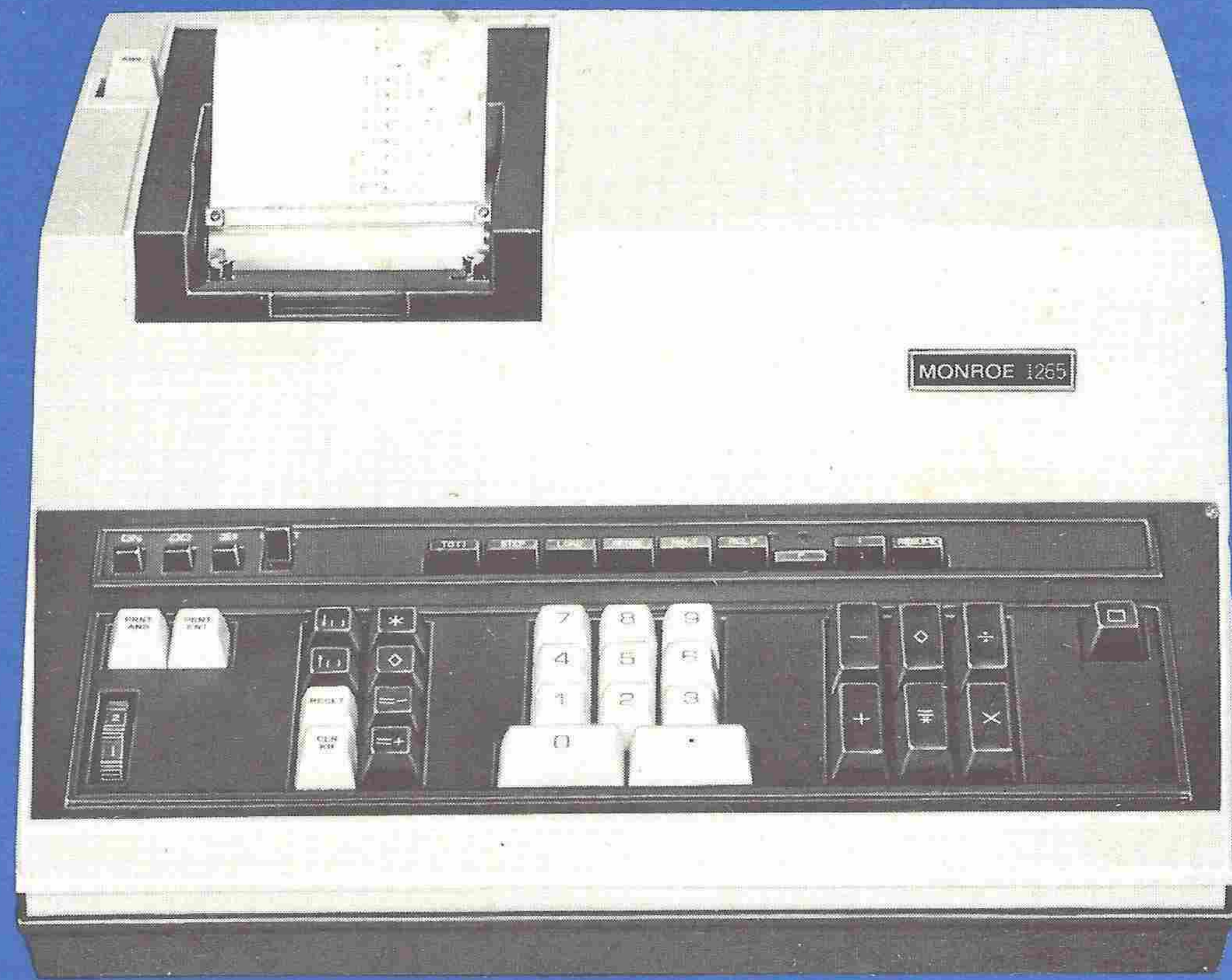
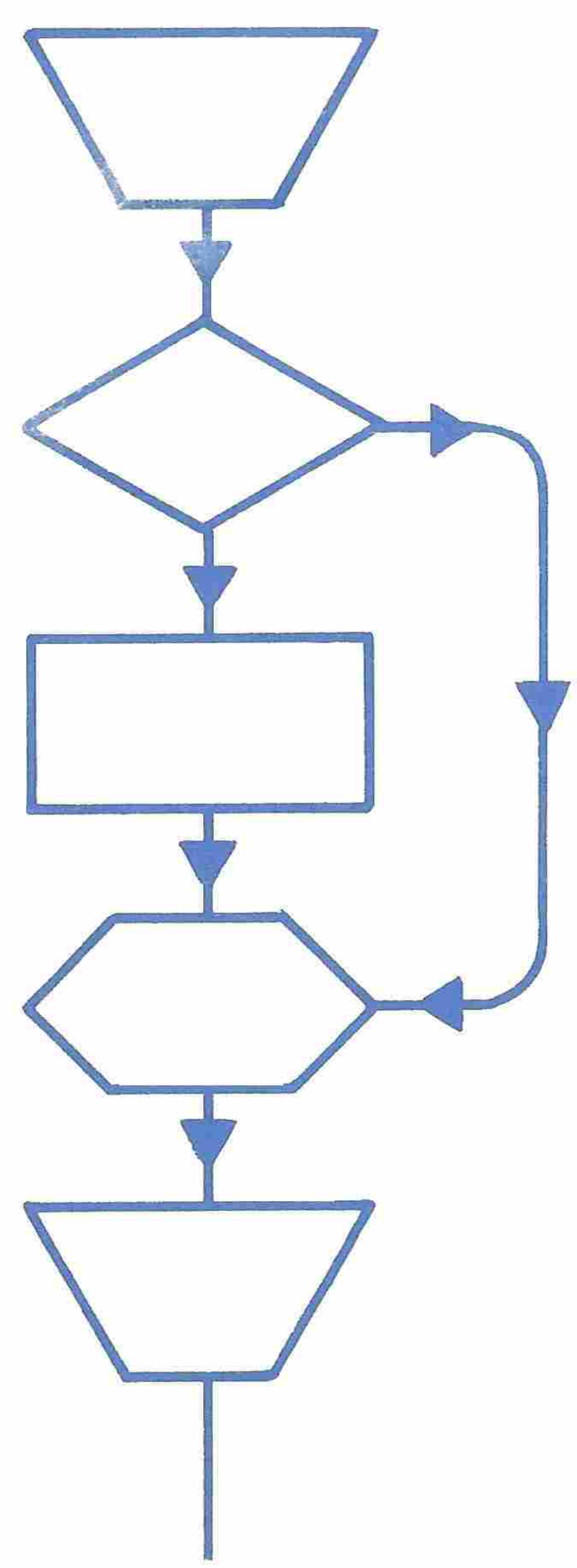


