4 batteries at the same time.

Batteries are available from your local Radio Shack store, Cat. No. 23-1521

4. Hook the tabs of the back cover into the slots on the Computer.
5. Push the back cover in slightly while replacing the screws. (Fig. 3)
6. Push the Reset switch on the back cover to clear the Computer. (Fig. 4) Use a ball-point pen to press the Reset switch.
7. Press the $\boxed{\text{OFF}}$ and $\boxed{\text{ON}}$ keys to clear the Computer. When the batteries are correctly intalled " >DEG RUN . " will be displayed.

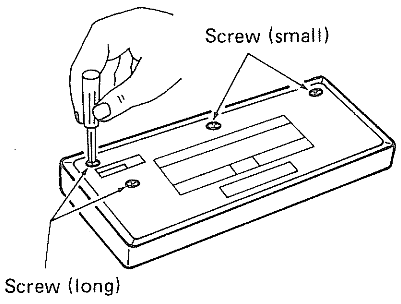


Fig. 1

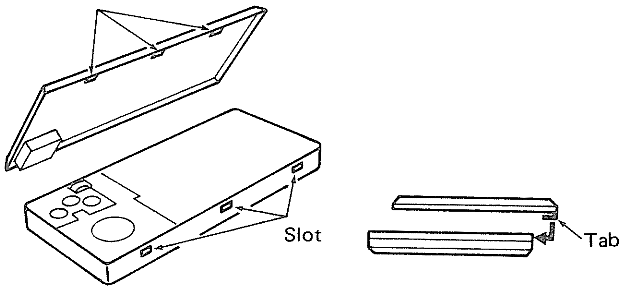


Fig. 3

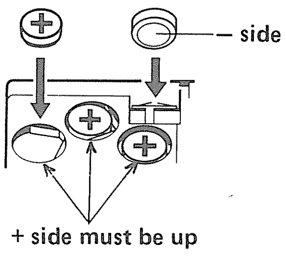


Fig. 2

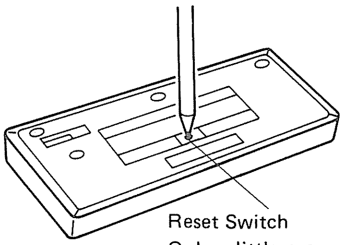


Fig. 4 Only a little pressure is needed. Do not use a pencil or other material that could break and leave a residue.

NOTE: Do not dispose of batteries in a fire.

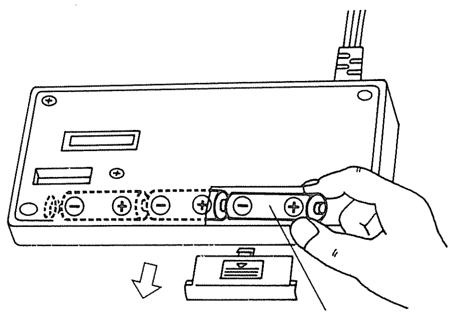
IMPORTANT: The batteries must be of specified type (675). When other type, such as 675E batteries are used, the display operation will become improper.

CONNECTING THE CASSETTE INTERFACE

You can obtain the optional Cassettle Interface for the Pocket Computer by ordering Cat. No. 26-3503. Using this Casstte Interface will allow you to store programs and data from the Pocket Computer onto standard cassette tapes (of course you'll also need a Cassette recorder such as we sell for this Pocket Computer system —— check with your local Radio Shack store). Once on tape, you can load these programs and data back into the Computer with a simple procedure.

Replacing the Batteries

If Remote countrol of the Cassette recorder is not functioning normally when using the Cassette Interface, it's time for fresh batteries.

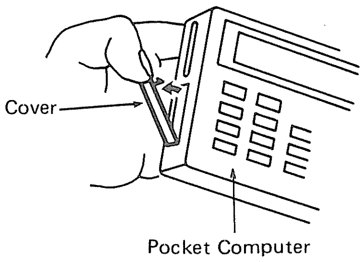


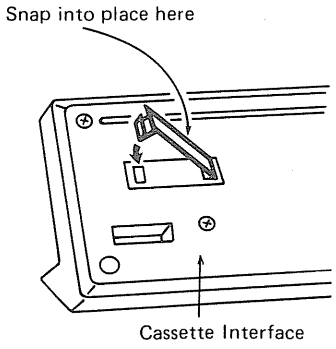
Three "AA" penlight cells.

Notes:
- Once a year you should replace the batteries in the Cassette Interface. Use only the Alkaline type (they maintain their voltage for a longer period of time).
- Always replace all three batteries at the same time.
- Never leave weak or dead batteries in any battery-operated device (they can leak damaging chemicals). If you are not going to use the Interface for a month or more, remove the batteries.

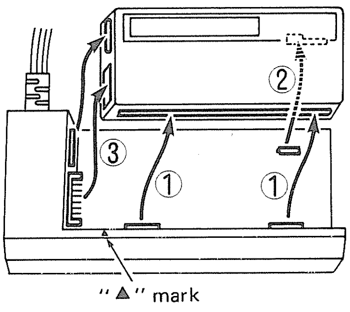
Connecting the Pocket Computer to the Cassette Interface

1. **Turn off your Pocket Computer by pressing $\boxed{\text{OFF}}$ key.**
2. Remove the cover from the left side of your Computer, and snap it into place on the bottom of the Cassette Interface.

