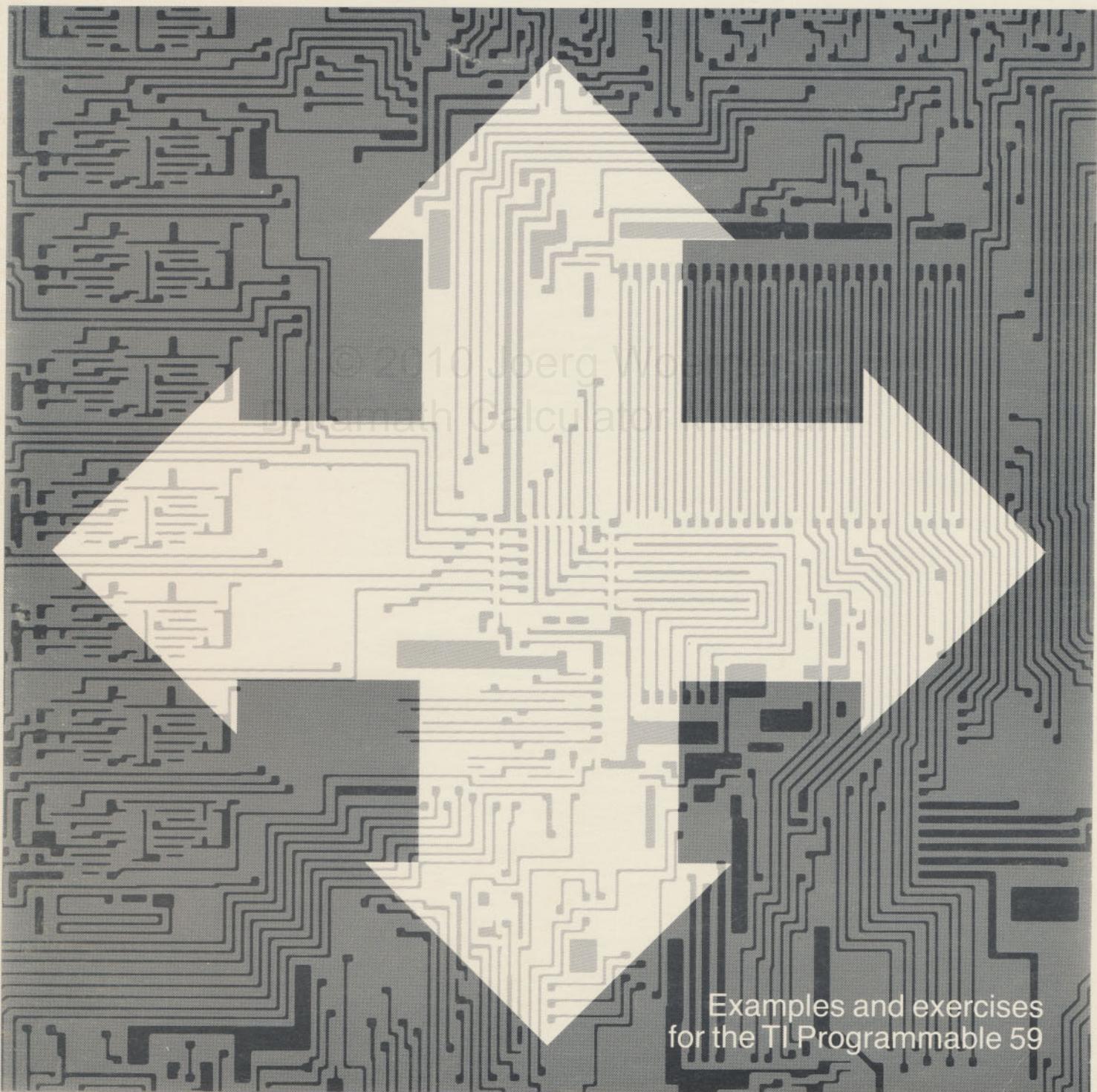


# TI Programmable 59 Workbook

PROPERTY OF: DATAMATH CALCULATOR MUSEUM

Educational examples and exercises for use with your TI Programmable 59





# TI Programmable 59 Workbook

© 2010 Joerg Woerner

This workbook has been prepared and is designed to help you understand how to use your TI Programmable 59. It contains examples and exercises with answers so that you may check yourself as you progress through the workbook while using your Programmable 59.

Prepared By Texas Instruments Learning Center



TEXAS INSTRUMENTS  
INCORPORATED  
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

First Edition  
Second Printing

Copyright © 1978

Texas Instruments Incorporated, All Rights Reserved

Unless otherwise noted, this publication, or parts thereof, may not be reproduced in any form by photographic, electrostatic, mechanical, or any other method, for any use, including information storage and retrieval.

For conditions of use and permission to use materials contained herein for publication in other than the English language, apply to Texas Instruments Incorporated.

For permissions and other rights under this copyright, please write Texas Instruments Learning Center, P.O. Box 225012 MS-54, Dallas, Texas 75265.



TEXAS INSTRUMENTS  
INCORPORATED  
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

## PREFACE

This workbook is intended to follow course material presented in training sessions and to give you sample problems for practice as you get to know your TI Programmable 59. As pointed out in the training sessions, the material covered is not comprehensive but has been designed to lead you rapidly into becoming familiar with the calculator, its functional groupings of keys, how to solve basic problems, and various features of your calculator.

Throughout the workbook, reference is made to the owner's manual, *Personal Programming*, which contains comprehensive information on using the calculator and programming. At the top of each page you will also find a *Personal Programming* reference section and page that should help if you encounter any difficulty in understanding the sample problems or exercises.

You will find a worked out sample problem to demonstrate each of the covered functions of your calculator. After you've studied the problem, solve the exercise corresponding to it. Be sure to have your calculator handy as you go through this workbook. Work each exercise on your calculator and write your results in this workbook. You will find solutions to these exercises in the Appendices.

© 2010 Joerg Woerner  
Datamath Calculator Museum



# CONTENTS

	<i>Page</i>
INTRODUCTION . . . . .	1
I BASIC FUNCTIONS THROUGH BEGINNING PROGRAMMING . . . . .	I-1
1. Basic Four-Function Group . . . . .	I-1
Addition . . . . .	I-1
Subtraction . . . . .	I-1
Multiplication . . . . .	I-2
Division . . . . .	I-2
Correcting an Entry Error . . . . .	I-3
2. Algebraic Operating System Entry Method . . . . .	I-4
3. Mathematical Functions . . . . .	I-6
4. Data Memories . . . . .	I-11
5. Summary Exercise (Keystroke Solutions) . . . . .	I-16
6. Beginning Programming . . . . .	I-18
7. Recording Programs on Magnetic Cards . . . . .	I-23
8. Reading a Program from Magnetic Cards . . . . .	I-24
9. Memory Partitioning . . . . .	I-25
10. Learn Mode . . . . .	I-26
11. <i>Solid State Software</i> <sup>TM</sup> Library Programs . . . . .	I-33
II INTERMEDIATE PROGRAMMING . . . . .	II-1
1. Variable Value Entry . . . . .	II-1
Enter Variable Values Through Display . . . . .	II-1
Enter Variable Values Using Data Memories . . . . .	II-3
2. Labels . . . . .	II-6
User-Defined Labels . . . . .	II-6
Program Labels . . . . .	II-11
3. Unconditional Branching . . . . .	II-13
4. Conditional Branching – Decision Making by Comparison . . . . .	II-18
5. Conditional Branching – Decision Making with Flags . . . . .	II-22
6. Conditional Branching – Decision Making for Looping . . . . .	II-29
III ADVANCED PROGRAMMING . . . . .	III-1
1. Indirect Addressing . . . . .	III-1
Data Memories . . . . .	III-1
Program Memories . . . . .	III-4
2. Alphanumeric Printing . . . . .	III-8
3. <i>Solid State Software</i> Module . . . . .	III-10
Appendices	
A Exercise Solutions – Section I . . . . .	A-1
B Exercise Solutions – Section II . . . . .	B-1
C Exercise Solutions – Section III . . . . .	C-1



# INTRODUCTION

This workbook is organized according to the following five functional groupings of keys:

1. Basic four functions
2. *AOS<sup>TM</sup>* entry method – algebraic operating system entry method
3. Mathematical functions
4. Data memories
5. Programming capabilities.

Before you attempt any programming, you should be familiar with how to use the first four functional groups of keys. This course is designed to lead you through these operations as quickly as possible. Experience has shown that a working familiarity with these operations makes programming a simpler matter.

© 2010 Joerg Woerner  
Datamath Calculator Museum



# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
I-4 II-2 V-10 VI-5 VII-3 VIII-1

## 1. BASIC FOUR-FUNCTION GROUP

MULTIPLICATION

These are the keys that provide you with the capability to add, subtract, multiply, and divide.

### ADDITION

Example 1-1:  $2 + 6 = ?$

Solution:

Press	Display
2	2
[ + ]	2.
6	6
[ = ]	8. (answer)

Exercise 1-1:  $6 + 12 = ?$

Press	Display
6	6
[ + ]	6.
12	12
[ = ]	18. (answer)

### SUBTRACTION

Example 1-2:  $12 - 5 = ?$

Solution:

Press	Display
12	12
[ - ]	12.
5	5
[ = ]	7. (answer)

Exercise 1-2:  $99 - 24 = ?$

Press	Display
99	99
[ - ]	99.
24	24
[ = ]	75. (answer)

# I Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING

I-4 II-2 V-10 DRV SII M

## MULTIPLICATION

Example 1-3:  $12 \times 13 = ?$

Solution:

Press	Display
12	12
[ X ]	12.
13	13
[ = ]	156. (answer)

## DIVISION

Example 1-4:  $192 \div 8 = ?$

Solution:

Press	Display
192	192
[ ÷ ]	192.
8	8
[ = ]	24. (answer)

Exercise 1-3:  $24 \times 36 = ?$

Press	Display
[ ]	
[ ]	8
[ ]	6
[ = ]	864. (answer)

Exercise 1-4:  $1890 \div 21 = ?$

Press	Display
[ ]	
[ ]	21
[ ]	1890
[ = ]	90. (answer)

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
II-2 V-3

## CORRECTING AN ENTRY ERROR

You can correct erroneous numerical entries with the [CE] key. The correction must be made immediately after keying in the erroneous value and *prior* to pressing any nonnumerical key. If a nonnumeral key is pressed after the erroneous entry, press [CLR] and enter the expression.

**Example 1-5:** You want to solve  $12.1 \times 7.8 = ?$  but as you enter the values, you accidentally press 7.9 instead of 7.8.

Solution:

Press	Display
12.1	12.1
[ X ]	12.1
7.9	7.9
[CE]	0
7.8	7.8
[ = ]	94.38 (answer)

**Exercise 1-5:** You want to solve  $21.9 + 10.3 = ?$  but as you enter the values, you accidentally enter 10.6 instead of 10.3.

Press	Display
[ ]	
[ ]	2
[ ]	1
[ ]	0
[ ]	6
[ = ]	32.2 (answer)

(answer) 32.2

(answer) .04

Note: As mentioned in the glossary, when calculating an internal result for the expression shown above, the calculator uses floating-point arithmetic.

# I Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING

II-3 V-11, 12

## 2. AOS\* ENTRY METHOD — ALGEBRAIC OPERATING SYSTEM ENTRY METHOD

This feature of TI programmable calculators integrates algebraic hierarchy and parentheses with the basic four-function operation to let you evaluate complete expressions.

The advantage of the *AOS* entry method is that: *If you can write the mathematical expression, you can enter it exactly as it is written — left to right.* The calculator automatically solves the problem according to the correct mathematical rules.

Example 2-1:  $2 + 8 \times 4 = ?$

Solution:

Press	Display
2	2
[ + ]	2.
8	8
[ X ]	8.
4	4
[ = ]	34. (answer)

Exercise 2-1:  $16 - 2 \div 2 = ?$

Press	Display
1	1
6	6
-	6.
2	2
[ ÷ ]	2.
2	1.
[ = ]	15. (answer)

Example 2-2:  $(2 + 8) \times 4 = ?$

Solution:

Press	Display
[ ( ]	0.
2	2
[ + ]	2.
8	8
[ ) ]	10.
[ X ]	10.
4	4
[ = ]	40. (answer)

Exercise 2-2:  $(16 - 2) \div 2 = ?$

Press	Display
1	1
6	6
-	6.
2	2
[ ÷ ]	2.
2	1.
[ = ]	7. (answer)

Note: As parentheses are closed, you get an interim result for the expression enclosed in parentheses.

\*Trademark of Texas Instruments Incorporated

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
II-3 V-11, 12

Exercise 2-3:  $(25 \times 4) + 2 + ((6.2 - 3) - 1) = ?$

Press      Display

0 2 5 × 4 + 2 + ( 6 . 2 - 3 ) - 1 = ?

104.2 (answer)

Exercise 2-4:  $10 \times (14.85 + (21.2 - 8.1) - (2.6 + 3.2)) = ?$

Press      Display

8 1 0 × ( 1 4 . 8 5 + ( 2 1 . 2 - 8 . 1 ) - ( 2 . 6 + 3 . 2 ) ) = ?

8

[S<sub>x</sub>]

221.5 (answer)

Press      Display

3 2 1 5 × ( 1 4 . 8 5 + ( 2 1 . 2 - 8 . 1 ) - ( 2 . 6 + 3 . 2 ) ) = ?

3

[ = ]

Exercise 3-3:  $\sqrt{256} = ?$

Press      Display

2 5 6 [ √ ] = ?

2

[ = ]

# Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

II-9, 10, 12 V-15-21

### 3. MATHEMATICAL FUNCTIONS

Often used but usually complex mathematical computations have been built into your calculator to give you quick accurate solutions with just a few keys. These functions have been added for your convenience. Use them as you need them.

The functions covered by this course are by no means all that are available. The ones included here are among the most frequently used. See your owner's manual for more in-depth coverage.

Example 3-1:  $8^2 = ?$

Press	Display
8	8
[ $x^2$ ]	64. (answer)

Example 3-2:  $3^4 = ?$

Press	Display
3 [ $y^x$ ]	3.
4	4
[ = ]	81. (answer)

Example 3-3:  $\sqrt{225} = ?$

Press	Display
225	225
[ $\sqrt{x}$ ]	15. (answer)

Exercise 3-1:  $42^2 = ?$

Press	Display
	1764. (answer)

Exercise 3-2:  $2^9 = ?$

Press	Display
	512. (answer)

Exercise 3-3:  $\sqrt{5726} = ?$

Press	Display
	75.67033765 (answer)

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
II-9, 10, 12 V-15-21

Example 3-4:  $\sqrt[4]{1296} = ?$

Note:  $[\sqrt[x]{y}] = [\text{INV}] [y^x]$

Solution:

Press                  Display

1296 [INV] [y<sup>x</sup>]      1296.  
4                            4  
[ = ]                      6. (answer)

Example 3-5:  $\sin 30^\circ = ?$

Notes:

1. Calculator turns on in degree mode.
2. Calculator stays in chosen angular mode until changed.
3. Angular mode affects only angular calculations — there's no effect on other calculations.

Solution:

Press                  Display

[2nd] [Deg]  
30                        30  
[2nd] [sin]              0.5

Example 3-6:  $10 + 3^4 = ?$

Press                  Display

10 [ + ]              10.  
3 [y<sup>x</sup>]              3.  
4                        4  
[ = ]                   91. (answer)

Exercise 3-4:  $\sqrt[100]{1470} = ?$

Press                  Display

1.075655429 (answer)

Exercise 3-5:  $\cos 47^\circ = ?$

Press                  Display

.6819983601 (answer)

Exercise 3-6:  $\tan 3 \text{ radians} = ?$

Press                  Display

-.1425465431 (answer)

Exercise 3-7:  $2 + \sqrt{8} = ?$

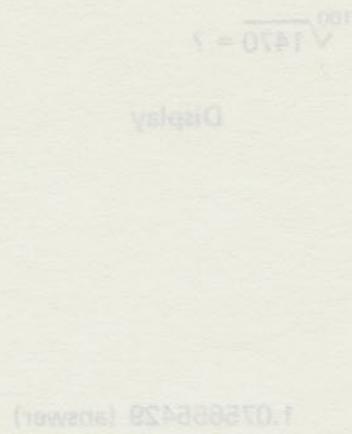
Press                  Display

4.828427125 (answer)

# I Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

II-9, 10, 12 V-15-21



**Example 3-7:** Round off 2.6379 to two decimal points.

*Notes:*

1. [Fix] \* function allows you to round off display to any number of desired digits
2. Calculator maintains specified digits in display until changed
3. Calculations are internally performed using the full decimal accuracy

**Solution:**

Press	Display
2.6379	2.6379
[2nd] [Fix] 2	2.64

**Example 3-8:** Restore 2.64 in above example to 2.6379.

**Solution 1:**

Press	Display
[2nd] [Fix]	2.64
[ 9 ]	2.6379 (answer)

\*Denotes second function key.

**Exercise 3-8:**  $\sin 45^\circ + \cos 45^\circ = ?$

Press      Display

1.414213562 (answer)

**Exercise 3-9:** Round off 10.79816 to three decimal points.

Press      Display

10.798 (answer)

**Exercise 3-10:** Restore 10.798 in Exercise 3-9 to 10.79816.

Press      Display

10.79816 (answer)

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
II-9, 10, 12 V-15-21

Solution 2:

Press	Display
[INV]	2.64
[2nd] [Fix]	2.6379 (answer)

Note: Either of above key sequences will restore display to full decimal accuracy of calculations.

Example 3-9: Enter 242,100 in scientific notation of  $2.421 \times 10^5$ .

Solution:

Press	Display
2.421	2.421
[EE]	2.421 00
5	2.421 05 (answer)

Note: Results of all calculations will be displayed in EE format until sequence [INV] [EE] is used to return to floating decimal mode or you press [CLR].

Example 3-10: Change above answer to engineering notation.

Solution:

Press	Display
[2nd] [Eng]	242.10 03 (answer)

Note: Results of all calculations will be displayed in engineering format until the sequence [INV] [2nd] [Eng] is used to return to floating decimal mode. [CLR] does not return display to floating decimal mode in this case.

Exercise 3-11: Enter 29,810 in scientific notation of  $2.981 \times 10^4$ .

Press	Display
2.981	2.981
4	2.981 00
[Inv] [Eng]	2.981 04 (answer)

Exercise 3-12: Change answer in Exercise 3-11 to engineering notation.

Press	Display
[2nd] [Eng]	29.81 03 (answer)

# I Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

II-9, 10, 12 V-15-21

**Example 3-11:** Drop the decimal portion of the number 2.7981.

Solution:

Press	Display
2.7981	2.7981
[2nd] [Int]	2. (answer)

**Example 3-12:** Drop the integer part of the number 4.79.

Solution:

Press	Display
4.79	4.79
[INV] [2nd] [Int]	0.79 (answer)

Note: The [Int]\* key used in Examples 3-11 and 3-12 not only changes the display, but also changes the numbers internally that are used for further calculations.