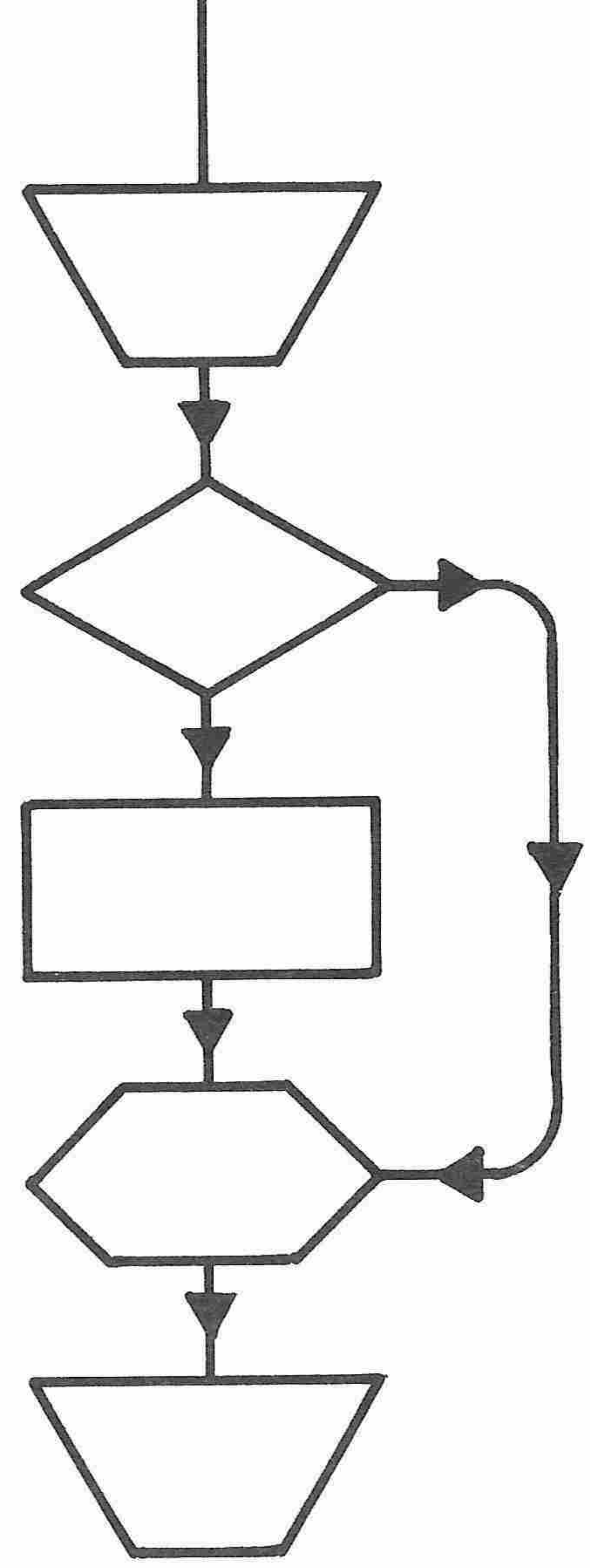
Monroe model 1265

operating and programming instructions



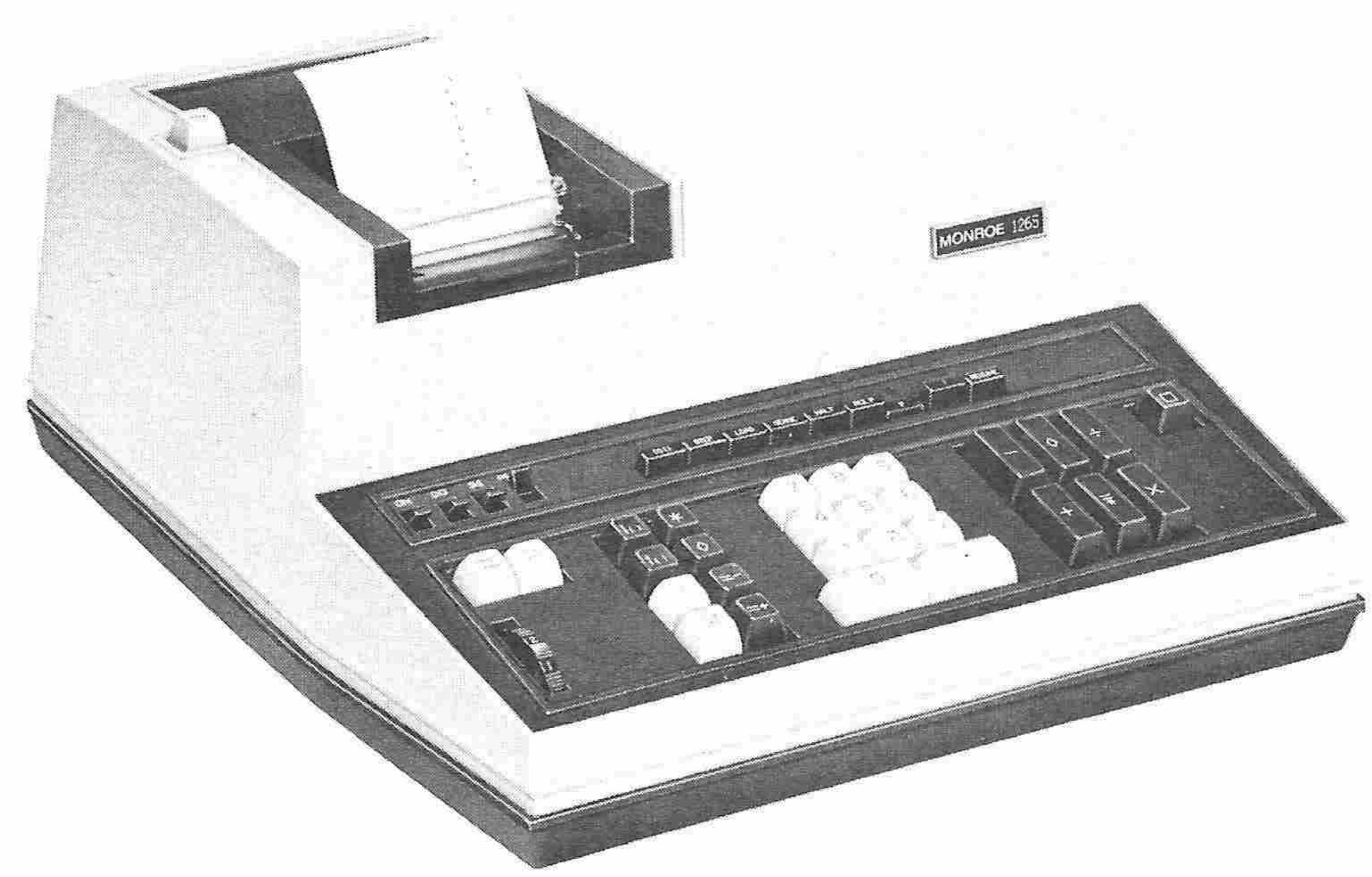
MONROE

THE CALCULATOR COMPANY



Monroe model 1265

operating and programming instructions



MONROE

THE CALCULATOR COMPANY

CONTENTS

CONTENTS

INTRODUCTION

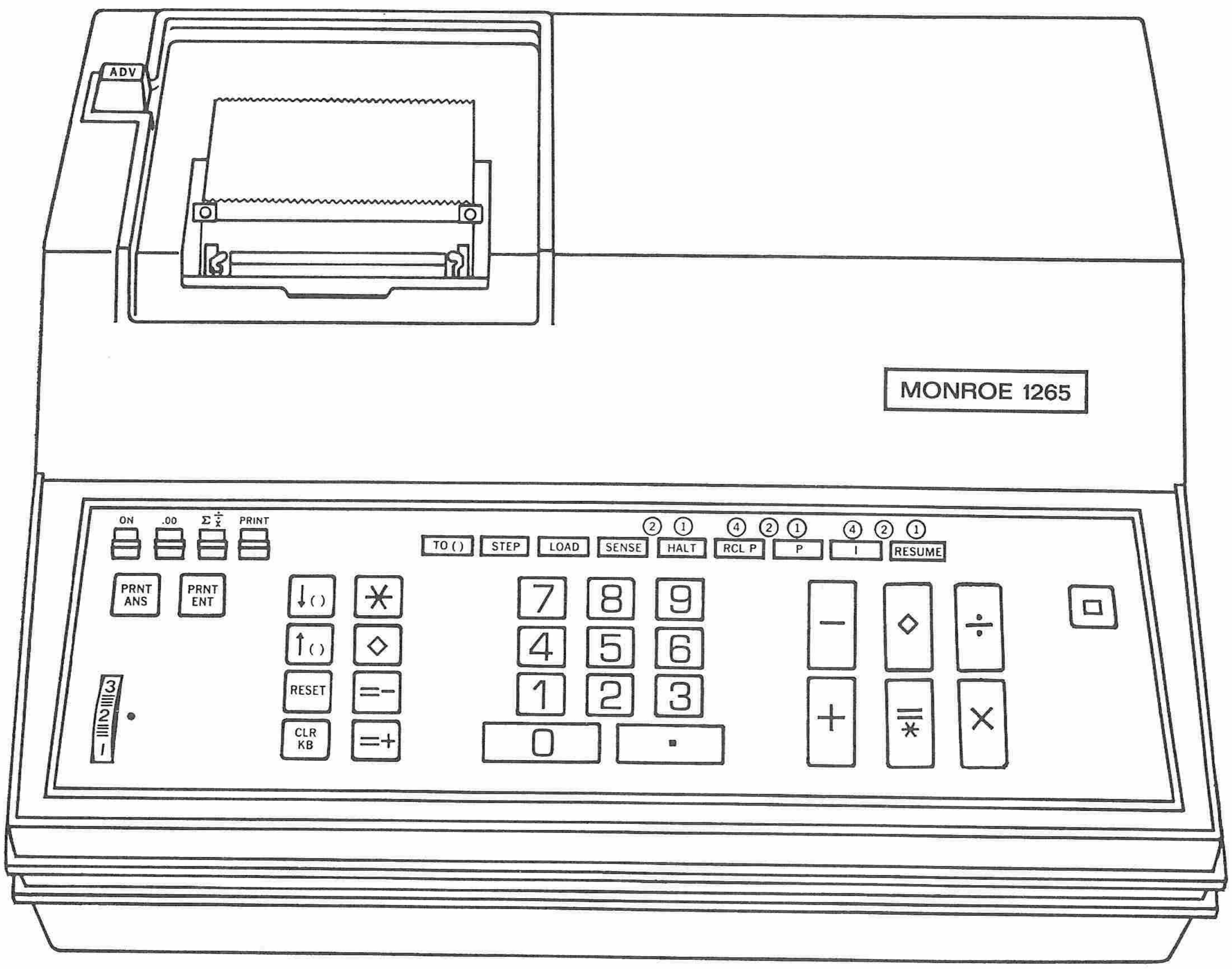
The Model 1265 is a compact and versatile electronic programmable calculator with a rapid, reliable printing mechanism. It is a simple calculator to use, making the transition from an ordinary desk calculator easy.

Keyboard programming with the Learn Mode Programmer (LEMP) eliminates repetitious operations. All the basic keyboard functions can be programmed, along with the ability to branch unconditionally, which allows for subroutines and the storing of more than one program.

Additional programming power and flexibility are achieved with the optional Monroe CR-1 card reader. The programming commands available through the card reader permit more powerful programs to be written and punched on cards, for permanent storage.

The operating controls are logically arranged in groups to provide the easiest and simplest operation possible.

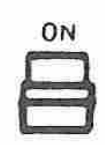
MONROE ELECTRONIC PROGRAMMABLE PRINTING CALCULATOR MODEL 1265



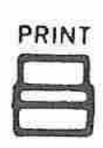
BASIC OPERATING INSTRUCTIONS

Set-up Group

The pre-setting of the Decimal Wheel, Double Zero Switch, Automatic Summation Switch and the Print Switch provide the operator with flexible control over the printed output.



Power Switch   Move to ON position. The calculator is ready for operation with all registers clear.



Print Switch   This switch should be in the UP position for the printing of all manual operations.

Decimal Wheel   The number of decimal places printed in entries and results is determined by the setting of the Decimal Wheel. Settings are 0 through 9. The maximum of thirteen decimal digits may be entered but only nine decimal digits will print.

When adding or subtracting, if the number of decimal digits entered exceeds the setting of the Decimal Wheel, the excess digits will not be printed or calculated.

When multiplying or dividing, the 1265 accepts all decimal digit entries and will calculate on a floating basis, but only the number of decimals selected on the Decimal Wheel will print. All results are automatically rounded.

The Decimal Wheel may be changed at any time without effecting the decimal accuracy of any numbers present in the calculator.

Double Zero Round-off Switch When in the UP position this switch allows for the printing of results that are rounded off two places less than the selection on the Decimal Wheel. For example: with the Decimal Wheel at 4, results are rounded off to two decimal places; with the Decimal Wheel at 5, results are rounded off to three decimal places, etc.

Example 125 items @ 2.45 3/4 ea. = ?

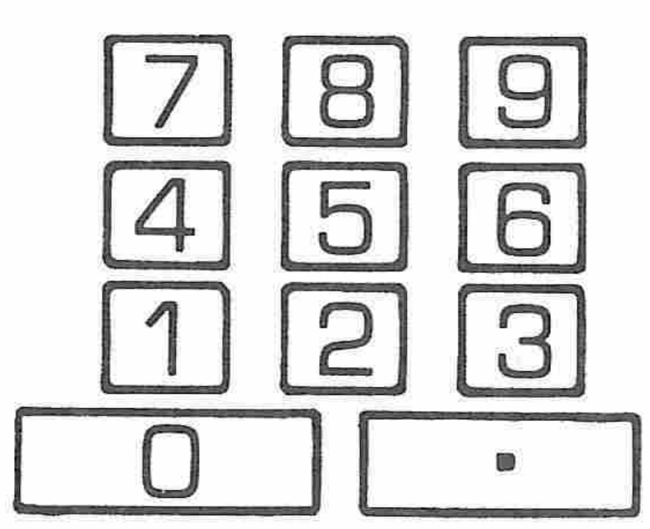
Decimal Wheel at 4

.00 Switch UP

Depress keys 125 x 2.4575 $\overset{=}{*}$

```
125•0000   X
  2•4575=
307•1900   *
```

Automatic Summation Switch With this switch in the UP position, multipliers or dividends are automatically accumulated in storage register 1.



Keyboard The 1265 has a standard 10-key keyboard and decimal point key. Numbers are indexed in the same sequence as they are written, including the decimal point, whenever it appears in a number.

The maximum entry of decimal digits is 13, with a combined total of 14 digits of whole and decimal digits.

Basic Function Group

These keys perform the basic arithmetic functions of addition, subtraction, multiplication and division. They are conveniently located to allow for easy operation.

[+] Plus Key Adds the number in the keyboard to the adding register; the number may be repeated by repetitive depressions of the plus key.

[−] Minus Key Subtracts the number in the keyboard from the adding register; the number may be repeated by repetitive depressions of the minus key.

Example 5 + 7 + 3 - = ?

Depress keys 5 + 7 + 3 - $\overset{=}{*}$

```
5•00   +
7•00   +
3•00   -
9•00 *
```

[×] Multiply Key Sets up the number in the keyboard as a multiplier. Can be used instead of the equals key for intermediate results that are to be used as multipliers. The intermediate result will not print.

Example 12 x 13 = ?

Depress keys 12 x 13 $\overset{=}{*}$

```
 12•00   X
 13•00=
156•00 *
```

The multiplier is retained to allow for constant multiplication.

Example 12 x 13 = ?

12 x 16 = ?

12 x 24 = ?

Depress keys 12 x 13 $\overset{=}{*}$

16 $\overset{=}{*}$

24 $\overset{=}{*}$

```
  12•00   X
  13•00=
 156•00  *
  16•00=
 192•00  *
  24•00=
 288•00  *
```

| ÷ |

Divide Key Sets up the number in the keyboard as a dividend. Can be used instead of the equals key for intermediate results that are to be used as dividends. The intermediate result will not print unless.

Example 12 ÷ 5 = ?

Depress keys 12 ÷ 5 $\overset{=}{*}$

```
12•00   ÷
 5•00=
 2•40  *
```

| ◇ |

Sub-Total Key Prints the total in the adding register but does not clear the register.

| $\overset{=}{*}$ |

Equals/Total Key After a number has been set up by the multiply or divide key, this key multiplies or divides by the number in the keyboard to complete the multiplication or division.

The Equals/Total Key also prints the total in the adding register and clears the register. After a printing of the total this key becomes inoperable until another operation is performed.

Register Control Group

This group includes the keys that control the flow of data to and from the storage registers, the automatic accumulation memory keys, and the clearance keys.

$\boxed{\downarrow()}$ Store Key The seven storage registers are designated as 1 through 7. The depression of this key, followed by a digit key, will store and print the number in the keyboard followed by the identification of the selected register. The accumulating memory may also be used as a storage register by using this store key followed by the digit 0.

$\boxed{\uparrow()}$ Recall Key Depression of this key followed by the appropriate digit key will recall the stored number to the keyboard and print it with the register identification. The contents of the register remain unchanged.

$\boxed{=+}$ Equal Plus Key After a number has been set up by either the x key or the ÷ key, this key will multiply or divide by the number in the keyboard to complete the multiplication or division, print the individual result and add the result to the accumulating memory. This key is also used to print the total in the adding register and automatically add it to the accumulating memory.

Example (11 x 13) + (22 x 14) + (17 x 12) = ?

Decimal on "0"

Depress RESET

11 x 13 =+

22 x 14 =+

17 x 12 =+

*

```
    0 • ^
   11 •    X
   13 • =+
  143 •  *
   22 •    X
   14 • =+
  308 •  *
   17 •    X
   12 • =+
  204 •  *
  655 • *M
```

[=-] Equals Minus Key After a number has been set up by either the x key or the ÷ key, this key will multiply or divide by the number in the keyboard to complete the multiplication or division, print the individual result and subtract the result from the accumulating memory.

This key is also used to print the total in the adding register and automatically subtract it from the accumulating memory.

Example (11 x 13) + (22 x 14) - (17 x 12) = ?

Decimal on "0"

Depress RESET

11 x 13 =+

22 x 14 =+

17 x 12 =-

*

```
    0 • ^
   11 •    X
   13 • =+
  143 •  *
   22 •    X
   14 • =+
  308 •  *
   17 •    X
   12 • =-
  204 •  *
  247 • *M
```

◇ **Memory Sub-Total Key** Prints the total in the accumulating memory but does not clear it.

✱ **Memory Total Key** Prints and clears the total in the accumulating memory.

RESET **Reset Key** Clears all registers except storage registers 2 through 7.

CLR KB **Clear Keyboard Key** Clears an incorrect keyboard entry.

□ **Special Function Key** This key is used with digit keys 6, 7, 8, and 9 to operate programs that are permanently stored in the calculator. The programs are as follows:

□ , 6 is amount and percent change

<u>Example</u>

This Year	Last Year	Amt. Diff.	Percent Change
579090	509912	69178	13.57 Inc.
96630	109063	12433	11.40 Dec.

Decimal Wheel at 2

Depress RESET 579090 + 509912 □ 6
96630 + 109063 □ 6

```
        0.00^
   579090.00   +
   509912.00□6
    69178.00   *
       13.57   *

    96630.00   +
   109063.00□6
 -  12433.00   *
 -     11.40   *
```

□ , 7 is Percent Minus

<u>Example</u> $125.50 less 5 - 12.5=$104.32

Decimal Wheel at 2

Depress RESET 125.5 + 5 □ 7 12.5 □ 7

```
     0.00^
   125.50   +
 -   5.00□7
 -   6.28   *
   119.22   *
 -  12.50□7
 -  14.90   *
   104.32   *
   104.32   *
```

□ , 8 is Percent Plus

Example $155.00 plus 5% = $162.75

Decimal Wheel at 2

Depress RESET 155 + 5 □ 8 $\overset{=}{*}$

```
  0•00Λ
155•00   +
  5•00□8
  7•75  *
162•75  *
162•75  *
```

□ , 9 is Square Root

Example $\sqrt{625}$

Decimal Wheel at 0

Depress RESET 625 □ 9

```
  0•Λ
625•□9
 25•  *
```

The Special Function key is also used with digits 0 through 5 for conditional branching, which will be explained in the programming section.

ADV   Electric paper advance

Error and Overflow Conditions

Illegal keyboard data entries and illegal operations are flagged by an error or overflow signal. When such a condition occurs, the word ERROR or OVERFLOW will print on the tape and the keyboard will lock, inhibiting the entry of any data and execution of any operations. The error or overflow condition can be cleared by depressing either $\overset{\text{CLR}}{\text{KB}}$ or RESET. Depressing $\overset{\text{CLR}}{\text{KB}}$ clears only the keyboard; depressing RESET clears the keyboard and registers 0 and 1.

```
                 0•00   ÷
                 0•00=
ERROR•••••••••••      9
                 0•00C
```

```
  1234567899•00   X
123456789155•00=
OVERFLOW•••••••   •
           0•00C
```

INTRODUCTION TO PROGRAMMING

The Model 1265 Programmable Printing Calculator, with its "learn mode programmer", offers the advantages of computer-like programming power while, at the same time, retaining all the calculating ease and keyboard simplicity of the basic calculator. The learn mode programmer, commonly called the LEMP, "learns" how to solve a problem as it is entered through the keyboard. Thereafter, the same problem may be solved repeatedly for new values with very little manual intervention.

The learning capability of the LEMP is made possible by the addition to the basic calculator of 128 programming steps with branching and looping features. These programming steps are independent of the data storage registers.

Programming the calculator from the keyboard follows essentially the same procedures as normal keyboard operation except that a few additional LEMP keys are used to set the LEMP in operation and establish the program starting point. Eight starting points, called branch points, are provided for subroutine programming so that operations that are used repeatedly can be performed without going through the entire program sequence each time. The branch points also permit more than one program to be stored in the calculator at one time.