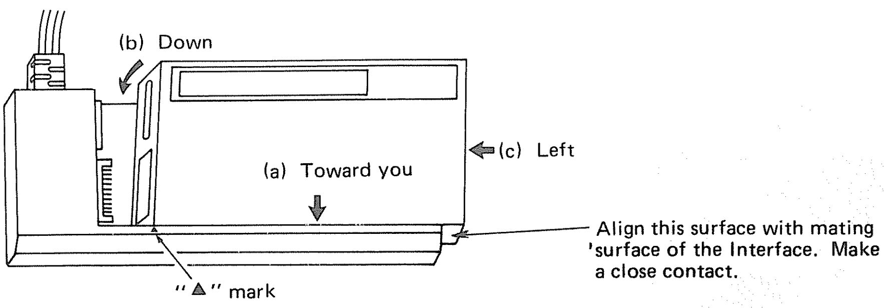




3. Fit projecting parts on the Cassette Interface in the grooves of the Computer as shown below.

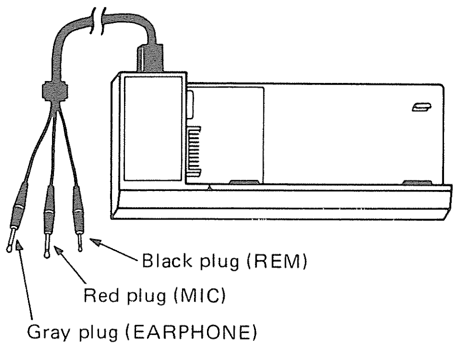


4. Slide the Pocket Computer carefully to fit securely onto the Cassette Interface (match triangular marks (▲) on the Computer and Cassette Interface).


(b) Down
(c) Left
(a) Toward you
Align this surface with mating 'surface of the Interface. Make a close contact.
"▲" mark

5. If parts do not mate properly, do not force. Carefully shift Computer left or right to be sure all mating surfaces are correct.

Note: Before attaching or removing the Computer from the Interface, be sure to turn off the Computer with the $\boxed{\text{OFF}}$ key. If the Computer is connected or disconnected with power ON, all keys may be inoperative. In this case, press the ALL RESET switch on the bottom of the Computer. This will clear the entire Computer.

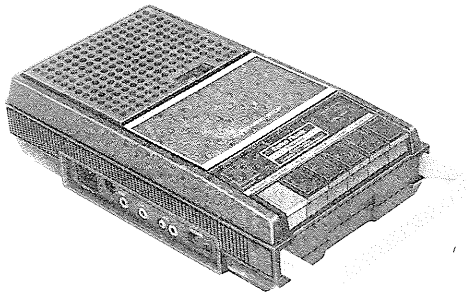


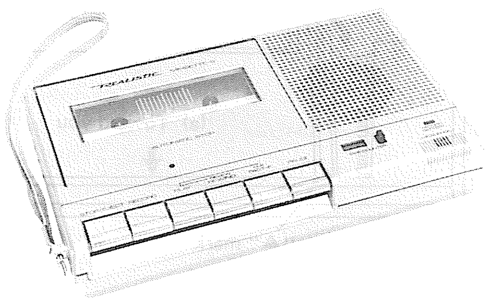
Connecting the Cassette Interface to a Tape Recorder

Only three connections are necessary:

1. Connect red plug into the MIC jack on the Cassette Recorder.
2. Connect gray plug into the EArphone jack on the Recorder.
3. Connect the black plug into the REMote jack on the Recorder.

While most Cassette Recorders can be used for Recording and Playing back programs and data for the TRS-80 Pocket Computer, we urge you to use one of the Recorders we recommend (such as our CTR-80A or Minisette-9).


CTR-80A


Minisette-9

OPERATING THE CASSETTE INTERFACE & RECORDER

Recording onto magnetic tape

See Tape Notes on page 101.

1. Enter a program or data into the Computer.
2. Load tape into the tape recorder.
   Determine the position on the tape where you want to record the program.
   - When using a tape, be sure the tape moves past the clear leader (non-magnetic mylar material).
   - When using a tape already partially recorded, search for a location where no recording is.
3. Connect the Interface's red plug to the tape recorder's MIC jack and the black plug to the REM jack.
4. Simultaneously press record and play buttons on the tape recorder (to put it in record mode). (The tape recorder will be stopped.)
5. Input recording instructions (CSAVE statement, PRINT # statement), and press the ENTER key for execution. (For CSAVE and PRINT # see pages 74 and 78.)

   To record the program:
   First press the MODE key to set the unit to "RUN" mode (or "DEF" mode). Next push the following keys: C S A V E SHFT " FILE NAME SHFT " ENTER.
   (To write the contents of data memory onto tape, push as follows;
   P R I N T SHFT # ENTER .)
   Eg. "RUN" mode C S A V E SHFT " A A SHFT " ENTER

   When you press the ENTER key, tape motion will begin, leaving about a 6-second none-signal blank. (Beep tone is recorded.) After that, the file name and its contents are recorded.
6. When the recording is complete, the PROMPT symbol (>) will be displayed and the tape recorder will automatically stop. Now you have your program on tape (it still is in the Pocket Computer also).
   When data is to be automatically recorded by program execution (PRINT # statement, not manual operation), set up steps 1 thru 4 before executing the program.

   To aid you in locating programs on tapes, use the tape counter on the recorder.

Loading from a magnetic tape

See Tape Notes on page 101.
To load, transfer, or read out programs and data from magnetic tape into the Pocket Computer, use the following procedure.

1. Load tape in the tape recorder. Position tape just before the portion to be read out.
2. Connect the gray plug to the EAR jack on the tape recorder, and the black plug to the REM jack jack.
   [In using a tape recorder having no REM terminal, press the PAUSE button to make a temporary stop.]
3. Push the PLAY button on the tape recorder (to put unit in playback mode). (The tape should be stopped.)

Set the VOLUME control to 6 to 8.

Set Tone to maximum treble (10).

4. Input transfer instructions (CLOAD statement, INPUT # statement), and press ENTER key for execution. (See pages 75 and 79.)

   To transfer the program:

   Put the unit into "RUN" mode (or "DEF" mode) with the MODE key. Then push the following keys; C L O A D SHFT " FILE NAME SHFT " ENTER . (To load the contents of the data memory, push as follows; I N P U T SHFT # ENTER .)

   Eg. "RUN" mode C L O A D SHFT " A A SHFT " ENTER

   The specified file name will be automatically searched for and its contents will be transferred into the Pocket Computer.

5. When the program has been transfered the Computer will automatically stop the tape motion and display the PROMPT (>) symbol.

   To transfer data (CHAIN statement, INPUT # statement) in the course of execution of a program, set up steps 1 thru 3 prior to executing the program.

