TI Programmable 58/59

Math/Utilities

Using the power of your *Solid State Software*™ module

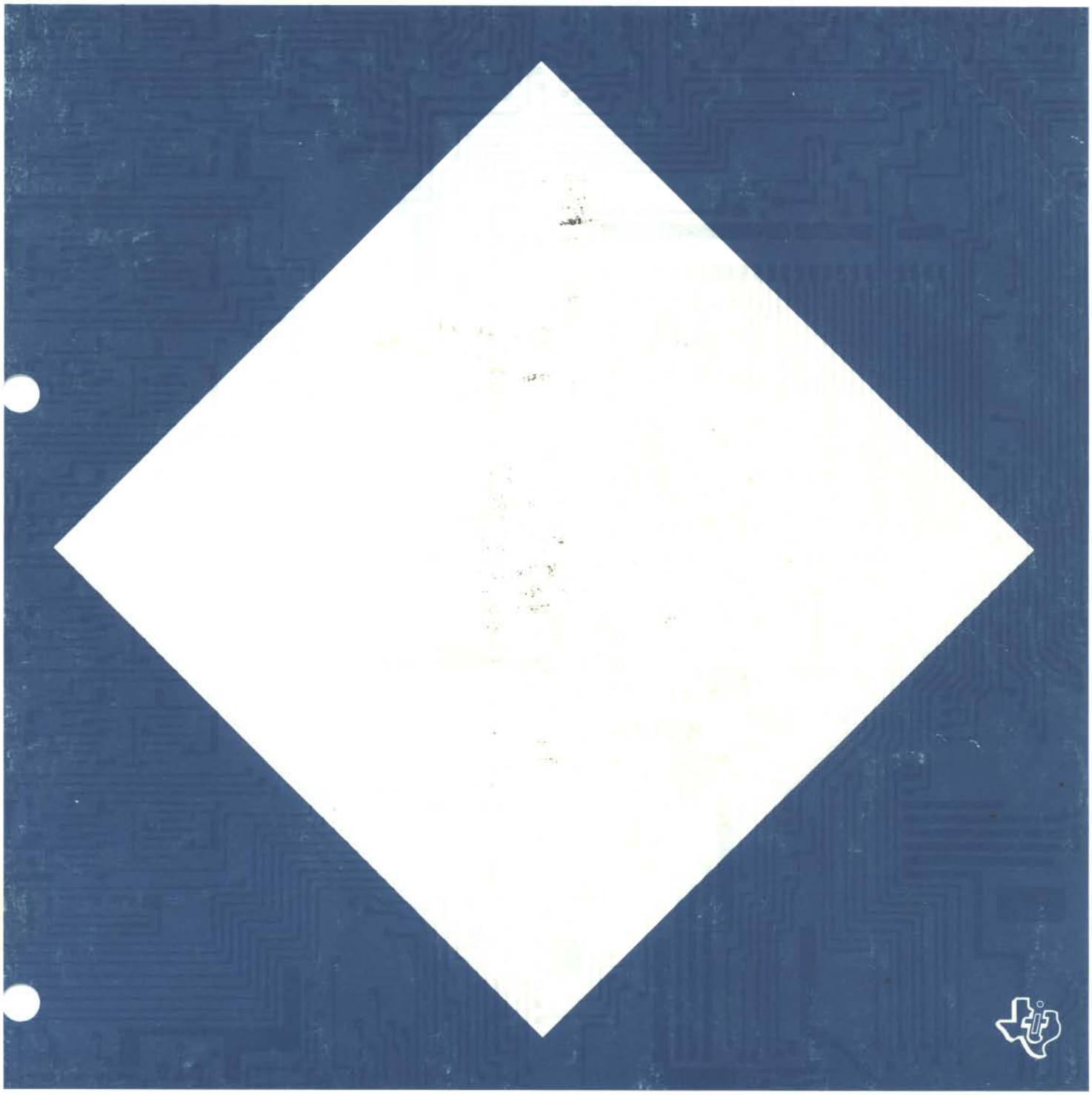


TABLE OF CONTENTS

INTRODUCTION

The Math/Utilities Module provides an easily accessible library of programs which range from utility programs to advanced mathematical routines. In addition to their use as independent programs, many of the programs can be used in whole or in part as sub-routines to programs that you write. Within seconds, you can install this *Solid State Software** module which tailors your calculator to perform a variety of frequently used functions.

USING THIS MANUAL

Following this brief introduction, you will find the description, principal equations, user instructions, and example problems for each of the 21 programs in the Math/Utilities Library. Each program is easily identified by the "MU" number in the upper corner of the page. This number corresponds with the call number you use to tell the calculator which program in the *Solid State Software* module you wish to use.

The primary reference point in this manual for each program is the User Instructions. These user instructions are also available for you in the handy pocket guide furnished with the library. The program description and sample problems should be used when you first run a program, to help you understand its full capabilities and limitations. Nonmagnetic label cards to identify the user-defined keys are also included in the library. Carefully remove the cards from the sheet and insert them in the card carrying case for convenient storage. Note that a special holder has been built into the case for storage of the library module.

When using the *Solid State Software* programs as subroutines to your own programs, you will also want to check Register Contents and Program Details for the programs.

USING THE OPTIONAL PRINTER

If you have the optional PC-100A or PC-100C printer†, a printed record of entries and results is automatic. The User Instructions and example problems are marked to show exactly which values are printed in addition to being displayed.

Use the Calculator Mounting procedure in the PC-100A or PC-100C Owner's Manual to mount your calculator on the printer. The switch called out in Step 2 (PC-100A only) should be set to "OTHER" for your calculator. Always turn the calculator and printer off before mounting or removing the calculator.

*Trademark of Texas Instruments

†Note: The TI Programmable 58 and TI Programmable 59 will not operate on the PC-100 print cradle.

INTRODUCTION

TIPS FOR RUNNING PROGRAMS

Before you begin using the *Solid State Software* programs on your own, here are a few things to keep clearly in mind until you become familiar with your calculator.

1. Press [CLR] before running a program if you are not sure of the status of the calculator. (To be completely sure of calculator status, turn it off and on again – but remember that this will clear the program memory.)

2. Some programs will leave the calculator in fix-decimal format. In that event, you should press [INV] [2nd] [fix] before running another program if this format is not desired.

3. There is no visual indication of which *Solid State Software* program has been called. If you have any doubts, the safest method is to call the desired program with [2nd] [Pgm] mm, where mm is the two-digit program number. The calculator will remain at this program number until another program is called, [RST] is pressed or the calculator is turned off.

4. A flashing display normally indicates an improper key sequence or that a numerical limit has been exceeded. When this occurs, always repeat the program sequence and check that each step is performed as directed by the User Instructions. Any unusual limits of a program are given in the User Instructions or related notes. The In Case of Difficulty portion of Appendix A in *Personal Programming* may be helpful in isolating a problem.

5. Some of the *Solid State Software* programs may run for several minutes depending on input data. If you desire to halt a running program, press the [RST] key. This is considered as an emergency halt operation which returns control to the main memory. A program must be recalled to be run again.

USING *SOLID STATE SOFTWARE* PROGRAMS AS SUBROUTINES

Any of the *Solid State Software* programs may be called as a subroutine to your own program in the main memory. Either of two program sequences may be used: 1) [2nd] [Pgm] mm (User-Defined Key) or 2) [2nd] [Pgm] mm [SBR] (Common Label). Both will send the program control to program mm, run the subroutine sequence, and then automatically return to the main program without interruption. Following [2nd] [Pgm] mm with anything other than [SBR] or a user-defined key is not a valid key sequence and can cause unwanted results.

It is very important to consider the Program Details at the end of each program description for any program called as a subroutine. You must plan and write your own program such that the data registers, flags, subroutine levels, parentheses levels, T-register, angular mode, etc., used by the called subroutine are allowed for in your program. In addition, a Register Contents section of each program description provides a guide to determine where data is

or must be located to run the program. A sample program that calls a *Solid State Software* program as a subroutine is provided in the PROGRAMMING CONSIDERATIONS section of *Personal Programming.*

If you need to examine and study the content of a *Solid State Software* program, you can download as described in the following paragraphs.

DOWNLOADING *SOLID STATE SOFTWARE* PROGRAMS

If you need to examine a *Solid State Software* program, it can be downloaded into the main program memory.* This will allow you to single step through a program in or out of the learn mode. It also allows using the program list or trace features of the optional printer. The only requirement for downloading a *Solid State Software* program is that the memory partition be set so there is sufficient space in the main program memory to receive the downloaded program. The key sequence to download a program is [2nd] [CP] [2nd] [Pgm] mm [2nd] [Op] 09, where mm is the program number to be downloaded. This procedure places the requested program into program memory beginning at program location 000. The downloaded program writes over any instructions previously stored in that part of program memory. Remember to press [RST] before running or tracing the downloaded program.

Please note that MU-07 cannot be downloaded in the TI Programmable 58 due to its length. Also, the memory partition must be reset from the power-up condition in the TI-58 for programs MU-02, 03, 04, 05, 13, 17, 18, and 20. The sequence to repartition the memory for MU-02 is 1 [2nd] [Op] 17. The sequence for MU-03, MU-04, and MU-05 is 0 [2nd] [Op] 17. The sequence for MU-13, MU-17, MU-18, and MU-20 is 2 [2nd] [Op] 17. Repartitioning must be performed before the downloading sequence.

The partition must be changed from the power-up condition in the TI Programmable 59 for MU-07. The key sequence to repartition the memory for MU-07 is 3 [2nd] [Op] 17.

REMOVING AND INSTALLING MODULES

The Securities Analysis Module can easily be installed in the calculator or replaced with another. It is a good idea to leave the module in place in the calculator except when replacing it with another module. Be sure to follow these instructions when you need to remove or replace a module.

CAUTION

Be sure to touch some metal object before handling a module to prevent possible damage by static electricity.

1. **Turn the calculator OFF.** Loading or unloading the module with the calculator ON may cause the keyboard or display to lock out. Also, shorting the contacts can damage the module or calculator.

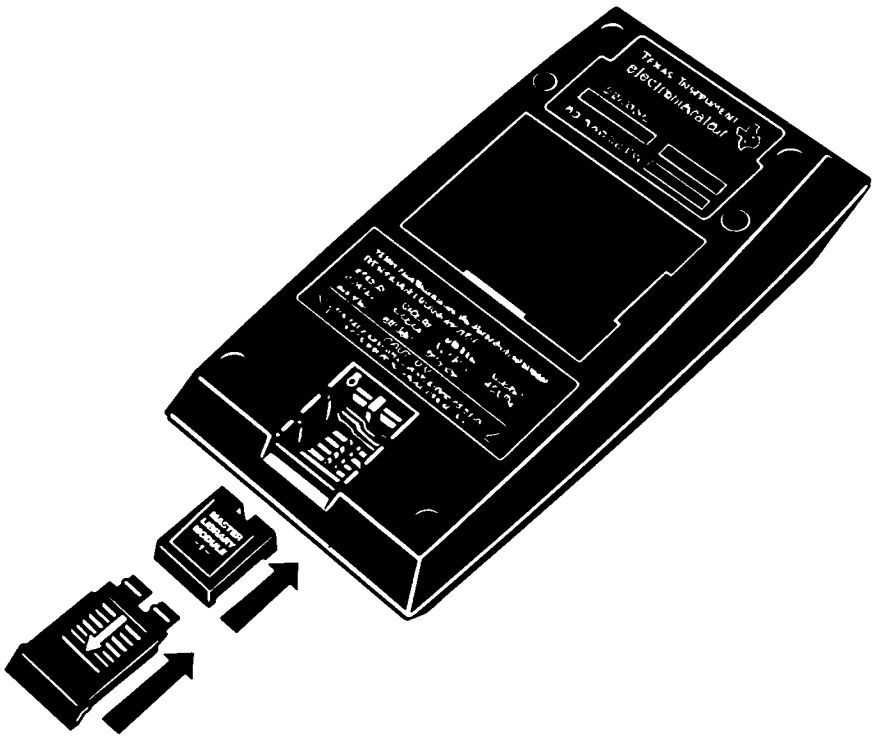
*Unless the library is a protected special-purpose library or the program is too large to fit in the main memory.