**Exercise 3-13:** Drop the decimal portion of the number 107.24.

Press	Display
	2.84
	2.8326 (answer)

**Exercise 3-14:** Drop the integer part of the number 11.9247.

Press	Display
	11.9247
	11.9247
	0.9247 (answer)

\* Denotes second function key.

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
II-6, 7 V-22-25

## 4. DATA MEMORIES

Data memories provide you with storage locations in which you can store variable values or the results of interim calculations. The calculator holds these values until you are ready to recall them to the display. The data memories are also used to perform "memory arithmetic".

### Example 4-1: Storing and recalling.

- (a) Store 100 in memory 01
- (b) Store 500 in memory 02
- (c) Recall value from memory 01
- (d) Recall value from memory 02

Solution:

Press	Display/Comments
(a) 100	100 (Value to be stored)
[STO]	100. (Instruction to store)
01	100. (Memory number/storage location)
(b) 500	500
[STO]	500.
02	500.
(c) [RCL]	500.
01	100.
(d) [RCL]	100.
02	500.

### Notes:

1. Values remain in memory even after they are recalled.
2. You can "write over" value in memory by storing a new number in the memory. The old number is discarded and the new number is stored.

### Exercise 4-1: Storing and recalling.

- (a) Store 175 in memory 29
- (b) Store 214 in memory 12
- (c) Recall value from memory 29
- (d) Recall value from memory 12

Solution:

Press	Display
(a) 300	300
[SUM]	00
01	01
[STO]	01
02	02
(b) 300	300
[RCL]	10
01	10
[RCL]	175.
02	175. (answer)
[RCL]	214.
01	214. (answer)

# Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

II-6, 7 V-22-25

**Example 4-2:** Add to value in memory.

- (a) Store 200 in memory 01
- (b) Add 300 to memory 01
- (c) Recall result from memory 01

Solution:

Press	Display	Comments
(a) 200 [STO] 01	200 200. 200.	{ Store 200 in memory 01
(b) 300 [SUM] 01	300 300. 300.	{ Sum 300 to memory 01 value
(c) [RCL] 01	300. 500.	Recall result to display

*Note:* The result of "memory arithmetic" can be seen in the display *only* by recalling it. Calculations are performed internal to the calculator. *This note applies to remaining data memory problems also.*

**Example 4-3:** Subtract from value in memory.

- (a) Store 500 in memory 06
- (b) Subtract 200 from memory 06
- (c) Recall result from memory 06

Solution:

Press	Display
(a) 500 [STO] 06	500
(b) 200 [INV] [SUM] 06	500. 200
(c) [RCL] 06	300.

**Exercise 4-2:** Add to value in memory.

- (a) Store 212 in memory 21
- (b) Add 42 to memory 21
- (c) Recall result from memory 21

Solution:

Press	Display
(a) 212 [STO] 21	212
(b) 42 [SUM]	212. 254.
(c) [RCL] 21	254.

**Exercise 4-3:** Subtract from value in memory.

- (a) Store 100 in memory 11
- (b) Subtract 29 from memory 11
- (c) Recall result from memory 11

Solution:

Press	Display
(a) 100 [STO] 11	100
(b) 29 [INV] [SUM] 11	100. 71.
(c) [RCL] 11	71.

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
II-6, 7 V-22-25

**Example 4-4:** Multiply value in memory.

- (a) Store 200 in memory 08
- (b) Multiply memory 08 by 6
- (c) Recall result from memory 08

Solution:

Press	Display
(a) 200	200
[STO] 08	200.
(b) 6	6
[Prd] * 08	6.
(c) [RCL] 08	1200.

**Example 4-5:** Divide value in memory.

- (a) Store 400 in memory 15
- (b) Divide memory 15 by 10
- (c) Recall result from memory 15

Solution:

Press	Display
(a) 400	400
[STO] 15	400.
(b) 10	10
[INV] [Prd] * 15	10.
(c) [RCL] 15	40.

**Exercise 4-4:** Multiply value in memory.

- (a) Store 75 in memory 09
- (b) Multiply memory 09 by 15
- (c) Recall result from memory 09

Solution:

Press	Display
(a) 75	75
[STO] 09	75.
(b) 15	15
[Prd] * 09	15.
(c) [RCL] 09	1125. (answer)

**Exercise 4-5:** Divide value in memory.

- (a) Store 999 in memory 27
- (b) Divide memory 27 by 33
- (c) Recall result from memory 27

Solution:

Press	Display
(a) 999	999
[STO] 27	999.
(b) 33	33
[INV] [Prd] * 27	33.
(c) [RCL] 27	30.27272727 (answer)

\*Denotes second function key.

# I Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

II-6, 7 V-22-25

**Example 4-6:** Exchange value in display with value in memory.

- (a) Store 100 in memory 01  
Store 200 in memory 02  
Store 300 in memory 03  
(b) Using exchange key, reverse order so 300 is in memory 01, 200 is in memory 02, 100 is in memory 03.

Solution:

Press	Display
(a) 100	100
[STO] 01	100.
200	200
[STO] 02	200.
300	300
[STO] 03	300.

- (b) Reverse order. With 300 in display:

Press	Display
[Exc] * 01	100.

The 300 in the display was placed in memory 01; the 100 in memory 01 was brought to display.

Press	Display
[Exc] * 03	300.

The 100 is the display was placed in memory 03; the 300 in memory 03 was brought to the display.

Verify that the order was reversed.

Press	Display
[RCL] 01	300.
[RCL] 02	200.
[RCL] 03	100.

\*Denotes second function key.

**Exercise 4-6:** Exchange value in display with value in memory.

- (a) Store 41 in memory 11  
Store 97 in memory 20  
Store 147 in memory 26  
(b) Using exchange key, move 41 to memory 20; 97 to memory 26; 147 to memory 11.

Solution:

Press	Display
(a) 41	41
.97	97
[STO] 26	97
147	147

Press	Display
[RCL] 11	41
[RCL] 20	97
[RCL] 26	147
[RCF] 11	41
[RCF] 20	97
[RCF] 26	147

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
II-6, 7 V-22-25 OF, E, S-II

**Example 4-7:** Clear all data memories simultaneously. (Places 0 in all data memories.)

**Solution:**

Press	Display
[CMs]*	000.000

You can verify you have cleared all data memories by recalling data memories used above.

Press	Display
-------	---------

[RCL] 01	0.
[RCL] 02	0.
[RCL] 03	0.

**Exercise 4-7:** Clear all data memories simultaneously.

**Solution:**

Press	Display
-------	---------

[CMs]*	000.000
--------	---------

(Recalling any of the memories used displays a zero when all memories have been cleared.)

©2010 Joerg Woerner  
Datamath Calculator Museum

Note: Since data memory is the memory internal to the calculator it must be in conversion mode (i.e., memory).

Solve using Solution 1 and 2 approaches to expressions below.

**Solution 1:**

Press	Display
2 [ = ]	200.000
1 [ + ]	1.00
1 [ x ]	1.00
1200.000 [ = ]	1200.000
2 [ = ]	200.333333 (answer)

\* Denotes second function key.

# I Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

II-2, 3, 10 V-10, 11, 21

### 5. SUMMARY EXERCISE (KEYSTROKE SOLUTIONS)

Remember from earlier discussions that if you can write the mathematical expressions required to solve problems, you can solve the problem by keying each expression into the calculator exactly as it is written, left to right.

**Example 5-1:** You have decided to place \$1500 into a savings account paying 6% interest, compounded annually. Assuming you make no withdrawals, how much money will you have in 5 years?

Mathematical expression:

$$FV = PV \times (1 + i)^n$$

where:

FV = future value of investment

PV = present value (amount invested)

i = interest rate (as a decimal)

n = period of investment

Solution 1:

Key in expression exactly as written, substituting variable values for PV, i, and n.

$$FV = 1500 \times (1 + .06)^5$$

Press	Display
1500 [ X ] [ ( ]	1500.
1 [ + ]	1.
.06 [ ) ] [ y <sup>x</sup> ]	1.06
5 [ = ]	2007.338366 (answer)

**Exercise 5-1:** You are considering the purchase of a new home. You anticipate financing \$45,000 at a 9% annual interest rate for 30 years. What is your monthly payment for principal and interest?

Mathematical expression:

$$PMT = PV \times (i \div (1 - (1 + i)^{-n}))$$

where:

PV = present value (amount financed)

PMT = monthly payment

i = periodic interest rate

n = term of loan

*Note:* Since payments are monthly, interest rate and term of note must be in consistent terms (i.e., monthly).

Solve using Solutions 1 and 2 approaches to example problem.

Solution 1:

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
II-2, 3, 6, 10 V-10, 11, 21, 23

Solution 2:

- (a) Designate a data memory for each variable (i.e., memory 1 for PV; memory 2 for i; memory 3 for n)
- (b) Store variable values in designated data memories.

Press      Display

1500 [STO] 01      1500.  
.06 [STO] 02      0.06  
5 [STO] 03      5.

- (c) Key in expression exactly as written, substituting "recall instructions" for variable values.

$$FV = RCL\ 01 \times (1 + RCL\ 02)^{RCL\ 03}$$

Press      Display

[RCL] 01 [ X ] [ ( ] 1500.  
1 [ + ]            1.  
[RCL] 02 [ ) ] [y<sup>x</sup>] 0.06  
[ ] [ = ]            1.06  
[RCL] 03 [ = ]            5.  
[ = ]            2007.338366 (answer)

Solution 2:

8. BEGINNING PROGRAMMING

$FV = P(1 + I)^n$

# I Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING

IV-1-9 V-43-47

## 6. BEGINNING PROGRAMMING

Problem: You have decided to place \$1500 in a savings account that pays 6% interest, compounded annually. Assuming you make no withdrawals, how much money will you have in 5 years?

Mathematical expression:  $FV = PV \times (1 + i)^n$

This is the same problem that you just solved manually on your calculator from the keyboard.

You can easily *program* your calculator to handle this problem. First, the calculator must be in the LEARN mode. Press the [LRN] key to enter the LEARN mode; press it again to leave the LEARN mode. Notice the unique display format 000 00 when in the LEARN mode. This unique display can consist of any combination of numbers, but the format remains the same. You will become more familiar with the LEARN mode in Section II. For now, it is important that you know how to enter and leave the LEARN mode.

In the example on the next page, the left and center columns are the keystroke solutions to the problem from the previous section. The right column is our program.

Notice all we did was add a stop instruction ([R/S]) and a reset instruction ([RST]) to our previous solution in order to complete the program. To store your program instructions, enter the LEARN mode, key in the instructions, and exit the LEARN mode. Your program instructions are then stored in the calculator's program memory.

After you enter the LEARN mode and start keying in the instructions, the left-hand set of digits in the display advances sequentially as you press instruction keys. These three digits indicate the position of the "program pointer", telling you which program instruction number or storage pigeonhole you are ready to fill. Notice we've numbered our instructions starting with 000. When you first enter the LEARN mode, the "pointer" is at 000. When you press [RCL], that instruction is placed in 000, and the pointer advances to 001, indicating the next pigeonhole to be filled. Notice just before you exit the LEARN mode, the pointer is setting at 015.

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
IV-1-9 V-43-47

I

## Previous Keystoke Solutions

## Programming Sequence

### Solution 1

### Solution 2

Press	Press	Press
1500	[RCL]	000 [RCL]
.06	01	001
[X]	[X]	002 [X]
[()	[()	003 [()
1	1	004 1
[+]	[+]	005 [+]
5	[RCL]	006 [RCL]
[y <sup>x</sup> ]	[y <sup>x</sup> ]	007 02
[=]	[RCL]	008 [ ) ]
	03	009 [y <sup>x</sup> ]
	[=]	010 [RCL]
		011 03
		012 [=]
		013 [R/S]
		014 [RST]
		015 [LRN] (exit LEARN mode)

Now that the program has been written and entered, we are about ready to run it. Look at your program instructions again. When the variable values are needed, they are to be recalled from a previously designated memory, so we need to ensure that the correct variable values are stored in these memories. For this problem:

Press	Display	Comments
1500 [STO] 01	1500.	Store PV value in memory 01
.06 [STO] 02	0.06	Store i value (as a decimal) in memory 02
5 [STO] 03	5.	Store n value in memory 03

Before *running* the program, remember where the program pointer was setting, step 015. Whenever the [R/S] key is pressed to start a program, execution starts sequentially from the current location of the program pointer. Since our program instructions start at 000, not 015, we must reset the pointer to 000.

Press [RST] to relocate the pointer to 000.

# Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

IV-1-9 V-43-47

The program is stored in program memory, the variable values are stored in data memory, and the pointer is reset to step 000 in the program memory. To run the program

Press [R/S] (Answer = 2007.338366)

*Notes:*

1. If [R/S] had been pressed before [RST] in the above example, the pointer would have started from step 015. Since there are no instructions after 015, you would have received a "flashing 0." To clear it, press [CE] to stop the flashing and [RST] to relocate the pointer to step 000. Then press [R/S] to run the program.
2. You may ask the question "Why did I have to press [RST] at the keyboard when it was in the program instructions?" Think about the location of the program pointer. When you first keyed in the program, the pointer stopped at step 015, below your instructions. Pressing [RST] was necessary to move the pointer to step 000. Once the pointer is reset and [R/S] is pressed, the pointer will stop at the [R/S] program instruction. When [R/S] is pressed again, the pointer proceeds sequentially and the very first program instruction it encounters is [RST], which will relocate the pointer to step 000.
3. When [RST] is pressed at the keyboard, the pointer only resets to step 000. However, when [RST] is encountered in a program during execution, it resets to 000 and automatically executes from that point.

With the program now in program memory, suppose you wanted to know how much \$3000 invested today would be worth in five years at 6% interest, compounded annually. Since 6% and five years don't change and are already stored in data memories 02 and 03, you only need to store the new PV (\$3000) in data memory 01 and start the program.

Press	Display	Comments
3000 [STO] 01	3000.	Store new PV in memory 01
[R/S]	4014.676733	New FV

(R/S) key is pressed to start or exit a program, execute or store a step in memory, or move the pointer to 000 or 015. The user can then use the pointer to 000 to start another instruction or 015 to locate the pointer to 000.

# Basic Functions Through Beginning Programming

I

PERSONAL PROGRAMMING  
IV-1-9 V-43-47

(beginning) → F-8 select E

**Exercise 6-1:** You are considering the purchase of a new home. You anticipate financing \$45,000 at a 9% annual interest rate for 30 years.

- What is your monthly payment (principal and interest)?
- What is your monthly payment if you have to finance at 9½% interest?

Write, enter, and run the program to solve this problem.

Mathematical expression:

$$\text{PMT} = \text{PV} \times (i \div (1 - (1 + i)^{-n}))$$

**Hints:**

- Write your manual keystroke solution from Exercise 5-1.
- Identify your variables (numbers that may change) and assign a memory number that will contain each.
- Replace the numbers in your manual keystroke solution with [RCL] 01 or nn (memory number you assigned).
- Write a R/S and RST in at end of your program.
- Get into the LEARN mode, key in your program, and exit the LEARN mode.
- Store your variable values in the memories you selected.
- Press [RST] and [R/S] to run your program the first time.
- Store new variable values in the appropriate memories (only those that change) and press [R/S].

© 2010 Joerg Woerner  
Datamath Calculator Manual

# I Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

IV-1-9 V-43-47

Exercise 6-1: You are considering the purchase of a new home. You anticipate incurring \$15,000 at a 3% interest rate for 30 years.

(a) What is your monthly payment (including principal and interest)?

(b) What is your monthly payment if you pay off the balance in 15 years?

(c) What is the present value of this house?

(d) What is the future value of this house?

(e) What is the present value of this house?

MS-Personal calculator:

$$(PMT(i + f) - f) + i \times PV = FV$$

Hints:

1. What about using calculate solution from exercise 5-1.

Exercise 6-1.

2. In reality you always (unfortunately) have to pay a down payment as well as a monthly sum per year.

What about using

3. Besides the numbers, how many keys do you have?

4. What is R12 and R2T in the case of your

calculator?

5. Get into the EASY mode key in your

calculator, and exit the EASY mode.

6. Store your variables always in the memory

or selection

7. Use [RCL] and [RS] of the four buttons

the first time

8. Store new variables always in the appropriate

memory (only those three commands) and

press [RS].

## Exercise 6-1: (Workspace)

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING

VII-1-3

8. RECORDING A PROGRAM

## 7. RECORDING PROGRAMS ON MAGNETIC CARDS

To practice recording programs on magnetic cards, use the program from the previous section. Turn to page VII-2 in *Personal Programming* for in-depth instructions.

These instructions are summarized below. Our previous program was 15 steps long, however 240 steps can be placed on one side of a magnetic card. Consequently our program will fit on "Side 1."

### 1. Key sequence

Enter              1              Card side being recorded

Press              [2nd]

Press              [Write]

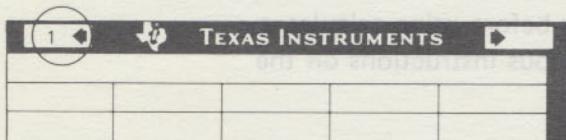
### 2. Insert card into lower slot on right-hand side

### 3. Motor will start and pull card through

### 4. Remove card from left side of calculator

### 5. Program is permanently recorded

### 6. Write card side number in block opposite arrow



### 7. Write name of your program on the magnetic card to identify it.

### 8. Turn calculator off and proceed to next section.

*Note:* If card is properly recorded a stable "1" (for this example) will appear in the display. A flashing display indicates a potential operator error. Refer to owner's manual for types of errors possible.

## CAUTION

Always remove card from calculator before using calculator again. This will avoid placing extraneous instructions on the card or erasing instructions.

# I Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
VII-1, 5, 6

## 8. READING A PROGRAM FROM MAGNETIC CARDS

Now read the program into program memory from the card you just recorded. A summary of the steps required to read a program is shown below. For detailed information, see page VII-5 of *Personal Programming*.

1. Turn calculator on
2. Enter 1 (side of card to be read) into display
3. Insert card into lower slot on right-hand side
4. Motor will start and pull card through
5. Remove card
6. Program has been loaded into calculator memory
7. Solve previous problems by storing variable values, pressing [RST], and pressing [R/S]. Check your answer to be sure the program was stored and run correctly.

*Note:* Same as on previous page.

### CAUTION

Always remove card from calculator before using calculator again. This will avoid placing extraneous instructions on the card or erasing instructions.

### CAUTION

Always remove card from calculator before using calculator again. This will avoid placing extraneous instructions on the card or erasing instructions.

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
V-22, 29, 42 VII-2, 3

## 9. MEMORY PARTITIONING

The TI-59 has a "variable" memory partition. This means you may designate how you want to allocate the memory registers between program instruction steps and data memories according to specific rules.

The partition must be changed in steps of 10 data memories. Increasing data memories decreases program steps and vice versa. Since there are an equivalent eight program steps per data memory, an increase of 10 data memories results in a reduction of 80 program steps.