   Notes:
   - If an error occurs (error code "5" is displayed), start over from the beginning. If the error continues, adjust volume up or down slightly.
   - If the error code is not displayed but tape motion continues (while the Pocket Computer displays the symbol "RUN"), transferring is improper. Press ON key (to "break") to stop the tape. Repeat steps.
   - If the error remains or the tape continues to run after several attempts to correct the problem, try clearning and demagnetizing the Recorder's tape head (see Recorder's Owners Manual).

Program Verification

See tape Notes on page 101.
After loading or transferring a program to or from tape, you can verify that the program on tape and program in the Pocket Computer are identical (and thus be sure that everything is OK before continuing your programming or execution of programs).

1. With cassette in the recorder, operate the tape motion controls to position tape at the point just before the appropritate file name to be checked.
2. Connect gray plug to EARphone and black plug to REMote jacks.
3. Press PLAY button of recorder.
4. Input a CLOAD? statement and start execution with ENTER key. Do this as follows: Press MODE key to set unit to "RUN" (or "DEF" mode).
   Enter the following key sequence —

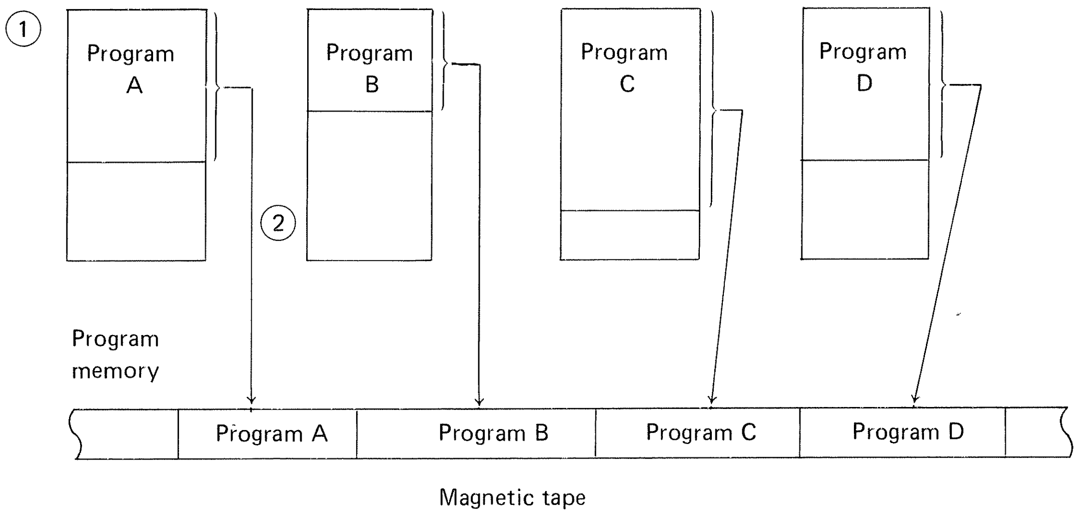
   C L O A D SHFT ? SHFT " A A SHFT " ENTER

   The Pocket Computer will automatically search for the specified file name and will compare the contents on tape with the contents in memory.
5. If the programs are verified as being identical, a prompt symbol will be displayed on the Pocket Computer.

   If the programs differ, execution will be interrupted and an Error code 5 will be displayed. If this occurs, try again.

Editing Programs on Magnetic Tape

You can use the Pocket Computer to Edit and merge Programs on tape by using the CLOAD1 statement.

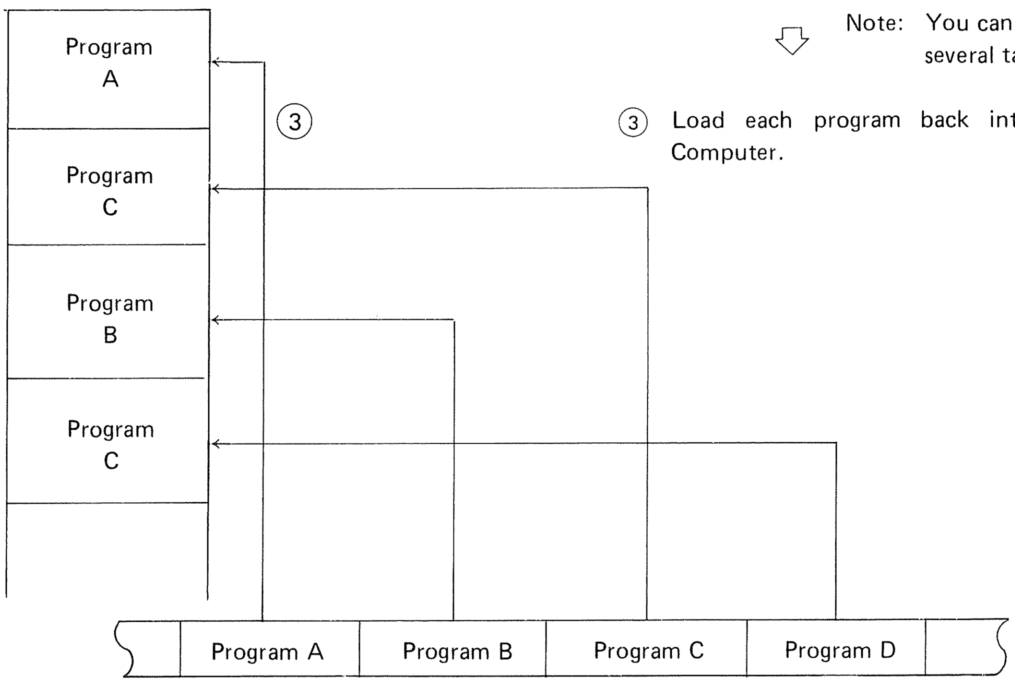


① Write each program into the Computer.

② Save each program onto magnetic tape.

Note: You can use several tapes.

③ Load each program back into the Computer.



CLOAD 1 statement

This statement is an instruction to transfer (load) programs from magnetic tape into the Computer. The program from magnetic tape is loaded into the program memory **in addition to** programs already stored in the Computer.