2. Slide out the small panel covering the module compartment at the bottom of the back of the calculator. (See Diagram below.)

3. Remove the module. You may turn the calculator over and let the module fall out into your hand.

4. Insert the module, notched end first with the labeled side up into the compartment. The module should slip into place effortlessly.

5. Replace the cover panel, securing the module against the contacts.



Don't touch the contacts inside the module compartment as damage can result.

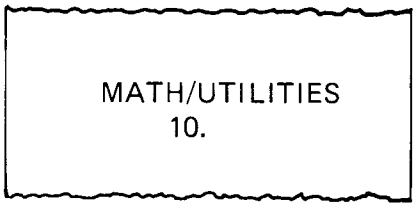
MATH/UTILITIES MODULE CHECK

This program performs the following functions separately.

1. Library Module Check
2. Linear Regression Initialization

Library Module Check

When you want to know which of your *Solid State Software* modules is in the calculator without physically looking at it, you can call the Library Module check portion of the routine. If the Math/Utilities Module is in the calculator, the number 10. will be displayed. This number is unique to the Math/Utilities Library (other optional libraries use other identifying digits). If the calculator is attached to a PC-100A or PC-100C print cradle, the following will be printed:

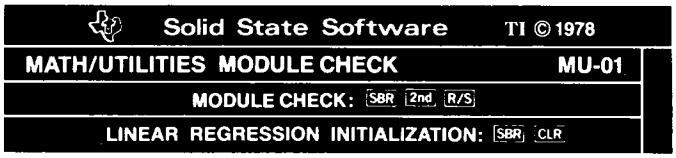


If these results are not obtained, there is probably a malfunction in the calculator or the *Solid State Software* module. First, install the Master Library module and try running the diagnostic program, ML-01. If this program does not run properly, refer to Appendix A of *Personal Programming* for an explanation of the various procedures to be followed when you have difficulties.

Linear Regression Initialization

This routine initializes the calculator for linear regression by clearing data registers R_{01} through R_{06} and the T-register. It should be used whenever linear regression or other built-in statistics functions are to be started. You can also use the routine at any time to clear these registers selectively without disturbing any other registers.

MU-01



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
	Library Module Check			
A1	Select Program		**[2nd] [Pgm] 01**	
A2	Run Module Check		**[SBR] [2nd] [R/S]**	10.[1]
	Initialize Linear Regression			
B1	Select Program		**[2nd] [Pgm] 01**	
B2	Initialize Linear Regression		**[SBR] [CLR]**	0.

NOTES: 1. The number 10. indicates the Math/Utilities Library.

PROMPTER

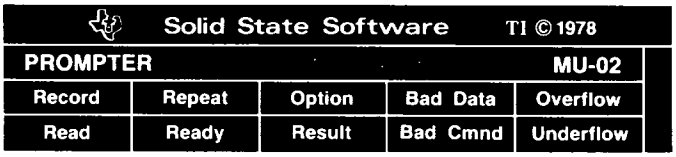
There are times when printed messages would be a helpful addition to a program. This program allows you to easily incorporate *standard* prompting messages into a program with a few simple steps. By calling Program 02 from the Math/Utilities Library, you can readily have the following messages printed: "ENTER CARD N", "READY", "REPEAT", "RESULT", "OPTION", "BAD COMMAND", "BAD DATA", "UNDERFLOW", and "OVERFLOW".

Note that "ENTER CARD N" is the prompting message used for both reading and recording magnetic cards on the TI Programmable 59. When using this message for reading program cards, the following sequence *must be* included in the first bank on the first card you read: **[2nd]** **[Lbl]** [=] **N** **[INV]** **[SBR]**. In this case, N is the number of banks you wish to read. The program will call for you to enter each bank in order with the message, "ENTER CARD N". At that time simply enter the card as you normally would to read it. Be sure you have repartitioned the calculator so that it is the same as the partitioning recorded on the cards.

You must enter the bank requested or the display will show the number of the bank you tried to read flashing. If a misread occurs, the program halts and will show a flashing zero in the display. If either of the above error conditions occur, you may repeat the instruction by pressing [R/S] and reinserting the card. *Do not clear the error condition before pressing* [R/S].

When every bank has been read, the following message will be printed: "PRESS RST R/S". Note that pressing **[RST]** will take you out of the module. Then pressing [R/S] will begin execution of the program you have just read into program memory at location 000. This allows the programmer to continue prompting or to begin calculations as needed.

MU-02



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select Program		**[2nd] [Pgm] 02**	0.
2	Prompt reading of magnetic cards. Print "ENTER CARD N".[1]		**[A]**	Bank No.
3	Prompt recording of selected bank. Print "ENTER CARD N".[2]	Bank No.	**[2nd] [A′]**	Bank No.
4	Print "READY".		**[B]**	0.
5	Print "REPEAT".		**[2nd] [B′]**	0.
6	Print "RESULT".		**[C]**	0.
7	Print "OPTION".		**[2nd] [C′]**	0.
8	Print "BAD COMMAND".		**[D]**	0.
9	Print "BAD DATA".		**[2nd] [D′]**	0.
10	Print "UNDERFLOW".		**[E]**	0.
11	Print "OVERFLOW".		**[2nd] [E′]**	0.

NOTES:

1. Be sure **[2nd] [Lbl] [=] N [INV] [SBR]** is a subroutine in the first bank read. Repartition the calculator as necessary to read in the desired number of cards. If an error occurs in reading the card (flashing display), press **[R/S]** and insert the card again. *Do not clear the error condition before pressing* **[R/S]**.
2. Records bank N on the card inserted.

Example: Suppose you wish to evaluate $f(x) = \sqrt{x^2 - x - 2}$ and you want to have a prompting message to point out to the user when $x^2 - x - 2 < 0$, since an error condition occurs when you attempt to find the square root of a negative number. The printer will prompt with a ready command when the program is ready for another input.

STEP	PROCEDURE	ENTER	PRESS	DISPLAY	PRINTOUT
1	Enter learn mode.		[LRN]	000 00	
2	Enter steps for function.		[2nd] [Lbl] [A] [STO] 1 [( ] [x^2] [−] [RCL] 1 [−] 2 [ )] [2nd] [CP] [INV] [2nd] [x≥t] [B] [$\sqrt{x}$] [2nd] [Prt] [2nd] [Adv] [2nd] [Pgm] 02 [B] [R/S] [2nd] [Lbl] [B] [$\sqrt{x}$] [2nd] [Prt] [2nd] [Pgm] 02 [2nd] [D′] [2nd] [Adv] [2nd] [Pgm] 02 [B] [R/S]	035 00	
	Exit learn mode.		[LRN]	0	
3	Compute f(4)	4	[A]		3.16227766
				0.	READY
4	Compute f(1)	1	[A]		1.414213562 ?
					BAD DATA
				0.	READY

MU-02

Register Contents

R_{00} Used	R_{05}	R_{10}	R_{15}	R_{20}	R_{25}
R_{01}	R_{06}	R_{11}	R_{16}	R_{21}	R_{26}
R_{02}	R_{07}	R_{12}	R_{17}	R_{22}	R_{27}
R_{03}	R_{08}	R_{13}	R_{18}	R_{23}	R_{28}
R_{04}	R_{09}	R_{14}	R_{19}	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
A	0		1	1			x	6,9
B								4,9
C								2,9
D								2,6,9
E								9
A′	0		1	1				6,9
B′								2,9
C′								2,9
D′								2,9
E′								9

ALPHA MESSAGES

Printing and storing alpha messages is sometimes time consuming and tedious. The following program when used with the PC-100A or PC-100C can save you time when you desire to store alpha messages on magnetic cards to reprint at a later time or when you want to incorporate prompting messages into your program.

This program uses a phone-pad entry method for entering letters and some symbols and utilizes labels to enter digits, spaces, other symbols and the following special messages: "ENTER", "PRESS", "PRESS SBR". These characters can be arranged in any order with the exception of the special messages which should be placed only at the beginning of a line.

MU-03

Solid State Software TI © 1978				
ALPHA MESSAGES				MU-03
New Line	Print	Edit	ENTER	PRESS; SBR
Left	Center	Right	Number	Space

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program		[2nd] [Pgm] 03	
2	Initialize		[SBR] [CLR]	20.
3a	To enter characters from the diagram: STU 7 VWX 8 YZ% 9 JKL 4 MNO 5 PQR 6 ABC 1 DEF 2 GHI 3 ,'? 0 Key in the number beneath the character, then • If character is on the left • If character is in the center • If character is on the right		 [A] [B] [C]	 See note 1.
3b	To enter a digit (0–9)	digit	[D]	See note 1.
3c	To enter the following symbols: + – X / = . ( )		 [SBR] [+] [SBR] [–] [SBR] [X] [SBR] [÷] [SBR] [=] [SBR] [·] [SBR] [(] [SBR] [)]	See note 1.
3d	To enter a character not listed, enter the print code from page 5 of the Quick Reference Guide	code	[SBR] [SBR]	See note 1.
3e	To leave a blank space		[E]	See note 1.
3f	The following messages may be entered at the beginning of a line: ENTER PRESS PRESS SBR		 [2nd] [D′] [2nd] [E′] [2nd] [E′] [R/S]	 See note 1.
4	To begin a new line (See note 2.)		[2nd] [A′]	
5	To enter a specific line **(Go to Step 3)**	Line No.	[2nd] [C′]	20.
6	To print a specific line	Line No.	[2nd] [B′]	0.
7	To print lines XX through YY (See note 3.)	XX.YY	[2nd] [B′]	0.

NOTES:

1. The number of characters remaining for entry on that line is displayed.
2. This instruction is used when a line contains fewer than 20 characters. When 20 characters have been entered, a new line is automatically started.
3. Two digits are required for each line number following the decimal. For example, to print lines 2 through 5, enter 2.05. When [**B′**] is used to print more than one line, entering additional lines without resetting the line pointer via [**C′**] begins storage following the last line printed.
4. If you discover an error during or after entering a line, enter the line number, press [**2nd**] [**C′**], and reenter the line.
5. Once a line is entered, the pointers are set to enter the next consecutive line.
6. Under normal operation each line is printed as it is entered. To avoid printing, press [**2nd**] [**St flg**] **1** after initialization.
7. Entry begins with line 1 after initialization.
8. Four data registers are used to store each line of your messages, beginning with R_{04}. If your message is K lines long, registers 0 through (4K + 3) must be left available for program use. Check the partitioning to be sure you have enough data registers available.
9. If you want the calculator to begin storing your message in a higher register, you may enter a line number K. This causes the calculator to leave lines 1 through (K-1) blank and begin storing the message in register 4K. R_{00} through R_{03}, as well as the T-register, must be left available for program use.