**Example 9-1:** Display memory partition immediately after turning calculator on.

*Note:* Op Code 16 displays only the current partition; it doesn't change it.

Solution:

Press	Display/Comments
[Op] * 16	(Display = 479.59) Last program step number → Last data memory number

**Example 9-2:** Change memory partition to provide 20 data memories.

*Note:* Op Code 17 is always used to change the partition.

Solution:

Press	Display / Comments
2 [Op] * 17	2 Number of blocks of 10 data memories 799.19

\*Denotes second function key.

**Exercise 9-1:** Change memory partition to provide 100 data memories.

Solution:

Press	Display/Comments
10 [Op] * 17	10 Number of blocks of 10 data memories 799.19

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
I-4 IV-6-9 V-43, 44

## 10. LEARN MODE

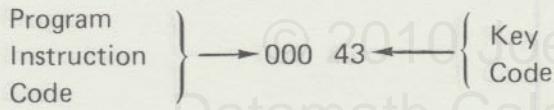
Earlier you learned that to place program instructions into the calculator, it must be in the LEARN mode. You enter the LEARN mode by pressing [LRN] and you exit it by pressing [LRN] once more. You know you are in the LEARN mode by the unique display format, 000 00.

You can *only* enter program instructions when the calculator is in the LEARN mode.

When in the LEARN mode, the calculator interprets all keystrokes as program instructions *except* "edit" keys:

Single step	[SST]
Back step	[BST]
Insert	[Ins] *
Delete	[Del] *

The unique LEARN mode display represents the following:



There are certain rules which tell you how to "read" the key codes. You should note that certain merged code functions deviate from these rules. See owner's manual, page V-51.

Key code rules:

1. All numbered keys are represented by their appropriate number (2-digit format)
2. All other keys are represented by 2 digits

1st digit: Row number (Row 1 at top)  
2nd digit: Column number (Primary function – column 1 at left)  
(2nd function – column 6 at left)

Examples:

Key Code	Key	Row/Column
18	[ C' ] *	1 8
43	[RCL]	4 3
02	2	(Not applicable)

\*Denotes second function key

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
I-4 IV-6-9 V-43, 44, 51-54

**Example 10-1:** Program the following expression. (Since you will need this program for following exercises, do not turn your calculator off.)

Display	Press	Comments
	[LRN]	Enter LEARN mode
000 00	[RCL]	
001 00	01	
002 00	[ + ]	
003 00	[RCL]	
004 00	02	
005 00	[ = ]	
006 00	[R/S]	
007 00	[LRN]	Exit LEARN mode

**Exercise 10-1:** You are going to program the expression for finding the payment on a home mortgage. Enter the LEARN mode and key in the expression as follows.

Display	Instruction
000 00	[RCL]
001 00	01
002 00	[ X ]
003 00	[ ( ]
004 00	[RCL]
005 00	02
006 00	[ ÷ ]
007 00	[ ( ]
008 00	1
009 00	[ - ]
010 00	[ ( ]
011 00	100
012 00	[ + ]
013 00	[RCL]
014 00	02
015 00	[ ) ]
016 00	[ y <sup>x</sup> ]
017 00	[RCL]
018 00	03
019 00	[ +/- ]
020 00	[ ) ]
021 00	[ ) ]
022 00	[ = ]
023 00	[R/S]
024 00	

The expression you just entered is incorrect. The correct expression is:

$$PMT = PV \times (i \div (1 - (1 + i)^{-n}))$$

1. Edit the program keyed in above against the correct program, making the necessary changes.
2. Write the correct program.

# Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

IV-21 V-48, 51-54

**Example 10-2:** Reset program pointer to 000 and verify the repositioning.

**Solution:**

1. While out of the LEARN mode, press [RST]
2. Enter LEARN mode to verify, press [LRN] (Display = 000 43)

**Note:** 000 000 00 500

1. Key Code "43" is the [RCL] key which is the instruction placed in program location 000.
2. The pointer not only shows the program location, but also what is currently stored in that location. When you initially started to key in the expression, the display read 000 00. The pointer was at 000, and 00 was stored there. When you keyed in the recall instruction, the pointer automatically moved to the next location. You could not see the instruction you keyed into 000. Now that you've reset the pointer to 000, it shows the instruction "43" [RCL] is stored there, which is what you initially keyed into that location.

00 050  
[43] 00 150  
1 = 1 00 550  
[25] 00 550  
00 450

**Exercise 10-1: (continued)**

**Hint:** You will have to:

1. Make one insertion
2. Make one deletion
3. Make two corrections.

**Solution:**

1. To edit the incorrect program

Press	+	Display
[RCL]	00	00 500
00	00	00 400
1 = 1	00	00 300
[RCL]	00	00 200

2. The correct key sequence:

Display	Instructions
---------	--------------

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
IV-21 V-48, 51-54

**Example 10-3:** "Edit" your program by "single stepping" forward, verifying you keyed in the correct instructions.

*Note:* The "single step" key [SST] can be pressed while in the LEARN mode and not be remembered as a program instruction.

**Solution:**

Keystroke	Display	Instruction
[SST]	000 43	[RCL]
[SST]	001 01	01
[SST]	002 85	[ + ]
[SST]	003 43	[RCL]
[SST]	004 02	02
[SST]	005 95	[ = ]
[SST]	006 91	[R/S]

**Example 10-4:** “Back step” through your program.

*Note:* The "back step" key [BST] can also be pressed while in the LEARN mode and not be remembered as a program instruction.

**Solution:**

Keystroke	Display
[BST]	006 91
[BST]	005 95
[BST]	004 02
[BST]	003 43
[BST]	002 85
[BST]	001 01
[BST]	000 43

Get out of LEARN mode – press [LRN]

# I Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING

IV-21 V-48, 51-54 V-15-VI

**Example 10-5:** Go to step 279 and verify the pointer relocated to 279.

*Note:* Single stepping and back stepping through a program can be time consuming, if you want to go to a specific step anywhere in your program. You can use the [GTO] instruction in editing, *but you must be out of the LEARN mode*. In the LEARN mode, [GTO] will be remembered as an instruction, outside the LEARN mode, it is not.

Solution:

Be sure the calculator is not in the LEARN mode.

Press	Display	Comments
[GTO] 279	0.	
[LRN]	279 00	Enter LEARN mode to verify reposition
[LRN]	0.	

**Example 10-6:** Assume you really meant to have a “–” at step 002 instead of the “+” keyed in. Make the correction.

Solution:

While out of the LEARN mode:

Press	Display	Comments
[GTO] 002	0.	Go to step 002
[LRN]	002 85	Enter LEARN mode
[ – ]	003 43	Make correction by writing over previous instruction
[BST]	002 75	Verify correction

Example 10-5: "Edit" your program by single stepping, then "exit" to return to the beginning, leaving any key pressed in the current instruction.

Notes: Type "next step" key [BS2] can be used to move while in the LEARN mode and not be remembered as a program instruction.

Keystroke	Display	Instruction
[RC1]	000 43	
01	001 01	[T22]
[ + ]	003 88	[BS2]
[RC1]	003 43	[BS2]
05	004 05	[BS2]
[ – ]	000 00	[BS2]
[BS2]	000 00	[BS2]
01	001 00	[BS2]

Example 10-6: "Edit" your program by single stepping, then "exit" to return to the beginning, not be remembered as a program instruction.

Keystroke	Display	
01	000 00	[BS2]
02	000 00	[BS2]
03	000 00	[BS2]
05	000 00	[BS2]
00	000 00	[BS2]
01	001 00	[BS2]
03	000 00	[BS2]

Get out of LEARN mode - press [LRN]

# Basic Functions Through Beginning Programming

---

**PERSONAL PROGRAMMING**  
IV-21 V-48, 51-54

**Example 10-7:** Assume you erroneously left out two instructions when keying in your program. Insert those instructions.

What You Wanted			What You Have in Memory		
[RCL]	000	43	[RCL]	000	43
01	001	01	01	001	01
[ - ]	002	75	Insert 2	[ - ]	002 75
2	003	02		[RCL]	003 43
[ + ]	004	85	[ + ]	02	004 02
[RCL]	005	43		[ = ]	005 95
02	006	02		[R/S]	006 91
[ = ]	007	95			
[R/S]	008	91			

### *Notes:*

1. You must make insertions in the LEARN mode. The [Ins] \* keystrokes are not remembered as instructions.
  2. When you press [Ins] \* you move all instructions below the pointer down one position in the program memory.

Solution: Single step to the position where you want to insert your first instruction (2 at step 003).

Press	Display	Comments
[Ins] *	003 00	(Opens space for the 2)
[Ins] *	003 00	Opens space for the +)
2	004 00	
[ + ]	005 43	

\* Denotes second function key.

# I Basic Functions Through Beginning Programming

## PERSONAL PROGRAMMING

IV-21 V-48, 51-54

### Example 10-7: (Continued)

To verify that you entered the new instructions, back step two steps.

Press	Display	Comments
[BST]	004 85	(Indicates you entered a +)
[BST]	003 02	(Indicates you entered a 2)

#### CAUTION

If the program memory is completely filled and you insert instructions, you will push the last instructions out and they will be lost.

Press	Display	Comments
000	000 [AC]	
01	001 01	
02	002 1 -	
03	003 [AC]	{
04	004 02	S
05	005 00	I + ]
06	006 13	
07	007 00	
08	008 00	
09	009 00	
10	010 00	
11	011 00	
12	012 00	
13	013 00	
14	014 00	
15	015 00	
16	016 00	
17	017 00	
18	018 00	
19	019 00	
20	020 00	
21	021 00	
22	022 00	
23	023 00	
24	024 00	
25	025 00	
26	026 00	
27	027 00	
28	028 00	
29	029 00	
30	030 00	
31	031 00	
32	032 00	
33	033 00	
34	034 00	
35	035 00	
36	036 00	
37	037 00	
38	038 00	
39	039 00	
40	040 00	
41	041 00	
42	042 00	
43	043 00	
44	044 00	
45	045 00	
46	046 00	
47	047 00	
48	048 00	
49	049 00	
50	050 00	
51	051 00	
52	052 00	
53	053 00	
54	054 00	
55	055 00	
56	056 00	
57	057 00	
58	058 00	
59	059 00	
60	060 00	
61	061 00	
62	062 00	
63	063 00	
64	064 00	
65	065 00	
66	066 00	
67	067 00	
68	068 00	
69	069 00	
70	070 00	
71	071 00	
72	072 00	
73	073 00	
74	074 00	
75	075 00	
76	076 00	
77	077 00	
78	078 00	
79	079 00	
80	080 00	
81	081 00	
82	082 00	
83	083 00	
84	084 00	
85	085 00	
86	086 00	
87	087 00	
88	088 00	
89	089 00	
90	090 00	
91	091 00	
92	092 00	
93	093 00	
94	094 00	
95	095 00	
96	096 00	
97	097 00	
98	098 00	
99	099 00	
00	100 00	
01	101 00	
02	102 00	
03	103 00	
04	104 00	
05	105 00	
06	106 00	
07	107 00	
08	108 00	
09	109 00	
10	110 00	
11	111 00	
12	112 00	
13	113 00	
14	114 00	
15	115 00	
16	116 00	
17	117 00	
18	118 00	
19	119 00	
20	120 00	
21	121 00	
22	122 00	
23	123 00	
24	124 00	
25	125 00	
26	126 00	
27	127 00	
28	128 00	
29	129 00	
30	130 00	
31	131 00	
32	132 00	
33	133 00	
34	134 00	
35	135 00	
36	136 00	
37	137 00	
38	138 00	
39	139 00	
40	140 00	
41	141 00	
42	142 00	
43	143 00	
44	144 00	
45	145 00	
46	146 00	
47	147 00	
48	148 00	
49	149 00	
50	150 00	
51	151 00	
52	152 00	
53	153 00	
54	154 00	
55	155 00	
56	156 00	
57	157 00	
58	158 00	
59	159 00	
60	160 00	
61	161 00	
62	162 00	
63	163 00	
64	164 00	
65	165 00	
66	166 00	
67	167 00	
68	168 00	
69	169 00	
70	170 00	
71	171 00	
72	172 00	
73	173 00	
74	174 00	
75	175 00	
76	176 00	
77	177 00	
78	178 00	
79	179 00	
80	180 00	
81	181 00	
82	182 00	
83	183 00	
84	184 00	
85	185 00	
86	186 00	
87	187 00	
88	188 00	
89	189 00	
90	190 00	
91	191 00	
92	192 00	
93	193 00	
94	194 00	
95	195 00	
96	196 00	
97	197 00	
98	198 00	
99	199 00	
00	200 00	
01	201 00	
02	202 00	
03	203 00	
04	204 00	
05	205 00	
06	206 00	
07	207 00	
08	208 00	
09	209 00	
10	210 00	
11	211 00	
12	212 00	
13	213 00	
14	214 00	
15	215 00	
16	216 00	
17	217 00	
18	218 00	
19	219 00	
20	220 00	
21	221 00	
22	222 00	
23	223 00	
24	224 00	
25	225 00	
26	226 00	
27	227 00	
28	228 00	
29	229 00	
30	230 00	
31	231 00	
32	232 00	
33	233 00	
34	234 00	
35	235 00	
36	236 00	
37	237 00	
38	238 00	
39	239 00	
40	240 00	
41	241 00	
42	242 00	
43	243 00	
44	244 00	
45	245 00	
46	246 00	
47	247 00	
48	248 00	
49	249 00	
50	250 00	
51	251 00	
52	252 00	
53	253 00	
54	254 00	
55	255 00	
56	256 00	
57	257 00	
58	258 00	
59	259 00	
60	260 00	
61	261 00	
62	262 00	
63	263 00	
64	264 00	
65	265 00	
66	266 00	
67	267 00	
68	268 00	
69	269 00	
70	270 00	
71	271 00	
72	272 00	
73	273 00	
74	274 00	
75	275 00	
76	276 00	
77	277 00	
78	278 00	
79	279 00	
80	280 00	
81	281 00	
82	282 00	
83	283 00	
84	284 00	
85	285 00	
86	286 00	
87	287 00	
88	288 00	
89	289 00	
90	290 00	
91	291 00	
92	292 00	
93	293 00	
94	294 00	
95	295 00	
96	296 00	
97	297 00	
98	298 00	
99	299 00	
00	300 00	
01	301 00	
02	302 00	
03	303 00	
04	304 00	
05	305 00	
06	306 00	
07	307 00	
08	308 00	
09	309 00	
10	310 00	
11	311 00	
12	312 00	
13	313 00	
14	314 00	
15	315 00	
16	316 00	
17	317 00	
18	318 00	
19	319 00	
20	320 00	
21	321 00	
22	322 00	
23	323 00	
24	324 00	
25	325 00	
26	326 00	
27	327 00	
28	328 00	
29	329 00	
30	330 00	
31	331 00	
32	332 00	
33	333 00	
34	334 00	
35	335 00	
36	336 00	
37	337 00	
38	338 00	
39	339 00	
40	340 00	
41	341 00	
42	342 00	
43	343 00	
44	344 00	
45	345 00	
46	346 00	
47	347 00	
48	348 00	
49	349 00	
50	350 00	
51	351 00	
52	352 00	
53	353 00	
54	354 00	
55	355 00	
56	356 00	
57	357 00	
58	358 00	
59	359 00	
60	360 00	
61	361 00	
62	362 00	
63	363 00	
64	364 00	
65	365 00	
66	366 00	
67	367 00	
68	368 00	
69	369 00	
70	370 00	
71	371 00	
72	372 00	
73	373 00	
74	374 00	
75	375 00	
76	376 00	
77	377 00	
78	378 00	
79	379 00	
80	380 00	
81	381 00	
82	382 00	
83	383 00	
84	384 00	
85	385 00	
86	386 00	
87	387 00	
88	388 00	
89	389 00	
90	390 00	
91	391 00	
92	392 00	
93	393 00	
94	394 00	
95	395 00	
96	396 00	
97	397 00	
98	398 00	
99	399 00	
00	400 00	
01	401 00	
02	402 00	
03	403 00	
04	404 00	
05	405 00	
06	406 00	
07	407 00	
08	408 00	

# Basic Functions Through Beginning Programming

PERSONAL PROGRAMMING  
I-3 III-1-3 E-I-III-C-1

## 11. SOLID STATE SOFTWARE LIBRARY PROGRAMS

*Solid State Software* library programs are available to provide you with easy instant access to powerful program solutions, even if you don't have extensive knowledge of the math or programming involved. All you need to do is to identify the type of problem you want to work in the appropriate library manual, and follow the keystoke instructions spelled out for you.

You can solve the problems in this section with the *Master Library* and module received with your calculator.

**Example 11-1:** You have decided to invest \$1500 in a savings account paying 6% interest, compounded annually. Assuming no withdrawals, how much money will you have in 5 years?

*Notes:*

1. This is a compound interest problem.
2. Program 18 on page 60 of the *Master Library* manual will solve the problem.
3. Detailed user instructions are on page 61 of the library manual.

Solution:

Press	Display	Comments
[CP] *	0.	Clear program memory
[Pgm] * 18	0.	Select program
[ E' ]	0.00	Initialize program
5 [ A ]	5.00	Enter n
6 [ B ]	6.00	Enter %i
1500 [ C ]	1500.00	Enter PV
0 [ D ]	2007.34	Calculate FV

*Note:* When you are using a *Solid State Software* program, you will find it very helpful to insert the plastic program label card in the window above the user-defined label keys.

**Exercise 11-1:** You want to know how much you need to invest today in a savings account paying 5½% interest compounded annually in order to have \$10,000 at the end of 10 years.

**Hint:** This is a compound interest problem. You know the future value and are looking for the present value.

Solution:

\*Denotes second function key.

I Basic Functions Through Beginning Programming

---

## PERSONAL PROGRAMMING

1-3 III-1-3

2MAB009

**Exercise 11-2:** You are considering buying a house. You anticipate financing \$50,000 for 30 years at 9%. What will be your monthly payment (for principal and interest)?

**Hint:**

1. This is an annuity problem.
  2. Convert your interest and number of periods to monthly form.

**Solution:**

## 1. VARIABLE VALUE ENTRY

You can enter variable values into programs in two ways:

- (1) Through the display
- (2) Recall them from data memories.

**Example 1-1:** You have decided to place \$1500 in a savings account that pays 6% interest compounded annually. Assuming no withdrawals, how much money will you have in 5 years?

Mathematical expression:

$$FV = PV \times (1 + i)^n$$

### ENTER VARIABLE VALUES THROUGH DISPLAY

#### Program Development

1. List instructions that will not change
2. Place imaginary box where variable values are needed
3. Place [R/S] instruction just before variable value is needed
4. Number instructions

**Exercise 1-1:** Your first child was just born. You want to invest an amount that will provide at least \$10,000 for college 18 years from now. You can get 6½% interest, compounded annually. How much do you have to invest now on a one-time basis? Program so variable values are entered through the display.