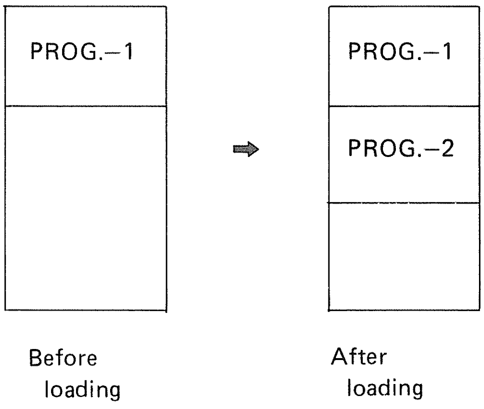
General form CLOAD 1 "File name" ENTER

1. **DEF, RUN and PRO mode**

   The CLOAD 1 instruction automatically references specified file names and transfers the corresponding programs from magnetic tape into the Computer's program memory, along with programs already stored in memory.

   Example: PRO mode
   CLOAD 1 "PROG–2" ENTER

   Contents of the program memory



2. **RESERVE mode**

   Same as for CLOAD statement.

● When the statement "CLOAD 1" is entered with the ENTER key, the statement is converted to a 2-step code. When the statement is recalled, one space is automatically displayed between "CLOAD" and "1".
The abbreviation for "CLOAD 1" is "CLO. 1" or "CLOA. 1".

Editing and execution of multi-program forms

The line numbers of each program must be arranged in numerical order.
The following example will help to explain this function.

File name
"PROG–1" →

```
  5 ▲A▲____________________
 20 GOTO 500
 30 ▲B▲____________________
          :
100 GOTO 30
          :
200 GOSUB ▲C▲:__________
          :
300 A = 100
```

File name
"PROG–2" →

```
 10 ▲D▲____________________
 20 GOTO 5
 30 ▲C▲__________: RETURN
          :
100 GOTO 30
          :
200 GOTO ▲B▲
          :
500 END
```

Program correction

You can insert or delete a program line in the final program only.
In the above example you can only edit or correct parts of "PROG.–2" program.

```
          :
          :
100 GOTO 30
          :
150 PRINT A, B
          :
200 GOTO "B"
          :
500 END
```

Checking the program

Set the Computer in PRO (program) mode.

1. LIST ⟨ Expression ⟩ [ENTER]

- Displays the starting program line corresponding to the contents of the ⟨ expression ⟩ by searching all the program lines from the starting point of the program presently stored in memory.
- An error (error code: 2) will occur when the program has no program line corresponding to the contents of ⟨ Expression ⟩, or when the contents of ⟨ Expression ⟩ is smaller than the line number at which program search begins.

Key opration	Display
LIST 30 [ENTER]	LIST 30_
	30 : "B"____________
LIST 500 [ENTER]	LIST 500_
	500 : END
LIST 10 [ENTER]	LIST 10
	2. . . . .

2. Display of the next or preceding program line by pressing the [↓] or [↑] key during program line display.

Key operation	Display	Note
	30 : "C"____________: RETURN	
[↑]	20 : GOTO 5	
[↑]	10 : "D"____________________	
[↑]	300 : A = 100	Preceding program
[↓]	10 : "D"____________________	Starting point of the next program
[↑]	300 : A = 100	

Execution of the program

(Program execution by using the RUN and DEBUG command.)
Set the Computer to the DEF or RUN mode.

RUN (DEBUG) 〈 Expression 〉 ENTER

Executes the starting program line corresponding to the contents of the 〈 expression 〉 by seaching all the program lines from the starting point of the program presently stored in memory.

Error conditions are the same as for "LIST 〈 Expression 〉 ENTER ".

Key operation	Execution start line	
RUN ENTER	5 ▵ A ▵________	
RUN 30 ENTER	30 ▵ B ▵________	
RUN 10 ENTER		Error (Error code: 2) occurs.
RUN SHFT ▵ B SHFT ▵ ENTER	30 ▵ B ▵________	
DEBUG SHFT ▵ D SHFT ▵ ENTER	10 ▵ D ▵________	

GOTO statement, GOSUB statement

- If the contents of the 〈 Expression 〉 is smaller than the program line which is being executed now, the program jumps to the desired program line by searching from the present line back to the starting point.
  If larger, the program jumps to the desired program line by searching from the present line to the final line.
- An error (error code: 2) will occur when the program has no program line corresponding to the contents of 〈 Expression 〉, or when the contents of 〈 Expression 〉 is smaller than or larger than the line number where program search begins.

Program		Note
	5 "A"________	
	20 GOTO 500	Jumps to line 500 of "PROG–2"
	30 "B"________	
	50 GOTO 150	Error occurs (code: 2)
"PROG.–1"	100 GOTO 30	Jumps to line 30 of "PROG.–1"
	200 GOSUB "C" :________	Jumps to line 30 of "PROG.–2" with sub-routine.
	300 A = 100	
	10 "D"________	
	20 GOTO 5	Jumps to line 5 of "PROG.–1"
	30 "C"________: RETURN	Jumps to line 200 of "PROG.–1" with sub-routine
"PROG.–2"		After the program jumps, it returns to the next statement.
	100 GOTO 30	Jumps to the line 30 of "PROG.–2"
	150 PRINT A, B	
	200 GOTO "B"	Jumps to the line 30 of "PROG.–1"
	300 GOTO 7	Error occurs (code: 2)
	500 END	

Others

The program will be executed from the starting program line after executing the final line of the preceding program.
In above example line 10 of "PROG.–2" will be executed, after ending line 300 of the "PROG.–1".

Tape Notes

1. If you use a recorder other than Radio Shack's CTR-80A, you may have to remove the REM plug before you are able to operate Fast-Forward and Rewind functions. Also, other difficulties may arise with normal functions.
2. Always use only the highest quality tape for program and data storage (economy grade audio type tape may not provide the proper characteristics for digital recordings). To insure best results, use Radio Shack's C-20 Cassettes, especially made for recording computer programs.
3. Keep the tape heads and tape handling parts clean – use a cassette cleaner/degamnetizer tape to keep everything clean.
4. Volume level can be very important when reading in data from the recorder; make slight adjustments as required to obtain error-free data transfer. A slight adjustment either up or down may result in perfect results every time.
5. Tapes made on one recorder may not function well on another recorder (e.g. a recording made with a Sony unit and played back on a CTR-80A may not function without errors). Use only the same model recorder for both recording and playback (if possible, use the same unit).
6. Be sure all connections between the Pocket Computer and Cassette Interface are secure. And be sure the connections between Interface and Recorder are secure and dirt-free.
7. If problems occur when using AC power for the recorder, use battery power instead (sometimes the AC power connection also adds some "hum" to the signal which upsets proper digital recordings).
8. Tone control – set to maximum treble (10).
9. Volume setting – for most Recorders from Radio Shack a setting between 4 and 10 should work out well (use 7 and leave it there). Other Recorders may not have this same range; try 6 to 8. If you consistenty have problems loading programs from cassette, adjust volume up or down slightly and try again.