The addition of the optional card reader expands the programming power and flexibility of the calculator by permitting access to internal features that are not represented by keys on the keyboard.

KEYBOARD PROGRAMMING

The transition from manual operation to keyboard programming requires an understanding of the operating registers of the calculator and the nine LEMP keys and switches. The important features of the operating registers can be summarized as follows:

E-Register   This is the entry register, which receives all keyboard data input and retains the results of each calculation.

A-Register   This is the accumulator register. It is used in all calculations. Results of arithmetic operations are always placed in the A-register as well as the E-register.

M-Register   This register holds the multiplier during multiplication and the divisor during division; this action allows for constant multiplication and division.

Registers 0 through 7   These registers are used to store data for later use. Register 0 is the automatic accumulation register, and register 1 is used for accumulating dividends or multipliers when the $\overset{\div}{\Sigma_x}$ switch is in the UP position.

PROGRAM MEMORY DIAGRAM*

Branch Point	Step	Branch Point	Step	Branch Point	Step	Branch Point	Step
0	00	2	32	4	64	6	96
	01		33		65		97
	02		34		66		98
	03		35		67		99
	04		36		68		100
	05		37		69		101
	06		38		70		102
	07		39		71		103
	08		40		72		104
	09		41		73		105
	10		42		74		106
	11		43		75		107
	12		44		76		108
	13		45		77		109
	14		46		78		110
	15		47		79		111
1	16	3	48	5	80	7	112
	17		49		81		113
	18		50		82		114
	19		51		83		115
	20		52		84		116
	21		53		85		117
	22		54		86		118
	23		55		87		119
	24		56		88		120
	25		57		89		121
	26		58		90		122
	27		59		91		123
	28		60		92		124
	29		61		93		125
	30		62		94		126
	31		63		95		127

*Program steps numbered consecutively beginning with 00.

There are nine LEMP keys and switches. Two of these, TO( ) and HALT, are essential in entering any program.

The 128 program steps are divided into eight groups of 16 steps each as shown in the diagram, and a program may be entered at the first step in any one of these groups. These entry steps are called branch points. Numeral keys 0 through 7 are used to specify which branch point has been selected.

TO ( ) Depressing this key followed by the depression of a numeral key, 0 through 7, selects the next step to be executed. For example, depressing TO( ) key followed by numeral 2, selects step 32 (branch point 2) as the next step to be executed. TO( ) followed by 6 selects step 96 (branch point 6) as the next step to be executed.

HALT This key is depressed when loading a program to instruct the calculator to stop for a keyboard entry during programmed operation.

LOAD Will lock when depressed and puts the calculator in the learn-mode. Each keyboard depression is recorded as a program step. This key must be released at the completion of the program loading.

RESUME When the LOAD key is released, the calculator is ready to execute the program, but actual program execution does not begin until the RESUME key is depressed after the depression of the TO( ) and branch point numeral key; where program was loaded. The program will operate and stop at the first HALT instruction. RESUME must be depressed after each HALT instruction to advance the program.

[RCL P] This instruction is used at the end of a subroutine to return the program from which the branch was made to the subroutine.

[STEP] Will lock when depressed and is used for the execution of a program one step at a time, as when checking or debugging a program. RESUME is used to advance the program.

[P] Will lock when depressed. With the P and STEP keys both down, the Program Monitor Lights will display, in octal code, the number of the next program step to be executed.

[I] Will lock when depressed. To read a particular instruction, the STEP, I, and LOAD...keys must be down. The Program Monitor Lights will display the instruction in octal code.

[SENSE] Will lock when depressed. This key is used only in conjunction with the card reader.

<u>Print Control Switch and Keys</u>

PRINT When in the UP position, all entries and functions will print on the tape.

When in the DOWN position, no printing will take place in any operation except as selected by the operator or through programmed selection.

[PRNT ENT] This key is depressed after a HALT in a program to print the contents of the keyboard with an E symbol.

PRNT ANS This key is depressed after the completion of a calculation, or the recall of data from storage, to print the contents of keyboard with an A symbol.

This sample problem is programmed as follows:

$$\frac{(A \times B) - C}{D} =$$

B = 23.75 a constant, and is part of the program; each digit requires a program step as does the decimal point.

A, C, and D are variables

The program must start at one of the branch points. For this problem, the branch point is 0.

On a coding sheet list the functions in the COMMAND column as shown in Figure 1 (next page). The column labled ADDRESS is used to keep track of the program steps. This is important when more than one program is to be entered, to insure that the programs do not overlap. (The function of the CODE column is explained later under "DEBUGGING.")

The TO( ), 0 instruction at the end directs the program back to the starting branch point. This instruction requires just one program step.

PROGRAM

Figure 1
Sample Coding Sheet

$$\frac{(A \times B) - C}{D} =$$

NO.________ PAGE 1 OF 1

DATE________ MODEL 1265

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS: E A M 0 1 2 3 4 5 6 7 8 9
0	0	0	0	HALT	4	0	1	Enter A
			1	PRT E	0	2	6	
			2	x	0	7	0	
			3	2	0	0	2	
			4	3	0	0	3	B (constant)
			5	.	0	1	2	
			6	7	0	0	7	
			7	5	0	0	5	
	0	1	0	=*	0	2	0	
			1	PRT A	0	2	7	
			2	+	0	6	0	
			3	HALT	4	0	1	Enter C
			4	PRT E	0	2	6	
			5	-	0	6	2	
			6	=*	0	2	0	
			7	÷	0	7	2	
1	0	2	0	HALT	4	0	1	Enter D
			1	PRT E	0	2	6	
			2	=*	0	2	0	
			3	PRT A	0	2	7	
			4	TO (0)	7	4	0	
			5					
			6					
			7					
			0					
			1					
			2					
			3					
			4					
			5					
			6					
			7					

Programmed by________

Loading a Program

To load the program just written on the coding sheet, the following procedure is used:

LOAD key must be UP

Depress RESET

Depress TO( ), 0

Depress LOAD key

Depress HALT

Depress PRNT ENT

Depress x

Depress 2

Depress 3

Depress .

Depress 7

Depress 5

Depress $\overset{=}{*}$

Depress PRNT ANS

Depress +

Depress HALT

Depress PRNT ENT

Depress -

Depress $\overset{=}{*}$

Depress ÷

Depress HALT

Depress PRNT ENT

Depress $\overset{=}{*}$

Depress PRNT ANS

Depress TO( ), 0

Depress LOAD key to release

Executing a Program

To execute the program just loaded, apply the following values:

A = 12 $\qquad\qquad \dfrac{(12 \times 23.75) - 15.5}{8.45} = 31.893$

C = 15.5

D = 8.45

```
 12.000E
285.000A
 15.500E
  8.450E
 31.893A
```

Print Switch DOWN

Decimal Wheel at 3

Depress RESET

Depress TO( ), 0

Depress RESUME

Enter 12

Depress RESUME

Enter 15.5

Depress RESUME

Enter 8.45

Depress RESUME

The program returns to the first HALT instruction for the entry of additional variables.

When writing a program, it is important to remember the following characteristics of the 1265:

Storage and recall instructions require two program steps; ↓( ) X, ↑( ) X.

The TO( ) X instruction requires only one program step.

X = Numerals 0 through 7

If registers 0 and 1 are being used for accumulation, a programmed RESET will clear these registers. If it is necessary to use one of the registers, clear the E-register with CLR KB and then store 0 into the desired register.

Writing a program using the .00 for specific decimal round-off, with entries that exceed the decimal wheel selection and that are to be printed and used for calculation, entries should be stored before the calculation, then recalled and printed. For example: decimal wheel at 4, .00 UP, and an entry of 5 decimals; if entry is printed first only 4 decimals will remain in the E - register for calculation, which will effect the result; because print out is controlled by the selection of decimals on the decimal wheel.

The following section contains three sample programs to illustrate the various techniques involved in keyboard programming.

Standard Deviation

Ungrouped Data

Program Loading Procedure

Entry Branchpoint 0

Switch Settings $\Sigma_x \div$ UP

Print, .00, DOWN

Decimal Wheel at 4

Depress TO( ), 0

Depress LOAD

Depress each key as shown in the COMMAND column on pages 1 and 2 of the program sheet.

Release LOAD key after last key has been depressed.

Formula $\sigma = \sqrt{\frac{N(\Sigma X^2) - (\Sigma X)^2}{N}}$

Example X_1 2

X_2 1

X_3 5.6

X_4 3

Operating Procedure

Depress TO( ), 0

Depress RESUME

Set 2 depress RESUME

prints with E

Set 1 depress RESUME

prints with E

```
2.0000E
1.0000E
5.6000E
3.0000E
0.0000E
2.9000A
2.9300A
1.7117A
```

Program loops back to start, continue with the balance of the X values. After last entry, depress RESUME. Program will branch to calculate mean, variance, and deviation.

PROGRAM Standard Deviation
Ungrouped Data

NO. ______ **PAGE** 1 **OF** 2

DATE ______ **MODEL** 1265

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE	REMARKS (E A M 0 1 2 3 4 5 6 7 8 9)
0	0	0	0	0		
			1	HALT		enter x
			2	PRT E		
			3	□		go to branchpoint 1 if E = 0
			4	1		
			5	X		
			6	=+		
			7	↑( )		
	0	1	0	2		
			1	+		
			2	1		N count
			3	+		
			4	=*		
			5	↓( )		
			6	2		
			7	TO(0)		
1	0	2	0	↑( )		
			1	1		
			2	÷		
			3	↑( )		
			4	2		
			5	=*		
			6	PRT A		read mean
			7	X		
	0	3	0	=*		
			1	↓( )		
			2	3		
			3	↑( )		
			4	0		
			5	÷		
			6	↑( )		
			7	2		

Programmed by ______

PROGRAM Standard Deviation
Ungrouped Data

NO. ______ PAGE 2 OF 2

DATE ______ MODEL 1265

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE	REMARKS
2	0	4	0	=*		
			1	+		
			2	↑( )		
			3	3		
			4	-		
			5	=*		
			6	PRT A		read variance
			7	□		
	0	5	0	9		
			1	PRT A		read deviation
			2	RESET		
			3	↓( )		
			4	2		
			5	TO(0)		
			6			
			7			
			0			
			1			
			2			
			3			
			4			
			5			
			6			
			7			
			0			
			1			
			2			
			3			
			4			
			5			
			6			
			7			

Programmed by ______

Radians To Degrees Conversion

Program Loading Procedure

Entry Branchpoints   0 for Radians to Degrees

2 for Degrees to Radians

Switch Settings   Print, .00, $\Sigma\div$, DOWN

Decimal Wheel at 7

Depress TO( ), 0

Depress LOAD

Depress each key as shown in the COMMAND column on page 1 of the program sheet

Release LOAD key after last key has been depressed

Depress TO( ), 2

Depress LOAD

Depress each key as shown in the COMMAND column on page 2 of the program sheet

Release LOAD key after last key has been depressed

<u>Example</u>   Convert .488692 Radians to Degrees

Convert 28 Degrees to Radians

Operating Procedure

	Depress TO( ), 0	
	Depress RESUME	0•4886920E
Set .488692	Depress RESUME	27•9999887A
	prints with E	
Read 27.9999887	degrees prints with A	
	Depress TO( ), 2	
	Depress RESUME	28•0000000E
Set 28°	Depress RESUME	0•4886922A
	prints with E	
Read .4886922	radians prints with A	

PROGRAM Radians to Degrees

NO.______ PAGE 1 OF 2

DATE______ MODEL 1265

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS
0	0	0	0	HALT	4	0	1	Enter radians
			1	PRT E	0	2	6	
			2	X	0	7	0	
			3	1	0	0	1	
			4	8	0	1	0	Degrees
			5	0	0	0	0	
			6	÷	0	7	2	
			7	3	0	0	3	
	0	1	0	.	0	1	2	
			1	1	0	0	1	
			2	4	0	0	4	
			3	1	0	0	1	π
			4	5	0	0	5	
			5	9	0	0	9	
			6	2	0	0	2	
			7	7	0	0	7	
1	0	2	0	=*	0	2	0	
			1	PRT A	0	2	7	
			2	TO(0)	7	4	0	
			3					
			4					
			5					
			6					
			7					
	0	3	0					
			1					
			2					
			3					
			4					
			5					
			6					
			7					

Programmed by ______

PROGRAM Degrees to Radians

NO.________ **PAGE** 2 **OF** 2

DATE________ **MODEL** 1265

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS E	A	M	0	1	2	3	4	5	6	7	8	9
2	0	4	0	HALT	4	0	1		Enter degrees											
			1	PRT E	0	2	6													
			2	X	0	7	0													
			3	3	0	0	3													
			4	.	0	1	2													
			5	1	0	0	1													
			6	4	0	0	4													
			7	1	0	0	1		π											
	0	5	0	5	0	0	5													
			1	9	0	1	1													
			2	2	0	0	2													
			3	7	0	0	7													
			4	÷	0	7	2													
			5	1	0	0	1													
			6	8	0	1	0		Degrees											
			7	0	0	0	0													
3	0	6	0	= *	0	2	0													
			1	PRT A	0	2	7													
			2	TO(2)	7	4	2													
			3																	
			4																	
			5																	
			6																	
			7																	
	0	7	0																	
			1																	
			2																	
			3																	
			4																	
			5																	
			6																	
			7																	

Programmed by________________

Present Value of Interest Bearing Loan
(Simple Interest)

Program Loading Procedure

Entry Branchpoint 1

Switch Settings Print, .00, $\Sigma_x^{\div}$ DOWN

Decimal Wheel at 4

Depress TO( ), 0

Depress LOAD

Depress each key as shown in the COMMAND column on page 1 of program sheet.

RELEASE LOAD key after depressing TO( ), 1 on step 007.

Depress TO( ), 1

Depress LOAD

Depress each key as shown in the COMMAND column on page 1 starting with step 020, and page 2 of program sheet.

RELEASE LOAD key after last key has been depressed.

Formulas: 1. $MV = P\,(1 + R_1\ T_1)$

2. $RV = \dfrac{MV}{1 + R_2\ T_2}$

where

MV = Maturity Value

P = Principal of Loan

R_1 = Rate of interest on loan

T_1 = Time on loan

PV = Present Value of loan

R_2 = Rate of interest on maturity value (current rate)

T_2 = Time remaining on loan at revaluation date

Example A loan of $1500 at 6% for 60 days is revaluated 30 days later at 7%.