Mathematical expression:

$$PV = FV \div (1 + i)^n$$

Program Entry

1. Enter LEARN mode — press [LRN]

2. Key in program instructions

3. Exit LEARN mode — press [LRN]

# Intermediate Programming

## PERSONAL PROGRAMMING

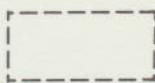
I-4 II-1-4

II-2-II

### Keystroke Solution

### Program Instructions

1500



[ X ]

000 [ X ]

[ ( ]

001 [ ( ]

1

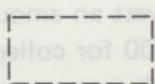
002 1

[ + ]

003 [ + ]

.06

004 [R/S]



[ ) ]

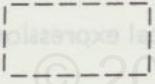
005 [ ) ]

[y<sup>X</sup>]

006 [y<sup>X</sup>]

007 [R/S]

5



[ = ]

008 [ = ]

009 [R/S]

### Program Entry

1. Enter LEARN mode – press [LRN]
2. Key in program instructions
3. Exit LEARN mode – press [LRN]

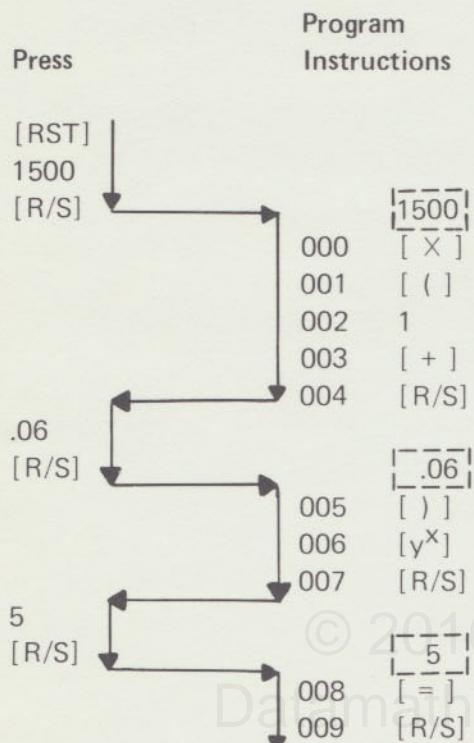
# Intermediate Programming

II

PERSONAL PROGRAMMING  
IV-8, 9

14.8.9

## Program Execution



## ENTER VARIABLE VALUES USING DATA MEMORIES

### Program Development

1. Designate specific data memory for each variable value
2. List instructions that will not change
3. In place of variable value, substitute instruction to recall value from designated memory
4. Add [R/S] at end
5. Number instructions

## Program Execution

Program Instructions	Registers	Solution
000 [RCR]	b1 000	200
001 100		
002 [X]	005	1 X 1
003 [1]	000	1 1
004 [+]	004	+
005 [RCR]	000	00.
006 05	001	
007 [1]	800	{ }
008 [y <sup>x</sup> ]	000	[y <sup>x</sup> ]
009 [RCR]	a	a
010 03	011	
011 [=]	012	=
012 [RS]	013	
013 [RS]		

Exercise 1-2: Solve same problem as Exercise 1-1 but use data memories for variable entry.

Keystroke Solution	Program Instructions
1500	000 [RCL] { PV
-	001 01 }
[ X ]	002 [ X ]
[ ( ]	003 [ ( ]
1	004 1
[ + ]	005 [ + ]
.06	006 [RCL] { i
	007 02 }
[ ) ]	008 [ ) ]
[y <sup>x</sup> ]	009 [y <sup>x</sup> ]
5	010 [RCL] { n
	011 03 }
[ = ]	012 [ = ]
	013 [R/S]

## Program Entry

1. Enter LEARN mode – press [LRN]
  2. Key in program instructions
  3. Exit LEARN mode – press [LRN]

© 2010 Joerg Woerner  
press [LRN] 

# Intermediate Programming

II

PERSONAL PROGRAMMING  
IV-8, 9

## Program Execution

5. LABELED

Press

1500 [STO] 01  
.06 [STO] 02  
5 [STO] 03  
[RST]  
[R/S]

refer to Program Execution

000 [RCL]  
001 01  
002 [ X ]  
003 [ ( ]  
004 1  
005 [ + ]  
006 [RCL]  
007 02  
008 [ ) ]  
009 [y<sup>X</sup>]  
010 [RCL]  
011 03  
012 [ = ]  
013 [R/S]

(Answer = 2007.338366)

USER-DEFINED LABELS

User-defined labels

labeled labels (refers to as common labels in RPN programming)

USER-DEFINED LABELS

Example 5-7: Refer to the sample code

blocks from Section II-1, which contains the following code:  
blocks instruction with point D. Then, the program continues at label E.  
as follows:  
storing to block step 012.

Next, follow instructions to first you can enter  
the assembly by pressing a "user-defined key".  
and key in these instructions starting at step 000.  
This means you will include the  
instruction, as part of your program.

What are the names of the variables in the  
code? blocks on a numeric calculator. And finally,  
execute the program.

# Intermediate Programming

PERSONAL PROGRAMMING

IV-11-14

0.8-VI

## 2. LABELS

Labels are used to establish reference points in a program. There are two types of labels:

User-defined labels

Program labels (referred to as common labels in *Personal Programming*).

### USER-DEFINED LABELS

User-defined labels are associated with user-defined keys: A, B, C, D, E, and A' through E'. When a user-defined key is pressed, the program pointer returns to step 000, searches for the corresponding label, and the program automatically starts executing at the first instruction following the label.

**Example 2-1:** Refer to the savings account problem from Section II-1 and rewrite the program instructions with its beginning "labeled" as reference point D. Then key in the program starting at program step 015.

Next, develop instructions so that you can enter the variables by pressing a "user-defined key" and key in these instructions starting at step 000. This means you will include the "store instructions" as part of your program.

Write the names of the variable values in the correct blocks on a magnetic card. And finally, execute the program.

**Exercise 2-1:** Program the problem in Exercise 1-1 so that

FV is entered on [ D ]

i is entered on [ B ]

n is entered on [ C ]

and PV is calculated by pressing [ A ].

Record your program and appropriately label the magnetic card.

# Intermediate Programming

PERSONAL PROGRAMMING  
IV-11-14

## Example 2-1: (continued)

Solution:

### Program Instructions

Location	Keystroke	Description
015	[Lbl] *	Establish reference point
016	{ [D]	
017	[RCL]	
018	01	
019	[X]	
020	[ ( ]	
021	1	
022	[ + ]	
023	[RCL]	
024	02	
025	[ ) ]	
026	[y <sup>x</sup> ]	
027	[RCL]	
028	03	
029	[ = ]	
030	[R/S]	

### Program Entry

1. Press [GTO] 015
2. Enter LEARN mode – press [LRN]
3. Key in program instructions
4. Exit LEARN mode – press [LRN]

The following sequence of instructions will allow you to enter a variable value and press a “user-defined key” instead of manually storing variable values from the keyboard.

## Exercise 2-1: (continued)

Location	Keystroke	Description
000	* [P1]	
001	[ A ]	
002	[STO]	
003	[ 0 ]	
004	[R/S]	
005	* [P1]	
006	[ 8 ]	
007	[STO]	
008	[ 5 ]	
009	[R/S]	
010	* [P1]	
011	( [C]	
012	[STO]	
013	[ 3 ]	
014	[R/S]	

\*Denotes second function key.

## Program Instructions

(beginning) Example 5-1:

Location	Keystroke
000	[Lbl] *
001	[ A ]
002	[STO]
003	01
004	[R/S]
005	[Lbl] *
006	[ B ]
007	[STO]
008	02
009	[R/S]
010	[Lbl] *
011	[ C ]
012	[STO]
013	03
014	[R/S]

## Program Entry

1. Press [RST]
2. Enter LEARN mode – press [LRN]
3. Key in program instructions
4. Exit LEARN mode – press [LRN]

© 2010 Joerg Woerder  
Datamath Calculator Museum

\* Denotes second function key.

# Intermediate Programming

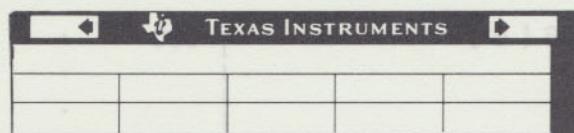
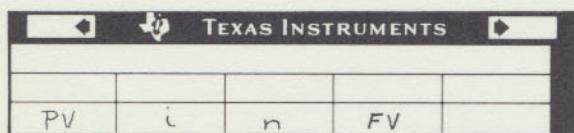
II

PERSONAL PROGRAMMING  
IV-11-14 VII-2

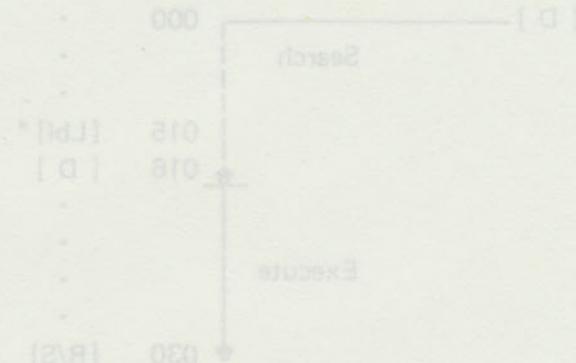
Variable	Assigned Data Memory	Assigned User-Defined Key
PV	01	[ A ]
i	02	[ B ]
n	03	[ C ]
FV	-	[ D ]

Variable	Assigned Data Memory	Assigned User-Defined Key
PV	000	[ A ]
i	100	[ B ]
n	200	[ C ]
FV	300	[ D ]

## MAGNETIC CARD



© 2010 Joeig Woerner  
Datamath Calculator Museum

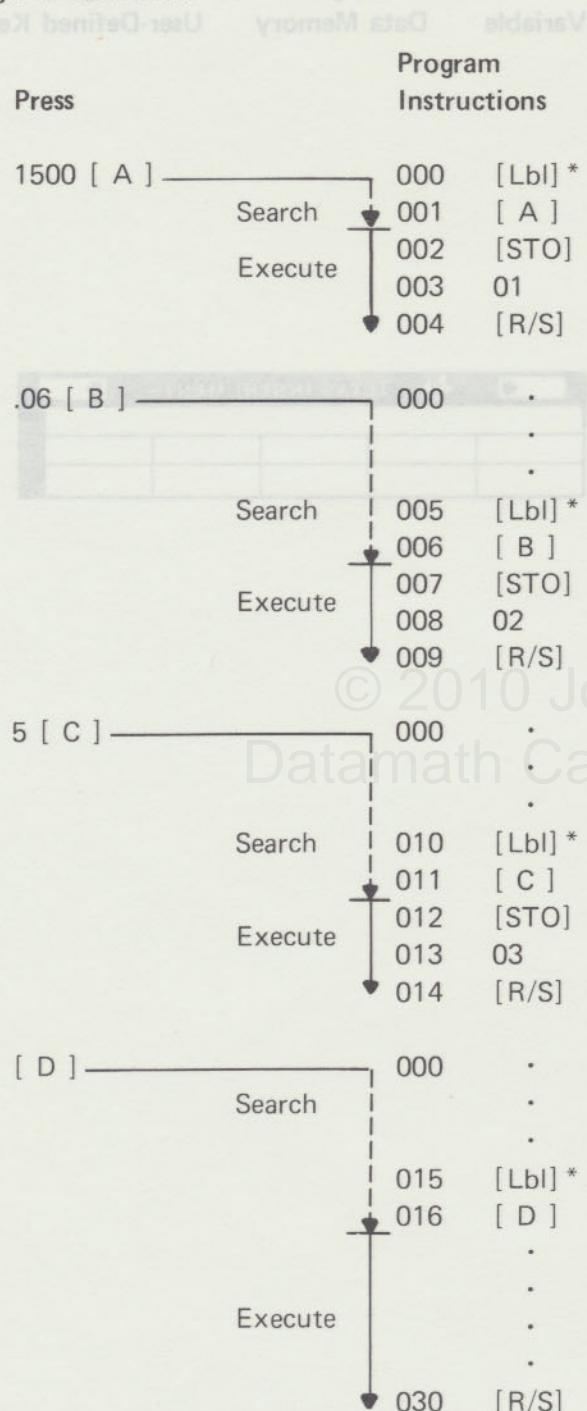


Answer = 500.3288

© 2010 Joeig Woerner All rights reserved

PERSONAL PROGRAMMING  
IV-11-14 VII-2 V-11-11 VI

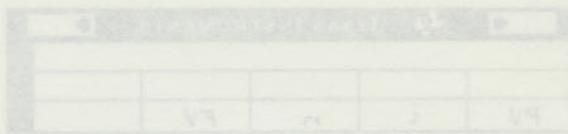
## Program Execution



(Answer = 2007.338366)

banglaA	banglaB	A Bangla	Asian
User-Defined Key	User-Defined Key	User-Defined Key	User-Defined Key
[ A ]		10	V9
[ B ]		50	1
[ C ]		ED	a
[ D ]		-	V7

MAGNETIC CARD



© 2010 Joerg Woerner

Datamath Calculator Museum

\*Denotes second function key.

## PROGRAM LABELS

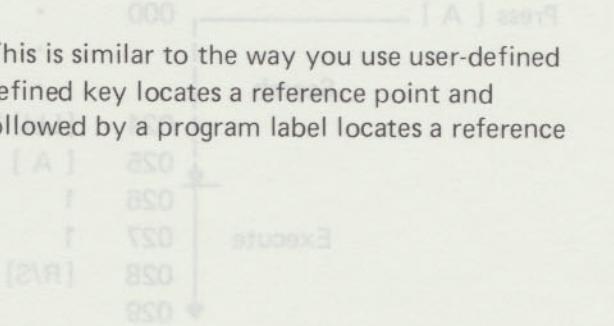
Program Execution

Program labels are also referred to as "common" labels in *Personal Programming*. Only the following keys cannot be used as labels: [2nd], [LRN], [Ins] \*, [Del] \*, [SST], [BST], [Ind] \* and the numbers 0-9. You should also avoid using [R/S] as a label because of its ability to start program execution.

You can assign program labels to create reference points. This is similar to the way you use user-defined keys, but it has one important difference. Pressing a user-defined key locates a reference point and automatically starts program execution. Pressing [GTO] followed by a program label locates a reference point, but does *not* execute.

### Program Entry

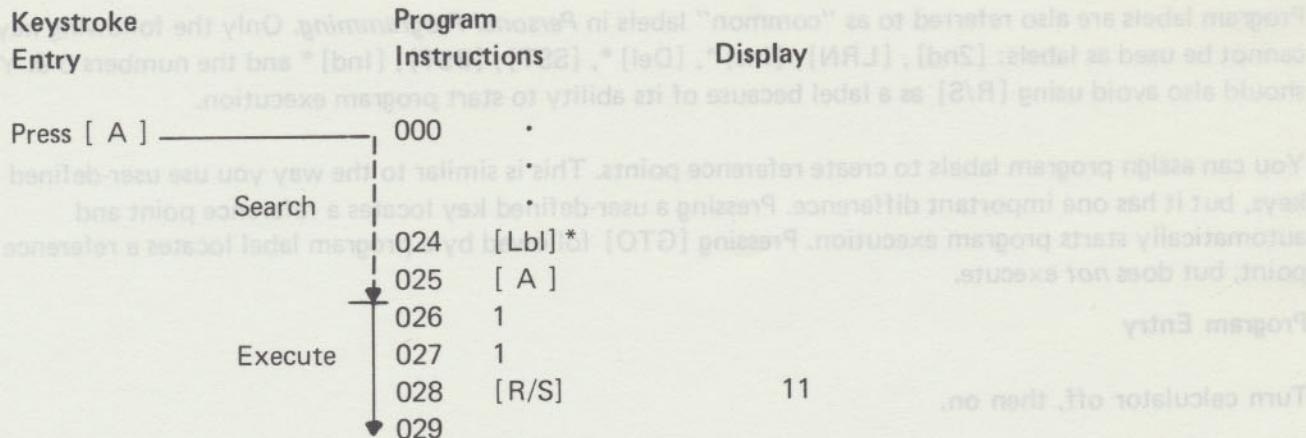
Turn calculator off, then on.



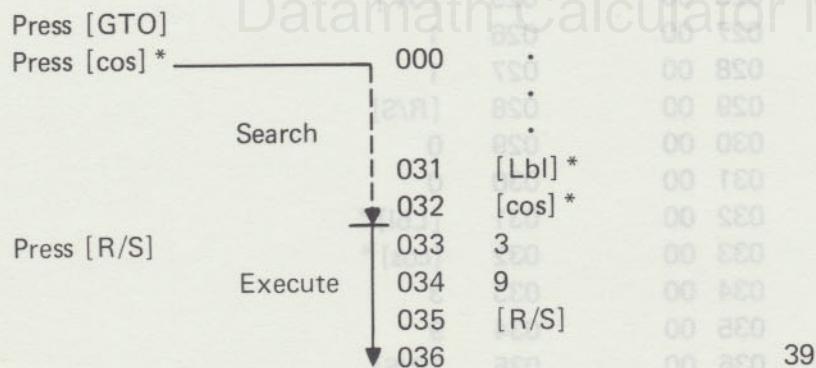
Press	Display	Program Instructions
[GTO] 024	0.	
[LRN]	024 00	
[Lbl] *	025 00	024 [Lbl] *
[ A ]	026 00	025 [ A ]
1	027 00	026 1
1	028 00	027 1
[R/S]	029 00	028 [R/S]
[SST]	030 00	029 0
[SST]	031 00	030 0
[Lbl] *	032 00	031 [Lbl] *
[cos] *	033 00	032 [cos] *
3	034 00	033 3
9	035 00	034 9
[R/S]	036 00	035 [R/S]
[LRN]	0.	

\*Denotes second function key.

## Program Execution



Enter LEARN mode. Verify that the pointer is at step 029. The calculator located label A, began execution automatically at step 026 and continued until it encountered the [R/S] instruction. Exit LEARN mode.



Enter LEARN mode. Verify pointer is at step 033. The calculator located label [cos] \* but did not begin execution. Exit LEARN mode. Press [R/S], display shows 39. Program halts at 035. Verify program counter is now at location 036 by pressing [LRN], then press it again.

\*Denotes second function key.

### 3. UNCONDITIONAL BRANCHING

Unconditional branching instructions are program instructions which cause the program pointer to relocate to another program step and commence sequential execution from the new location. Unconditional branching instructions are:

[RST] Reset  
[GTO] Go To  
[SBR] Subroutine

**Example 3-1:** Write a program that will count by 5.

Write it so that when the program is stopped:

- (a) Counting will start at 5 when the keyboard sequence [RST] [R/S] is pressed, and
  - (b) Counting will continue from last displayed number when the keyboard sequence [GTO] [cos] \* [R/S] is pressed (or when [SBR] [cos] \* is pressed).

**Solution:**

## **Program Instructions**

Enter LEARN mode – press [LRN]

```

000    0
001    [Lbl] *
002    [cos] *
003    [ + ]
004    5
005    [ = ]
006    [Pause]
007    [GTO]
008    [cos] *

```

Exit LEARN mode – press [LRN]

\*Denotes second function key.