SOME SAMPLE USER'S PROGRAMS

We've included some sample programs for your own benefit and entertainment. They will give you just a brief idea of the types of programs which can be run on your TRS-80 Pocket Computer.

NOTE: Please do not request special custom-written programs to be provided for your own applications needs. Radio Shack does not have the facilities to provide such services.

Notes for entering program listings:

1. Before entering program lines, set Computer to "PRO" (program) mode and enter [N] [E] [W] [ENTER] to clear out any existing program(s).
2. Press [ENTER] at the end of each program line. This automatically forces a colon to be added after each program line (no need to add that colon).
3. If you want to add a blank space inside quote marks, you'll have to enter one ([SPC] key). However it is not necessary to add a space between commands which use two or more characters.

   The following examples should help you

→ Numeric 0

→ A colon will automatically be entered by the Computer directly behind each line number (after you press [ENTER]).

→ Press [SHFT] key and then [W] key.

→ No need to press [SPC] key.

→ Press [ENTER] key, which enters the line into the Computer's memory.

```
10: "A": PAUSE "MANNING"
20: INPUT "R = "; R, "H = "; H, "L = "; L, "N = "; N
30: I = H/L: C = R ^(1/6)/N
```

→ Here you must press the [SPC] key.

→ Press [SHFT] key and then [√] key.

→ Use [Exp] key to enter this sign.

→ Numeric 0

```
410: IF (ABS F >= E - 4) + (ABS G >= E - 4) <> 0LET A = A+
     F: B = B + G: PAUSE F, G: GOTO 400
420: GOSUB 900
```

→ This indicates the letter O key.

4. After you enter the contents of the program list into the Computer and you execute the program, set the Computer in the DEF mode.

Title BIORHYTHM

[Formula]

The idea of biorhythm is that man's physical, emotional and intellectual conditions have a rhythm or cycle, from the very day of his birth.

Physical (P):	a cycle of 23 days
Emotional/Sensitivity (S):	a cycle of 28 days
Intellect (I):	a cycle of 33 days

The result of calculation with respect to date on which these conditions are the worst is as follows:

P:	0, 11 or 12
S:	0, 14
I:	0, 16 or 17

Calculation can be made, however, only for date of birth on or after March 1, 1900.

[Example]

To find the biorhythm on 3rd of March, 1977 for one whose date of birth is 4th of February, 1954:

[Operation]

C LOAD ▽I 1▽ ENTER

Display:	P: AFTER 1.5	(Means the date 1.5 days after the target date (March 3rd) is the worst day for physical condition.
	S: TODAY	(Means the target date is the worst day for emotion/sensitivity.)
	I : AFTER 3.5	(Means the date 3.5 days after the target date is the worst for the intellectual condition.)

	Input	Display	Note
1	SHFT A	BIRTHDAY?	
2	1954 ENTER	?	
3	2 ENTER	?	
4	4 ENTER	TARGET ?	
5	1977 ENTER	?	
6	3 ENTER	?	
7	3 ENTER	P AFTER 1.5	
8	ENTER	S TODAY	
9	ENTER	I AFTER 3.5	
10	SHFT B		
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			

Title BIORHYTHM

Memory content		
A	1	✓
B	2	✓
C	3	✓
D	4	Day
E	5	
F	6	
G	7	
H	8	
I	9	
J	10	
K	11	
L	12	
M	13	Month
N	14	N
O	15	
P	16	✓
Q	17	✓
R	18	
S	19	
T	20	
U	21	
V	22	
W	23	
X	24	✓
Y	25	Year
Z	26	

```
 10: "A":INPUT "BIRTHDAY ?",Y,M,D
 20: GOSUB 500
 30: X=N
 40: "B":INPUT "TARGET ?",Y,M,D
 50: GOSUB 500
 60: P=N-X
 70: A$="P":B=23
 80: GOSUB 540
 90: A$="S":B=28
100: GOSUB 540
110: A$="I":B=33
120: GOSUB 540
130: END
500: IF M-3>=0LET M=M+1:GOTO 520
510: Y=Y-1:M=13+M
520: N=INT (365.25*Y)+INT (30.6*M)+D
530: RETURN
540: C=P-INT (P/B)*B:BEEP 2
550: IF C>B/2LET Q=B-C:GOTO 590
560: IF B/2=CPRINT A$;" TODAY":RETURN
570: IF C=0PRINT A$;" TODAY":RETURN
580: Q=B/2-C
590: PRINT A$;" AFTER";USING "######.#";Q
600: RETURN
  337
```

Title GUESS-NUMBER GAME

[Formula]

This program lets you guess a number which the Computer generates. The number could be 1 to 4 digits.

When you guess correctly, the Computer beeps 5 times and the display shows "CONGRATULATION" together with trial number.

When you guess incorrectly, the display shows the following comments.

2	6578	ABBC	(In this case, Computer generated 4587 and your guess was 6578)
trial No.	guess No.	comments	

A: number and position both are right
B: number is right but position is not right
C: neither number nor position is right

The sequence of the comments (ABBC) does not relate the positions of the number you guessed.

[Operation]

At DEF Mode

N SHFT A When executing this instruction, the Computer generates a 4 digit number. N could be any number.

Guess number ENTER If the number entered is more than 5 digits, a sound will beep twice and display will show "EXCESSIVE INPUT". In this case re-enter a 4-digit number.