Depress TO( ), 1

Depress RESUME

```
1500.0000E
   6.0000E
  60.0000E
1515.0000A
1515.0000E
   7.0000E
  30.0000E
1506.2637A
```

Set 1500 depress RESUME

prints with E

Set 6 depress RESUME

Set 60 depress RESUME

6 will print with E

60 will print with E

Read 1515.00 prints with A - Maturity Value

Program will go back to beginning for calculation of Present Value.

Depress RESUME

Set 7 depress RESUME

Set 30 depress RESUME

7 will print with E

30 will print with E

Set 0 depress RESUME

Read 1506.26 will print with A - Present Value

PROGRAM Present Value of Interest Bearing Loan. (Simple Interest)

NO.________ PAGE 1 OF 2

DATE________ MODEL 1265

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS
0	0	0	0	↑( )				
			1	1				
			2	÷				
			3	↑( )				
			4	5				
			5	= *				
			6	PRT A				read revaluation - Present Value
			7	TO(1)				
	0	1	0					
			1					
			2					
			3					
			4					
			5					
			6					
			7					
1	0	2	0	HALT				enter loan or depress RESUME for revaluation
			1	PRT E				
			2	↓( )				
			3	1				
			4	HALT				enter interest rate
			5	↓( )				
			6	2				
			7	HALT				enter days
	0	3	0	↓( )				
			1	3				
			2	TO(2)				branch for interest calculation
			3	HALT				depress RESUME or 0 for revaluation
			4	□				go to branchpoint 0 if E=0
			5	0				
			6	PRT A				read maturity value
			7	TO(1)				

Programmed by ________

PROGRAM Present Value of Interest Bearing Loan. (Simple Interest)

NO. ______ **PAGE** 2 **OF** 2

DATE ______ **MODEL** 1265

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE	REMARKS
2	0	4	0	↑( )		
			1	2		
			2	PRT E		
			3	÷		
			4	1		
			5	0		
			6	0		
			7	=*		
	0	5	0	↓( )		
			1	4		
			2	↑( )		
			3	3		
			4	PRT E		
			5	÷		
			6	3		
			7	6		
3	0	6	0	0		
			1	X		
			2	↑( )		
			3	4		
			4	=*		
			5	+		
			6	1		
			7	+		
	0	7	0	=*		
			1	↓( )		
			2	5		
			3	X		
			4	↑( )		
			5	1		
			6	=*		
			7	RCLP		back to 033

Programmed by ______

INTRODUCTION

BASIC CARD PROGRAMMING

The Monroe Model CR-1 card reader is a separate module which can be plugged into the 1265 for the automatic entry of programs. The CR-1 transmits codes from punched cards into the program memory of the 1265. The card reader instructions include not only the keyboard functions, but also a large repertoire of computer-like machine instructions that greatly increase the programming power of the system.

Among the additional instructions available through card programming are: Conditional branches, Single-step store or recall operations for each register, and the ability to manipulate the contents of the working registers.

Most card reader programs will make use of the basic card instructions described in this section.

Documentation

Unless the program is very brief, it is best to write it first on a coding form in the same way that you would write a keyboard program. Remember, though, that you can intermix both keyboard and machine instructions in your program. The instruction codes for the keyboard instructions are given in Table 1 in the Learn Mode Programmer portion of this manual. Those for the machine instructions are given later in this section.

The cards used in the card reader are made especially for the Monroe programmable calculators. A maximum of 40 instructions may be punched on a card, although it is recommended that 32 instructions be punched per card. The first 32 columns of the card are sectioned in four groups of eight columns each, to correspond with the octal addressing of the program memory. Nine rows of each column are divided into three groups (representing three octal digits) for the octal instruction codes with three holes per octal digit. With the card oriented with the guide numbers (4, 2, 1) upright, the row down the left-hand side under the letter V is used for validating each punched code. A validation hole must be punched for each code to be read. Adjacent to the validation row is another set of prescored holes. A punch in this position causes the cancellation of an instruction, even when the validation hole has been punched.

Beginning at the left-hand end of the card, punch the octal instruction codes according to the table of octal codes and octal digits in the LEMP programming description. A hole represents a 1; absence of a hole represents a zero.

Punch a validation hole for each valid instruction. An octal code of 000 may be entered by punching the validation hole only.

To eliminate an unwanted instruction, punch the hole next to the validation hole.

To eliminate an unwanted hole you may cover the hole with black masking tape.

Loading a Punched Card Program

To load a punched card program into the calculator, use the following procedure:

Depress the RESET key unless important information is stored in Register A, M, 0, 1, or E. The CLR KB key may be used to clear the E-register only.

With the LOAD key up, depress the TO( ) key.

Depress one of data entry keys (0 through 7), depending on the branch point at which the program is to start.

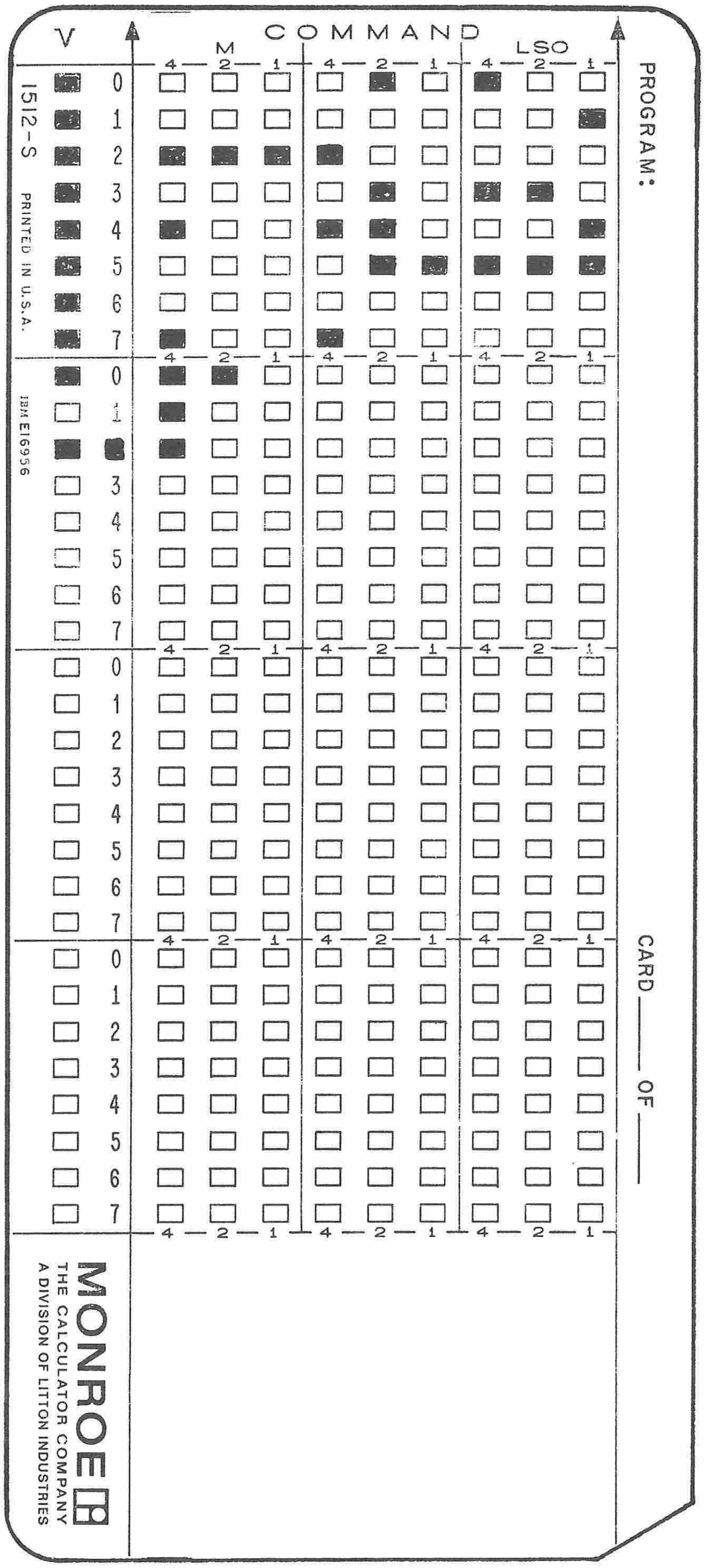
Depress the LOAD key.

Insert the card in the mouth of the card reader with the leading edge of the card forward and the printed side up. Move the card into the slot until the card reader catches the card and slides it through automatically. If the program is contained on more than one card, feed the cards in the order in which the program is to be read into memory.

Release the LOAD key.

Below is a sample Monroe program card. Its layout was designed to conform to the address and command coding systems used in the 1265. Darkened holes represent sample codes punched in the card.

Address	Command
000	024
001	001
002	740
003	026
004	461
005	037
006	000
007	440
010	600
011	
012	
013	
014	
015	
016	
017	



Notes:

1. When codes are cancelled, the addresses no longer conform to the graphics of the card.

2. Command at address 011 not read; no validation punch.

3. Command at address 012 not read; cancellation punch.

Executing a Punched Card Program

The operating procedure for executing a punched card program is identical to that of executing a learn mode program:

Depress the RESET key unless important information is stored in register A, M, 0, 1, or E. The $\frac{CLR}{KB}$ key may be used to clear the E-register only.

With the LOAD key released and the STEP key up, depress the TO( ) key.

Depress one of data entry keys (0 through 7), depending on the branch point at which the program has been loaded.

Depress the RESUME key. The program will be executed until a HALT instruction is encountered. At each HALT enter data as directed in the program instructions and depress the RESUME key. Whenever the program halts, the RESUME key must be depressed to continue program execution.

BASIC MACHINE INSTRUCTIONS

The following list gives the command, name, octal codes, and a brief description of the basic machine instructions that can be used with the card reader. You may notice that some of the machine instructions appear to perform the same functions as certain keyboard instructions; for example, the Store and Recall machine instructions and the ↓( ) and ↑( ) keyboard instructions. Although you can use either set of instructions for storing and recalling, the machine instructions are more efficient, since they require only one program step each instead of two program steps for each keyboard instruction.

Certain functions require a combination of codes rather than a single code. For these functions the combination is listed.

Command	Name	Code	Description
TO ( )	Branch	74X	Branches to branch point X, where X equals 0 through 7. Sets the P-counter to the address following the branch instruction. If the additional program memory is installed,* the TO ( ) codes will be 75X for the branch points in the additional memory.
RCL P	Recall P	557	Recalls the P count to return the program to the main routine at the end of a subroutine. Control is transferred to the program location immediately following the Branch instruction that initiated the subroutine execution.
JUMP	JUMP	6XX	Causes the program to jump to the address defined by the digits XX in the instruction. These digits represent the two least significant octal digits in the address; therefore, a Jump instruction in locations 000 through 077 takes the program to an address within that set of 64 steps. For the same reason a Jump in locations 100 through 177 takes the program to a location between 100 and 177.
HALT	HALT	401	Stops program execution until the RESUME key is pressed to restart program execution.
NOP	No Operation	456	Causes the program to space through its address without performing any operation. Is useful to fill in unused code positions on the program cards without zero entry instructions.

* This applies to the Model 1265W-1 which has double the program memory of Model 1265.

Command	Name	Code	Description
STR (0)	Store 0	457	
STR (1)	Store 1	440	
STR (2)	Store 2	441	
STR (3)	Store 3	443	Copies the contents of the E-register into the specified register. The contents of the E-register remain unchanged.
STR (4)	Store 4	444	
STR (5)	Store 5	445	
STR (6)	Store 6	446	
STR (7)	Store 7	447	
STR (A)	Store A	455	
STR (M)	Store M	454	
RCL (0)	Recall Register 0	477	
RCL (1)	Recall Register 1	460	
RCL (2)	Recall Register 2	461	Recalls the contents of the specified register into the E-register. The contents of the specified register are not changed.
RCL (3)	Recall Register 3	463	
RCL (4)	Recall Register 4	464	
RCL (5)	Recall Register 5	465	
RCL (6)	Recall Register 6	466	
RCL (7)	Recall Register 7	467	
RCL (A)	Recall A-Register	475	
RCL (M)	Recall M-Register	474	
XCEA	Exchange E and A	402	Exchanges the contents of the E-register and the A-register.
XCMA	Exchange M and A	403	Exchanges the contents of the M-register and the A-register.
CLR A	Clear A	424	Clears the A-register to zero.

Command	Name	Code	Description
SHLA	Shift A Left	421	Shifts the numeric portion of the A-register one digit position to the left. New digit appearing on the right will always be zero. The left hand digit will be lost. The exponent is not affected. The value in the register therefore, is usually changed.
SHRA	Shift A Right	420	Shifts the numeric portion of the A-register one digit position to the right. New digit appearing on the left will always be zero. The least significant digit will be lost.
SHRE	Shift E Right	422	Shifts the numeric portion of the E-register one digit position to the right. New digit appearing on the left will always be zero. The least significant digit will be lost.
INXA	Increment ex-* ponent of A	414	Increments the exponent of the A-register by one. Has the effect of having multiplied the value in the A-register by ten.
INXE	Increment ex- ponent of E	415	Increments the exponent of the E-register by one. Has the effect of having multiplied the value in the E-register by ten.
DCXA	Decrement ex- ponent of A	416	Decrements the exponent of the A-register by one. Has the effect of having divided the value in the A-register by ten.
DCXE	Decrement ex- ponent of E	417	Decrements the exponent of the E-register by one. Has the effect of having divided the value in the E-register by ten.
NORM	Normalize	706	Equalizes the exponents in the E- and A-registers. The larger number retains its original exponent.

Note: The Normalize instruction should generally precede any "IF" statement comparing registers E and A.

* The 1265 operates internally in scientific notation although all printed figures are automatically converted to the decimal setting. However, exponents can be controlled utilizing card reader instructions.

SET FLAG "IF" COMMANDS

Command	Name	Code	Description
**SFE >A	Set Flag If E Greater Than A	706 431	Compares the contents of the E-register with the contents of the A-register and sets Flag 1 if the contents of the E-register are greater.
**SFA >E	Set Flag If A Greater Than E	706 430	Compares the contents of the A-register with the contents of the E-register and sets Flag 1 if the contents of the A-register are greater.
**SFA≠E	Set Flag If A Not Equal to E	706 432	Compares the contents of the E-register with the contents of the A-register and sets Flag 1 if the two are not equal.