PERSONAL PROGRAMMING  
IV-43-51 V-55-60

**Example 3-2:** Write a program that will count starting at 3, then add 2, then 3, then 2, etc.

Use a subroutine in this program. The program should run when you press [RST] [R/S].

Solution:

#### Program Instructions

Enter LEARN mode — press [LRN]

```

000  0
001  [Lbl] *
002  [cos] *
003  [ + ]
004  3
005  [ = ]
006  [Pause] *
007  [SBR] } Subroutine
008  [sin] *   call
009  [GTO]
010  [cos] *
011  [Lbl] *
012  [sin] *
013  [ + ]
014  2         } Subroutine
015  [ = ]      instructions
016  [Pause] *
017  [INV] [SBR]

```

Exit LEARN mode — press [LRN]

**Example 3-3:** Modify above program so

- It will count by three when [RST] [R/S] is pressed
- It will add two to whatever value is in the display and stop when [GTO] [sin] \* [R/S] or [SBR] [sin] \* is pressed.

*Note:* Observe that [INV] [SBR] acts as a stop in (b).

Solution:

Delete the subroutine call in previous problem (i.e., delete [SBR], delete [sin] \*).

\*Denotes second function key.

#### 3. UNCONDITIONAL BRANCHING

Unconditional branching instructions are to simpler programs what goto statements are to programming languages like C.

[RST] Reset

[GTO] Go To

[SBR] Subroutine

**Example 3-2:** Write a program that will count by 2 starting at 3 until the display is stopped.

(a) Counting will start at 3 when the key sequence [R/S] [cos] \* [R/S] is pressed, and stop when [GTO] [cos] \* [R/S] is pressed for when [R/S] [cos] \* [R/S] is pressed.

```

Enter LEARN mode — press [LRN]
000  0
001  [Lbl] *
002  [cos] *
003  [ + ]
004  2
005  [ = ]
006  [Pause] *
007  [GTO]
008  [cos] *

```

Exit LEARN mode — press [LRN]

Deletes unnecessary instruction part

**Example 3-4:** You want to investigate the two situations below. Write one program that uses subroutines and one without subroutines to assist you. Notice the savings in program steps the subroutine provides.

1. If you invest \$2000 in a savings account paying 6% compounded annually, how much will you have in 5 years?

$$FV = PV \times (1 + i)^n$$

2. How much would you have to invest today in a savings account paying 6% compounded annually in order to have \$5000 in 5 years?

$$PV = FV \div (1 + i)^n$$

Write the program so that you store the variables in data memories. Store PV in data memory 01, i in 02, n in 03, and FV in 04 by entering them into memory from the keyboard. Then press [ E ] to calculate FV and [ E' ]\* to calculate PV.

**Exercise 3-1:** You want to finance a car through your credit union. The car will require \$5200 financing. The credit union has a 10.8% APR interest rate and will finance for 42 months. How much will your monthly payments be?

Mathematical expression:

$$PMT = PV \div ((1 - (1 + i)^{-n}) \div i)$$

Notice the interest (i) appears twice in the expression. Write your program so that you can enter i as an annual rate (as a percent) and the program will convert it to a monthly decimal rate.

Make the interest conversion to a monthly decimal value a subroutine. Input the variables with user-defined keys: PV with [ A ], i with [ B ], n with [ C ], and press [ D ] to calculate PMT.

\*Denotes second function key.

# Intermediate Programming

PERSONAL PROGRAMMING  
IV-43-51 V-55-60

**Solution:** To calculate  $V_A = V_R \times (1 + \frac{R_A}{R_R})$

## Program Development and Entry

Program Instructions without Subroutine	Program Instructions with Subroutine
Enter LEARN mode	Enter LEARN mode
Press [LRN]	Press [LRN]
000 [Lbl] *	000 [Lbl] *
[ E ]	[ E ]
[RCL]	[RCL]
01 [ X ]	01 [ X ]
005 [ ( ] 005 [SBR]	[cos] *
1 [ + ] [R/S]	[R/S]
[RCL]	[Lbl] *
02 [ E' ] *	[ E' ] *
010 [ ) ] 010 [RCL]	[RCL]
[y <sup>x</sup> ]	04 [ ÷ ]
[RCL]	[RCL]
03 [ = ] [cos] *	[cos] *
015 [R/S] 015 [ ( ]	[ ( ]
016 [Lbl] *	1 [ + ]
[ E' ] *	[RCL]
[RCL]	02 [RCL]
04	02
020 [ ÷ ] 020 [ ) ]	[ ) ]
[ ( ] [y <sup>x</sup> ]	[y <sup>x</sup> ]
1 [RCL]	[RCL]
[ + ]	03 [ = ]
[RCL]	[RCL]
025 02 [INV] [SBR]	025 [INV] [SBR]
[ ) ] Exit LEARN mode	Exit LEARN mode
[y <sup>x</sup> ]	Press [LRN]
[RCL]	
03	
[ = ]	
031 [R/S]	
Exit LEARN mode	
Press [LRN]	

\*Denotes second function key.

**Solution:** To calculate  $V_A = V_R \times (1 + \frac{R_A}{R_R})$

$$V_A = V_R \times (1 + \frac{R_A}{R_R})$$

How much would you have to invest  
today in savings account paying 6%  
compounded monthly in order to have  
\$10000 in 5 years?

$$(1 + 0.06)^5 = 1.33$$

What is the present value of a  
mortgage of \$100,000 at 6%  
interest rate over 30 years?  
Type  $(1 - e^{-0.06 \times 30}) / 0.06$  to calculate PV.

$$(1 - e^{-0.06 \times 30}) / 0.06$$

To calculate PV.

# Intermediate Programming

II

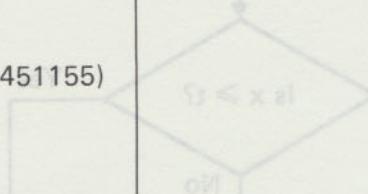
PERSONAL PROGRAMMING  
IV-53-51 V-55-60

## Program Execution (both programs)

Press

(x) Condition testing instructions require the position pointer to either of two locations to evaluate a test combining two values. The display value (x) is combined with either value (t)

1. 2000 [STO] 01  
.06 [STO] 02  
5 [STO] 03  
[ E ] (answer = 2676.451155)
2. .06 [STO] 02  
5 [STO] 03  
5000 [STO] 04  
[ E' ] \* (answer = 3736.290864)



Example #1: Press 6 in calculator

Example #2: Press 6 in calculator

Solution:

Press 6 (x &lt; t)

Model: TI-83 just has the condition testing instruction `IF`. No `THEN` or `END` is needed. The display value will be stored in `t`.

© 2010 Joerg Woerner  
Datamath Calculator Museum

Exercise #5: From the keyboard combine 200 with the 512 placed in the x-register above. If 203 is less than x, press 0 to display zero. If 203

hints: Review page 1A-00 of owner's manual for other decision-making logic for combinations.

Example #5: From the keyboard combine 200 with the space register above (B). If 6 is greater than or equal to x, press 153.

Solution:

Press 6 (x &lt; t = 153)

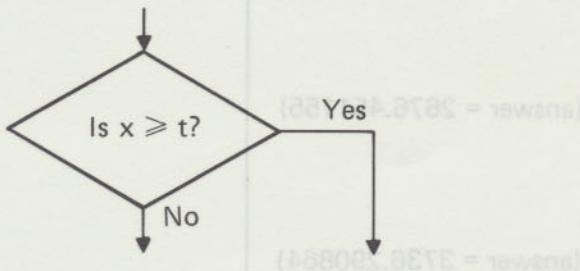
Enter FEEARN name or vanity number (Press [PRN] — display = 153 00)

\*Denotes second function key.

\*Designates second function key

## 4. CONDITIONAL BRANCHING — DECISION MAKING BY COMPARISON

Conditional branching instructions relocate the program pointer to *either of two locations based on result of a test comparing two values*. The display value (x) is compared to the test register value (t).



**Example 4-1:** Place 5 in t-register.

Solution:

Press 5 [ $x \geq t$ ]

*Note:* The [ $x \geq t$ ] key acts like the data memory [Exc] \* key except it only exchanges the display value with the t-register.

**Example 4-2:** From the keyboard compare 6 with the above t-register value (5). If 6 is *greater than or equal* to t, branch to program step 123.

Solution:

Press 6 [ $x \geq t$ ] \* 123

Enter LEARN mode to verify transfer (press [LRN] – display = 123 00).

**Exercise 4-1:** Place 215 in t-register.

**Exercise 4-2:** From the keyboard compare 203 with the 215 placed in the t-register above. If 203 is *less than* t, branch to program step 072.

**Hint:** Review page IV-60 of owner's manual for other decision-making tests for comparisons.

\*Denotes second function key.

**Example 4-3:** A customer receives a 2% discount if his order is \$100 or more. What is his invoice amount if he orders 20 items priced at \$7.25 each?

- (1) Let your program multiply numbers of items times the unit price to get the invoice amount. This amount should be compared to the break price for a discount.
- (2) Display the order price for 2½ seconds before making the comparison.

## Program Development

```
000 [Lbl] *  
    [ A ]  
    [x ≥ t]  
003 [R/S]  
004 [Lbl] *  
    [ B ]  
    [STO]  
01  
008 [R/S]  
009 [Lbl] *  
    [ C ]  
    [STO]  
02  
013 [R/S]  
014 [Lbl] *  
    [ D ]  
    [STO]  
03  
018 [R/S]  
019 [Lbl] *  
    [ E ]  
    [RCL]  
02  
    [ X ]  
    [RCL]  
03  
    [ = ]
```

**Exercise 4-3:** A customer receives a 2% discount if his order is over \$100 and a 3% discount if it is over \$250. What is his invoice amount if he

- (1) orders 30 items at \$7.25?

- (2) orders 40 items at \$7.25?

Flash "ORDER AMOUNT BEFORE  
DISCOUNT."

## Program Development

```
0  
30  
038 [R/S]  
[ X ] 030  
[ ]  
[ ]  
[ - ]  
[ RCL ]  
01  
01  
[ ]  
[ ]  
[ = ]  
044 [R/S]  
Program End
```

© 2010 Joerg Woerner

Datamath Calculator Museum

\*Denotes second function key.

Denotes second function key.

# Intermediate Programming

## PERSONAL PROGRAMMING

IV-57-60 V-62, 63

### Example 4-3: (continued)

#### Program Development

```
034      [R/S]          Program Development
035      [X]
036      [(]
037      1
038      [-]
039      [RCL]
040      01
041      [)]
042      [=]
043      [R/S]
```

Exercise 4-3: A customer places a \$24.95 discount on his order of over \$100 and a \$10.00 off order if it is over \$250. When he runs his program he gets a discount of \$10.00 and a \$10.00 discount.

Exercise 4-3: A customer places a \$24.95 discount on his order of over \$100 and a \$10.00 off order if it is over \$250. When he runs his program he gets a discount of \$10.00 and a \$10.00 discount.

Program Development

*	[T/P]	000
T	[A]	
S	[x]	
x	[R/S]	000
*	[T/P]	000
I	B	
E	[S/TD]	
R	0	
A	[R/S]	800
*	[T/P]	800
C	[C]	
E	[S/TD]	
S	0	
R	[R/S]	000
*	[T/P]	000
D	I	
O	[S/TD]	
T	0	
E	[R/S]	800
*	[T/P]	800
E	I	
R	[S/TD]	
S	0	
X	[R/S]	000
*	[T/P]	000
R	C	
O	[R/S]	000
T	I	

#### Program Entry

1. Enter LEARN mode — press [LRN]
2. Key in instructions
3. Exit LEARN mode — press [LRN]

\*Denotes second function key.

\*Denotes second function key.

# Intermediate Programming

II

PERSONAL PROGRAMMING  
IV-57-60 V-62, VI

Program Execution

TEXAS INSTRUMENTS				
Break	Disc	Units \$	Qty	Invo Amt

Program Execution

TEXAS INSTRUMENTS				
Break	Disc	Units \$	Qty	Invo Amt

Press

100 [ A ]  
.02 [ B ]  
7.25 [ C ]  
20 [ D ]

} Enter variables

[ E ] – Calculate invoice amount

145. Flashes (Amount  
Without Discount)

142.1 (answer)  
(Discounted Invoice)

Leave this program in memory for next section.

Exercises 5-1: From the keyboard  
Enter field B if it is set, press key to  
begin step 51.

Solutions

TEXAS INSTRUMENTS				
Break	Disc	Units \$	Qty	Invo Amt

Program Execution

TEXAS INSTRUMENTS				
Break	Disc	Units \$	Qty	Invo Amt

Exercises 5-1: From the keyboard

Enter field A if it is set, press key to

begin step 51.

Solutions

TEXAS INSTRUMENTS				
Break	Disc	Units \$	Qty	Invo Amt

In LEARN mode set A pointer move

to step 318

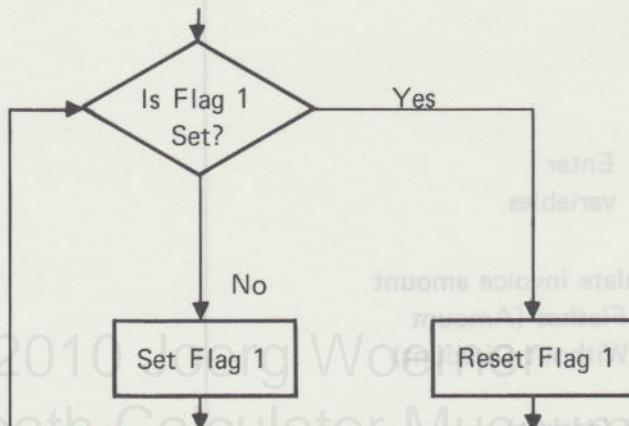
((LRN) – display = 318 00)

\* Shows working function key

## 5. CONDITIONAL BRANCHING — DECISION MAKING WITH FLAGS

You can use branching instructions to relocate the program pointer to either of two locations based on testing the condition of a "flag". A "flag" is simply a switch. It is either "on" or "off". In calculator terminology, it is "set" or "not set".

When writing your program, you must specify the number of the flag to be "set" or "not set" as well as the flag to be tested.



### Example 5-1: From the keyboard

- (1) Set flag 4
- (2) Test flag 4; if it is set, branch to program step 379.

Solution:

Press

1. [St flg] \*
- 4
2. [If flg] \*
- 4
- 379

In LEARN mode verify pointer moved to step 379  
([LRN] – display = 379 00)

### Exercise 5-1: From the keyboard

- (1) Set flag 5
- (2) Test flag 5; if it is set, branch to program step 211.

Solution:

\* Denotes second function key.

# Intermediate Programming

II

PERSONAL PROGRAMMING

IV-61-67

18-18-VI

**Example 5-2:** From the keyboard

- (1) "Reset" flag 4 to "not set" condition
- (2) Test flag 4 under these two conditions:
  - (a) If it is "set," branch to program step 125
  - (b) If it is "not set," branch to program step 125.

Solution:

- Press
1. [INV][V] [St flg] \* 4
  2. [If flg] \* 4 125

Pointer did not move from location of previous example. Flag 4 was not set when tested; branching did not occur. Verify in LEARN mode.

([LRN] — display = 379 00)

- Press
3. [INV]  
[If flg] \*  
4  
125

Instruction was to branch if flag is "not set". Pointer moved to step 125.

Verify in LEARN mode.

([LRN] — display = 125 00)

\*Denotes second function key.

**Exercise 5-2:** From the keyboard

- (1) "Reset" flag 5 to "not set" condition
- (2) Test flag 5; if it is "not set," branch to program step 227.

Solution:

Press

1. [INV][V] [St flg] \* 5
2. [If flg] \* 227

Your store gives a 10% discount on all orders over \$100. However, certain customers get a 20% discount on all orders over \$200. A store customer gets a 25% discount on all purchases over \$300. Write this as a program.

# Intermediate Programming

PERSONAL PROGRAMMING  
IV-61-67

**Example 5-3:** Modify the instructions of the discount problem of preceding section (Section II.4) to solve the following problem.

Your store gives a 2% discount on all orders over \$100. However, catalog customers get a 5% discount on all orders over \$100. A store customer and a catalog customer each purchase five items at \$8.98 each. What should they be invoiced?

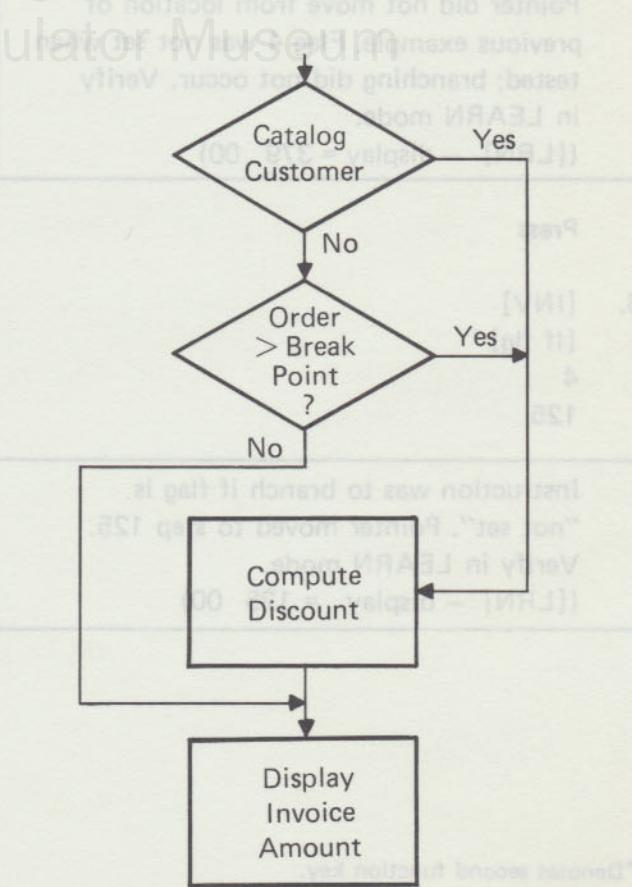
**Exercise 5-3:** Your store gives a 2% discount on all orders over \$100, unless it is a catalog customer, in which case he gets a 2% discount regardless of order size. A store customer and a catalog customer each purchase five items at \$8.98 each. What should they be invoiced?

Use a flag to designate catalog customer.

*Note:* You should make sure the flag used is reset somewhere after it is tested. This prevents the next customer to be invoiced from getting a catalog discount if he is not entitled to it.

Compute the amount of purchase. Flash "ORDER AMOUNT WITHOUT DISCOUNT."