Input	Display	Note		Input	Display	Note
1 2 3 4 5 6			11	ENTER	GUESS NUMBER =	
SHFT A	GUESS NUMBER GAME		12	2607 ENTER	CONGRATULATION TRIAL = 5	
	GUESS NUMBER =		13	ENTER	GUESS NUMBER =	
1 2 3 4 ENTER	1 1234 BCCC		14	5678 ENTER	1 5678 ACCC	
ENTER	GUESS NUMBER =		15	(	(	
5 6 7 8 ENTER	2 5678 ABCC		16			
ENTER	GUESS NUMBER =		17	)	)	
9 0 1 2 ENTER	3 9012 BBCC		18			
ENTER	GUESS NUMBER =		19			
6 0 7 2 ENTER	4 6 0 7 2 BBBB		20			

Title GUESS-NUMBER GAME

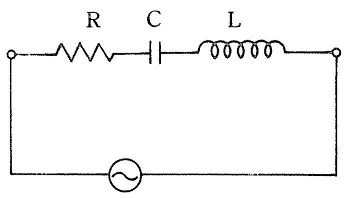
Memory content		
A	1	
B	2	Index
C	3	
D	4	Index
E	5	4 digits random No.
F	6	Input data
G	7	
H	8	No. of comment A
I	9	No. of comment B
J	10	
K	11	
L	12	Trail No.
M	13	A(13) Random No.
N	14	
O	15	
P	16	A(16)
Q	17	A(17) Guess No.
R	18	
S	19	
T	20	A(30)
U	21	For random No.
V	22	
W	23	For judgement
X	24	
Y	25	
Z	26	

```
 10: C=INT (A/10)
 20: A(B)=A-C*10
 30: A=C
 40: RETURN
 50: "A":AREAD F:U=0
 60: E=F:PAUSE "GUESS NUMBER GAME"
 70: L=0:A=E+1234:A=A*√A
 80: E=INT A-INT (A/E4)*E4:A=E
 90: FOR B=16TO 13STEP -1
100: GOSUB 10
110: NEXT B
120: INPUT "GUESS NUMBER =";F
130: IF F>E4BEEP 2:PAUSE "EXCESSIVE INPUT":
     GOTO 120
140: U=F+U:L=L+1
150: A=F:H=0:I=0
160: FOR B=20TO 17STEP -1
170: GOSUB 10
180: NEXT B
190: FOR B=17TO 20
200: FOR C=13TO 16
210: A=B-9:IF A(C)=A(B)LET H=H+1:
     C=16
220: NEXT C
230: NEXT B
240: FOR B=17TO 20
250: C=B-4:A=B-9
260: IF A(C)=A(B)LET I=I+1:H=H-1
270: NEXT B
280: IF E=FGOTO 320
290: V=I*10+H+500:GOSUB V
300: PRINT USING "###";L;USING "######";
     F;"  ";W$
310: GOTO 120
320: BEEP 5:PRINT "CONGRATULATION
     TRIAL=";USING "###";L
330: E=U+E:GOTO 70
340: END
500: W$="CCCC":RETURN
501: W$="BCCC":RETURN
502: W$="BBCC":RETURN
503: W$="BBBC":RETURN
504: W$="BBBB":RETURN
510: W$="ACCC":RETURN
511: W$="ABCC":RETURN
512: W$="ABBC":RETURN
513: W$="ABBB":RETURN
520: W$="AACC":RETURN
521: W$="AABC":RETURN
522: W$="AABB":RETURN
530: W$="AAAC":RETURN
531: W$="AAAB":RETURN
```

Title IMPEDANCE IN A SERIES CIRCUIT

[Formula]

Impedance in a series circuit



$$Z = |\dot{Z}| = \sqrt{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2} \quad [\Omega]$$

$$\theta = \tan^{-1}\left(\frac{\omega L - \frac{1}{\omega C}}{R}\right) \quad [^\circ]$$

$$\dot{Z} = R + j\left(\omega L - \frac{1}{\omega C}\right)$$

[Example]

$$\begin{cases} L = 25 & [\mathrm{mH}] \\ C = 10 & [\mu F] \\ R = 5 & [\Omega] \\ f = 50 & [\mathrm{Hz}] \end{cases} \qquad \begin{cases} Z = 310.5 & [\Omega] \\ \theta = -89.08 & [^\circ] \\ x = 5 & \\ y = -310.5 & \\ \therefore \dot{Z} = 5 - 310.5\,j & \end{cases}$$

[Operation]

C LOAD ▽ E 1 ▽ ENTER

Input	Display	Note		Input	Display	Note
SHIFT A	L =		11			
(L)25E−3 ENTER	C =		12			
(C)10E−6 ENTER	R =		13			
(R)5 ENTER	F(HZ) =		14			
(f)50 ENTER			15			
	X 5		16			
ENTER	Y −310.455 ···		17			
ENTER	Z 310.496 ···		18			
ENTER	PHASE −89.077 ···	(θ)	19			
			20			

Title IMPEDANCE IN A SERIES CIRCUIT

Memory content		
A	1	
B	2	
C	3	C
D	4	
E	5	
F	6	f
G	7	
H	8	
I	9	√
J	10	
K	11	
L	12	L
M	13	
N	14	
O	15	
P	16	
Q	17	
R	18	R
S	19	
T	20	
U	21	
V	22	
W	23	
X	24	Phase
Y	25	
Z	26	Z

```
 10: "A":DEGREE :INPUT "L=";L
 20: INPUT "C=";C
 30: INPUT "R=";R
 40: INPUT "F(HZ)=";F
 50: F=2*π*F
 60: L=L*F
 70: C=C*F
 80: I=L-1/C
 90: Z=√(R*R+I*I)
100: X=ACS (R/Z)
110: IF 0>ILET X=-X
120: USING
130: PRINT "X",R
140: PRINT "Y",I
150: PRINT "Z",Z
160: PRINT "PHASE", X
170: END
  172
```

Title DAYS BETWEEN DATES

[Formula]

This program figures the number of days between two dates. Leap years are taken into account.