Example: Enter the following messages into the calculator:

ENTER LENGTH, PRESS A
ENTER WIDTH, PRESS B
PRESS C, AREA =

ENTER	PRESS	DISPLAY	COMMENTS
	[2nd] [Pgm] 03		Select program.
	[SBR] [CLR]	20.	Initialize.
	[2nd] [D′]	14.	ENTER
4	[C]	13.	L
2	[B]	12.	E
5	[B]	11.	N
3	[A]	10.	G
7	[B]	9.	T
3	[B]	8.	H
0	[A]	7.	,
6	[A]	6.	P
6	[C]	5.	R
2	[B]	4.	E
7	[A]	3.	S
7	[A]	2.	S
	[E]	1.	
1	[A]	20.	A

ENTER LENGTH, PRESS A is printed.

	[2nd] [D′]	14.	ENTER
8	[B]	13.	W
3	[C]	12.	I
2	[A]	11.	D
7	[B]	10.	T
3	[B]	9.	H
0	[A]	8.	,
6	[A]	7.	P
6	[C]	6.	R
2	[B]	5.	E
7	[A]	4.	S
7	[A]	3.	S
	[E]	2.	
1	[B]	1.	B
	[2nd] [A′]	20.	

ENTER WIDTH, PRESS B is printed.

	[2nd] [E′]	14.	PRESS
1	[C]	13.	C
0	[A]	12.	,
1	[A]	11.	A
6	[C]	10.	R
2	[B]	9.	E
1	[A]	8.	A
	[SBR] [=]	7.	=
	[2nd] [A′]	20.	

PRESS C, AREA = is printed.

Do not turn calculator off. These messages will be used in a program that you will enter into the calculator. With the TI-59, the program and the messages can be recorded on magnetic cards for future use. The example is a simple prompting program for calculating the area of a rectangle.

ENTER	PRESS	DISPLAY	COMMENTS
	[RST] [LRN]	000 00	Exit module, enter learn mode.
	[2nd] [Lbl] [E]		
	1 [2nd] [Pgm] 03		
	[2nd] [B′] [R/S]		
	[2nd] [Lbl] [A]		
	[STO] 27 [2nd] [Prt]		
	2 [2nd] [Pgm] 03		
	[2nd] [B′] [R/S]		
	[2nd] [Lbl] [B]		
	[STO] 28 [2nd] [Prt]		
	3 [2nd] [Pgm] 03		
	[2nd] [B′] [R/S]		
	[2nd] [Lbl] [C]		
	[RCL] 27 [×]		
	[RCL] 28 [=]		
	[2nd] [Prt] [R/S]	037 00	
	[LRN]	0	Exit learn mode.

Use the program to find the area of a rectangle of length 36 and width 27.

	[E]		ENTER LENGTH, PRESS A is printed.
36	[A]	36.	**36.** is printed.
			ENTER WIDTH, PRESS B is printed.
27	[B]	27.	**27.** is printed.
			PRESS C, AREA = is printed.
	[C]	972.	**972.** is printed.

Register Contents

R_{00} 5 Count (PTR)	R_{05}	R_{10}	R_{15}	R_{20}	R_{25}
R_{01} Used	R_{06}	R_{11}	R_{16}	R_{21}	R_{26}
R_{02} 20 Count	R_{07}	R_{12}	R_{17}	R_{22}	R_{27}
R_{03} IND	R_{08}	R_{13}	R_{18}	R_{23}	R_{28}
R_{04}	R_{09}	R_{14}	R_{19}	R_{24}	R_{29}

R_{04} and up are used for data, four registers for each line.

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
A	0–3	1	1				x	
B	0–3	1	1				x	
C	0–3	1	1				x	
D	0–3	1	1				x	
E	0–3	1	1				x	
A′*	0–3	1	1	1			x	
B′	0							
B′**	0–3	1	1	1			x	
C′	0–3			1				
D′	0,2–3							
E′	0,2–3							
+	0–3	1	1				x	
–	0–3	1	1				x	
X	0–3	1	1				x	
÷	0–3	1	1				x	
=	0–3	1	1				x	
(	0–3	1	1				x	
)	0–3	1	1				x	
•	0–3	1	1				x	
CLR	0–3	1		1				

*Does not run in scientific notation.
**Single line print command.

PRINTER FORMATTING

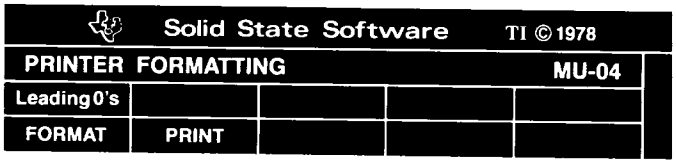
Computer print statements which combine alpha messages and data on the same line can be simulated when this program is used in conjunction with Program MU-03, Alpha Messages. This is accomplished in three steps:

1. Store the alpha message, leaving space for data, using MU-03.

2. Enter data into the line using the format statement of this program.

3. Print the combined line.

As shown in the example, each line can have several separate alpha and data fields, limited only by the maximum of twenty characters per line.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Store alpha message using MU-03, leaving space for data.[1,2]			
2	Select program.		[2nd] [Pgm] 04	
3a	Enter data.	Data	[x≷t]	T-Register
3b	Enter format statement (see note 3) • Without leading zeros • With leading zeros (Repeat step 3 as needed).		[A] [2nd] [A′]	0. 0.
4	Print line.	Line No.	[B]	0.

Notes: 1. It is assumed that the alpha code has been entered using Program MU-03. (See the Example problem).

2. Line 1 of MU-03 may not be used because data registers 4 through 7 are used in this program, MU-04.

3. The format statement is a number consisting of four elements which describe the data entered in step 3a:

(Skip)(Left) . (Right)(Line)

Skip – Number of spaces data is indented from left side of tape.
Left – Number of digits to left of decimal.
Right – Number of decimal digits.
Line – Line number (same as corresponding alpha message).

Each of the elements must be two digits (leading zero if needed).

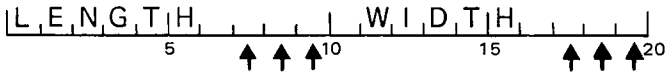
4. If the data is too large positively or negatively to fit the format specified, asterisks are printed in the spaces allowed for the data.

5. When the data is negative, one space to the left of the decimal is allocated to the minus sign.

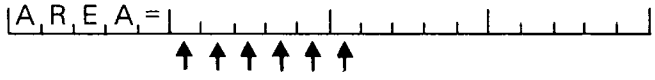
6. The decimal point and decimal digits, if any, are not printed when Right = 0.

MU-04

Example:



Store this alpha message in line 2 of Alpha Messages, MU-03.



Store this message in line 3.

These messages are entered using the following procedure. Then write a prompting program similar to the example in MU-03.

ENTER	PRESS	DISPLAY	COMMENTS
	[2nd] [Pgm] 03		Select program.
	[SBR] [CLR]	20.	Initialize.
2	**[2nd] [C′]**	20.	Enter line 2.
4	**[C]**	19.	L
2	**[B]**	18.	E
5	**[B]**	17.	N
3	**[A]**	16.	G
7	**[B]**	15.	T
3	**[B]**	14.	H
	[E]	13.	Space.
	[E]	12.	Space.
	[E]	11.	Space.
	[E]	10.	Space.
	[E]	9.	Space.
8	**[B]**	8.	W
3	**[C]**	7.	I
2	**[A]**	6.	D
7	**[B]**	5.	T
3	**[B]**	4.	H
	[2nd] [A′]	20.	**LENGTH** **WIDTH** is printed.
3	**[2nd] [C′]**	20.	Enter line 3.
1	**[A]**	19.	
6	**[C]**	18.	
2	**[B]**	17.	
1	**[A]**	16.	
	[SBR] [=]	15.	
	[2nd] [A′]	20.	**AREA** = is printed.

Do not turn calculator off. Enter a prompting program to calculate the area of a rectangle, placing the input and output data at the positions indicated by the arrows in the diagrams shown above. Input values will be limited to three digits maximum, the output to six digits.

ENTER	PRESS	DISPLAY	COMMENTS
	[RST]		Exit library program.
	[LRN]	000 00	Enter learn mode.
	[2nd] [Lbl] A		
	[STO] 27		
	[x≷t]		
	703.0002		Format statement.
	[2nd] [Pgm] 04		
	[A] [INV] [SBR]		
	[2nd] [Lbl] [B]		
	[STO] 28		
	[x≷t]		
	1703.0002		Format statement.
	[2nd] [Pgm] 04		
	[A]		
	2 [2nd] [Pgm] 04		
	[B] [INV] [SBR]		
	[2nd] [Lbl] [C]		
	[RCL] 27 [X]		
	[RCL] 28 [=]		
	[x≷t]		
	506.0003		Format statement.
	[2nd] [Pgm] 04		
	[A]		
	3 [2nd] [Pgm] 04		
	[B] [INV] [SBR]	064 00	
	[LRN]	20.	

Now use the program to calculate the area of a rectangle of length 366 and width 273.

366	[A]	0.	Enter length.
273	[B]	0.	**LENGTH 366 WIDTH 273** is printed.
	[C]	0.	**AREA = 9918** is printed.

Register Contents

R_{00} 5 Count	R_{05} Counter	R_{10}	R_{15}	R_{20}	R_{25}
R_{01} Used	R_{06} Data	R_{11}	R_{16}	R_{21}	R_{26}
R_{02} 20 Count	R_{07} Counter	R_{12}	R_{17}	R_{22}	R_{27}
R_{03} IND	R_{08}	R_{13}	R_{18}	R_{23}	R_{28}
R_{04} Format	R_{09}	R_{14}	R_{19}	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm.	Spec. Func. Used	x≷t	Fix Decimal Format
A	0–7+	1–4	2	2	03	DMS	x	
A'	0–7+	1–4	2	2	03	DMS	x	
B	0		1					

SUPERPLOTTER

Plotting and graphing functions, equations, and relationships is a useful tool for many different situations. The Superplotter program allows you to plot or graph up to 10 different functions simultaneously. The graph can be of any size and precision on both the independent and dependent variables. For the independent variable you only need to enter the starting value (x_0), the increment Δx, and the number of points (per function) you wish to print. For the dependent variable, enter the minimum value (y_{min}), the maximum value, and the number of tapes desired. The output values will be properly scaled for printing with the horizontal increments being $(y_{max} - y_{min})/(20 \times \#\text{Tapes})$.

The tapes are printed sequentially with the tape containing the least positive (or most negative) values being printed first. The tapes can then be cut apart and positioned side-by-side to properly locate the various segments. Alignment marks are printed each 10 vertical increments to aid in positioning the tapes. The end of each segment is marked with a dotted line.

Each of the functions you want to print must be defined by a series of keystrokes and entered as a subroutine in the calculator's program memory. If you need help in this area, see *Personal Programming*, pages IV-46 to IV-51. The subroutines must be identified by the labels indicated in the user instructions.

When the program calls the user-defined subroutine, x is in the T-register. When the subroutine is completed, the alpha code for the character you want printed should be in the T-register and f(x) should be in the display. See *Personal Programming*, page VI-7 for alpha codes. For example, the following subroutine would be used for printing "." in plotting the function f(x) = 2x.

Keystrokes	Comments
[2nd] [Lbl] [A']	Label for 1st function.
40	Alpha code for "."
[( ]	
[x≷t]	Swap alpha code and x value.
[×] 2 [ )]	Multiply x by 2.
[INV] [SBR]	End subroutine.