**Hint:** The flow diagram below may be helpful.



## Intermediate Programming

11

---

## PERSONAL PROGRAMMING

### IV-61-67

Program Development		Exercise B-3: (continued)	
Previous Instructions	Changes or Additions	Program Development	Program Development
000 [Lbl] *			
[ A ]			
[x ≥ t]			
003 [R/S]			
004 [Lbl] *			
[ B ]			
[STO]			
01			
008 [R/S]			
009 [Lbl] *			
[ C ]			
[STO]			
02			
013 [R/S]			
014 [Lbl] *			
[ D ]			
[STO]			
03			
018 [R/S]			
019 [Lbl] *			
[ E ]			
[RCL]			
02			
[ X ]			
[RC L]			
03			
[ = ]			
[Pause] *			
031 [Pause] *		[If flg] *	
[x ≥ t] * insert 1			
[x <sup>2</sup> ] *		[x <sup>2</sup> ]	
[R/S]			

\*Denotes second function key.

### Example 5-3: (continued)

Previous		Changes or Additions
<b>Instructions</b>		
038	[Lbl] *	
	[x <sup>2</sup> ]	
040	[ x ]	
	[ ( ]	
	1	
	[ - ]	
	[RCL]	
	01	
	[ ) ]	
	[ = ]	
048	[R/S]	[INV] [St flg] *
		1
		[R/S]
		[Lbl]
		[ A' ] *
		[St flg] *
		1
056		.
		0
		5
		[STO]
		01
061		[R/S]

### Exercise 5-3: (continued)

\*Denotes second function key.

## Intermediate Programming

11

---

PERSONAL PROGRAMMING  
IV-61-67

## Program Entry

Press [GTO] 044  
Enter LEARN mode – press [LRN]  
[INV]  
[St flg] \*  
1  
[R/S]  
[Lbl] \*  
[ A' ] \*  
[St flg] \*  
1  
. .  
0  
5  
[STO]  
01  
[R/S]  
Exit LEARN mode – press [LRN]  
Press [GTO] 032  
Enter LEARN mode – press [LRN]  
[Ins] \*  
[Ins] \*  
[Ins] \*  
[Ins] \*  
[If flg] \*  
1  
[x<sup>2</sup>]  
[x ≥ t] \*  
[x<sup>2</sup>]  
[R/S]  
[Lbl] \*  
[x<sup>2</sup>]  
Exit LEARN mode – press [LRN]

\*Denotes second function key.

# Intermediate Programming

PERSONAL PROGRAMMING  
IV-61-67 TB-TS-VI

## Program Execution

TEXAS INSTRUMENTS				
CATALOG				
Break	Disc	Unit \$	Qty	Inv Amt

Press

100 [ A ]  
.02 [ B ]  
7.25 [ C ]  
20 [ D ]

To calculate store customer invoice price

[ E ] (Answer = 142.10)

To calculate catalog customer invoice price

[ A' ] \*  
[ E ] (Answer = 137.75)

## Program Execution

TEXAS INSTRUMENTS				
CATALOG				
Break	Disc	Unit \$	Qty	Inv Amt

Invoice amount for store customer:

Press                          Answer

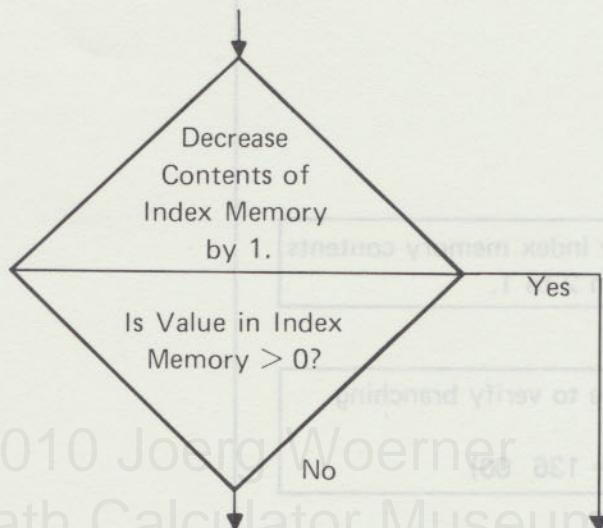
Invoice amount for catalog customer:

Press                          Answer

\*Denotes second function key.

## 6. CONDITIONAL BRANCHING — DECISION MAKING FOR LOOPING (DSZ)

The “decrement and skip on zero” instruction relocates the pointer to either of two locations based on a test of whether the contents of an “index” data memory is zero or not. When the calculator encounters the DSZ instruction, it reduces the value in the “index” memory by one before the test is made. Data memories 00 through 09 can be used as “index” memories.



© 2010 Joerg Woerner  
Datamath Calculator Museum

**Example 6-1:** Let data memory 05 be the index memory.

- (1) Store 2 in memory 05
- (2) From the keyboard, using DSZ instruction, branch to program step 136 if contents of index memory are greater than zero.
- (3) Using DSZ instruction a second time, branch to program step 124 if contents of index memory are greater than zero.

**Exercise 6-1:** Let data memory 00 be the index memory.

- (1) Store 5 in memory 00
- (2) Will branching occur on the fourth DSZ instruction?
- (3) What will contents of data memory 00 be after the four DSZ instructions?

# Intermediate Programming

## PERSONAL PROGRAMMING

IV-71-79 V-63-65 VI-13-14

### Example 6-1: (continued)

Solution:

Press

1. 2
- [STO]
- 05
2. [Dsz] \*
- 5
- 136

[RCL] 05 to verify index memory contents  
were decreased from 2 to 1.

Enter LEARN mode to verify branching  
occurred.  
([LRN] – display = 136 00)

Press

- [Dsz] \*
- 5                00      (1) Store 5 in memory 00
- 124              (2) Will branching occur on this jump?

[RCL] 05 again to verify index memory  
contents were decreased from 1 to 0.

Enter LEARN mode to verify branching  
*did not* occur.  
([LRN] – display = 136 00)

### Exercise 6-1: (continued)

The “decrement and skip on zero” instruction located here causes the test of register 5 to carry on until memory 00 is reached.

Press

1. 2
- [STO]
- 05
2. [Dsz] \*
- 5
- 136

[RCL] 05 to verify index memory contents  
were decreased from 2 to 1.

Enter LEARN mode to verify branching  
occurred.  
([LRN] – display = 136 00)

Press

- [Dsz] \*
- 5                00      (1) Store 5 in memory 00
- 124              (2) Will branching occur on this jump?

[RCL] 05 again to verify index memory  
contents were decreased from 1 to 0.

Enter LEARN mode to verify branching  
*did not* occur.  
([LRN] – display = 136 00)

\*Denotes second function key.

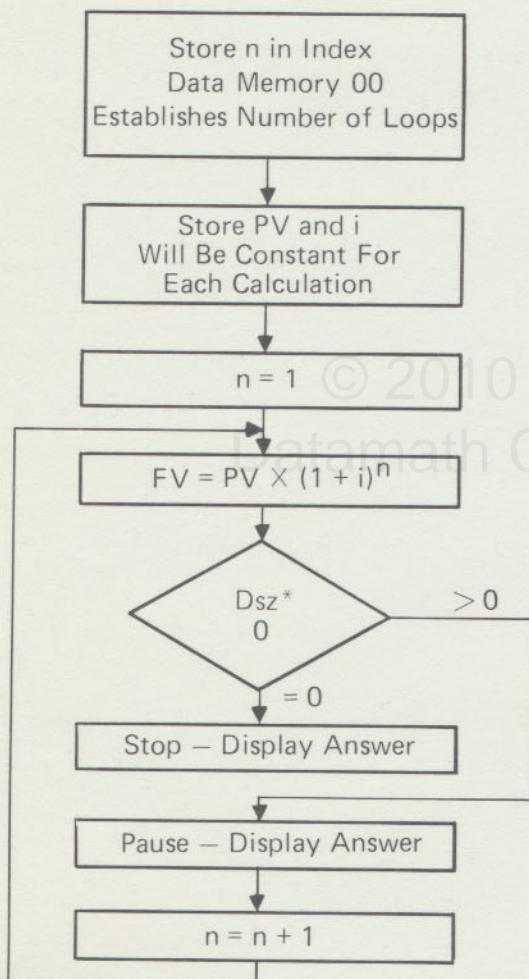
## Intermediate Programming

11

**PERSONAL PROGRAMMING**

**Example 6-2:** If you invest \$1000 in a savings account paying 6% compounded annually, how much will it be worth at the end of each year for the next five years? Use the DSZ instruction to set up five automatic loops.

## Flow Diagram



**Exercise 6-2:** Write a program that will count by 1 up to a total of 10 and then start over again. Use data memory 00 as the index memory and use decrement and skip on zero (DSZ) looping.

\*Denotes second function key.

# Intermediate Programming

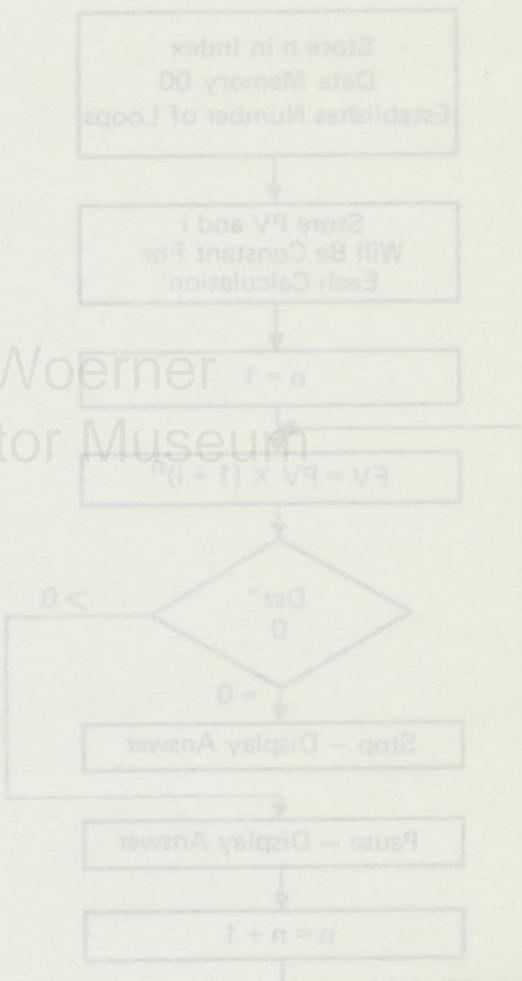
PERSONAL PROGRAMMING  
IV-71-79 V-63-65

## Program Development

000	[Lbl]	*	038	[Pause]	*
	[ A ]			[Pause]	*
	[STO]			[RCL]	
	00		03		
004	[R/S]			[ + ]	
005	[Lbl]	*		1	
	[ B ]			[ = ]	
	[STO]			[STO]	
	01		03		
009	[R/S]			[GTO]	
010	[Lbl]	*		0	
	[ C ]		049	20	
	[STO]				
02					
014	[R/S]				
015	[Lbl]	*			
	[ E ]				
	1				
	[STO]				
	03				
020	[RCL]				
	01				
	[ X ]				
	[ ( ]				
	1				
	[ + ]				
	[RCL]				
	02				
	[ ) ]				
	[y <sup>x</sup> ]				
	[RCL]				
	03				
	[ = ]				
	[Dsz]	*			
	0				
	0				
	38				
037	[R/S]				

\*Denotes second function key.

Example 8-5: N-1000 Gauss-Jordan Elimination  
Simplifying row operations by using the DSE instruction to set the value  
of one row to zero after the first two rows have been swapped.  
Row Displays



# Intermediate Programming

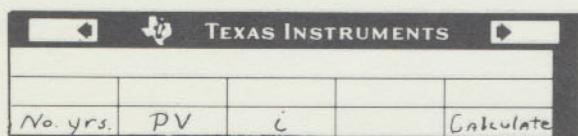
II

PERSONAL PROGRAMMING  
IV-71-79 V-63-65

## Program Entry

1. Enter LEARN mode – press [LRN]
2. Key in program instructions
3. Exit LEARN mode – press [LRN]

## Program Execution



Press

5 [ A ]  
1000 [ B ]  
.06 [ C ]  
[ E ]

} Enter  
variables

Calculate FVs

© 2010 Joerg Woerner  
Datamath Calculator Museum

Answers: 1060.              1st Year  
1123.6              2nd Year  
1191.016              3rd Year  
1262.47696              4th Year  
1338.225578              5th Year



The first two sections of this book introduced you to the basic functions of the TI Programmable 59 calculator as well as the most commonly used programming techniques. This section goes a little further into programming to show you some of your calculator's potential to solve even the most complex problems.

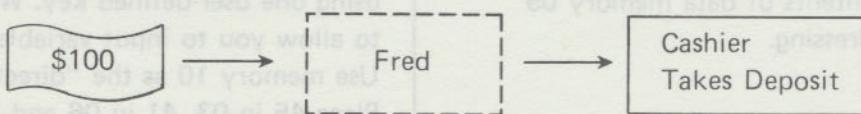
The topics we have selected for this section include indirect addressing, alphanumeric printing, and solid-state software. The information here by no means covers all the aspects of these selected topics or advanced programming techniques. Please refer to your owner's manual, *Personal Programming*, for more detailed information.

## 1. INDIRECT ADDRESSING

### DATA MEMORIES

One way you can use "indirect addressing" is with operations involving data memories. Instead of addressing a data memory directly, there are times when you may wish to address the memory "indirectly" through another data memory. When you use indirect addressing, the first data memory you specify *does not* contain the data you want; what it does contain is the number of another data memory which does contain the data you want.

For example take the statement, "Go deposit \$100 with the cashier. Fred will tell you who the cashier is." Your ultimate objective is to deposit \$100 with the cashier, but you have to go to Fred first who will tell you where the cashier is located. You take an indirect route to the cashier by seeing Fred first.



Direc<sup>t</sup>ts you  
to Cashier

(001 = suffav) 00 \* [ba] {RCF}

**Example 1-1:** Store 100 in data memory 09 using indirect addressing. Let data memory 00 be the "directing" memory.

Solution:

- (1) Store 9 in memory 00

9 [STO] 00

- (2) Place 100 in memory 09 with indirect addressing

100 [STO] [Ind] \* 00

The calculator goes to memory 00, finds a 9 there and then goes to memory 09 where it stores 100.

(3) [RCL] 00 (value = 9)

(4) [RCL] 09 (value = 100)

**Example 1-2:** As a continuation of the above exercise, recall contents of data memory 09 using indirect addressing.

Solution:

- (1) Clear display – press [CLR]

- (2) Press

[RCL] [Ind] \* 00 (value = 100)

\*Denotes second function key.

**Exercise 1-1:** From the keyboard store 1 directly in data memory 00. Then using indirect addressing store 2 in data memory 01 and 3 in data memory 02. Recall and verify the contents of all three data memories.

Solution:

## J. INDIRECT ADDRESSING

### DATA MEMORIES

**Exercise 1-2:** Load data memories 03, 06 and 09 using one user-defined key. Write your program to allow you to input variables on key [ C ]. Use memory 10 as the "directing" data memory. Place 45 in 03, 41 in 06 and 247 in 09.

Solution:

**Example 1-3:** Load data memories 01, 02 and 03 using one user-defined key. Write a program that will allow you to store 24 in memory 01, 75 in memory 02 and 136 in memory 03 using the following key sequence:

Press

24 [ A ]  
75 [ A ]  
136 [ A ]

Solution:

## Program Development

000	[LBL] *	}	Initialization. Places 1 in "directing" memory initially.
	[ E ]		
1			
[STO]			
00			

005	[R/S]
006	[LBL] *
	[ A ]
	[STO] [Ind] *
00	

010	[RCL]
	00
	[ + ]
	1
	[ = ]

015	[STO]
	00

017	[R/S]
-----	-------

## Program Entry

1. Enter LEARN mode – press [LRN]
2. Key in program instructions
3. Exit LEARN mode – press [LRN]

\*Denotes second function key.

**Exercise 1-3:** Write a program to allow you to indirectly "load" consecutive data memories, and specify which data memory you will load first. Use memory 00 as the "directing" data memory. Then load 10 into memory 10, 11 into memory 11 and 12 into memory 12.

Solution:

00	[S <sub>1</sub> ]
01	[RCL] 01
02	[RCL] 02
03	[RCL] 03

Values all numbers were stored in correct  
order

PROGRAM MEMORIES

Yon can also use indirect addressing  
with address 10 for a data memory, but a direct  
address 10 for a data memory will also  
work.

00

01

02

03

04

05

06

07

08

09

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

**Program Execution**

Press [E] to execute the program.

**Display/Comments**

- [E] 1. Memory to be loaded

- |           |    |
|-----------|----|
| 24 [ A ]  | 2. |
| 75 [ A ]  | 3. |
| 136 [ A ] | 4. |

Verify all numbers were stored in correct locations.

- |          |                |
|----------|----------------|
| [RCL] 01 | (value = 24.)  |
| [RCL] 02 | (value = 75.)  |
| [RCL] 03 | (value = 136.) |

**PROGRAM MEMORIES**

You can also use indirect addressing in conjunction with branching instructions. When you do this, your first address is for a data memory, not a program step number. However, the ultimate address is the program step number stored in the data memory which you first addressed.

Normal GTO:

3-digit program step location

Press

[GTO]

{  
1  
94}

Program pointer moves to:  
010

00

Indirect GTO:

2-digit number of data memory }

Press

[GTO]

[Ind]\*

01

Program pointer moves to:  
010

Data memory 01

194

194

00

\*Denotes second function key.

**Example 1-4:** Use data memory 01 to "direct" the pointer to program step 194.

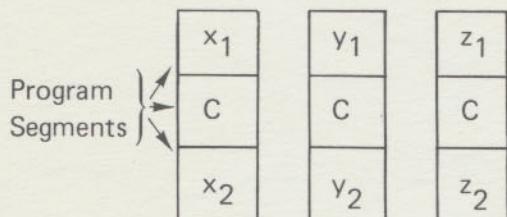
Solution:

Press	Comments
[RST]	Set pointer to 000
194 [STO] 01	Place ultimate location in directing memory
[GTO] [Ind] * 01	

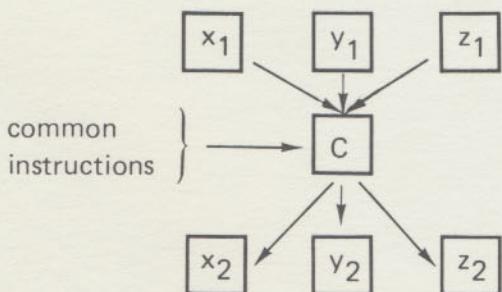
Verify pointer went to program step 194.  
Enter LEARN mode.  
([LRN] – display = 194 00)

## Demonstrated Conceptual Use

1. Assume your program has 3 segments, each with a set of instructions (c) identical to those in the other segments.



2. You prefer to write the common set of instructions only once to save program steps.



\*Denotes second function key.