[Example]

From 10/5/1976 to 2/20/1977: 138 days
From 10/5/1976 to 11/15/1977: 406 days

[Operation]

```
C LOAD ▽H1▽ ENTER
                      SHFT  A
        Reference year      ENTER
        Reference month     ENTER
        Reference day       ENTER
  ┌───▶ Appointed year      ENTER
  │     Appointed month     ENTER
  │     Appointed day       ENTER      Number of days displayed
  └─────                    ENTER
```

NOTE: To set a reference date, start operation from A

Input	Display	Note		Input	Display	Note
SHFT A	START YEAR=		11			
1976 ENTER	MONTH=		12			
10 ENTER	DAY=		13			
5 ENTER	END YEAR=		14			
1977 ENTER	MONTH=		15			
2 ENTER	DAY=		16			
20 ENTER			17			
	DAYS 138		18			
			19			
			20			

Title	DAYS BETWEEN DATES	

Memory content		
A	1	
B	2	
C	3	
D	4	
E	5	
F	6	
G	7	Month for SUB.
H	8	Year for SUB.
I	9	Day for SUB.
J	10	No. of days 1
K	11	
L	12	
M	13	
N	14	
O	15	
P	16	
Q	17	
R	18	Start year
S	19	Start month
T	20	Start day
U	21	End year
V	22	End month
W	23	End day
X	24	No. of days wanted
Y	25	
Z	26	✓

```
 10: "A":Y=0
 20: INPUT "START YEAR=";R,"MONTH=";S,"DAY="
    ;T
 30: INPUT "END YEAR=";U,"MONTH=";V,"DAY=";W
 40: IF Y=1LET H=R+1925:GOTO 60
 50: H=R
 60: G=S:I=T
 70: GOSUB 500
 80: J=I
 90: IF Y=1LET H=U+1925:GOTO 110
100: H=U
110: G=V:I=W
120: GOSUB 500
130: X=I-J
140: PRINT "DAYS",X
150: GOTO 30
500: IF G-3>=0LET Z=-(G-3)*30.6-.5:GOSUB 600
    :I=I-Z:GOTO 530
510: H=H-1
520: Z=(-(G-3)-12)*30.6-.5:GOSUB 600:I=I-Z
530: Z=H*365.25:GOSUB 600:I=I+Z
540: Z=H/100:GOSUB 600:I=I-Z
550: Z=H/400:GOSUB 600:I=I+Z
560: I=I-307:RETURN
600: X=INT ABS Z:Z=SGN Z*X:RETURN
  385
```

Title	RANDOM NUMBERS	

[Formula]

Generate random numbers according to the congruence method.

$$x_{n+1} = 23x_n - \mathrm{int}\left(\frac{23x_n}{10^8+1}\right) \times (10^8+1)$$

x_0 is an arbitrary 8-digit integer.

[Example]

Clear all memories, and generate ten random numbers with the initial value = 0.

[Operation]

CLOAD "B11" ENTER

Note: Make sure memory X is loaded with no character. If the memory is loaded with a character, an error will occur.

Input	Display	Note		Input	Display	Note
SHFT A	INITIAL VALUE =		11	ENTER	9. 13784262	
0 ENTER	NUMBER =		12	ENTER	10. 17038023	
10 ENTER	1 10100381		13			
ENTER	2 32308761		14			
ENTER	3 43101496		15			
ENTER	4 91334399		16			
ENTER	5 691156		17			
ENTER	6 15896588		18			
ENTER	7 65621521		19			
ENTER	8 9294968		20			

Title RANDOM NUMBERS

Memory content		
A	1	i
B	2	$23x$
C	3	
D	4	
E	5	10^8+1
F	6	
G	7	
H	8	
I	9	
J	10	
K	11	
L	12	
M	13	
N	14	n
O	15	
P	16	
Q	17	
R	18	
S	19	
T	20	
U	21	
V	22	
W	23	
X	24	
Y	25	
Z	26	

```
 10: "A":INPUT "INITIAL VALUE =";Z
 20: INPUT "NUMBER=";N
 30: X=ABS (439147+X+Z)
 40: E=E8+1
 50: FOR A=1TO N
 60: B=23*X
 70: X=B-INT (B/E)*E
 80: BEEP 2:PRINT A,X
 90: NEXT A
100: END
  121
```

Title	NORMAL DISTRIBUTION AND PERCENTILE	

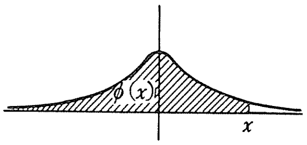
[Formula]

Determine the normal distribution function $\phi(x)$ and its inverse function (percentile) according to Hasting's best approximate equation.

- $\phi(x)$

Suppose, $\phi(x) = \int_{-\infty}^{t} \phi_t \, dx$

$$\phi_t = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$



$$t = \frac{1}{1+px}$$

we obtain,

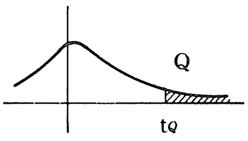
$$\phi(x) = 1-\phi_t(c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5)$$

$P = 0.2316419$	$c_1 = 0.31938153$
$c_2 = -0.356563782$	$c_3 = 1.78147937$
$c_4 = -1.821255978$	$c_5 = 1.330274429$

- Percentile

$$x = \sqrt{\ln Q^{-2}}$$

$$t_Q \fallingdotseq x - \frac{c_0 + c_1 x + c_2 x^2}{1 + d_1 x + d_2 x^2 + d_3 x^3}$$



$c_0 = 2.515517$	$d_1 = 1.432788$
$c_1 = 0.802853$	$d_2 = 0.189269$
$c_2 = 0.010328$	$d_3 = 0.00138$

[Example]

$\phi(x)$ .... $x = 2$; percentile .... $Q = 0.05$

[Operation]

CLOAD ▽B3▽ ENTER

	Input	Display	Note		Input	Display	Note
1	2 SHFT A	P 9.77249 E-01	$\phi(x)$	11			
2				12			
3				13			
4	0.05 SHFT B	TQ 1.645361125	t_Q	14			
5				15			
6				16			
7				17			
8				18			
9				19			
0				20			