By plotting this simple function using several different increments for x and f(x), you can easily become familiar with the various features of the program.

Solid State Software TI © 1978				
SUPERPLOTTER				MU-05
			# Functions	# Pts → Plot
X_0	Δx	Y_{min}	Y_{max}	# Tapes

USER INSTRUCTIONS

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Enter subroutines for user-defined functions. (See Note 1)			
2	Select Program.		[2nd] [Pgm] 05	
3	Enter initial x value.	x_0	[A]	x_0
4	Enter x increment.	Δx	[B]	Δx
5a	Enter minimum y value.[2]	y_{min}	[C]	y_{min}
5b	Enter maximum y value.[2]	y_{max}	[D]	y_{max}
5c	Enter number of tapes desired.[2]	# Tapes	[E]	max. pos. #
6	Enter number of functions.	# Functions	[2nd] [D′]	# Functions
7	Enter number of points to be plotted per function and print graphs.	# Pts.	[2nd] [E′]	Graph[3]

NOTES: 1. Function 1 – Lbl A′ 4 – Lbl D′ 7 – Lbl B 10 – Lbl E
Function 2 – Lbl B′ 5 – Lbl E′ 8 – Lbl C
Function 3 – Lbl C′ 6 – Lbl A 9 – Lbl D

2. Steps 5a, 5b, and 5c must be performed in sequence. If any input is changed, the entire sequence must be repeated.

3. If two functions cross at the same point, an "x" is printed unless both functions are using the same character.

Example 1: Set up the Superplotter program to plot the three biorhythm cycles simultaneously, plotting points each day for the three functions. The biorhythm cycles are all sine functions differing only in period. The cycles, their periods, and the symbols to be used for plotting are shown below.

Biorhythm Cycle	Period	Symbol
Physical	23 days	P
Emotional	28 days	E
Intellectual	33 days	I

The first step is to write the subroutines to define the functions. The keystrokes are:

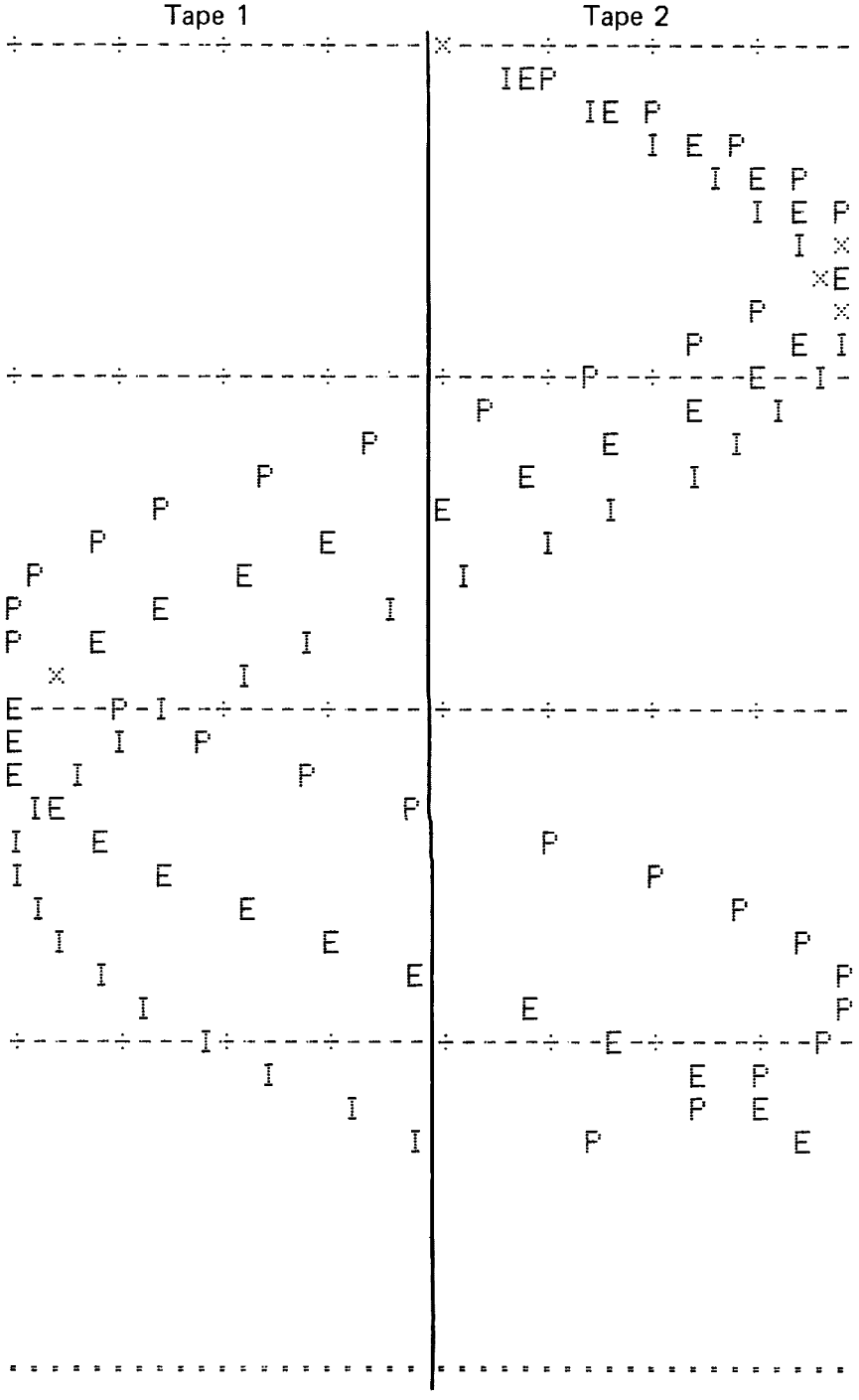
Keystrokes	Comments
[2nd] [Lbl]	
[2nd] [A′]	Label 1st Function.
33	Alpha Code for P
[x≷t]	x to display, code to T-register
[STO] 00	Store x in R_{00}
23	Period of P cycle
[GTO] [SBR]	Go to Label SBR
[2nd] [Lbl]	
[2nd] [B′]	Label 2nd Function
17	Alpha Code for E
[x≷t]	x to display, code to T-register
[STO] 00	Store x in R_{00}
28	Period of E cycle
[GTO] [SBR]	Go to Label SBR
[2nd] [Lbl]	
[2nd] [C′]	Label 3rd Function
24	Alpha Code for I
[x≷t]	x to display, code to T-register
[STO] 00	Store x in R_{00}
33	Period of I cycle
[2nd] [Lbl] [SBR]	Label SBR
[2nd] [Deg]	Select degree mode
[( ] [1/x] [X]	
[RCL] 00	
[X] 360 [ )]	Convert x to degrees
[2nd] [sin]	sin x
[INV] [SBR]	End subroutine with return

These subroutines will allow the three functions to be plotted as x is incremented (0, 1, 2, . . .). The plots will start on day 0 and progress through day 33 (34 days) so you will want to plot 34 points for each function. The sine function varies between +1 and −1, so for readability, spread this range over two tapes. The complete procedure for making this plot is as follows.

ENTER	PRESS	DISPLAY	COMMENTS
	[RST]		Exit module program
	[LRN]	000 00	Select learn mode
	[2nd] [Lbl] [A′]		Enter subroutines described earlier.
	. . .		
	[INV] [SBR]	046 00	

ENTER	PRESS	DISPLAY	COMMENTS
	[LRN]		Leave learn mode
	[2nd] [Pgm] 05		Select program
0	[A]	0.	Initial x value
1	[B]	1.	x increment
1	[+/–] [C]	–1.	minimum y value
1	[D]	1.	maximum y value
2	[E]	39.	Enter number of tapes
3	[2nd] [D′]	3.	Number of functions
34	[2nd] [E′]	0.	Number of points and start plot.

The output tapes are:

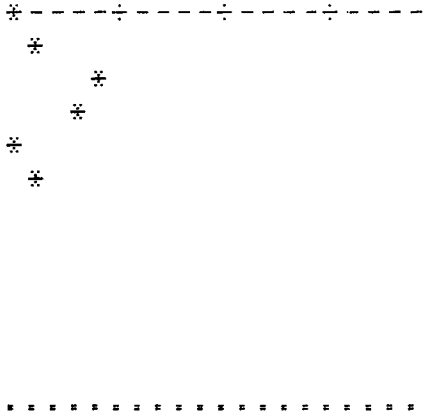


Example 2: Use the Superplotter program to plot a histogram from data stored in a block of contiguous data registers. For the example, store the following data in the registers shown. In practice, the data might be the output of a sorting program, and might reside in any block of registers not used by the Superplotter program.

Register	Data
01	1
02	2
03	5
04	4
05	1
06	2

ENTER	PRESS	DISPLAY	COMMENTS
	[RST]		Exit module program
	[LRN]	000 00	Enter learn mode
	[2nd] [Lbl] [2nd] [A′]		Enter subroutine
	51		
	[x≷t]		
	[STO] 00		
	[RCL] [2nd] [Ind] 00		
	[INV] [SBR]	010 00	x is used as pointer to retrieve data.
	[LRN]		Exit learn mode
1	[STO] 01		Store data
2	[STO] 02		
5	[STO] 03		
4	[STO] 04		
1	[STO] 05		
2	[STO] 06		
	[2nd] [Pgm] 05		Select program
1	[A]	1.	Initial x value
1	[B]	1.	x increment
1	[C]	1.	Minimum y value
20	[D]	20.	Maximum y value
1	[E]	19.	Number of Tapes
1	[2nd] [D′]	1.	Number of Functions
6	[2nd] [E′]	0.	Number of points, and plot.

The output tape is:



Register Contents

R_{00}	R_{05}	$R_{10}x_i$	R_{15}Op 01 Load	R_{20}Used	R_{25}#Tapes(save)
R_{01}	R_{06}	$R_{11}\Delta x$	R_{16}Op 02 Load	R_{21}#Functions(save)	R_{26}
R_{02}	R_{07}	$R_{12}y_{min}$	R_{17}Op 03 Load	$R_{22}y_0$	R_{27}
R_{03}	R_{08}#Functions(ctr)	$R_{13}(y_{max}-y_{min})$	R_{18}Op 04 Load	R_{23}10 Line Ctr	R_{28}
R_{04}	R_{09}#Pts (ctr)	R_{14}Pointer	R_{19}Used	R_{24}#Tapes (ctr)	R_{29}

Program Details

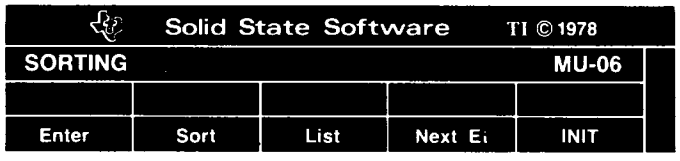
Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	$x \gtrless t$	Fix Decimal Format
A	10							
B	11							
C	12							
D	12,13			1				
E	13,24,25	1		3			x	
D′	21							
E′*	8-25	1	1	2		DMS	x	9

*Does not run in scientific notation.

SORTING

This program will quickly order a list of up to 99 numerical elements (59 with the TI-58) using an advanced technique known as the Shell sort. When used in conjunction with the PC-100A, the program will print the ordered list with each element accompanied by its ordinal number.

The sorted numbers are stored sequentially in the data registers beginning with R_{01}.