Compare to Zero

Command	Name	Code	Description
**SFE=0	Set Flag If E Equals Zero	733 437	Sets Flag 1 if the value in the E-register is equal to zero.
**SFE ≤0	Set Flag If E Zero to Negative	733 436	Sets Flag 1 if the value in the E-register is equal to or less than zero.
**SFA≠0	Set Flag If A Not Equal to Zero	705 434	Sets Flag 1 if the value in the A-register is not equal to zero.
**SFA ≤0	Set Flag If A Zero or Negative	705 435	Sets Flag 1 if the value in the A-register is equal to or less than zero.

OTHER CONDITIONAL FLAGS

Command	Name	Code	Description
**SFR_o>A	Set Flag If R_o Greater Than A	433	Compares the numeric portion of register 0 with the numeric portion of the A-register without regard to their exponents and sets Flag 1 if the contents of register 0 are greater.
**SFSNS	Set Flag on SENSE	523	Sets Flag 1 if the SENSE key is depressed.

Skip Commands

Command	Name	Code	Description
SFK1	Skip on Flag 1	540	Causes the program to skip the step immediately following if Flag 1 is set. The flag is reset by this instruction.

** Generally, the SKF 1 instructions should immediately follow an "IF" statement which would set the flag. This is because most of the mathematical functions automatically reset Flag 1.

Command	Name	Code	Description
SKX=0	Skip If Exponent Zero	535	Causes the program to skip the step immediately following if the exponent of the A-register is zero.
SKXP	Skip If Exponent Positive Zero	536	Causes the program to skip the step immediately following if the exponent of the A-register is zero or positive.

Expansion of Power

The expanded power in basic card reader programming results from additional functions not available on the keyboard: a more flexible system of branching, a reduction in the number of steps required to perform certain operations (i.e. storage and recall), conditional branching operations, and manipulation of data in the E- and A-registers.

Saving Program Steps

Using the card reader Store and Recall instructions instead of the equivalent keyboard instructions saves program steps. When a ↑( ) or ↓( ) function is entered, the instruction code takes one program step; the numeral entry representing the affected register takes another. In the machine instructions there are separate single codes for Store 0 through Store 7 and for Recall 0 through Recall 7, each instruction occupying only one program step.

Jump

The Jump instruction in the basic card reader codes allows many more branching possibilities than the TO ( ) branch instruction. With the Jump instruction, the program may branch from any address in a 64 step set (octal 00 through 77) to any other step within that same 64 step set, regardless of branch points. For example: in the layout below, a 651 instruction at the octal address 004, (A) causes a branch to step 051, (B). The same instruction (651) at step 104, (C) causes a branch to step 151, (D). In each case the two digits following the 6 in the Jump code are the same as the last two digits in the destination address.

BRANCH POINT	ADDRESS	BRANCH POINT	ADDRESS	BRANCH POINT	ADDRESS	BRANCH POINT	ADDRESS
0	0 0 0	2	0 4 0	4	1 0 0	6	1 4 0
	1		1		1		1
	2		2		2		2
	3		3		3		3
	A 0 0 4		4		C 1 0 4		4
	5		5		5		5
	6		6		6		6
	0 0 7		0 4 7		1 0 7		1 4 7
	0 1 0		0 5 0		1 1 0		1 5 0
	1		B 0 5 1		1		D 1 5 1
	2		2		2		2
	3		3		3		3
	4		4		4		4
	5		5		5		5
	6		6		6		6
	0 1 7		0 5 7		1 1 7		1 5 7
1	0 2 0	3	0 6 0	5	1 2 0	7	1 6 0
	1		1		1		1
	2		2		2		2
	3		3		3		3
	4		4		4		4
	5		5		5		5
	6		6		6		6
	0 2 7		0 6 7		1 2 7		1 6 7
	0 3 0		0 7 0		1 3 0		1 7 0
	1		1		1		
	2		2		2		
	3		3		3		
	4		4		4		
	5		5		5		
	6		6		6		
	0 3 7		0 7 7		1 3 7		1 7 7

64 STEP SET

64 STEP SET

Keyboard Debugging

You can test your program by executing it with sample values with known results and checking the calculator result to see that it compares with the known results. If the results do not agree, you can verify the program to see whether it is stored correctly, or you can execute it step by step to check each operation.

In order to test and debug a program, you must know the octal numbering system, because it is used in this manual as a form of shorthand to refer to program steps and keyboard codes. A short introduction to the octal system is given here for your review.

Any number written in binary form (ones and zeros) may easily be converted into an octal number by dividing the number into groups of three and using the following table:

Group	Octal Digit
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

As an example of octal translation, the number 100000001 is divided into groups of three and written as follows:

Stored number	100	000	001
Octal number	4	0	1

There is a three-digit octal instruction code for each key on the keyboard. The codes for all of the keys on the keyboard are shown in table 1 before the Exercises at the end of this section.

Key	Code	Stored Number
2	002	000 000 010
+	060	000 110 000

The eight monitor lights of the LEMP show an octal code. A lighted indicator represents a 1 and an unlighted indicator represents a 0. Notice that there are two groups of three lights and one group of two lights. The two groups of three display the two right-most octal digits in the code. The two left-most lights display the left-most octal digit with the 4 position omitted for reasons of internal structure of the calculator. For example, the following codes are displayed as shown at the right. An asterisk represents a lighted indicator and a circle represents an unlighted indicator.

	421	421	421
003	oo	ooo	o**
056	oo	*o*	**o

When the P key and the LOAD key are depressed, the monitor lights display the address of the next program step. When the I key and the LOAD key are depressed, the monitor lights display the instruction code of the

Key	Instruction Code	Program Step
5	005	000
PRT ENT	026	001
x	070	002
6	006	003
PRT ENT	026	004
=	020	005
* PRT ANS	027	006

current program step. For example, assume that the preceeding program has been loaded into memory.

The indicators display the codes shown at the right with the P key or the I key depressed. The octal codes are given to make the comparison clear.

P key depressed	I key depressed
001	005
002	026
003	070
004	006

All of the basic keyboard codes have a 000 in the left-most octal group; for example, 000 111 010 (072) for the ÷ key. The LEMP keys, however, have octal codes with 4, 5, or 7 in the left-hand position. The left-hand 1 is necessarily omitted as shown below.

HALT		(Code 401)	100	000	001	oo ooo oo*
TO ( )	3	(Code 743)	111	100	011	** *oo o**

Each program step has an address identified by three octal digits. The first step is 000 and, in a 128 step program memory, the last step is 177.

Within the program memory there are three units of grouping steps.

Octals - An octal is a grouping of eight steps.

Sets - A set is a grouping of eight octals or 64 steps. Within each set the two right hand digits of the addresses will range from 00 to 77.

Branch Points - A branch point is a starting point for a program or subroutine. These starting points are spaced at 16 step intervals.

On this page is a program memory diagram showing the octal addresses for 128 steps.

PROGRAM MEMORY DIAGRAM

(Octal Addresses)

Branch Point	Address	Branch Point	Address
0	0 0 0	2	0 4 0
	1		1
	2		2
	3		3
	4		4
	5		5
	6		6
	0 0 7		0 4 7
	0 1 0		0 5 0
	1		1
	2		2
	3		3
	4		4
	5		5
	6		6
	0 1 7		0 5 7
1	0 2 0	3	0 6 0
	1		1
	2		2
	3		3
	4		4
	5		5
	6		6
	0 2 7		0 6 7
	0 3 0		0 7 0
	1		1
	2		2
	3		3
	4		4
	5		5
	6		6
	0 3 7		0 7 7

64 STEP SET

Branch Point	Address	Branch Point	Address
4	1 0 0	6	1 4 0
	1		1
	2		2
	3		3
	4		4
	5		5
	6		6
	1 0 7		1 4 7
	1 1 0		1 5 0
	1		1
	2		2
	3		3
	4		4
	5		5
	6		6
	1 1 7		1 5 7
5	1 2 0	7	1 6 0
	1		1
	2		2
	3		3
	4		4
	5		5
	6		6
	1 2 7		1 6 7
	1 3 0		1 7 0
	1		1
	2		2
	3		3
	4		4
	5		5
	6		6
	1 3 7		1 7 7

64 STEP SET

The branch points in octal notation are as follows:

Branch Point	Decimal Step	Octal Step
0	000	000
1	016	020
2	032	040
3	048	060
4	064	100
5	080	120
6	096	140
7	112	160

Verifying A Program

To verify a stored program, the LOAD and I keys are depressed and the RESUME key is used to advance the calculator through the program steps so that the operator may observe the stored instruction codes. The procedure is as follows:

Depress the RESET key unless important information is stored in registers A, M, O, or 1. If the contents of any of these registers must be saved, the CLR KB key may be used to clear the E-register only.

Depress the I key.

With the LOAD key up, depress TO( ), and a branch point number.

Depress the LOAD key. The monitor lights will display 74(n) where n represents the starting branch point of the program.

Figure 1

PROGRAM Sample Coding Sheet

$$\frac{(A \times B) - C}{D} = 31.893 \qquad A=12,\; C=15.5,\; D=8.45$$

NO.______ PAGE 1 OF 1

DATE______ MODEL 1265

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS
0	0	0	0	HALT	4	0	1	Enter A
			1	PRT E	0	2	6	
			2	X	0	7	0	
			3	2	0	0	2	B(constant)
			4	3	0	0	3	
			5	.	0	1	2	
			6	7	0	0	7	
			7	5	0	0	5	
	0	1	0	=*	0	2	0	
			1	PRT A	0	2	7	
			2	+	0	6	0	
			3	HALT	4	0	1	Enter C
			4	PRT E	0	2	6	
			5	-	0	6	2	
			6	=*	0	2	0	
			7	÷	0	7	2	
1	0	2	0	HALT	4	0	1	Enter D
			1	PRT E	0	2	6	
			2	=*	0	2	0	
			3	PRT A	0	2	7	
			4	TO(0)	7	4	0	
			5					
			6					
			7					
			0					
			1					
			2					
			3					
			4					
			5					
			6					
			7					

REMARKS columns: E A M 0 1 2 3 4 5 6 7 8 9

Programmed by ______

Depress the RESUME key. The monitor lights will display the instruction code in the branch point location of the program.

Continue depressing the RESUME key. The instruction codes will be displayed successively in the monitor lights.

As an example of program verification, load the program on the sample coding sheet in figure 1. After releasing the LOAD key, the procedure is as follows:

Depress RESET

Depress I

Depress TO ( )

Depress 0

This sets the program to branch point 0, (Step 000)

	Indicators	Octal Code
Depress LOAD	* * * o o o o o	740**
Depress RESUME		
Verify code in Step 000	o o o o o o o *	401
Depress RESUME		
Verify code in Step 001	o o o * o * * o	026
Depress RESUME		
Verify code in Step 002	o o * * * o o o	070
Depress RESUME		
Verify code in Step 003	o o o o o o * o	002
Depress RESUME		
Verify code in Step 004	o o o o o o * *	003

** The 9th light must be assumed for codes whose left hand digit is a 4, 5, 6, or 7.

Depress RESUME

Verify code in Step 005   o o o o * o * o   012

Depress RESUME

Verify code in Step 006   o o o o o * * *   007

Depress RESUME

Verify code in Step 007   o o o o o * o *   005

Depress RESUME

Verify code in Step 010   o o o * o o o o   020

Depress RESUME

Verify code in Step 011   o o o * o * * *   027

Depress RESUME

Verify code in Step 012   o o * * o o o o   060

Depress RESUME

Verify code in Step 013   o o o o o o o *   401

Depress RESUME

Verify code in Step 014   o o o * o * * o   026

Depress RESUME

Verify code in Step 015   o o * * o o * o   062

Depress RESUME

Verify code in Step 016   o o o * o o o o   020

Depress RESUME

Verify code in Step 017   o o * * * o * o   072

Depress RESUME

Verify code in Step 020   o o o o o o o *   401

Depress RESUME

Verify code in Step 021	o o o * o * * o	026
Depress RESUME		
Verify code in Step 022	o o o * o o o o	020
Depress RESUME		
Verify code in Step 023	o o o * o * * o	027
Depress RESUME		
Verify code in Step 024	* * * o o o o o	740

If at any time you want to check the step number, depress the P key. The indicators will display the address of the next instruction to be executed. A program command and its address may not be displayed after the same depression of the RESUME key because the address indication is always one step ahead of the command indication.

Testing A Program

To test the operation of a program, depress the STEP key and use the RESUME key to execute the program one step at a time while you check the tape for results. The following procedure should be used to test a program step by step:

Be sure that the LOAD key is off.

Depress the P key.

Depress the RESET key.

Depress the TO ( ) key.

Depress one of the data entry keys 0 through 7, depending on the starting location of the program. The LEMP indicators will display the step number corresponding to selected branch point.

Depress the STEP key.

Depress the RESUME key. The instruction at the branch point will be executed. The LEMP indicators will display the number of the next step to be executed.

Continue depressing the RESUME key. The instructions in successive steps will be executed one at a time. For example, see figure 1 when step 001 is displayed in octal lights, enter A, depress RESUME key. Enter sample values at HALT steps. The tape will show the contents of the E-register at the end of each instruction. With the P key depressed, the LEMP indicators will display the step number of the next instruction to be executed. If the I key and the LOAD key are depressed, the LEMP indicators will display the code of the last instruction executed. Be sure to release the LOAD and I keys and depress the P key before continuing with the step procedure.

As an example of testing a program, the program in figure 1 should be checked step by step as follows, using the sample values: A = 12, C = 15.5, and D = 8.45

Release LOAD

Depress P

Depress RESET

Depress TO( )

Depress 0

The program is set at branch point (0) or Step 000.

Depress STEP

In stepping through a program, enter each variable at the HALT step.

If a program step is found to contain the wrong instruction, you can change the instruction by using the following procedure:

Make sure that the LOAD key is up.

Depress the P key.

Depress the TO ( ) key.

Depress one of data entry keys 0 through 7, depending on the branch point for the program block that contains the step to be changed. For example, to change Step 045, depress the 2 key, because 2 is the branch point for the program block beginning with Step 040.

Depress the LOAD key.

Depress the RESUME key repeatedly until the address of the step to be changed appears in the monitor lights.

Depress the key corresponding to the correct instruction to be stored as a program step. The new instruction is now stored. Be sure you release the LOAD key before entering a TO ( ) instruction to execute the program.