Title	NORMAL DISTRIBUTION AND PERCENTILE

Memory content

A	1	
B	2	
C	3	
D	4	
E	5	
F	6	
G	7	
H	8	
I	9	
J	10	
K	11	
L	12	
M	13	
N	14	
O	15	
P	16	p
Q	17	T_Q
R	18	
S	19	
T	20	
U	21	
V	22	
W	23	
X	24	
Y	25	working area
Z	26	Input area

```
 10: "A":AREAD Z
 20: Y=1/(1+.2316419*Z)
 30: A=.31938153:B=-.356563782
 40: C=1.78147937:D=1.330274429
 50: E=-1.821255978
 60: F=C+Y*(D*Y+E):P=1-EXP (-.5*Z*Z)/√(2*π)*
    Y*(A+Y*(B+Y*F))
 70: BEEP 2:PRINT "P",P
 80: END
 90: "B":AREAD Z
100: Y=√LN (1/Z/Z)
110: A=2.515517:B=.802853:C=.010328
120: D=1.432788:E=.189269:F=.00138
130: Q=Y-(A+Y*(B+C*Y))/(1+Y*(D+Y*(E+F*Y)))
140: BEEP 2:PRINT "TQ",Q
150: END
  319
```

itle	

Input	Display	Note		Input	Display	Note
			11			
			12			
			13			
			14			
			15			
			16			
			17			
			18			
			19			
			20			

Title	

Memory content			Line number	Statements
A	1			
B	2			
C	3			
D	4			
E	5			
F	6			
G	7			
H	8			
I	9			
J	10			
K	11			
L	12			
M	13			
N	14			
O	15			
P	16			
Q	17			
R	18			
S	19			
T	20			
U	21			
V	22			
W	23			
X	24			
Y	25			
Z	26			

YOUR OWN PROGRAM

Title	

	Input	Display	Note		Input	Display	Note
				11			
				12			
				13			
				14			
				15			
				16			
				17			
				18			
				19			
				20			

Title	

Memory content			Line number	Statements
A	1			
B	2			
C	3			
D	4			
E	5			
F	6			
G	7			
H	8			
I	9			
J	10			
K	11			
L	12			
M	13			
N	14			
O	15			
P	16			
Q	17			
R	18			
S	19			
T	20			
U	21			
V	22			
W	23			
X	24			
Y	25			
Z	26			

INDEX

Page

A

Absolute Value (ABS) 23
Addition and Subtraction 18
Angle Conversions (DMS, DEG) 22
Angular Symbol 16
Appendix. 87
AREAD Statement 69
Arithmetic Calculation 18
Arithmetic Operator 18

B

Battery Indicator. 16
Battery Replacement 89
BEEP Statement 68

C

Calculations with Parentheses 20
CHAIN Statement. 76
Character Variable. 45
Checking Reserve Programs 83
Checking Stored Programs. 36
Checking the Program (after CLOAD1) 98
CLEAR Statement. 68
CLOAD Statement 75
CLOAD? Statement. 75
CLOAD1 Statement. 96
Command Statement 70
Computation Range. 17
Configuration of Reserve Programs 84
Connecting the Cassette Interface. 90
Connecting the Cassette Interface to a Tape Recorder 92
Connecting the Pocket Competer to the Cassette Interface 91
CONT Command. 71
Correction of Reserve Programs 83
CSAVE Statement. 74
Cursor. 14
Cursor positioning. 28

D

DEBUG Command 71
Debugging Programs. 41
Definable Mode (DEF) 13
Defined Progeam. 42
DEGREE Statement 69
Deleting Lines. 39
Deleting Reserve Programs 84
Display 14
Display System 16

E
Editing Porgrams on Magnetic Tape 95
Editing Expressions 26
END Statement. 68
Error Codes 85
Executing Programs 39
Execution of the Program (after CLOAD1) 99
Exponential Function (EXP) 22

F
Features 1
Fixed Memory 46
Flexible Memory 46
Flow Chart. 67
FOR Statement. 64
Functions 3
Functions of Keys 9

G
GOTO Statement 59
GOSUB Statement. 62
GRAD Statement 69

H

I
IF Statement 60
Indirect Designation. 47
INPUT # 79
INPUT Statement 52
Inputting Data 17
Inputting to Variables 48
Inserting Lines 38
Integer (INT) 23
Introduction. 1
Inverse Trigonometric Functions (ASN, ACS, ATN). 21

J

K
Keyboard. 8

L
Label 35
LET Statement. 51
Levels of Pending Operation 29
Line 35
Line Number 35
LIST Command 72
Logarithmic functions (LN LOG) 22
Loading from the Magnetic Tape 93
Logic Functions 24
Logic Operator 18

M
Manual Calculations. 18
MEM Command 73
Memory Calculations 25
Mode 13
Mode Symbol 16

N

Name Label 1
NEW Command 73
NEXT Statement. 64
Number of input Characters 16
Numerical Variable 45

O

Operating the Cassette interfece and recorder 93
Organization of Program Memory. 36

P

Partial correction. 37
PAUSE Statement 57
Power Calculations 20
PRINT # Statement 78
PRINT Statement 54
Priority of Calculations. 29
Program Correction 37
Programmed Calculations 31
Program mode (PRO). 13
Program Memory. 47
Program Statement 51
Prompt Symbol. 14
Program Verification 94

Q

R

RADIAN Statement. 69
Rapid Debugging. 41
Recall Functions. 26
Recalling the Contents of Variables. 49
Recording onto Magnetic Tape. 93
REM Statement 70
Reservable Keys 80
Reserve Memory for Reservable Keys 80
Reserve Program Mode (RESERVE) 13
RETURN Statement 62
RUN Command 70
RUN Mode (RUN). 13

S

Scientific Functions. 21
Second Functions 13
Shift Symbol 16
Sign Function (SGN) 23
Specifications. 88
Specifying Variables. 46
Square Roots 22
Statement 4
Statement for optional cassette interface 74
Steps. 35
STOP Statement 68

T

Table of Error Codes 86
Table of Functions and Statements. 3
Tape Notes. 101
Trigonometric Function (SIN, COS, TAN) 21

U

USING Statement 57

V

Variables 45

W

Writing Programs. 34

X

Y

Z

RADIO SHACK A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
280-316 VICTORIA ROAD RYDALMERE, N.S.W. 2116	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

1A1 PRINTED IN JAPAN