USER INSTRUCTIONS

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		[2nd] [Pgm] 06	
2	Initialize.[1]	0	[E]	0.
3	Repartition if necessary.[2]			
4a	Enter first element.	E_1	[A]	E_1
4b	Enter next element (repeat).	E_i	[A] or [R/S]	E_i
5	Sort.		[B]	0.
6	List sorted data or display first sorted element.[3]		[C] or [R/S]	SE_1
7	Display next sorted element.		[D] or [R/S]	SE_i

NOTES:
1. Stores 0 in R_{00}.
2. Registers 0-n are used where n is the number of elements.
3. Data is printed using INV List function. Press R/S to stop printing and display first sorted element. If the printer is not attached, the first element is displayed immediately.

Example: Rearrange the following numbers into increasing numerical sequence: 10.6, 5.12, 11, 9.2, −4.3, −1.45, 0.4, 37, 0, 8.3.

ENTER	PRESS	DISPLAY	COMMENTS	PRINTOUT
	[2nd] [Pgm] 06		Select program.	
0	[E]	0.	Initialize.	
10.6	[A]	10.6	Enter elements.	
5.12	[A]	5.12		
11	[A]	11.		
9.2	[A]	9.2		
4.3	[+/−] [A]	−4.3		
1.45	[+/−] [A]	−1.45		
.4	[A]	0.4		
37	[A]	37.		
0	[A]	0.		
8.3	[A]	8.3		
	[B]	0.	Sort list.	
	[D]			−4.3 01
				−1.45 02
				0. 03
				0.4 04
				5.12 05
				8.3 06
				9.2 07
				10.6 08
				11. 09
		−4.3		37. 10

Register Contents

R_{00}	Pointer	R_{05}	R_{10}	R_{15}	R_{20}	R_{25}
R_{01}	Data*	R_{06}	R_{11}	R_{16}	R_{21}	R_{26}
R_{02}		R_{07}	R_{12}	R_{17}	R_{22}	R_{27}
R_{03}		R_{08}	R_{13}	R_{18}	R_{23}	R_{28}
R_{04}		R_{09}	R_{14}	R_{19}	R_{24}	R_{29}

*R_{01} and above are used for data storage.

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm.	Spec. Func. Used	x≷t	Fix Decimal Format
A	0,1+							
B	0,1+						x	
C	0							
D	0							
E	0							

DATA ARRAYS

The handling of arrays or matrices of data is always a time consuming job. In addition, the chance for human error increases with the amount of data, especially when repetitive calculations are performed on the data array. If you are involved in processing arrays of numbers arranged in rows and columns, this program may help save some of your valuable time.

This program can simplify handling of data arrays which have a large amount of fixed data and only a percentage of the data items change from time to time. Even greater savings are possible when a data array contains numerous intermediate and final results based on other data items of the array.

The large variety of ways in which a data array is used and manipulated have made it necessary to separate the control functions of this program in elemental blocks. After the basic data items are entered, you have almost unlimited capabilities in processing, changing and evaluating the data, either from the keyboard or under program control. Using functions of this program as subroutines to your own program make it possible to generate complex and detailed reports from your data. The alphanumeric capability of the optional printer allows you to produce a permanent record of a data array and results.

Setting Up A Data Array

The characteristic feature of a data array is that it consists of rows and columns of numbers. The general left-to-right handling of an array is to input data items first in row 1.

Beginning with column 1, and continuing input of each data item into columns 2, 3, 4, etc. Then, you enter data items in row 2, then row 3, etc.

DATA ARRAY

Rows	Columns				
	1	2	3	4	5
1					
2					
3					

The first requirement of this program is that you set the size of the array. For example, the simple array shown is a 3 x 5 array (3 rows and 5 columns). The important thing to remember in setting array size is that you count every row and column you intend to use, regardless of whether it will contain data items or data results. This program automatically assumes that the last column and last row will contain totals or data results of some sort. Therefore, if your basic input data is a 3 x 5 array, you will need to specify a 4 x 6 array for this program to allow for row and column results.

Before getting into the details of how this program handles a data array, a discussion of the relationship between array size and calculator memories is in order. Since each data item is stored independently and the basic program operations take seven data registers, the total data registers required is the number of rows times the number of columns plus seven. For example, a 3 x 5 array requires (3 x 5) + 7 = 22 data registers.

The impact of the number of required data registers is that the larger the array, the less the number of program steps available in the main memory for your program. Also, you will have to change memory partitioning as the size of the array increases. In fact, if any array is too large, it may not fit at all. The following chart will help you correlate array size to memory allocation.

ARRAY/MEMORY CHART

Data Array Items*	N Value for [2nd] [Op] 17	Program Steps Available TI-59	TI-58
93**	10	160	N/A
83**	9	240	N/A
73**	8	320	N/A
63**	7	400	N/A
53	**6**	**480**	0
43	5	560	80
33	4	640	160
23	**3**	720	**240**
13	2	900	320

Power up partitioning shown in bold type.

*Multiply number of rows times number of columns for number of array items.

**Arrays limited to TI-59 only.

If, for example, you had a data array that requires a total of 5 rows and 10 columns, you have 5 x 10 = 50 data array items. By looking up the nearest larger number of data array items in the Array/Memory Chart, which is 53, you can see that you must partition with an N value of 6 to accommodate 50 array items. This happens to be the power-up partitioning for the TI-59, but the TI-58 must be repartitioned.

Entering Data

The program will allow you to input row and column numbers larger than the array specified in Step 2a of the User Instructions. You may specify any row number you desire for data operations as long as the memory partitioning is set to accommodate an array with the largest row number you use. (See the previous Array/Memory Chart). **Using a column number larger than set in Step 2a will cause mislocated data.** Sequential access to column positions with [C] such as in steps 3b, 5b or 15b is protected and a flashing display will occur if you go past the last column.

Data Array Computations

The array computations in Steps 6 through 14 of the User Instructions are simple to use; however, take care to follow key sequences explicitly. Overlooking a decimal point, number entry, minus sign, etc., can modify a lot of data.

The row operations of Steps 10, 11 and 12 allow you to mathematically combine two rows (or one row and a constant) column by column and place the result in a third row. Note that the only math functions available are +, −, ×, ÷, and y^x.

As previously mentioned, the program automatically assumes the last row and last column of the data array are reserved for row and column results. This means that the last row and column are generally unaffected by data array computations. The exceptions are the computations covered by Steps 10 through 12 of the User Instructions. The + and − computations operate on all columns (including the last column) of the rows specified and place the result in Row C. In ×, ÷, and y^x computations, the data items (except the last) in the row containing the result (Row C) are automatically summed and placed in the last column. Step 2b of the User Instructions offers the option to cancel the last-row/column isolation such that all columns columns of each row and all rows of each column are operated on the same way.

One feature of the shift operations in Steps 13 and 14 is worth mentioning. When data in one row (Row A) is shifted and placed in another row (Row B), the data items in the row containing the shift result (Row B) remain intact unless replaced directly by the shift. For example, consider the following two rows of data:

	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	2	3	8	13
Row 2	9	6	11	26

If Row 1 is shifted right one position and the result is placed in Row 2, the result would be:

	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	2	3	8	13
Row 2	9	2	3	26

The 9 remains in place in Row 2 because the right shift does not have a defined data item from Row 1 to place in Col. 1 of Row 2. The last column is not affected because of the automatic isolation of the last column. If flag 0 had been set in Step 2b, the 8 would have replaced the 26 in Row 2.

Recall Array Data

If a printer is not available, the data array may be recalled to the display item by item. If the printer is connected, the specified row number is printed and followed by the data items from column one through the last column. The printer is fully functional only when Fix 0, 2 or 9 is selected.

Solid State Software TI © 1978				
DATA ARRAYS				MU-07
Constant	R.C (INIT)	R.C [→STO →RCL]	Shift [→R →LI]	PRD [→R →C]
Row_A • Row_B	Row_C [→STO →RCL]	Data [→STO →RCL]	Growth	SUM [→R →C]

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
	ENTERING DATA			
1	Select program.		[2nd] [Pgm] 07	
2a	Enter total number of rows and columns in data array.	R.C(INIT)[1]	[2nd] [B′]	R.C
2b	To cancel last row/column isolation automatically set by [2nd] [B′].		[2nd] [StFlg] 0	
	Entering Array Data			
3a	Enter row number of data to be entered.	Row Number	[2nd] [C′]	0.
3b	Enter data items column by column in left to right order.	Data Item	[C]	Data Item
3c	Repeat 3a and 3b for each row of data.			
	Entering Linear Growth Data			
4a	Enter row number of data to be entered.	Row Number	[2nd] [C′]	0.
4b	Enter first data item in row (column 1).	Data Item	[C]	Data Item
4c	Enter growth rate as decimal percent.	Growth[2]	[D]	Last Item
	Entering Single Data Items			
5a	Enter row and column number of data item.	R.C	[2nd] [C′]	R.C
5b	Enter data item.	Data Item[3]	[C]	Data Item
	DATA ARRAY COMPUTATIONS			
	Row and Column Totals			
6a	Compute the total of a row.	Row Number	[E]	Row Total[4]
6b	Store row total in last column of row.	Row Total	[C]	Row Total
7a	Compute the total of a column.	Col. Number	[+/−] [E]	Col. Total[4]
7b	Store column total in last row of column.	Col. Total	[C]	Col. Total
	Row and Column Products			
8a	Compute the product of data items in a row.	Row Number	[2nd] [E′]	Row Product[4]
8b	Store product of row in last column of row.	Row Product	[C]	Row Product
9a	Compute the product of data items in a column.	Col. Number	[+/−] [2nd] [E′]	Col. Product[4]
9b	Store product of column in last row of column	Col. Product	[C]	Col. Product

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
	Row A □ Row B = Row C (□ = +, −, X, ÷, or y^x)			
10a	Enter numbers of rows A and B.	Row A.Row B[5]	[A]	0.
10b	Enter number of row C.	Row C[6]	[B]	0.
10c	Compute Row A □ Row B = Row C[10]		[SBR] [□]	0.
	Row A □ Constant = Row C (□ = +, −, X, ÷, or y^x)			
11a	Enter constant value.	Constant	[2nd] [A′]	Constant
11b	Enter number of row A.	Row A	[A]	0.
11c	Enter number of row C.	Row C[7]	[B]	0.
11d	Compute Row A □ Constant = Row C[10]		[SBR] [□]	0.
	Constant □ Row B = Row C (□ = +, −, X, ÷, or y^x)			
12a	Enter constant value.	Constant	[2nd] [A′]	Constant
12b	Enter number of row B.	.Row B[5]	[A]	0.
12c	Enter number of row C.	Row C[7]	[B]	0.
12d	Compute Constant □ Row B = Row C[10]		[SBR] [□]	0.
	Shift Right Row A, result in Row B[4]			
13a	Enter numbers of rows A and B.	Row A.Row B[5]	[A]	0.
13b	Enter positions to be shifted and perform shift.	No. Shifts[4,8]	[2nd] [D′]	0.
	Shift Left Row A, result in Row B[4]			
14a	Enter numbers of rows A and B.	Row A.Row B[5]	[A]	0.
14b	Enter positions to be shifted and perform shift.	No. Shifts[4,8]	[+/−] [2nd] [D′]	0.
	RECALL ARRAY DATA			
	Display Output (Printer Not Connected)			
15a	Enter row number, recall first data item.	Row Number	[+/−] [B]	First Data Item
15b	Recall next data item in row.		[C]	Next Data Item
15c	Repeat 15b for each column.			
16	To print all data items in a row.	Row Number[9]	[+/−] [B]	Last Data Item

NOTES:

1. Column numbers from 1 through 9 must have a leading zero. A 4 x 6 array is entered as 4.06 [2nd] [B′]. Initialization does not clear stored data; however, **changing array size after entering data will result in mislocated data.**
2. For example, to make each column increase by 50%, enter .5 [D]. This function enters growth data into each column except the last (unless last-row/column isolation has been cancelled in Step 2b.).
3. Next data item in the same row can be entered by simply repeating Step 5b.
4. Operation does not include or affect the content of the last row (or column) unless last-row/column isolation has been cancelled in Step 2b.
5. Row numbers 1 through 9 must have a leading zero, such as .01 for row 1, etc. If a shift operation has the same row number for Row A and Row B, it is only necessary to enter the Row A number.
6. This step may be skipped if the row numbers for Row B and Row C are the same.