As an example of changing a program step, in the program in figure 1, assume that the ÷ key had been accidently depressed instead of the first X key during the loading process, thereby storing the wrong instruction code in Step 002. Correct the error as follows:

Release LOAD key.

Depress P

Depress RESET

Depress TO ( )	Monitor lights
Depress 0	o o o o o o o o

Observe Step 000 in monitor lights

Depress LOAD

Depress RESUME	
Observe Step 001 in neon indicators	o o o o o o o *
Depress RESUME	
Observe Step 002 in neon indicators	o o o o o o * o
Depress X	
The lights will show after X key is depressed	o o o o o o * *
Release LOAD key	
Depress TO ( )	
Depress 0	o o o o o o o o
Depress RESUME	
Program starts from beginning	

Programming Techniques

In order to write effective keyboard programs, you should have a grasp of three special programming techniques: writing subroutines, using constants, and looping.

Subroutines

If a function or sequence of steps is common to more than one program, the sequence may be stored at one of the branch points of the program memory as a subroutine. A RCL P instruction must be entered as the end of the subroutine so that program execution will return to the point from which the branch was made to the subroutine.

Each time the function of the subroutine is required, a TO ( ) instruction is inserted in the program with a numeral specifying the branch point where the subroutine is located. During program execution, when the program encounters the TO (N) instructions, the program jumps to that branch point and executes the subroutine. When the RCL P instruction is encountered at the end of the subroutine, the program jumps back to the step following the TO(N) instruction in the main program.

An example of a main program and a subroutine is shown in figure 2. In the program each time a TO(1) instruction is encountered, the program branches to the subroutine beginning at Step 020. At the end of the subroutine, the RCL P instruction causes a jump back to the step after that TO(1) instruction in the main program. In other words, the main program takes up where it left off.

In this example, the subroutine, which calculates a constant discount rate is called on twice in the main program. This eliminates the necessity of repeating this sequence of steps in the main program.

As sample values use:

Quantity = 10 items

Price = $5.00

Constant chain discount rate is less 5% - 5%.

```
10.00E
 5.00E
50.00A
45.12A
```

Set decimal on "2".

PROGRAM Figure 2 Main Program with Subroutine Sample **NO.**________ **PAGE** 1 **OF** 1

Invoice Extension with constant chain discount rates **DATE**________ **MODEL** 1265

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS
								E A M 0 1 2 3 4 5 6 7 8 9
0	0	0	0	HALT	4	0	1	enter quantity
			1	PRT E	0	2	6	
			2	X	0	7	0	
			3	HALT	4	0	1	enter price
			4	PRT E	0	2	6	
			5	=*	0	2	0	
			6	PRT A	0	2	7	
			7	+	0	6	0	
	0	1	0	TO(1)	7	4	1	to discount routine
			1	TO(1)	7	4	1	
			2	= +	0	2	2	accumulate net extension
			3	PRT A	0	2	7	
			4	TO(0)	7	4	0	back to beginning
			5					
			6					
			7					
1	0	2	0	5	0	0	5	constant discount rate
			1	◻	0	3	4	
			2	7	0	0	7	
			3	RCLP	5	5	7	back to step 011
			4					
			5					
			6					
			7					
			0					
			1					
			2					
			3					
			4					
			5					
			6					
			7					

Programmed by________

Constants

Constants, are stored by the programmer, either manually in a storage register or by programming the numeric entry.

A constant may be stored in registers 0 through 7 by entering it through the keyboard and then storing it with ↓( ) followed by the numeral of the desired storage register. Whenever the constant is required by the program, a ↑( ) instruction followed by the appropriate numeral returns the constant to the E-register for the desired operation.

A constant may be stored in the program memory by entering it as a subroutine beginning at one of the branch points. The constant must be followed by an RCL P instruction to return to the program that called for the constant. For example, the constant 409.83201 could be stored as follows:

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS												
								E	A	M	0	1	2	3	4	5	6	7	8	9
7	1	6	0	4	0	0	4													
			1	0	0	0	0													
			2	9	0	1	1													
			3	.	0	1	2													
			4	8	0	1	0													
			5	3	0	0	3													
			6	2	0	0	2													
			7	0	0	0	0													
			0	1	0	0	1													
			1	RCL P	5	5	7													

A TO (7) instruction would be entered in the program wherever this constant is required. The RCL P instruction takes the program back to the step following the TO (7) instruction.

Looping

The looping, or unconditional branching, capability of the LEMP makes possible the repeated execution of the same program as many times as desired. A TO ( ) instruction at the end of the program followed by the number of the branch point where the program began will always restart the program as soon as the last instruction is executed. With HALT instructions inserted in the program, different variables may be entered each time the program is executed.

The following routine will sum X^2 for as many values of X as are entered. After the last X value, depress *, PRNT ANS to print $\sum x^2$.

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS (E A M 0 1 2 3 4 5 6 7 8 9)
4	1	0	0	HALT	4	0	1	
	1	0	1	PRNT ENT	0	2	6	Enter and print X_i
	1	0	2	X	0	7	0	
	1	0	3	=+	0	2	2	Squares X and accumulates X^2 in RO
	1	0	4	TO (4)	7	4	4	Causes loop back to branch point 4
			5					

Summary of Keyboard and LEMP Instructions

Table 1 summarizes all keyboard and LEMP instructions. It lists each key, along with its command, instruction code, and functional description.

Table 1. Keyboard and LEMP Instructions

Key	Command	Code	Function
0	0	000	Enters the respective digit into the E-register. If a numeric key is the first entry after the calculator has completed an operation, the previous is cleared and the algebraic sign is automatically set to positive as the digit is entered. Subsequent numeric key entries do not alter the sign of the E-register. Up to 14 digits may be entered. Each result is calculated to 14-digit accuracy and rounded to the setting on the decimal wheel. If too many digits are entered, an ERROR condition occurs. The ERROR condition disables the keyboard until CLR KB or RESET is depressed.
1	1	001	
2	2	002	
3	3	003	
4	4	004	
5	5	005	
6	6	006	
7	7	007	
8	8	010	
9	9	011	
.	.	012	Terminates the integer part of a numeric entry and defines the start of the fractional part. If the decimal point is the first entry after the calculator has completed an operation, the value "0." is entered into the E-register.

Key	Command	Code	Function
CLR KB	CLR KB	033	Clears the contents of the E-register to 0 and also resets the overflow and error conditions. The CLR KB instruction does not terminate a previous arithmetic or algebraic sequence.
$\overset{=}{*}$	$\overset{=}{*}$	020	Executes the last entered arithmetic or algebraic function and terminates the algebraic sequence of operations. The $\overset{=}{*}$ key need only be used to <u>terminate</u> a particular computational sequence. The result is printed and stored in the E-register after decimal point alignment and may be used as the first entry of a subsequent arithmetic or algebraic sequence. Multiple $\overset{=}{*}$ key depressions without intervening data entries or instructions do not perform any arithmetic operations.
◇	◇	021	Causes the total of the addition register to be printed. Does not alter the contents of the adding register.
=+	=+	022	Causes the result of a multiplication or division to be added to the contents of register 0.
=-	=-	023	Causes the result of a multiplication or division to be subtracted from the contents of register 0.

Key	Command	Code	Function
□		034	Percent of Change; calculates and prints the
6	6	006	amount and percent of change between any two numbers.
□		034	Percent Minus; calculates the amount to be
7	7	007	subtracted and prints the percent, and amount in red; prints the new total in black. The new total remains in the A-register and may be used for additional calculations. The $\overset{=}{*}$ command should be given if the new total is not to be used again.
□		034	Percent Plus; calculates the amount to be add-
8	8	010	ed and prints the percent, amount, and new total. The new total remains in the A-register and may be used for additional calculations. The $\overset{=}{*}$ command should be given if the new total is not to be used again.
□		034	Square Root; calculates the square root of the
9	9	011	number in the E-register.

Key	Command	Code	Function
*	*	036	Causes the contents of the accumulating register (0) to be recalled. The contents are cleared from the accumulator but are available for further calculations.
RESET	RST	024	Resets the contents of registers E, A, M, 0, and 1 to zero. The RESET key also resets the overflow and error conditions. If an algebraic or arithmetic sequence is in progress when the RESET key is depressed, that sequence is terminated.
↓( )	↓( )	025	Depressing the ↓( ) key followed by a single digit (0 through 7) transfers the contents the E-register to the storage register identified by the numeric entry. The contents of the E-register remain unchanged.

Key	Command	Code	Function
PRNT ENT	PRT E	026	Depressing the PRT E key causes the contents of the E-register to be printed on the tape.
PRNT ANS	PRT A	027	Depressing the PRINT ANS key causes the contents of the A-register to be printed on the tape.
↑( )	↑( )	031	Depressing the ↑( ) key followed by a single digit (0 through 7) transfers the contents of the storage register identified to the E-register. The contents of the storage register remain unchanged.
◇	M ◇	032	Provides for sub-total of the accumulating register. The contents of that register remains unchanged.
+	+	060	Adds the contents of the E-register to the A-register. Does not terminate an arithmetic condition set up by a previous operation. The previous operation will be cleared.
−	-	062	Subtracts the contents of the E-register from the A-register. Does not terminate an arithmetic condition set up by a previous operation. The previous operation will be cleared.
×	x	070	Sets up a multiplication of the contents of the E-register times the next data entry. Terminates any algebraic operation previously set up using the result as the operand.

Key	Command	Code	Function
÷	÷	072	Sets up a division of the contents of the E-register by the next data entry. Terminates any algebraic operation previously set up using the result as the operand.
HALT	HALT	401	Stops program execution. The RESUME key is depressed to restart program execution.
RCL P	RCL P	557	Causes reentry to main program from the end of a subroutine.
TO ( )	TO ( )	74(n)	Causes a branch to program step n, where n is one of the branch points, 0 through 7.

Conditional Branching

Conditional branching is the method used to program the calculator to decide which of two routines to perform. This decision is usually based on whether or not a certain condition is met. The 1265 card reader codes provide ample means for setting up the desired test, testing for the condition and then jumping to the appropriate sequence of program steps.

To effect a conditional branch, codes are available for maneuvering values into the E and A registers to set up a test; setting a flag if the condition is met; and skipping a command which is usually a jump to a different routine.

The Skip instructions; Skip if Flag 1, Skip if Exponent Zero and Skip if Exponent Positive are usually followed by one or two Branch (6xx) instructions. The Skip instruction causes the next command to be ignored if the condition has been met. If the condition has not been met, the next command is executed.

The following sequence illustrates the branch sequence described above. The example is taken from a program to calculate the irregular final payment of a debt, after a series of regular and equal monthly payments.

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS
4	1	0	0	SFE$\leq$0	4	3	6	Is P$\leq$M?
			1	SKF 1	5	4	0	Yes, skip
			2	JUMP	6	1	0	No, check if n = N
			3	RCL(1)	4	6	0	
			4	STR(2)	4	4	1	Set M = P
			5	0	0	0	0	
			6	STR(1)	4	4	0	Set B = 0
			7	TO(7)	7	4	7	Print n, m, B_n
	1	1	0	TO(0)	7	4	0	Begin another problem

In step 100, Flag 1 is set if the difference between the Principal and the Monthly Payment is less than or equal to zero. In this case, the Jump instruction is ignored and the Monthly Payment and the Principal are equalized and the balance is set equal to zero. If Flag 1 is not set, implying that the Principal is greater than the Monthly Payment, the program jumps to Step 110 at which point it will return to the beginning to start a new problem.

The second type of conditional branch employs two successive Jump or Branch instructions immediately after the Skip instruction. The step to which the program branches is determined by the skip condition. If the skip condition is not met, the program performs the first jump step. If the skip condition is met, the second Jump or Branch instruction is executed. This type of conditional branch is illustrated on the next page.

			1	RCL 4	4	6	4	
			2	STR A	4	5	5	
			3	RCL 3	4	6	3	Is n - N?
			4	C Sub	7	6	7	
			5	RCL A	4	7	5	
			6	SFE=0	4	3	7	
			7	SKF 1	5	4	0	Yes, skip
5	1	2	0	JUMP	6	2	2	No, set P = P - M
			1	JUMP	6	0	3	Print n, M, P
			2					

In order to determine if the total number of payments to be made is equal to the payment number, the calculation n-N is performed, Flag 1 is set if n-N=0. If Flag 1 is set, the Skip step causes the first Jump instruction to be ignored and the Jump from step 121 is executed. If Flag 1 is not set, the first Jump is executed and the program continues with Step 122.

The example below illustrates a conditional branch operation in which two numbers are normalized in order to make a comparison.

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS
								E A M 0 1 2 3 4 5 6 7 8 9
1	0	2	0	RCL 2	4	6	1	Recall "x"
			1	STR A	4	5	5	Store "x" in A-register
			2	RCL (1)	4	6	0	Recall "a"
			3	Norm	7	0	6	
			4	SFE>A	4	3	1	Set Flag if a > x
			5	SKF 1	5	4	0	Skip if condition was met
			6	JUMP	6	5	7	No, jump to x > a calculation
			7					Yes, continue with a > x calculation

The variable x is recalled and placed in the A-register. The variable a is recalled and left in the E-register. After the exponents have been equalized by the NORM instruction, the numeric portions of the E- and A-register contents are compared and Flag 1 set if the E-register contents are greater. The Skip instruction, SKF1, tests the flag to determine whether the program will jump to the $\underline{x} \geq \underline{a}$ calculation or continue with the $\underline{a} \geq \underline{x}$ calculation.

The sequence below illustrates the use of the SENSE key to control branching. The example is taken from a statistical program to compute a standard deviation.

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS
								E A M 0 1 2 3 4 5 6 7 8 9
0	0	0	0	RST	0	2	4	Clears storage registers 0 and 1.
			1	HALT	4	0	1	Enter x value
			2	SFSNS	5	2	3	Set a flag if SENSE key is depressed
			3	SKF1	5	4	0	Skip if condition was met
			4	JUMP	6	3	3	NO - Jump to routine for summing x
			5	RCL(7)	4	6	7	YES - Calculate Mean, Variance, & Std. Dev.
			6	÷	0	7	2	
			7	RCL(2)	4	6	1	

Until the SENSE key is depressed, the program loops back continually to Step 001, after summing each x and its square. At Step 001 the next x value is entered. Flag 1 remains reset, and the Jump to Step 33 (Summing Routine) is executed every time data is entered. When all of the data has been accumulated, the operator depresses the SENSE key. As the program encounters the SFSNS instruction, Flag 1 is set. The SKF1 instruction causes the Jump instruction to be ignored, and the program starts operating on the summations to calculate Mean, Variance, and Standard Deviation beginning with Step 005.