**Exercise 1-4:** From the keyboard, use data memory 24 to "direct" the pointer to program step 329.

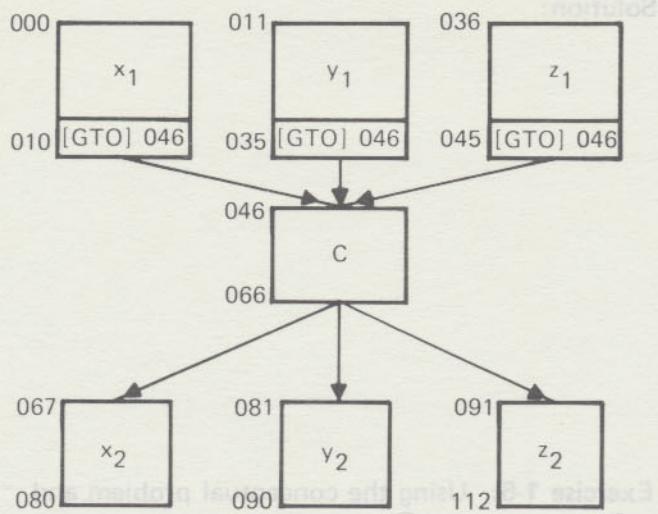
Solution:



**Exercise 1-5:** Using the conceptual problem and the program step numbers assigned to each segment, set up a program that would solve the problem. Identify x with label A, y with label B and z with label C.

By using labels, GTOs, STOs, indirect GTOs, R/Ss, and zeros for other instructions, you can construct a program that will follow the conceptual flow.

3. Converging from  $x_1, y_1$ , and  $z_1$  to C  
is no problem. We simply use a "Go To"  
statement at the end of each segment.



Examples 1: Use pointer memory to "glue" the pointers of procedure step 104.

Comments:  
• Set pointer to 000  
• Place null bytes  
• Position in  
• Occupied memory  
• X  
• 10 \* [ptr] - [OTD]

Actual pointer must be pointer step 104  
Enter LEARN mode  
([F10] — [left] = 104 00)

Demonstrated Connectional Use

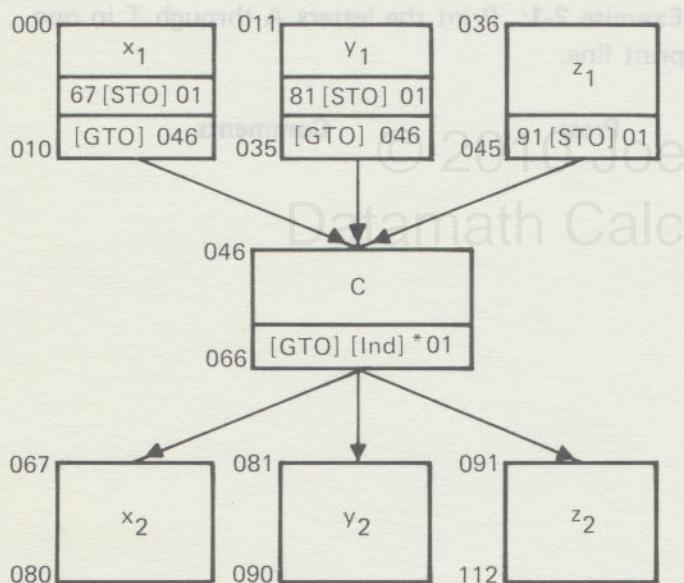
Assume four blocks have 3 columns  
each with 3 rows  
Information of three in-line outer statements



You better to write the column set of  
intersections out-of-order to save disk wear



4. Here's the problem: How does the pointer know to which segment it should diverge. You can tell the pointer which segment to go to by using indirect addressing of the program memories in the following manner. Notice that just prior to converging to "C" the program step number to which the program should return is stored in a "directing" data memory. When indirectly addressed at the end of "C", the pointer goes to the "directing" memory and then to the address corresponding to the value stored in the memory.



\*Denotes second function key.

## III-5. ALPHANUMERIC PRINTING

You can connect your calculator to the PC-400A printer via RS-232C interface. This allows you to print alphanumeric characters directly from your calculator. To do this, you will need to purchase a special adapter cable. This cable connects the RS-232C port on the back of your calculator to the serial port on the back of the PC-400A printer. Once connected, you can use the built-in printer command language to print anything you like.

For example, to print the string "Hello World", you would enter:

PRINT "Hello World"

Calculator:  
PRINT "Hello World"

Computer:  
Hello World

Output:  
Hello World

Calculator:  
PRINT "A" \* 10

Computer:  
AAAAAAAAAA

Output:  
AAAAAAAAAA

Calculator:  
PRINT "B" \* 10

Computer:  
BBBBBBBBBB

Output:  
BBBBBBBBBB

Calculator:  
PRINT "C" \* 10

Computer:  
CCCCCCCCCC

Output:  
CCCCCCCCCC

Calculator:  
PRINT "D" \* 10

Computer:  
DDDDDDDDDD

Output:  
DDDDDDDDDD

Calculator:  
PRINT "E" \* 10

Computer:  
EEEEEEEEEE

Output:  
EEEEEEEEEE

Calculator:  
PRINT "F" \* 10

Computer:  
FFFFFFFFF

Output:  
FFFFFFFFF

## 2. ALPHANUMERIC PRINTING

You can connect your calculator to the PC-100A Print Cradle and print alphanumeric information. You first enter a two-digit alphanumeric code in the display for each character you wish to print. One print line has 20 character positions (0-19) and you must use one of the following op codes to tell your calculator exactly where on the line you want your characters printed.

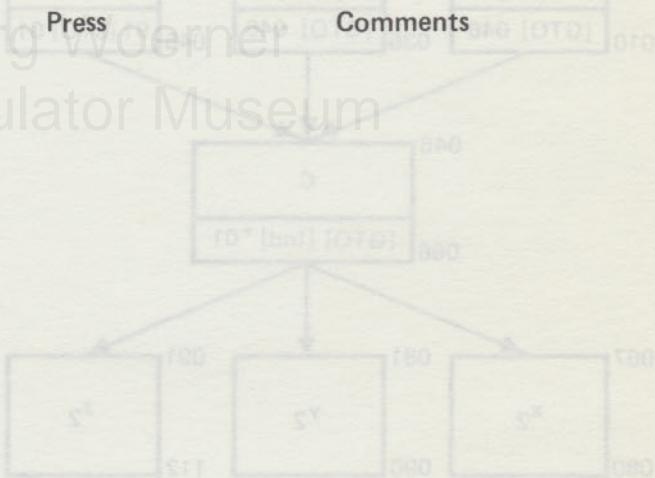
- [2nd] [Op] 01 – for far left quarter of line
- [2nd] [Op] 02 – for inside left quarter of line
- [2nd] [Op] 03 – for inside right quarter of line
- [2nd] [Op] 04 – for far right quarter of line

To print a line, press [2nd] [Op] 5.

**Example 2-1:** Print a line of 20 characters.

Press	Comments
[2nd] [Op] 00 1313131313	Clear the print buffer Place the code for 5 A's in the display
[2nd] [Op] 01	Place the A's in the first 5 positions (0-4)
[2nd] [Op] 02	Place A's in the second 5 print positions (5-9)
[2nd] [Op] 03	Third print positions (10-14)
[2nd] [Op] 04	Fourth print positions (15-19)
[2nd] [Op] 05	Print 1 line (consisting of 20 A's).

**Exercise 2-1:** Print the letters A through T in one print line.



**Example 2-2:** Print a number and a two-character name using op code 06.

Press	Comments
3324	Enter codes for letters P and I
[2nd] [Op] 04	Place in print positions 18 and 19
[2nd] [ π ]	Enter value of $\pi$ in the display
[2nd] [Op] 06	Print the value of $\pi$ and the name "pi".

**Exercise 2-2:** Print the value of  $2\pi$  and the name "2 X  $\pi$ ."

Press	Comments
-------	----------

© 2010 Joerg Woerner  
Datamath Calculator Museum

### 3. SOLID STATE SOFTWARE MODULE

Your TI Programmable 59 comes with a *Solid State Software* module already inserted in the back of the calculator. This is the Master Library module which we will use in the following examples.

For use in programs you are writing, you can "call" any labeled portion of a solid-state module program as a subroutine.

**Example 3-1:** Execute subroutine [Write] \* from program ML-01 (Diagnostic Program)

Solution:

Press	Display/Comments
[Pgm] * 01	0. Call program ML-01
[SBR] [Write] *	1. Execute subroutine [Write] *

*Note:* If you have your calculator connected to a PC-100A print cradle, it will print "MASTER", and the number "1."

**Example 3-2:** You can use the sequence [Pgm] \* mm [Op] \* 09 to download or transfer a library program from the *Solid State Software* module into your calculator's program memory. Then you may treat the program as though you had read it from a magnetic card or created it in the LEARN mode.

**Exercise 3-1:** Program steps 12-21 or ML-01 are part of the initialization program for the statistical functions within your calculator. These instructions can be used to clear specific data memories in sequential order, but do not clear the t-register.

Store 1, 2, 3, 4, 5, and 6 in data memories 01, 02, 03, 04, 05, and 06, respectively. Use subroutine [CLR] of ML-01 to clear data memories 01 through 04.

Press	Display/Comments
-------	------------------

\* Denotes second function key.

## Example 3-2: (continued)

- (a) If you have a PC-100A, produce on your printer a listing of the first 21 steps of the previous example; that is, the first steps of program ML-01.

Press	Display/Comments
[Pgm] * 01	0. Call ML-01
[Op] * 09	0. Download ML-01 into program memory
[RST]	0. Ensure that the program pointer is at step 000.
[List] *	0. Printer begins listing program steps

Note: Press [R/S] to stop printing. For this example, stop printing after step 21.

## Program Listing

The following sequence appears in program ML-01.

000	76	LBL	011	06	6
001	24	CE	012	42	STD
002	00	0	013	01	01
003	42	STD	014	00	0
004	09	09	015	72	ST*
005	60	DEG	016	01	01
006	58	FIX	017	97	DSZ
007	09	09	018	01	01
008	76	LBL	019	00	00
009	25	CLR	020	15	15
010	29	CP	021	92	RTN

\*Denotes second function key.

## Example 3-2: (continued)

- (b) If you don't have a PC-100A, verify downloading by single stepping through the program and checking it against the above listing.

Press	Display/Comments
[Pgm] * 01	0. Call ML-01
[Op] * 09	0. Download ML-01 into program memory
[RST]	0. Ensure program pointer is at step 000.
[LRN]	000 76 Enter LEARN mode
[SST]	001 24 Single step through instructions to verify program was downloaded.

Repeat [SST] until you have checked the first 21 steps.

\*Denotes second function key.

## (continued) : S-2. Program

10-JM memory to  
check no output. ADD-39 is now key W (1)  
set to enter IC test set to provide a testing  
area. This will allow a maximum of 10-M memory to

## Digital Comments Keys

10-JM Hg3 .0 10-\* [mg4]

00 \* [q0]

10-JM memory to

# Appendix

A

## EXERCISE SOLUTIONS — SECTION I

Exercise 1-1:  $6 + 12 = ?$

Press	Display
6	6
[ + ]	6.
12	12.
[ = ]	18. (Answer)

Exercise 1-2:  $99 - 24 = ?$

Press	Display
99	99
[ - ]	99.
24	24
[ = ]	75. (Answer)

Exercise 1-3:  $24 \times 36 = ?$

Press	Display
24	24
[ X ]	24.
36	36
[ = ]	864. (Answer)

Exercise 1-4:  $1890 \div 21 = ?$

Press	Display
1890	1890
[ ÷ ]	1890.
21	21
[ = ]	90. (Answer)

Exercise 1-5: You want to solve  $21.9 + 10.3 = ?$  but as you enter the values, you accidentally enter 10.6 instead of 10.3.

Press	Display
21.9	21.9
[ + ]	21.9
10.6	10.6
[ CE ]	0
10.3	10.3
[ = ]	32.2 (Answer)

Exercise 2-1:  $16 - 2 \div 2 = ?$

Press	Display
16	16
[ - ]	16.
2	2
[ ÷ ]	2.
2	2
[ = ]	15. (Answer)

Exercise 2-2:  $(16 - 2) \div 2 = ?$

Press	Display
[ ( ]	0.
16	16
[ - ]	16.
2	2
[ ) ]	14.
[ ÷ ]	14.
2	2
[ = ]	7. (Answer)

**Exercise 2-3:**

$$(25 \times 4) + 2 + ((6.2 - 3) - 1) = ?$$

Press	Display
[ ( ]	0.
25	25
[ X ]	25.
4	4
[ ) ]	100.
[ + ]	100.
2	2
[ + ]	102.
[ ( ]	102.
[ ( ]	102.
6.2	6.2
[ - ]	6.2
3	3
[ ) ]	3.2
[ - ]	3.2
1	1
[ ) ]	2.2
[ = ]	104.2 (Answer)

**Exercise 2-4:**

$$10 \times (14.85 + (21.2 - 8.1) - (2.6 + 3.2)) = ?$$

Press	Display
10	10
[ X ]	10.
[ ( ]	10.
14.85	14.85
[ + ]	14.85
[ ( ]	14.85
21.2	21.2
[ - ]	21.2
8.1	8.1
[ ) ]	13.1
[ - ]	27.95
[ ( ]	27.95
2.6	2.6
[ + ]	2.6
3.2	3.2
[ ) ]	5.8
[ ) ]	22.15
[ = ]	221.5 (Answer)

(Answer)

(Answer)

# Appendix A

A

**Exercise 3-1:**  $42^2 = ?$

Press	Display
42	42
[ $x^2$ ]	1764. (Answer)
( $\text{Answer}$ )	( $\text{INV}$ ) ( $\text{Ans}$ )

**Exercise 3-2:**  $2^9 = ?$

Press	Display
2 [ $y^x$ ]	2.
9	9
[ = ]	512. (Answer)

**Exercise 3-3:**  $\sqrt{5726} = ?$

Press	Display
5726	5726
[ $\sqrt{x}$ ]	75.67033765 (Answer)

**Exercise 3-4:**  $\sqrt[100]{1470} = ?$

Press	Display
1470 [ $\text{INV}$ ] [ $y^x$ ]	1470.
100	100
[ = ]	1.075655429 (Answer)

**Exercise 3-5:**  $\cos 47^\circ = ?$

Press	Display
[2nd] [Deg]	
47	47
[2nd] [cos]	.6819983601 (Answer)

**Exercise 3-6:**  $\tan 3 \text{ radians} = ?$

Press	Display
[2nd] [Rad]	
3	3
[2nd] [tan]	-.1425465431 (Answer)

**Exercise 3-7:**  $2 + \sqrt{8} = ?$

Press	Display
2 [ + ]	2.
8	8
[ $\sqrt{x}$ ]	2.828427125
[ = ]	4.828427125 (Answer)

**Exercise 3-8:**  $\sin 45^\circ + \cos 45^\circ = ?$

Press	Display
[2nd] [Deg]	
45	45
[2nd] [sin] [ + ]	.7071067812
45	45
[2nd] [cos]	.7071067812
[ = ]	1.414213562 (Answer)

**Exercise 3-9:** Round off 10.79816 to three decimal places.

Press	Display
10.79816	10.79816
[2nd] [Fix] 3	10.798 (Answer)

**Exercise 3-10:** Restore 10.798 in Exercise 3-9 to 10.79816.

Press	Display
[INV]	10.798
[2nd] [Fix] 9	10.79816 (Answer)
or [2nd] [Fix] 9	10.79816 (Answer)

**Exercise 3-14:** Drop the mantissa of the number 11.9247.

Press	Display
	11.9247
[INV] [2nd] [Int]	0.9247 (Answer)

**Exercise 3-11:** Enter 29,810 in scientific notation of  $2.981 \times 10^4$ .

Press	Display
2.981	2.981
[EE]	2.981 00
4	2.981 04 (Answer)

**Exercise 3-12:** Change answer in Exercise 3-11 to engineering notation.

Press	Display
[2nd] [Eng]	29.81 03 (Answer)

**Exercise 3-13:** Drop the decimal portion of the number 107.24.

(Note: Press [INV] [2nd] [Eng] to get out of engineering notation.)

Press	Display
107.24	107.24
[2nd] [Int]	107. (Answer)

**Exercise 4-1:**

Press	Display
175 [STO] 29	175.
214 [STO] 12	214.
[RCL] 29	175.
[RCL] 12	214.

**Exercise 4-2:**

Press	Display
212 [STO] 21	212.
42 [SUM] 21	42.
[RCL] 21	254.

**Exercise 4-3:**

Press	Display
100 [STO] 11	100.
29 [INV] [SUM] 11	29.
[RCL] 11	71.

# Appendix

A A

## Exercise 4-4:

Press	Display
75 [STO] 09	75.
15 [Prd] * 09	15.
[RCL] 09	1125.

[RCF]

## Exercise 4-5:

Press	Display
999 [STO] 27	999.
33 [INV] [Prd] * 27	33.
[RCL] 27	30.27272727

[RCF]

## Exercise 4-6:

Press	Display
41 [STO] 11	41.
97 [STO] 20	97.
147 [STO] 26	147.
[Exc] * 11	41.
[Exc] * 20	97.
[Exc] * 26	147.

[RCF]

## Exercise 4-7: Clear all memories simultaneously.

Press [CMs] \*

Verify by recalling memories used: 09, 11, 12, 20, 21, 26, 27, and 29.

\* Denotes second function key.

## Exercise 5-1:

## Keyboard Expression Solving

Solution 1

Press

45000  
[STO]  
01  
.09  
[ ÷ ]  
12  
[ = ]  
[STO]  
02  
30  
[ × ]  
12  
[ = ]  
[STO]  
03

Solution 2

Press

360  
[+/-]  
[ ) ]  
[ ) ]  
[ = ]  
(Answer =  
362.0801776)  
To solve for different  
variables, key in  
expression again  
substituting new  
values where needed.

45000

[ × ]

[ ( ]

.0075

[ ÷ ]

[ ( ]

1

[ - ]

[ ( ]

1

[ + ]

.0075

[ ) ]

[y<sup>x</sup>]

## Keyboard Expression Solving

Solution 1  
(Continued)

Press

[RCL]  
03  
[+/-]  
[ ) ]  
[ ) ]  
[ = ]  
(Answer =  
362.0801776)

Solution 2  
(Continued)

Press

[RCL]  
03  
[+/-]  
[ ) ]  
[ ) ]  
[ = ]  
(Answer =  
362.0801776)

To solve for different  
variables, store new  
values in appropriate  
data memory and  
key in expression  
again with appropriate  
“recalls” in place  
of variable values.