NOTES:
7. This step may be skipped if Row C number is the same as the row number in the previous step.
8. Data items in Row B that are not replaced by the shift remain unchanged.
9. Use Fix 0, 2, or 9.
10. For X, ÷, and y^x computations, the sum of Row C is automatically placed in the last column unless last-row/column isolation has been cancelled in Step 2b.

Example: You are evaluating a project for your company and you need to prepare a project budget showing the gross profit under two different assumptions concerning unit sales. Unit variable cost is estimated to be $25 per unit, independent of the number of units produced. Total identifiable fixed cost is $25,000 per month. The projected sales performance is as follows for the two assumptions:

	Unit Sales	
Sales Price	**1st Month**	**Growth**
$40	2000	10%/month
$50	1800	5%/month

Compare the monthly and annual sales, cost, and gross profit figures under these two conditions. To determine the size of the data array necessary to handle this problem, first consider that you need to show 12 months plus an annual total. Thus, 13 columns are needed. The input data items are sales, variable cost and fixed cost. The output data needed is gross profit. Therefore, 4 rows should work for this problem. Note in the following solution to the problem that row 4 is used for intermediate total costs (the sum of variable costs and fixed costs). Then row 4 is used again for the gross profit.

ENTER	PRESS	DISPLAY	COMMENTS
6	[2nd] [Op] 17	XX.59	Set partitioning
	[2nd] [Pgm] 07	XX.59	Select program
4.13	[2nd] [B′]	4.13	4 x 13 Data Array
	[2nd] [Fix] 2	4.13	Fixed 2 Decimal
$40 PRODUCT			
1	[2nd] [C′]	0.00	Select row 1
40	[X]	40.00	
2000	[=]	80000.00	1st month sales
	[C]	80000.00	Input to column 1
.1	[D]	228249.34	Growth factor 10%
1	[E]	1710742.70	Total annual sales
	[C]	1710742.70	Store in last column
1	[+/–] [B]	1710742.70*	See tape 1
2	[2nd] [C′]	0.00	Select row 2
25	[X]	25.00	
2000	[=]	50000.00	1st month variable cost
	[C]	50000.00	Input to column 1
.1	[D]	142655.84	Growth factor 10%
2	[E]	1069214.19	Total variable cost
	[C]	1069214.19	Store in last column

MU-07

ENTER	PRESS	DISPLAY	COMMENTS
3	[2nd] [C′]	0.00	Select row 3
25000	[C]	25000.00	Monthly fixed cost
0	[D]	25000.00	Growth factor 0%
3	[E]	300000.00	Total fixed cost
	[C]	300000.00	Store in last column
2.03	[A]	0.00	Add rows 2 & 3 and put result in row 4
4	[B]	0.00	
	[SBR] [+]	0.00	
4	[+/–] [B]	1369214.19*	See tape 2
1.04	[A]	0.00	Subtract row 4 from row 1 and put result in row 4
4	[B]	0.00	
	[SBR] [–]	0.00	
4	[+/–] [B]	341528.51*	See tape 3
$50 PRODUCT			
1	[2nd] [C′]	0.00	Select row 1
50	[X]	50.00	
1800	[=]	90000.00	1st month sales
	[C]	90000.00	Input to column 1
.05	[D]	153930.54	Growth factor 5%
1	[E]	1432541.39	Total annual sales
	[C]	1432541.39	Store in last column
1	[+/–] [B]	1432541.39*	See tape 4
2	[2nd] [C′]	0.00	Select row 2
25	[X]	25.00	
1800	[=]	45000.00	1st month variable cost
	[C]	45000.00	Input to column 1
.05	[D]	76965.27	Growth factor 5%
2	[E]	716270.69	Total variable cost
	[C]	716270.69	Store in last column
2.03	[A]	0.00	Add rows 2 and 3 and put result in row 4
4	[B]	0.00	
	[SBR] [+]	0.00	
4	[+/–] [B]	1016270.69*	See tape 5
1.04	[A]	0.00	Subtract row 4 from row 1 and put result in row 4
4	[B]	0.00	
	[SBR] [–]	0.00	
4	[+/–] [B]	416270.69*	See tape 6

*If printer is not connected, the first data item is displayed. Use [C] to obtain remaining data items in row.

$50 PRODUCT

Month	Tape 1 SALES 1.00	ROW	Tape 2 COSTS 4.00	ROW	Tape 3 GROSS PROFIT 4.00	ROW
1	80000.00		75000.00		5000.00	
2	88000.00		80000.00		8000.00	
3	96800.00		85500.00		11300.00	
4	106480.00		91550.00		14930.00	
5	117128.00		98205.00		18923.00	
6	128840.80		105525.50		23315.30	
7	141724.88		113578.05		28146.83	
8	155897.37		122435.86		33461.51	
9	171487.10		132179.44		39307.66	
10	188635.82		142897.38		45738.43	
11	207499.40		154687.12		52812.27	
12	228249.34		167655.84		60593.50	
Total	1710742.70		1369214.19		341528.51	

$40 PRODUCT

Month	Tape 4 SALES 1.00	ROW	Tape 5 COSTS 4.00	ROW	Tape 6 GROSS PROFIT 4.00	ROW
1	90000.00		70000.00		20000.00	
2	94500.00		72250.00		22250.00	
3	99225.00		74612.50		24612.50	
4	104186.25		77093.13		27093.13	
5	109395.56		79697.78		29697.78	
6	114865.34		82432.67		32432.67	
7	120608.61		85304.30		35304.30	
8	126639.04		88319.52		38319.52	
9	132970.99		91485.49		41485.49	
10	139619.54		94809.77		44809.77	
11	146600.52		98300.26		48300.26	
12	153930.54		101965.27		51965.27	
Total	1432541.39		1016270.69		416270.69	

MU-07

Register Contents

R_{00}	Used	R_{05}	Used	R_{10}		R_{15}		R_{20}		R_{25}	
R_{01}	Used	R_{06}	Used	R_{11}		R_{16}		R_{21}		R_{26}	
R_{02}	Used	R_{07}	Data*	R_{12}		R_{17}		R_{22}		R_{27}	
R_{03}	Used	R_{08}		R_{13}		R_{18}		R_{23}		R_{28}	
R_{04}	Used	R_{09}		R_{14}		R_{19}		R_{24}		R_{29}	

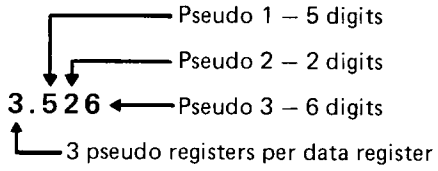
*R_{07} and above are used for data storage.

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm.	Spec. Func. Used	x≷t	Fix Decimal Format
A	0,3-6	1-6						
A′	2							
B(STO)	0,6+	6						
B(RCL)	0,2,6+	6						
B′	1,6	0		1				
C(STO)	0,4+	6						
C(RCL)	0	6						
C′	0,4,6+	6						
D	0,2-4,6+	0		1				
D′	0,3-6+	0,6						
E	0,1,3,4,6+	0,1,2,6	1	1				
E′	0,1,3,4,6+	0,1,2,6	1	1				
+	0-6	0-6		1				
−	0-6	0-6		1				
X	0-6	0-6		1				
÷	0-6	0-6		1				
y^x	0-6	0-6		1				

DATA PACKING

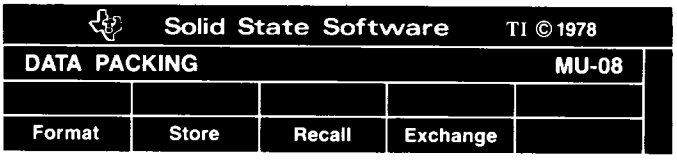
The effective number of available data registers can be increased by packing data. Using this program, each register starting with R_{04} is divided into a selected number of smaller registers as specified by a single format statement. For example, the following statement divides each data register into three pseudo registers and specifies the length of each pseudo register.



The lengths of higher numbered pseudo registers are also governed by this statement with the pattern being repeated for each three registers. Problems of correlating register number and length are avoided if all pseudo registers are assigned the same number of digits.

The format is flexible within the limits of 1 to 9 pseudo registers per data register. Maximum length of a pseudo register is 9 digits and the maximum combined length for all pseudo registers in a data register is 13 digits.

The program handles only integer values, and scaling of input values must be performed or programmed by the user. If a number is too large to fit the format, the most significant digits are truncated.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		[2nd] [Pgm] 08	
2	Enter format.[1]	Format	[A]	Format.
3a	Enter data.[2,3]	Data	[x≷t]	T-Register.
3b	Enter pseudo register number.	PR[4]		
	• Store.		[B]	Stored Value.
	• Recall.		[C]	Recalled Value.
	• Exchange.		[D]	Recalled Value.

NOTES:
1. Format is stored in R_{01} : N.xxx....Maximum length of a pseudo register is 9 digits. Maximum combined length is 13 digits.
2. Not necessary for Recall.
3. Only integer values may be entered. If the number entered is too large to fit the format, the most significant digits are truncated.
4. There is no pseudo register 0.

MU-08

Example: Set up format of 2 pseudo registers per data register. Make each pseudo register 5 digits. Store the following numbers in pseudo registers 1, 2, and 3, respectively: 12345, 23456, 34567. Recall these values; then exchange 2 with the value stored in pseudo register 2. Recall pseudo register 2.

ENTER	PRESS	DISPLAY	COMMENTS
	[2nd] [Pgm] 08		Select program.
2.55	[A]	2.55	Format statement.
12345	[x≷t] 1 [B]	12345.	Store in PR1
23456	[x≷t] 2 [B]	23456.	Store in PR2
34567	[x≷t] 3 [B]	34567.	Store in PR3
	1 [C]	12345.	Recall PR1
	2 [C]	23456.	Recall PR2
	3 [C]	34567.	Recall PR3
2	[x≷t] 2 [D]	23456.	Exchange PR2
	2 [C]	2.	Recall PR2

Register Contents

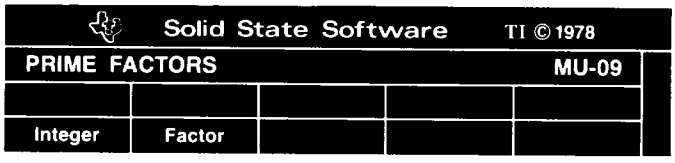
R_{00}	Used	R_{05}	R_{10}	R_{15}	R_{20}	R_{25}
R_{01}	Format	R_{06}	R_{11}	R_{16}	R_{21}	R_{26}
R_{02}	Pointer	R_{07}	R_{12}	R_{17}	R_{22}	R_{27}
R_{03}	Used	R_{08}	R_{13}	R_{18}	R_{23}	R_{28}
R_{04}		R_{09}	R_{14}	R_{19}	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
A	1							
B	0–3		1	2		DMS	x	
C	0–3		1	2		DMS	x	
D	0–3		1	2		DMS	x	

PRIME FACTORS

This program will determine all the prime factors of any positive integer. Note that any positive integer can be factored into a product of prime numbers.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		[2nd] [Pgm] 09	
2	Enter integer and find first prime factor.	Integer	[A]	Factor
3	Display next factor.[1]		[B]	Factor
4	Repeat step 3 until all factors are found.[2]		[B]	1.