Data Manipulation

Data in the A- and E-registers can be manipulated with the shift instructions. The A-register contents can be shifted one digit position to the left or right by the Shift A Left and Shift A Right instructions respectively. The E-register contents can be shifted one digit position to the right by the Shift E Right instruction. These "shift" operations do not alter the exponent of the affected register. On the other hand, the ICXE, ICXA, DCXE, and DCXA instructions enable the programmer to manipulate the exponents without losing significant digits. The value of this data manipulation capability is not as obvious as that of the other functions. In writing a program, however, these codes are often useful in stretching the programming power of the unit.

Debugging

The debugging procedure for punched card programs is the same as the keyboard program debugging procedure. The TO ( ) key is used to start the program at the desired branch point, and then the program can be verified using the LOAD key and the RESUME key or tested using the STEP key and the RESUME key.

Programming Techniques

Writing a program with card reader instructions involves the same techniques as those used in writing a program with keyboard instructions, except that the repertoire of operations is much larger and more efficient. It is good practice, as you write the longhand program, to record at each stage the contents of all registers used in the program.

Subroutines for card reader programs are written in the same manner as those for keyboard programs. The subroutines may be stored at any available branch point in the program memory. As in keyboard programming, a TO ( ) instruction must be stored at the point where the subroutine is needed, and the subroutine must end with a RCLP instruction.

Card reader programs requiring more than 32 steps can be punched on more than one card. When loading a program, the cards are run through the reader in succession without intervening keyboard operation. Be sure that you know the layout of the program so that your card reader program will not overlay another active program or go beyond the last step.

The SENSE key can be used whenever you want to control manually the path the program takes at any point. For example, if the program is written to determine one of two parameters, you can choose the parameter to be calculated by inserting a Set Flag on Sense Switch instruction at the point where the decision is to be made, followed by a Skip on Flag 1 instruction and the appropriate Jump or Branch instructions. A HALT instruction must precede the Set Flag instruction because program execution must be stopped to allow for the depression of the SENSE key. When the program is restarted with the RESUME key, the program will take either the normal path or the path called for by the Skip instruction.

The addition of the Model CR-1 card reader expands the capability of the 1265 immeasurably. It makes simple programs simpler, long programs shorter, and many otherwise impossible programs possible.

ADVANCED PROGRAMMING

The program examples in this section are given to show you how to use the advanced card instructions.

The following program sequence is an example of data manipulation using index 1 and index 2. The program is a subroutine that fills the E-register with nines.

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS
5	1	2	0	9	0	1	1	9 ⟶ Index 1
			1	XFIE	5	1	7	⟶ E at Index 2 location
			2	INI2	5	7	7	Add 1 to Index 2
			3	SKLD	5	3	4	Last Digit? (D14)
			4	JUMP	6	2	6	Yes, jump to Print
			5	JUMP	6	2	1	No, Load next Digit
			6	PRT E	0	2	6	Print all nines
			7	RCL P	5	5	7	Return to Main Program

The first step loads a 9 into index 1 and into digit position 1 of the E-register. A 2 is automatically loaded into index 2. The XFIE instruction transfers a 9 into position 2 of the E-register, as specified by index 2. The INI2 instruction adds one to the count in index 2. Since the last digit has not been reached, the second jump is executed to return to step 121 and load a 9 into digit position 3 of the E-register, as specified by index 2. This loop continues until the calculator recognizes the last digit, D14, and then the Jump to step 126 is executed to print the nines. The RCLP instruction at the end of the subroutine places in the P-counter the last address that was stored, thereby branching the program to the instruction following the one that called for the subroutine.

A similar subroutine can be used to enter an identification number into D1 of the E-register and fill the rest of the register with blanks. The identification number is printed at the program halt in the subroutine to tell the operator what type of data to enter at that point. An example of this type of subroutine is shown in figure 7.

The entry 1 in step 140 places a 1 in index 1 and in position D1 of the E-register. The first time through the subroutine, the identifier is 1 because register 7 is empty. During subsequent executions, the identifier is increased by 1 each time with the + RCL(7) operation. Index 2 is reset to 1 in step 146 and increased to 2 in the next step to point to position D2 in the E-register. Step 150 resets index 1 to 0, and step 151 decrements index 1, causing it to revert to a count of 15. The following instructions load 15's from index 1 into the E-register in the same manner as the 9's were loaded in the previous example. At the print command displays the identifier in digit 1 and a blank for every digit position containing a 15 will print.

The use of index 1 to select a storage register is also shown in figure 7. Register 7 contains the identifier in digit 1 and 15's in the remaining digit positions. The A-register contains data just entered from the keyboard. This sequence stores the data in the register selected by digit 1 of register 7.

The RCL(7) instruction recalls the identifier that has previously been stored. The STR(M) instruction transfers the identifier to the M- register. The data that has been manually entered is transferred to the E-register by the XCEA instruction. Index 2 is set to 1 by the RSI2 instruction. The DLIX instruction transfers the M-register digit selected by index 2 (digit 1) into index 1. The STRI instruction transfers the data in the E-register to one of storage registers 0 through 9 as specified by the number in index 1.

PROGRAM Figure 7 **NO.**________ **PAGE**____ **OF**____

DATE________ **MODEL**________

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS (E A M 0 1 2 3 4 5 6 7 8 9)
6	1	4	0	1	0	0	1	
			1	+	0	6	0	
			2	RCL 7	4	6	7	
			3	+	0	6	0	
			4	=	0	2	0	
			5	STR 7	4	4	7	After identification number is
			6	RSI 2	5	7	5	in the Entry Register, the
			7	INI2	5	7	7	remainder of the Register
	1	5	0	RSI1	5	1	2	is blanked to distinguish
			1	DCI1	5	1	4	the number as on
			2	XFIE	5	1	7	identifier
			3	INI2	5	7	7	
			4	SKLD	5	3	4	
			5	JUMP	6	5	7	
			6	JUMP	6	5	2	
			7	PRT E	0	2	6	Print identifier
7	1	6	0	STR A	4	5	5	
			1	HALT	4	0	1	Enter data
			2	PRT E	0	2	6	Print data
			3	SFA≠E	4	3	2	
			4	SKF1	5	4	0	
			5	CLRKB	0	3	3	
			6	STR A	4	5	5	
			7	RCL 7	4	6	7	Stores data in the same
	1	7	0	STR M	4	5	4	register as the identifier.
			1	XCEA	4	0	2	Stores zero in the Register
			2	RSI2	5	7	5	Corresponding to the
			3	LDIX	5	1	5	unknown.
			4	STRI	7	2	0	
			5	CLR A	4	2	4	
			6	RCL P	5	5	7	Subroutine compute
			7					

Programmed by________________

ADVANCED CARD INSTRUCTIONS

The following list gives the advanced card instructions, their codes, and descriptions.

Command	Name	Code	Description
DCI1	Decrement Index 1	514	Decreases the count of index 1 by one. If index 1 equals zero when the instruction is executed, the index count reverts to 15
DCXA	Decrement Exponent A	416	Subtracts 1 from the exponent contents of the A-register. Sets the overflow flag if the exponent exceeds -99. If there is no exponent in the A-register, this instruction moves the decimal point one digit position to the left
DCXE	Decrement Exponent E	417	Subtracts 1 from the exponent contents of the E-register. Sets the overflow flag if the exponent exceeds -99. If there is no exponent in the E-register, this instruction moves the decimal point one digit position to the left
INI1	Increment Index 1	513	Increases the count in index 1 by one. If index 1 equals 15 when the instruction is executed, the index count reverts to zero
INI2	Increment Index 2	577	Increases the count in index 2 by one. A count of binary 14, indicating digit position D14, goes to binary 15, indicating the exponent tens position. Index 2 counts to 31 before reverting to zero. Counts of 18 to 31 are meaningless
INXA	Increment Exponent A	414	Adds 1 to the exponent of the A-register. Sets the overflow flag if the exponent exceeds 99. If there is no exponent in the A-register, this instruction moves the decimal point one digit position to the right

Command	Name	Code	Description
INXE	Increment Exponent E	415	Adds 1 to the exponent of the E-register. Sets the overflow flag if the exponent exceeds 99. If there is no exponent in the E-register, this instruction moves the decimal point one digit position to the right
JUSA	Justify A	705	Adjusts the number in the A-register so that the most significant digit appears in the D1 position. If a right shift is required, the exponent is increased; if a left shift is required, the exponent is decreased; thus, the numerical value of the data remains the same.
JUSE	Justify E	733	Adjusts the number in the E-register so that the most significant digit appears in the D1 position. If a right shift is required, the exponent is increased; if a left shift is required, the exponent is decreased; thus, the numerical value of the data remains the same.
LDDP	Load Decimal Point	574	Loads a zero into index 1 if the data format switch on the keyboard is in position E, or loads 15 into index 1 if the switch is in the . position
LDIX	Load Index	515	The digit in the position of the M-register defined by index 2 is loaded into index 1. The contents of the M-register remain unaltered
NOP	No Operation	456	No operation takes place when this instruction is executed
PRI2	Preset Index 2	576	Presets index 2 to 15, representing the exponent tens position, D15
RCLI	Recall Per Index 1	722	Transfers the contents of one of registers 0 through 7 to the E-register. The selected register is specified by the contents of index 1.

Command	Name	Code	Description
RSI1	Reset Index 1	512	Resets index 1 to zero
RSI2	Reset Index 2	575	Resets index 2 to one, representing the first digit position, D1
SKD0	Skip on Digit Zero	537	Causes the program to ignore the step immediately following if the guard digit, position D0, of the A-register is anything other than 0 or binary 15.
SKI0	Skip if Index Zero	533	Causes the program to ignore the step immediately following if index 1 contains a zero
SKLD	Skip Except on Last Digit	534	Causes the program to ignore the step immediately following if index 2 contains any count other than 14, which represents the least significant digit position, D14
STRI	Store Per Index 1	720	Copies the contents of the E-register into one of registers 0 through 7. The selected register is specified by the contents of index 1. Uses five storage levels in the program storage register
XFIE	Transfer Index to E	517	Transfers the count contained in index 1 into the E-register at a digit position defined by the count in index 2. If one of digits 0 through 9 is entered in the program immediately before the XFIE instruction, this digit becomes the index count that is loaded into the E-register. If index 1 contains a count other than 0 through 9 at the time this instruction is given, subsequent arithmetic operations using the data will be useless
XFIM	Transfer Index to M	516	Transfers the count contained in index 1 into the M-register at a digit position defined by the count in index 2. If index 1 contains a count other than 0 through 9 at the time this instruction is given, subsequent arithmetic operations using the data will be useless

C Add, C Sub, C Mlt, C Div

The card reader arithmetic instructions are extremely important in advanced programming of the 1265. Their main purpose is to conserve 14 digit accuracy throughout the program. Keyboard arithmetic instructions, i. e. +; -; x; ÷, all go through a print cycle which aligns the digits for printing according to the decimal setting, and therefore can truncate significant digits. The card reader arithmetic instructions bypass this print cycle and are also unaffected by the double-zero round-off.

When it is necessary for a result to be rounded off, the last instruction before printing the answer should be a keyboard arithmetic instruction.

The limitations for using these instructions are as follows:

> Be sure that all information is stored in the correct register before entering instruction.
>
> Do not use an equals instruction.
>
> If the result is in the A-register, as after the C Add and C Sub commands; it may be necessary to enter a RCL (A) (475) instruction before continuing with the program.

The program in figure 8 is included to illustrate the technique of using card reader commands.

The program is a summation calculation. The input quantities are any number of positive variables, x and y. The output quantities are $\sum x$, $\sum x^2$, $\sum xy$, $\sum y$, $\sum y^2$, and the N-count.

Load the program at branch point 0 and execute as follows:

Depress TO( )

Depress 0

Depress RESUME

Enter x_i

Depress RESUME Read x_i

Enter y_i

Depress RESUME Read y_i

(Continue entering x and y values, depressing RESUME after each entry)

Depress SENSE switch

Line Prints

Read $\sum x$

Read $\sum x^2$

Read $\sum xy$

Read $\sum y$

Read $\sum y^2$

Read n

Double line prints

PROGRAM Figure 8 **NO.**________ **PAGE** 1 **OF** 3

DATE________ **MODEL**________

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS (E A M 0 1 2 3 4 5 6 7 8 9)
0	0	0	0	RST	0	2	4	
			1	STR 2	4	4	1	
			2	STR 3	4	4	3	Clear registers for accumulations
			3	STR 4	4	4	4	
			4	STR 5	4	4	5	
			5	HALT	4	0	1	Enter x_i
			6	SFSNS	5	2	3	
			7	SKF1	5	4	0	Has last entry been made?
	0	1	0	JUMP	6	1	2	No, jump to continue summations.
			1	JUMP	6	6	2	Yes, jump to recall results.
			2	STR A	4	5	5	
			3	STR M	4	5	4	
			4	PRT E	0	2	6	Print x_i
			5	RCL 0	4	7	7	
			6	C Add	7	3	0	Σx-.
			7	RCL A	4	7	5	
1	0	2	0	STR 0	4	5	7	
			1	RCL M	4	7	4	
			2	C MLT	7	1	4	
			3	RCL 1	4	6	0	Σx^2____.
			4	C Add	7	3	0	
			5	RCL A	4	7	5	
			6	STR 1	4	4	0	
			7	HALT	4	0	1	Enter y_i
	0	3	0	STR A	4	5	5	
			1	PRT E	0	2	6	Print y_i
			2	XCMA	4	0	3	
			3	RCL A	4	7	5	
			4	C MLT	7	1	4	
			5	RCL 2	4	6	1	Σxy____.
			6	C Add	7	3	0	
			7	RCL A	4	7	5	