[RCL]  
01  
[ × ]  
[ ( ]  
[RCL]  
02  
[ ÷ ]  
[ ( ]  
1  
[ - ]  
[ ( ]  
1  
[ + ]  
[RCL]  
02  
[ ) ]  
[y<sup>x</sup>]

# Appendix

A

### Exercise 6-1:

**Exercise 6-1: (continued)**

Programming		Press		Programming	
000 000 28	Press	000 000 160	[GTO] 000 000	000 000 017	Press
000 23		000 000 [ ÷ ]		000 000 018	03
000 23	45000	000 000 022		000 000 019	[+/-]
000 10	[STO]	000 000 * [DSE]		000 000 020	[ ) ]
000 13	01	000 000 [LRN] [GTO] 000 000		000 000 021	[ ) ]
000 12	.09	000 000 [RCI] 000 000		000 000 022	[ = ]
000 24	[ ÷ ]	000 000 [LRN] [GTO] 000 000		000 000 023	[R/S]
000 050 24	12	000 000 [ -/+ ] * [INT]		Exit LEARN mode Press [LRN]	
0	[ = ]	000 000 [LRN]		(a) To solve as stated	
	[STO]			Press [RST] Press [R/S]	
	02			(Answer = 362.0801776)	
	30			(b) To solve with new interest rates	
	[ X ]			Press	
	12			.0925	
	[ = ]			[ ÷ ]	
	[STO]			12	
	03			[ = ]	
	Enter LEARN mode Press [LRN]			[STO]	
000 000 05	000 000 [RCL]	000 000 000 000		02	
000 000 11	000 000 001	000 000 000 000		[RST]	
000 000 11	000 000 [ X ]	000 000 000 000		[R/S]	
000 000 11	000 000 [ ( ]	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 004 [RCL]	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 005 02	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 006 [ ÷ ]	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 007 [ ( ]	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 008 1	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 009 [ - ]	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 010 [ ( ]	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 011 1	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 012 [ + ]	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 013 [RCL]	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 014 02	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 015 [ ) ]	000 000 000 000		(Answer = 370.2039415)	
000 000 11	000 000 016 [y <sup>x</sup> ]	000 000 000 000		(Answer = 370.2039415)	

**Exercise 9-1:**

Press	Display / Comments
10	10 No. blocks of 10 data memories
[Op] * 17	159.99

Press [LRN]  
Exit LEARN mode

**Exercise 10-1:**

Press	Display
[GTO] 006 [LRN]	006 75
[ ÷ ]	007 53
[SST]	008 53
[Del] *	008 01
[LRN] [GTO] 13 [LRN]	013 43
[RCL] 02	015 54
[LRN] [GTO] 19 [LRN]	019 54
[Ins] * [+/-]	020 54
[LRN]	0

**2. The correct key sequence is:**

Display	Instruction
000 00	[RCL]
001 00	01
002 00	[ X ]
003 00	[ ( ]
004 00	[RCL]
005 00	000 [RCL] 02
006 00	01 100 [ ÷ ]
007 00	000 [ X ] 000 [ ( ]
008 00	003 [ ) ] 1
009 00	004 [RCL] 000 [ - ]
010 00	000 000 [ ( ]
011 00	000 [ ÷ ] 000 1
012 00	000 [ ) ] 000 [ + ]
013 00	000 000 [RCL]
014 00	000 [ - ] 000 02
015 00	000 [ ) ] 010 [ ) ]
016 00	000 010 [ y <sup>x</sup> ]
017 00	000 015 [RCL]
018 00	000 013 [RCL] 03
019 00	000 014 [ + / - ]
020 00	000 018 [ ) ]
021 00	000 019 [ * y <sup>x</sup> ]
022 00	000 020 [ = ]
023 00	[ R/S ]

\*Denotes second function key.

# Appendix

# A

## Exercise 11-1:

Solution:

Press	Comments
[Pgm] * 18	Select program
[ E' ] *	Initialize
10 [ A ]	Enter n
5.5 [ B ]	Enter i
10000 [ D ]	Enter FV
0 [ C ]	Calculate PV

(Answer: PV = 5854.31)

## Exercise 11-2:

Solution:

Press	Comments
[Pgm] * 19	Select program
[ E' ] *	Initialize
[ C' ] *	Select type annuity (ordinary)
30 [ X ] 12 [ = ] [ A ]	Enter n (as months)
9.5 [ ÷ ] 12 [ = ] [ B ]	Enter i (as monthly rate)
50000 [ D ]	Enter PV
0 [ C ]	Calculate monthly payment

© 2010 Joerg Woerner  
(Answer: PMT = 420.43)  
Datamath Calculator Museum

\*Denotes second function key.



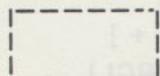
## EXERCISE SOLUTIONS — SECTION II

## Exercise 1-1:

## ENTER VARIABLE VALUES THROUGH DISPLAY

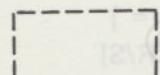
## Program Instructions

000	[ ÷ ]	50
001	[ ( ]	[ ) ]
002	1	[ x <sup>y</sup> ]
003	[ + ]	[ RCL ]
004	[ R/S ]	[ = ]



005	[ ) ]	[ RCL ]
006	[ y <sup>x</sup> ]	
007	[ R/S ]	
008	[ = ]	
009	[ R/S ]	

(Answer = 3218.896851)



## Program Entry

1. Enter LEARN mode — press [LRN]
2. Key in program instructions
3. Exit LEARN mode — press [LRN]

## Program Execution (Keyboard Entries)

## Press

[RST]  
10000 [MR]  
[R/S]  
.065 [R/S]  
18 [R/S]  
(Answer = 3218.896851)

## ENTER VARIABLE VALUES USING DATA MEMORIES

## Program Instructions

000	[RCL]	
001	04	
002	[ ÷ ]	
003	[ ( ]	
004	1 [ × ]	000
005	[ + ]	
006	[RCL]	
007	02	
008	[ ) ]	
009	[y <sup>x</sup> ]	
010	[RCL]	
011	03	
012	[ = ]	
013	[R/S]	

## Exercise 1-1: (continued)

## Program Entry

1. Enter LEARN mode — press [LRN]
2. Key in program instructions
3. Exit LEARN mode — press [LRN]

## Program Execution (Keyboard Entries)

Press	Comments
10000 [STO] 04	Store FV
.065 [STO] 02	Store i
18 [STO] 03	Store n
[RST] [R/S]	Calculate PV

(Answer = 3218.896851)

## Exercise 2-1: (continued)

## Program Instructions

- |     |                   |
|-----|-------------------|
| 015 | [Lbl] *           |
|     | [ A ]             |
|     | [RCL]             |
| 04  |                   |
|     | [ ÷ ]             |
|     | [ ( ]             |
|     | 1                 |
|     | [ + ]             |
|     | [RCL]             |
| 02  |                   |
|     | [ ) ]             |
|     | [y <sup>x</sup> ] |
|     | [RCL]             |
| 03  |                   |
|     | [ = ]             |
| 030 | [R/S]             |

## Program Execution (Keyboard Entries)

## Exercise 2-1:

Program Instructions	Press
000 [Lbl] *	000
[ D ]	04
[STO] [RCL]	05
04	00
[R/S]	00
005 [Lbl] *	00
[ B ]	00
[STO]	03
02 [ + ]	00
[R/S]	013
010 [Lbl] *	01
[ C ]	00
[STO]	03
014 [R/S]	03

- |             |
|-------------|
| 10000 [ D ] |
| .065 [ B ]  |
| 18 [ C ]    |
| [ A ]       |

(Answer = 3218.896851)

\*Denotes second function key.

## Exercise 3-1:

## Exercise 4-3:

## Exercise 3-1: (continued)

## Exercise 4-3:

Program  
Instructions

```

000      [RPN]      000
Enter LEARN mode [A]      035      [ ÷ ]
Press [LRN] [STO]      [SBR]
000      [Lbl] * 01      [cos] *
[RCI]      [ A ] [R/S]      [ ) ]
      [STO] [RPN]      [ = ]
      00      [R/S]      040      [R/S]
004      [Lbl] * 02      [Lbl] *
005      [B]      005      [cos] *
[RCI]      [STO] [RPN]      [ ( ]
      02      [R/S]      041      [RCL]
009      [Lbl] * 03      000      02
010      [C]      04      [ ÷ ]
[RCI]      [STO] [RPN]      1
      01      [R/S]      050      0
014      [D]      05      [ ) ]
015      [Lbl] * 06      054      [INV] [SBR]
[RCI]      [RPN]      Exit LEARN mode – press [LRN]
      01      [ ÷ ]
      02      [ ( ]
      03      [ ( ]
      1      [ + ]
      00      [SBR]
      025     04      050      PV
      1      [A]
      00      [R/S]      10.8      i
      [SBR]      050      030      42      n
      [cos] *
      03      [yx]      [RCI]
      030     06      [RCL]      Calculate PMT
      03      [x]
      00      [+/–]      00      PMT = 149.2299882
      [ ) ]

```

Program  
Instructions

```

035      [ ÷ ]
[SBR]
[cos] *
[ ) ]
[ = ]
[R/S]
[Lbl] *
[cos] *
[ ( ]
[RCL]
040      [R/S]
041      [Lbl] *
[cos] *
[ ( ]
[RCL]
045      [ ÷ ]
1
0
0
050      [ ÷ ]
1
2
[ ) ]
[INV] [SBR]
Exit LEARN mode – press [LRN]

```

Program  
Execution

## Press

5200	{	PV
[ A ]		
10.8	{	i
[ B ]		
42	{	n
[ C ]		
[ D ]		Calculate PMT

$$\text{PMT} = 149.2299882$$

\*Denotes second function key.

**Exercise 4-1:****Exercise 4-2:****Exercise 4-3:****Exercise 4-4:**

**Press**  
215 [ $x \geq t$ ]

**Program  
Execution**  
000

<b>Press</b>	<b>Program Execution</b>	<b>Program Development</b>
215 [ $x \geq t$ ]	000	[Lbl] * 037 [=] [A] [Pause] * 038 [LEARN mode — press LRN]
		[STO] [Pause] * 039 [LEARN mode — press LRN]
		01 [x $\geq t$ ] 040 [RCL]
		004 [R/S] [A] 041 [RCL]
		005 [Lbl] * 042 [=] [A'] * 043 [x $\geq t$ ]
		[STO] 044 [x $\geq t$ ] * 045 [RCL]
		02 046 [RCL]
		009 [R/S] 047 [x $\geq t$ ] 010 [Lbl] * 048 [x $\geq t$ ]
		[B] 049 [RCL]
		[STO] 050 [x $\geq t$ ] 051 [RCL]
		03 052 [x $\geq t$ ] 014 [R/S] 053 [x $\geq t$ ] * 015 [Lbl] * 054 [R/S]
		020 055 [X] 019 [R/S] 056 [ ] 020 [Lbl] * 057 [=] [C] 058 [—] [STO] 059 [RCL]
		05 060 [ ] 024 [R/S] 061 [ )] 025 [Lbl] * 062 [ =] [D] 063 [R/S] [STO] 064 [X] 06 065 [ (] 029 [R/S] 066 [ 1] 030 [Lbl] * 067 [—] [E] 068 [RCL] [RCL] 069 [ 03] 05 070 [ =] [X] 071 [R/S] [RCL] 072 [ 06] 06 073 [ ]

**Exercise 4-2:**

**Press**  
203 [INV] [ $x \geq t$ ] \* 072

Verify transfer by entering LEARN mode  
Press [LRN] — display = 072 00

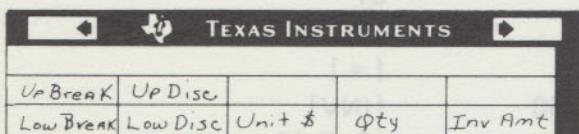
\*Denotes second function key.

## Exercise 4-3: (continued)

### Program Entry

1. Enter LEARN mode — Press [LRN] 000
2. Key in program instructions
3. Exit LEARN mode — press [LRN]

### Program Execution



### Press

100 [ A ]  
 250 [ A' ] \*  
 .02 [ B ]  
 .03 [ B' ] \*  
 7.25 [ C ]  
 30 [ D ]  
 [ E ] — Calculate invoice amount  
 (Answer = 213.15)

Enter variables

To find invoice price for 40 items after solving for 30 items, the only variable changing is "D". Use following key sequence:

### Press

40 [ D ]  
 [ E ]  
 (Answer = 281.3)

\*Denotes second function key.

## Exercise 5-1:

### Press

[St flg] *	* [P1]	000
5	[A]	
[If flg] *	[Sx]	
5	[RS]	000
211	[FP]	000
	[B]	

Pointer moves to program step 211.

Verify in LEARN mode.  
 ([LRN] — display = 211 00)

## Exercise 5-2:

### Press

[INV]	[RS]	013
[St flg] *	[CP1]	014
5	[D]	
[INV]	[RS]	018
[If flg] *	[TR1]	010
5	[E]	
227	[RCN]	05
	[S]	

Pointer moves to program step 227.

Verify in LEARN mode.  
 ([LRN] — display = 227 00)

## Exercise 5-3:

## Program Development

000	[Lbl] *	*	Press
	[ A ]		
	[x ≥ t]	*	
003	[R/S]		
004	[Lbl] *		
	[ B ]		
	[STO]		points move to line 01
01			Ans in LEARN mode
008	[R/S]	(LRN) – display –	
009	[Lbl] *		
	[ C ]		
	[STO]		Exercise 5-5:
02			
013	[R/S]	Press	
014	[Lbl] *		
	[ D ]		

## Exercise 5-3:

## Press

## Exercise 5-3: (continued)

## Program Development

038	[Lbl] *	22519	Press
	[x <sup>2</sup> ]		
			5 Key in program line [x]
			[ )
			1 [LRN]
			–
			[RCL]
01			Program Execution
			[ )]
			[ = ]
048			01
			[INV]
			[St flg] *
			1
			[R/S]
			[Lbl] *

© 2010 Joerg Woerner

Datamath Calculator Museum

## Exercise 5-3: (continued)

### Program Execution (continued)

Press

100 [ A ]  
.02 [ B ]  
8.98 [ C ]  
5 [ D ]

To calculate store customer invoice amount

[ E ] (Answer = 44.9)

To calculate catalog customer invoice amount

[ A' ] \*  
[ E ] (Answer = 44.002)

## Exercise 6-1:

- (1) Press 5 [STO] 00
- (2) Yes
- (3) One

## Exercise 6-2:

### Program Development

000	1
001	0
002	[STO]
003	00
004	0
005	[ + ]
006	1
007	[ = ]
008	[Pause] *

## Exercise 6-2: (continued)

009	[Dsz] *
010	0
011	0
012	05
013	[RST]

### Program Entry

1. Enter LEARN mode – press [LRN]
2. Key in program instructions
3. Exit LEARN mode – press [LRN]

### Program Execution

Press [R/S]

To stop: Press [R/S]

To start over from 1: Press [RST] [R/S]

To start from last number: Press [R/S]

\*Denotes second function key.



**EXERCISE SOLUTIONS — SECTION III****Exercise 1-1:**

Press

[CMs] \*

Comments

Ensure all memories

**Program Entry**

1. Enter LEARN mode — press [LRN]
2. Key in program instructions

© 2010 Joerg Woerner  
Datamath Calculator Museum

### **Exercise 1-3:**

## Program Development

```

000      [Lbl] *
[ E ]
[STO]
00      Key in floating instruction
004      [R/S]
005      [Lbl] * - Exit LEARN mode
[ A ]
[STO] [Ind] * - floating Execution
00
[RCL]
00
[ + ]
1
[ = ]
[STO]
00
[R/S]

```

## Program Entry

1. Enter LEARN mode – press [LRN]
  2. Key in program instructions
  3. Exit LEARN mode – press [LRN]

## Program Execution

Press	Display/Comments
[CMs] *	0. Clear data memories
10 [ E ]	10. Start loading in memory 10
10 [ A ]	11.
11 [ A ]	12.
12 [ A ]	13.
Verify data memories contain correct values.	
[RCL] 10	Value = 10.
[RCL] 11	Value = 11.
[RCL] 12	Value = 12.

\*Denotes second function key.

### **Exercise 1-4:**

[RST]  
329 [STO] 24  
[GTO] [Ind] \* 24  
Verify pointer went to program  
Enter LEARN mode.  
([LRN] — display = 329 0)

### **Exercise 1-5:**

## Program Development

000	[ Lbl ] *
001	[ A ]
.	.
.	.
004	6
005	7
006	[STO]
007	01
008	[GTO]
009	0
010	46
011	[ Lbl ] *

012	[ B ]
.	.
.	.
.	.
029	8
030	1
031	[STO]
032	01
033	[GTO]
034	0
035	46
036	[LbI] *

## Exercise 1-5: (continued)

## Program Development

037	[ C ]
.	• 00 [0T2] 1
.	• 00 [0T2] 2
.	• 00 [0T2] 3
.	• 00 [0T2] 4
042	9 80 [0T2] 5
043	1 80 [0T2] 6
044	[STO]
045	01
046	.
047	.
048	.
049	.
050	.
065	[GTO] [Ind] *
066	01
067	.
080	[R/S]
081	• 02 [RCI] 01
082	• 03 [RCI] 02
083	• 04 [RCI] 03
084	• 05 [RCI] 04
085	• 06 [RCI] 05
090	[R/S]
091	• 07 [RCI] 06
092	• 08 [RCI] 07
112	[R/S]

## Program Entry

1. Clear all memories — turn calculator off, then on
2. Enter LEARN mode — press [LRN]
3. Key in only the program instructions shown in the location shown.

\*Denotes second function key.

## Exercise 1-5: (continued)

*Note:* You may single step in the LEARN mode or use [GTO] out of LEARN mode to move from segment to segment. Zeros will remain in locations not filled with instructions.

4. Exit LEARN mode — press [LRN]

## Program Execution

Press [ A ]  $x_1$ , C, and  $x_2$  are executed. The pointer encounters [R/S] at 080 and when you press [LRN] the calculator displays 081 00.

Press [ B ]  $y_1$ , C and  $y_2$  are executed. The pointer will encounter [R/S] at 090 and when you press [LRN] the calculator displays 091 00.

Press [ C ]  $z_1$ , C, and  $z_2$  are executed. The pointer encounters [R/S] at 112 and when you press [LRN] the calculator displays 113 00.

*Note:* When not in the LEARN mode the calculator always displays "0" as an answer.

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803

33803&lt;/

**Exercise 2-1:****Exercise 3-1:****Exercise 3-1:****Exercise 4-1:**

Press	Comments
1314151617	Enter codes for A through E
[2nd] [Op] 01	Place in print positions 0-4
2122232425	Enter codes for F through J
[2nd] [Op] 02	Place in print positions 5-9

Press	Display/Comments
1 [STO] 01	1.
2 [STO] 02	2.
3 [STO] 03	3.
4 [STO] 04	4.
5 [STO] 05	5.
6 [STO] 06	6.

1. - 6. } Store values in data memories  
01-05 }

© 2010 Joerg Woerner  
Datamath Calculator Museum

This certifies that

has successfully completed a  
course of instruction in  
**TI-59 PROGRAMMING**  
developed by the  
**Texas Instruments Learning Center**

Instructor \_\_\_\_\_

Date \_\_\_\_\_

**TEXAS INSTRUMENTS**  
INCORPORATED





© 2010 Joerg Woerner  
Datamath Calculator Museum

© 2010 Joerg Woerner  
Datamath Calculator Museum

TEXAS INSTRUMENTS  
INCORPORATED  
DALLAS, TEXAS