NOTES:
1. Execution time increases with the magnitude of the number and the difference between prime factors.
2. A flashing 1 is displayed when all factors have been found.

MU-09

Example: Determine the prime factors of 987654321.

ENTER	PRESS	DISPLAY	COMMENTS	PRINTOUT
	[2nd] [Pgm] 09		Select program.	
987654321	[A]	3.	Enter integer and find first factor.	
	[B]	3.	Compute next factor.	
	[B]	17.	Compute next factor.	
	[B]	17.	Compute next factor.	
	[B]	379721.	Compute next factor. Note this calculation takes several minutes.	
	[B]	"1."	Flashing 1 indicates no more factors.	

Thus, $987654321 = (3)^2\,(17)^2\,(379721)$

Register Contents

R_{00}	R_{05}	R_{10}	Used	R_{15}	R_{20}	R_{25}
R_{01}	R_{06}	R_{11}	Used	R_{16}	R_{21}	R_{26}
R_{02}	R_{07}	R_{12}	Used	R_{17}	R_{22}	R_{27}
R_{03}	R_{08}	R_{13}		R_{18}	R_{23}	R_{28}
R_{04}	R_{09}	R_{14}		R_{19}	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	$x \gtrless t$	Fix Decimal Format
A	10-12	1,2		1			x	
B	10-12	1,2		1			x	

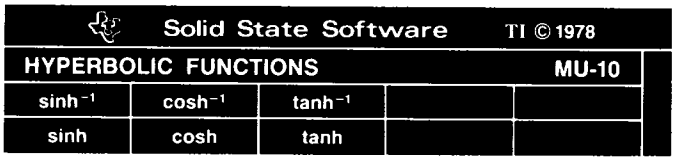
HYPERBOLIC FUNCTIONS

The program will compute the values of all the hyperbolic functions and the inverse hyperbolic functions. These functions are defined by the following formulas:

$$\sinh x = \frac{e^x - e^{-x}}{2} \qquad \sinh^{-1} x = \ln\left(x + \sqrt{x^2 + 1}\right)$$

$$\cosh x = \frac{e^x + e^{-x}}{2} \qquad \cosh^{-1} x = \sinh^{-1}\sqrt{x^2 - 1}$$

$$\tanh x = \frac{1 - e^{-2x}}{1 + e^{-2x}} \qquad \tanh^{-1} x = \tfrac{1}{2}\ln\left(\frac{1 + x}{1 - x}\right)$$



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		[2nd] [Pgm] 10	0.
2	Enter argument.	x	[A]	sinh x
3	Enter argument.	x	[2nd] [A′]	$\sinh^{-1}$ x
4	Enter argument.	x	[B]	cosh x
5	Enter argument.	x^1	[2nd] [B′]	$\cosh^{-1}$ x
6	Enter argument.	x	[C]	tanh x
7	Enter argument.	x^2	[2nd] [C′]	$\tanh^{-1}$ x

NOTES: 1. $x \geq 1$ for $\cosh^{-1}$
2. $|x| < 1$ for $\tanh^{-1}$

MU-10

Example: Find
1. sinh (.25) and $\sinh^{-1}$ [sinh (.25)]
2. cosh (.25) and $\cosh^{-1}$ [cosh (.25)]
3. tanh (.25) and $\tanh^{-1}$ [tanh (.25)]
4. $\cosh^{-1}$ (.25) and $\tanh^{-1}$ (2)

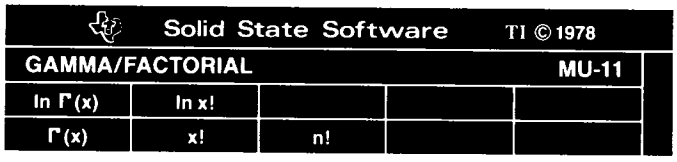
ENTER	PRESS	DISPLAY	COMMENTS	PRINTOUT
	[2nd] [Pgm] 10	0.	Select program.	
.25	[A]	.2526123168	sinh (.25)	
	[2nd] [A′]	0.25	$\sinh^{-1}$ (.2526123168)	
.25	[B]	1.0314131	cosh (.25)	
	[2nd] [B′]	0.25	$\cosh^{-1}$ (1.0314131)	
.25	[C]	.2449186624	tanh (.25)	
	[2nd] [C′]	0.25	$\tanh^{-1}$ (.2449186624)	
.25	[2nd] [B′]	"0.25"	Error – x<1	
	[CLR]	0	Clears error condition.	
2	[2nd] [C′]	".5493061443"	Error – x>1	
	[CLR]	0	Clears error condition.	

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
A				2				
A′				2				
B				2				
B′				2			x	
C				2			x	
C′				3			x	

GAMMA/FACTORIAL

This program will compute the value of the gamma function ($\Gamma(x)$) for $0 < x \leq 70$ and the ln $\Gamma(x)$ for $0 < x \leq 4.5535879 \times 10^{10}$. The program also computes x! for real numbers, including non-integers, ($0 \leq x \leq 69$) and computes n! for integers ($0 \leq n \leq 69$). In addition, ln x! can be computed for $0 \leq x \leq 4.5535879 \times 10^{10}$.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		**[2nd] [Pgm] 11**	
2a	Calculate $\Gamma(x)$[1]	x	**[A]**	$\Gamma(x)$
2b	Calculate ln $\Gamma(x)$[2]	x	**[2nd] [A′]**	ln $\Gamma(x)$
2c	Calculate x!3	x	**[B]**	x!
2d	Calculate ln x!4	x	**[2nd] [B′]**	ln x!
2e	Calculate n!5	n	**[C]**	n!

NOTES:
1. $0 < x \leq 70$
2. $0 < x \leq 4.5535879 \times 10^{10}$
3. $0 \leq x \leq 69$
4. $0 \leq x \leq 4.5535879 \times 10^{10}$
5. $0 \leq n \leq 69$ (Integers only)

MU-11

Example: Calculate $\Gamma(x)$ and $\ln\Gamma(x)$ for x = 6. Calculate x! and ln x! for x = 5. Then compute n! for n = 6.

ENTER	PRESS	DISPLAY	COMMENTS
	[2nd] [Pgm] 11		Select program.
6	**[A]**	120.	$\Gamma(6)$
6	**[2nd] [A′]**	4.787491743	$\ln\Gamma(6)$
5	**[B]**	120.	5!
5	**[2nd] [B′]**	4.787491743	ln 5!
5	**[C]**	120.	5!

Method Used

$\Gamma(x)$, $\ln\Gamma(x)$, x!, and ln x! are all calculated from the asymptotic series[1]:

$$\ln\Gamma(x) = \ln(x-1)! \approx (x-\tfrac{1}{2})\ln x - x + \tfrac{1}{2}\ln(2\pi) + \sum_{m=1}^{\infty} \frac{B_{2m}}{2m\,(2m-1)x^{2m-1}}$$

where B_{2m} are the Bernoulli numbers. For these calculations the sum is truncated at m = 5. The error caused by truncation is:

$$|\epsilon_{\ln}| \leqslant \frac{691}{360360\,x^{11}} \text{ for } \ln\Gamma(x) \text{ and } \ln x!$$

and

$$|\epsilon_{\Gamma(x)}| \leqslant e^{\ln\Gamma(x)}\,(e^{|\epsilon_{\ln}|} - 1) \text{ for } \Gamma(x) \text{ and } x!$$

where e is the base of natural logarithms and $\ln\Gamma(x)$ is the calculated value of the series.

For those values of x which yield unacceptably large error, one of the identities

(1) $\Gamma(x+1) = x\,\Gamma(x)$

or

(2) $(x+1)! = (x+1)x!$

should be used to enhance the accuracy of the calculation.

[1] *Handbook of Mathematical Functions,* Abramowitz and Stegun, National Bureau of Standards, 1964, p. 257.

For example:

$$(½)! = \frac{\sqrt{\pi}}{2} = .8862269255$$

Entering x = .5 and pressing [**B**] yields (½)! = .8862357531. Entering x = 4.5 and pressing [**B**] yields 52.34277779. Now repeatedly applying the identity given in (2) above, we obtain:

$$(.5)! = \frac{4.5!}{(4.5)(3.5)(2.5)(1.5)} = .8862269255$$

Note that one of the identities given above must be used to calculate $\Gamma(x)$ and x! for $x < 0$. Note that neither $\Gamma(x)$ nor x! are defined when x is a negative integer. Also, $\Gamma(0)$ is not defined.

Register Contents

R_{00}	R_{05}	R_{10} Used	R_{15}	R_{20}	R_{25}
R_{01}	R_{06}	R_{11}	R_{16}	R_{21}	R_{26}
R_{02}	R_{07}	R_{12}	R_{17}	R_{22}	R_{27}
R_{03}	R_{08}	R_{13}	R_{18}	R_{23}	R_{28}
R_{04}	R_{09} Used	R_{14}	R_{19}	R_{24}	R_{29}

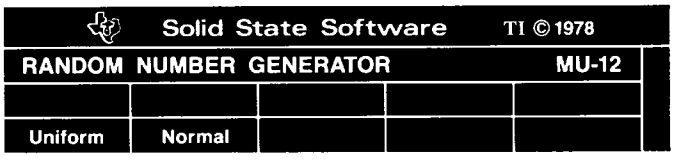
Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	$x \gtrless t$	Fix Decimal Format
A	10	0		2				
B	10	0		2				
C	9,10						x	
A′	10	0		2				
B′	10	0		2				

MU-12

RANDOM NUMBERS

With this program, you can let your calculator generate uniformly distributed random numbers in the range 0 to 0.99999. Also normally distributed random numbers are generated by using the inverse normal function of Program MU-13. These numbers will have a mean of zero and standard deviation of one.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		**[2nd] [Pgm] 12**	
2	Enter Seed ($0 < \text{Seed} \leq 199017$).	Seed	**[STO] 09**	Seed
3	Generate uniformly distributed numbers (one for each key push).		**[A]**	Random No.
4	Generate normally distributed numbers (one for each key push).		**[B]**	Random No.

Example: Using π as a seed, generate five uniformly distributed random numbers. Then generate five normally distributed random numbers.