Programmed by________

PROGRAM Figure 8

NO.______ PAGE 2 OF 3

DATE______ MODEL______

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS
2	0	4	0	STR 2	4	4	1	
			1	RCL M	4	7	4	
			2	C MLT	7	1	4	
			3	RCL 3	4	6	3	Σy^2 ______.
			4	C Add	7	3	0	
			5	RCL A	4	7	5	
			6	STR 3	4	4	3	
			7	XCMA	4	0	3	
	0	5	0	RCL 4	4	6	4	
			1	C Add	7	3	0	Σy ________.
			2	RCL A	4	7	5	
			3	STR 4	4	4	4	
			4	RCL 5	4	6	5	
			5	STR A	4	5	5	
			6	1	0	0	1	n-count ________.
			7	C Add	7	3	0	
3	0	6	0	RCL A	4	7	5	
			1	JUMP	6	0	4	Return to enter next x
			2	TO (5)	7	4	5	Branch to Print line-end of entries
			3	RCL 0	4	7	7	
			4	PRT A	0	2	7	Print Σx
			5	RCL 1	4	6	0	
			6	PRT A	0	2	7	Print Σx^2
			7	RCL 2	4	6	1	
	0	7	0	PRT A	0	2	7	Print Σxy
			1	RCL 4	4	6	4	
			2	PRT A	0	2	7	Print Σy
			3	RCL 3	4	6	3	
			4	PRT A	0	2	7	Print Σy^2
			5	RCL 5	4	6	5	
			6	PRT A	0	2	7	Print n
			7	TO (5)	7	4	5	

Programmed by ______

PROGRAM Figure 8 **NO.**________ **PAGE** 3 **OF** 3

DATE________ **MODEL**________

Branch Point	ADDRESS (P COUNT)			COMMAND	CODE			REMARKS (E A M 0 1 2 3 4 5 6 7 8 9)
4	1	0	0	PRT E	0	2	6	Print double line
			1	TO (0)	7	4	0	Return to begin new problem
			2	0	0	0	0	
			3	0	0	0	0	
			4	0	0	0	0	
			5	0	0	0	0	
			6	0	0	0	0	
			7	0	0	0	0	filler
	1	1	0	0	0	0	0	
			1	0	0	0	0	
			2	0	C	0	0	
			3	0	0	0	0	
			4	0	0	0	0	
			5	0	0	0	0	
			6	0	0	0	0	
			7	0	0	0	0	
5	1	2	0	CLRKB	0	3	3	
			1	RSI1	5	1	2	
			2	DCI1	5	1	4	
			3	DCI1	5	1	4	
			4	DCI1	5	1	4	
			5	DCI1	5	1	4	
			6	DCI1	5	1	4	Dotted line subroutine
			7	DCI1	5	1	4	
	1	3	0	RSI2	5	7	5	
			1	XFIE	5	1	7	
			2	SKLD	5	3	4	
			3	JUMP	6	3	6	
			4	INI2	5	7	7	
			5	JUMP	6	3	1	
			6	PRT E	0	2	6	
			7	RCL P	5	5	7	

Programmed by________

SPECIALIZED PROGRAMMING TECHNIQUES

General

This section is intended for the programmer with considerable experience on the calculator. The capabilities of the calculator discussed here are not ordinarily available to the operator because of the complexities involved. The instructions presented here are used in the permanently stored programs in the read-only memory and require a knowledge of the internal workings of the calculator and of the general operation of the internal programs. The specialized programming instructions must be used carefully to avoid changing or deleting data essential to the automatic operation of the calculator. It is also possible for some automatic calculator operations to conflict with a stored program if the specialized programming instructions are not used with caution.

The features provided by these instructions include: algebraic operations not available in the basic and advanced card reader instructions, additional comparison operations, 13 more flags and corresponding skip instructions.

The specialized programming instructions can be interspersed with the basic and advanced card reader instructions to provide more powerful programs. The instruction codes can be punched in a program card along with the other card reader instructions and loaded into the LEMP program memory in the same manner.

A complete list of the specialized programming instructions with their codes and descriptions is shown at the end of this section.

Name	Description	Set By	Reset By
Flag D	Transfer E to M flag	Keyboard logic	Automatic operation
Flag E	Transfer E to A flag	Keyboard logic	Automatic operation

Notes:

1. Flag 1 is set by following instructions: SFAE, SFEA, SFNE, SFRA, SFNZ, SFAN, SFEN, SFEO, and SFS6.
2. Reset by any automatic function except numeral and decimal point.
3. Reset by any Store, Recall, or Transfer instruction: that is, an instruction with a second octal digit of 4, 5, 6, or 7.

Flag 8, the align inhibit flag, may be used to inhibit an automatic alignment process that normally takes place if the exponent is negative. If flag 8 is set, the number is shifted right to eliminate the negative exponent, and the exponent is blanked.

Flags 2 through 7 should not normally be used when programming the calculator. The SFL8 instruction is entered to inhibit some unnecessary internal operations that use additional levels in the program storage register.

Table 3. Flag Usage

Function	Flags 1	2	3	4	5	6	7	8	A	B	C	D	E
RESET	X			X		X		X					
↑( )	X			X		X							
↓( )	X			X		X							
0 9	X	X	X	X	X		X						
.	X	X		X	X		X						
−	X			X		X		X	X	X	X	X	X
+	X			X		X		X	X	X	X	X	X
÷	X			X		X		X	X	X	X	X	X
×	X			X		X		X	X	X	X	X	X
=									X				

Flag Instructions

Fourteen flags are available in the calculator to record certain conditions for testing with Skip instructions. The Set Flag instructions should be used only by a programmer with a thorough understanding of flag operation in the internal programs. It is important that at the beginning of any keyboard function all flags used in that function are in the correct state.

Each flag has its specific function in automatic calculator operation. Some of the flags are set or reset automatically by the calculator. Table 2 shows the function of each flag in automatic operation and specifies the set and reset conditions for each flag.

Table 3 shows the flags that are used in the automatic execution of each keyboard function. If you use any of these flags in your program, you should ensure that the flag usage in your program does not conflict with the flag usage in any keyboard function that you include in the program.

Table 2. Flag Functions

Name	Description	Set By	Reset By
Flag 1	General purpose flag	See Note 1	SKF1 instruction
Flag 2	Decimal point flag	SFL2 instruction	See Notes 2 and 3
Flag 3	Exponent flag	SFL3 instruction	See Notes 2 and 3
Flag 4	Exponent and general purpose flag	SFL4 instruction	RFL4 instruction
Flag 5	Key flag	SFL5 instruction	See Notes 2 and 3
Flag 7	Sign flag	SFL7 instruction	See Notes 2 and 3
Flag 8	Align inhibit flag	SFL8 instruction	RFL8 instruction
Flag A	Equals flag	= key	Automatic operation

SPECIAL PROGRAMMING NOTES

The OFLOW indicator may be used to signify an operator error. The Set Overflow instruction provides the only means of setting the OFLOW indicator other than when the data to be printed exceeds the capacity of the calculator. The OFLOW indicator may be reset only by manually depressing the RESET or the CLR KB key.

In the fixed point mode, always enter a Normalize instruction before a Compare instruction because Compare instructions operate on the numeric portion of the number.

SPECIALIZED PROGRAMMING INSTRUCTIONS

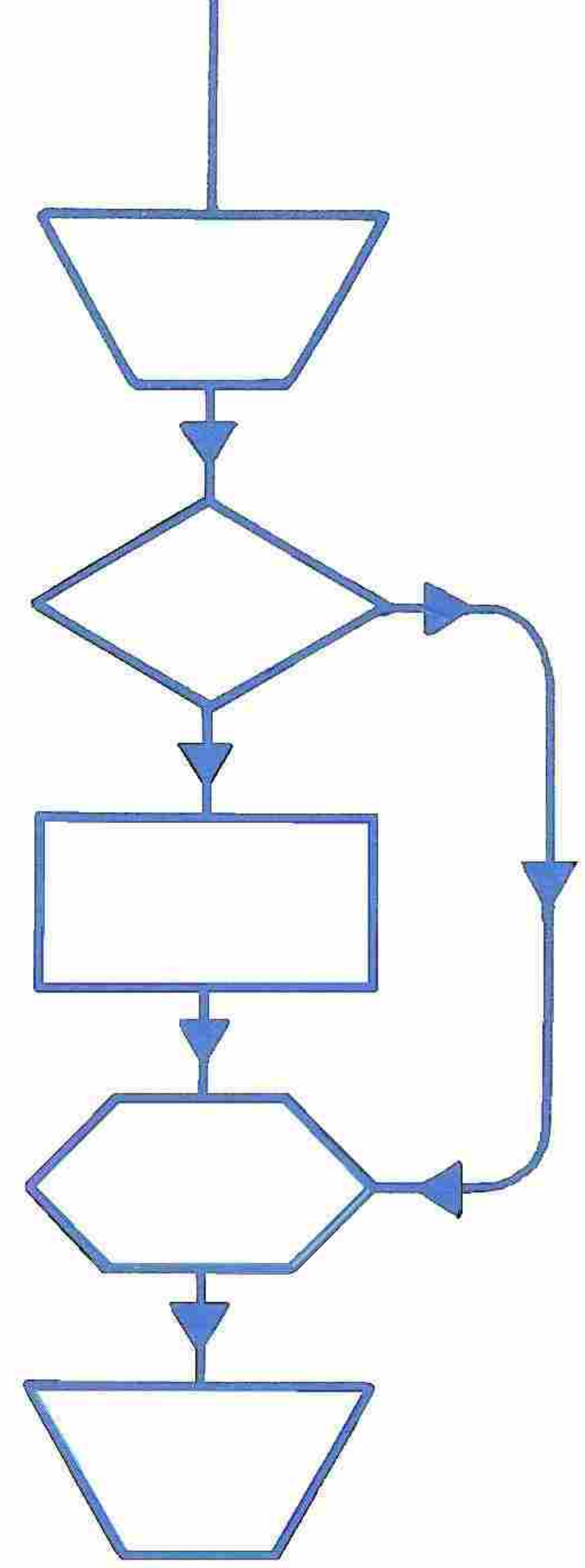
The following list gives the command, name, octal code, and a brief description of the specialized programming instructions:

Command	Name	Code	Description
ADDN	Add Numeric	410	Adds the numeric portion of the E-register contents to the numeric portion of the A-register contents and places the sum in the A-register. The exponent in the A-regis-remains unchanged.
ADDX	Add Exponents	412	Adds the exponent in the A-register to the exponent in the E-register and places the result in the exponent portion of the A-register. The numeric portion of the A-register contents remains unchanged.
ADRP	Add Repeat	426	If the contents of index 1 are not equal to zero, adds the E-register contents to the A-register contents, places the result in the A-register, and subtracts 1 from the contents of the index register.

Command	Name	Code	Description
CADD	Card Reader Add	730	Adds the E-register contents to the A-register contents and places the sum in the A-register. The contents of the E-and M-registers remain unchanged
CDIV	Card Reader Divide	710	Divides the M-register by the E-register contents and places the quotient in the E-, A-, and M-registers
CHGSGN	Change Sign	404	Changes the numerical sign of the E-register contents
CHSX	Change Exponent Sign	405	Changes the sign of the exponent in the E-register
CLRE	Clear E	423	Clears the E-register to zero
CMLT	Card Reader Multiply	714	Multiplies the M-register contents by the E-register contents and places the product in the E-and A-registers. The M-register contents remain unchanged
CSUB	Card Reader Subtract	731	Subtracts the E-register contents from the A-register contents and places the difference in the A-register. The contents of the E- and M-registers remain unchanged
NOP2	No Operation 2	476	No operation is performed
RCL(X)	Recall X	471	Recalls the contents of the X-register to the E-register. The contents of the X-register remain unchanged
RCL(Y)	Recall Y	472	Same as RCL(X) except that the Y-register contents are recalled
RCL(Z)	Recall Z	473	Same as RCL(X) except that the Z-register contents are recalled
RCXA	Recall A Exponent	407	Transfers the exponent from exponent store to the A-register. The numeric portion of the A-register contents remains unchanged
RFL4	Reset Flag 4	570	Resets flag 4

Command	Name	Code	Description
RFL6	Reset Flag 6	571	Resets flag 6
RFL8	Reset Flag 8	572	Resets flag 8
SBRP	Subtract Repeat	427	If the A-register contents are equal to or greater than the E-register contents, subtracts the E-register contents from the A-register contents, places the result in the A-register, and adds 1 to index 1
SFL2	Set Flag 2	561	Sets flag 2. Programmed in read-only memory to sense decimal point conditions during entry
SFL5	Set Flag 5	564	Sets flag 5. Programmed in read-only memory to sense numeral entry
SFL6	Set Flag 6	565	Sets flag 6
SFL8	Set Flag 8	567	Sets flag 8. Programmed in read-only memory to inhibit alignment procedure
SFS6	Set Flag on Switch 6	525	Sets flag 1 if the N switch is set to N
SKF2	Skip on Flag 2	541	Causes the program to ignore the next instruction in sequence if flag 2 is set
SKF3	Skip on Flag 3	542	Same as SKF2, but tests flag 3
SKF4	Skip on Flag 4	543	Same as SKF2, but tests flag 4
SKF5	Skip on Flag 5	544	Same as SKF2, but tests flag 5
SKF6	Skip on Flag 6	545	Same as SKF2, but tests flag 6
SKF7	Skip on Flag 7	546	Same as SKF2, but tests flag 7
SKF8	Skip on Flag 8	547	Same as SKF2, but tests flag 8
SKFD	Skip on Flag D	553	Same as SKF2, but tests flag D set by compiler to indicate transfers
SKFE	Skip on Flag E	554	Same as SKF2, but tests flag E set by compiler to indicate transfers

Command	Name	Code	Description
SOFL	Set Overflow	531	Sets OFLOW indicator
STR(X)	Store X	451	Transfers the contents of the E-register to the X-register. The contents of the E-register remain unchanged. Should be followed by a NOP instruction if possible
STR(Y)	Store Y	452	Same as STR(X) except that the contents of the Y-register are transferred. Should be followed by a NOP instruction if possible
STR(Z)	Store Z	453	Same as STR(X) except that the contents of the Z-register are transferred. Should be followed by a NOP instruction if possible
STXA	Store Exponent of A	406	Transfers the A-register exponent to exponent store. The contents of the A-register remain unchanged
SUBN	Subtract Numeric	411	Subtracts the numeric portion of the E-register contents from the numeric portion of the A-register contents and stores the difference in the A-register. The exponent in the A-register remains unchanged
SUBX	Subtract Exponent	413	Subtracts the exponent in the E-register from the exponent in the A-register and places the result in the A-register. The numeric portion of the A-register contents remains unchanged



Monroe model 1265

operating and programming instructions

MONROE
THE CALCULATOR COMPANY

1513-S Rev. Printed in U.S.A.

MONROE: Orange, New Jersey • Sales and Service Throughout the World