ENTER	PRESS	DISPLAY	COMMENTS
	[2nd] [Pgm] 12		Select program.
	[2nd] [π] [STO] 09	3.141592654	Enter seed.
	[A]	0.88598	Uniformly distributed numbers.
	[A]	0.08556	
	[A]	0.54427	
	[A]	0.21789	
	[A]	0.83883	
	[B]	−.2657858899	Normally distributed numbers.
	[B]	−1.185003769	
	[B]	−0.886537315	
	[B]	−.0791204506	
	[B]	−.5627623434	

Register Contents

R_{00}	R_{05}	R_{10} Used	R_{15}	R_{20}	R_{25}
R_{01}	R_{06}	R_{11} Used	R_{16}	R_{21}	R_{26}
R_{02}	R_{07}	R_{12}	R_{17}	R_{22}	R_{27}
R_{03}	R_{08}	R_{13}	R_{18}	R_{23}	R_{28}
R_{04}	R_{09} Seed	R_{14}	R_{19}	R_{24}	R_{29}

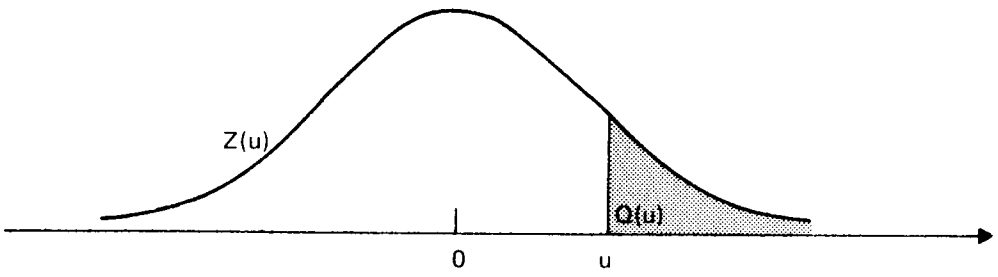
Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
A	9			2				
B	9-11	0	1	2	13		x	

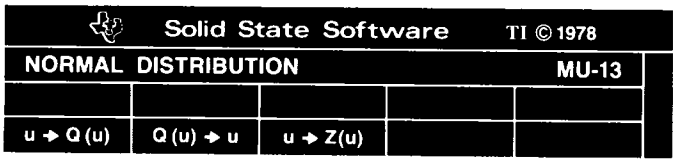
MU-13

NORMAL DISTRIBUTION

This program uses the standard normal distribution curve.



The program allows you to enter the normal variate (u) and will compute either Z(u) at that point or the area, Q(u), under the curve to the right of that point. Also, you may enter Q(u) and the program will compute u.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		[2nd] [Pgm] 13	
2a	Enter normal variate.	u[1]	[A]	Q(u)
2b	Enter Q(u).	Q(u)	[B]	u
2c	Enter normal variate.	u[1]	[C]	Z(u)

NOTES: 1. $|u| \leq 15.11$, display will flash for u outside this range.

Example: Compute Q(u) for u = 1, then use the Q(u) to recompute u. Also, compute Z(u) for u = 1.

ENTER	PRESS	DISPLAY	COMMENTS
	[2nd] [Pgm] 13		Select program.
1	[A]	.1586552596	Compute Q(u).
	[B]	.9999684605	Compute u (see note).
1	[C]	.2419707245	Compute Z(u).

NOTE: Because approximated values must be used for computation purposes, you may not always obtain the exact value you entered if you press [A] and [B] in succession.

Method Used

$$Z(u) = \frac{1}{\sqrt{2\pi}}\, e^{-u^2/2}$$

$$Q(u) = \frac{1}{\sqrt{2\pi}} \int_u^\infty e^{-t^2/2}\, dt$$

$$= Z(u)\,(b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5) + \epsilon(u)$$

where

$$|\epsilon(u)| < 7.5 \times 10^{-8}$$

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

$p = .2316419$
$b_1 = .319381530$
$b_2 = -.356563782$
$b_3 = 1.781477937$
$b_4 = -1.821255978$
$b_5 = 1.330274429$

MU-13

To compute u, given p = Q(u)

$$u = t - \frac{c_0 + c_1 t + c_2 t^2}{1 + d_1 t + d_2 t^2 + d_3 t^3} + \epsilon(p)$$

where

$$|\epsilon(p)| < 4.5 \times 10^{-4}$$

$$t = \sqrt{\ln \frac{1}{p^2}}$$

$c_0 = 2.515517$ $\quad$ $d_1 = 1.432788$
$c_1 = .802853$ $\quad$ $d_2 = .189269$
$c_2 = .010328$ $\quad$ $d_3 = .001308$

Reference: *Handbook of Mathematical Functions,* Abramowitz and Stegun, National Bureau of Standards, 1964, pp. 931-933.

Register Contents

R_{00}	R_{05}	R_{10} Used	R_{15}	R_{20}	R_{25}
R_{01}	R_{06}	R_{11} Used	R_{16}	R_{21}	R_{26}
R_{02}	R_{07}	R_{12}	R_{17}	R_{22}	R_{27}
R_{03}	R_{08}	R_{13}	R_{18}	R_{23}	R_{28}
R_{04}	R_{09}	R_{14}	R_{19}	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	$x \gtrless t$	Fix Decimal Format
A	10,11	0	1	3			x	
B	11	0		2			x	
C				1				

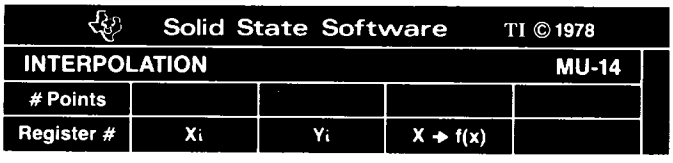
INTERPOLATION

This program fits an $(n-1)^{th}$ order polynomial to n input data points using Aitken's Method.[1] The resulting polynomial is then evaluated at an intervening point (x_0) to predict the value of $f(x_0)$.

Use as a Subroutine

User must store data in contiguous data registers in the order $x_1, x_2, \ldots, x_n; y_1, y_2, \ldots, y_n$; plus n additional working registers following y_n. User must also place address of x_1 in R_{10}, n in R_{11}, and x_0 in R_{12}. Value of $f(x_0)$ will be in the display upon return from the subroutine.

Reference: *Introduction to Numerical Methods,* Peter A. Stark, MacMillan, New York (1970), pp. 288-292.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		[2nd] [Pgm] 14	
2	Enter address of x_1 (See Note 1).	Register #	[A]	Register #
3	Enter number of (x,y) pairs	# Points	[2nd] [A′]	# Points
4a	Enter x value.	x_i	[B]	i
4b	Enter y value.	y_i	[C]	i
	Repeat steps 4a and 4b until the number of pairs in step 3 are entered.			
5	Enter value of x_0 and calculate $f(x_0)$	x_0	[D]	$f(x_0)$

NOTES: 1. Program requires three times the number of (x,y) pairs of contiguous data registers starting with R_{18} or above.

2. Step 5 may be repeated for interpolated values at additional points.

MU-14

Example: Interpolate for the area under the standard unit normal distribution for $z_0 = 0.8$ using five tabulated values (with a fourth degree polynomial).

ENTER	PRESS	DISPLAY	COMMENTS
	[2nd] [Pgm] 14		Select program.
18	[A]	18.	Enter x_1 address.
5	[2nd] [A′]	5.	Enter number of (x,y) pairs
.6	[B]	1.	z_1
.2257	[C]	1.	$f(z_1)$
.7	[B]	2.	z_2
.2580	[C]	2.	$f(z_2)$
.9	[B]	3.	z_3
.3159	[C]	3.	$f(z_3)$
1	[B]	4.	z_4
.3413	[C]	4.	$f(z_4)$
1.1	[B]	5.	z_5
.3643	[C]	5.	$f(z_5)$
.8	[D]	0.28811	Enter z_0, compute $f(z_0)$

Register Contents

R_{00}	R_{05}	R_{10} x_1 address	R_{15} Used	R_{20}	R_{25}
R_{01}	R_{06}	R_{11} # (x,y) pairs	R_{16} Used	R_{21}	R_{26}
R_{02}	R_{07}	R_{12} x_0	R_{17} Used	R_{22}	R_{27}
R_{03}	R_{08} Used	R_{13} Used	R_{18} *	R_{23}	R_{28}
R_{04}	R_{09} Used	R_{14} Used	R_{19}	R_{24}	R_{29}

*See User Instructions.

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
A	10,13							
A′	10,11,14							
B	10,13			1				
C	10,14			1				
D	12							
E	9-17			3				

ROOTS OF FUNCTIONS

One of the most widely used methods for approximating roots of a function of the form f(x) = 0 is the Newton-Raphson method. To use the program you must enter f(x) as a series of keystrokes. If you need help in entering a function as a subroutine, see *Personal Programming*, pages IV-46 – IV-51.

To run this program you must specify how close you want the result to be to the real root and enter this value as the maximum error (ϵ) desired. Also, you may enter a limit on the number of iterations.

For each root you must enter an initial guess, or approximation, of the root. You may find it helpful to draw a rough plot of f(x) in order to make good approximations. By limiting the number of iterations, you can protect yourself from those situations where this method fails to converge to a root.

MU-15

Solid State Software TI © 1978				
ROOTS OF FUNCTIONS				MU-15
	ϵ	(no limit)		
	Limit	$X_0 \rightarrow$ Root		

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Initialize		[RST]	
2	Enter learn mode.		[LRN]	000 00
3	Use [A′] as label.		[2nd] [Lbl] [2nd] [A′]	001 00 002 00
4	Enter f(x) as a series of keystrokes.[1]			
5	End f(x) with [INV] [SBR]		[INV] [SBR]	
6	Exit learn mode.		[LRN]	
7	Select program.		[2nd] [Pgm] 15	
8	Define maximum number of iterations allowed (optional).	Limit	[B]	Limit
9	Enter maximum error.	ϵ	[2nd] [B′]	ϵ
10a	Enter first approximation of root and calculate root to within ϵ, but do not exceed maximum number of iterations specified.[2] or	x_0	[C]	Root
10b	Enter first approximation of root and calculate root to within ϵ with no limit on the number of iterations.[2]	x_0	[2nd] [C′]	Root

NOTES:
1. Enter the subroutine using parentheses only. Do not use [=] or [CLR] in the subroutine for f(x).
2. The running time for the program depends on your choice for the initial guess, maximum error, and the nature of f(x).

Example 1: Find the roots of the function:

$$f(x) = x^3 - 4x^2 - x + 4$$

Use $\epsilon = 0.01$. Note that there are three real roots.

ENTER	PRESS	DISPLAY	COMMENTS	PRINTOUT
	[RST]	0		
	[LRN]	000 00	Enter learn mode.	
	[2nd] [Lbl]		Enter keystrokes for f(x)	
	[2nd] [A′]			
	[( ] [STO] 11			
	[X^2] [X] [RCL] 11			
	[−] 4 [X]			
	[RCL] 11 [X^2]			
	[−] [RCL] 11			
	[+] 4 [ )]			
	[INV] [SBR]	022 00		
	[LRN]	0	Exit learn mode.	
	[2nd] [Pgm] 15	0.	Select program.	
10	[B]	10.	Enter limit.	
.01	[2nd] [B′]	0.01	Enter ϵ.	
5	[C]	4.00000018	Enter x_0 and calculate 1st root (with limit).	
2	[2nd] [C′]	1.	Enter x_0 and calculate 2nd root (without limit).	
1.5	[+/−] [2nd] [C′]	−1.000047626	Enter x_0 and calculate 3rd root (without limit).	

Example 2: This program can be used to find the vertical asymptotes of a function such as:

$$g(x) = \frac{x - 1}{x^2\,(x-3)}$$

by finding the roots of

$$f(x) = \frac{1}{g(x)} = \frac{x^2\,(x-3)}{x - 1}$$

Find the asymptotes of g(x). Use $\epsilon = 0.001$.

ENTER	PRESS	DISPLAY	COMMENTS	PRINTOUT
	[RST]	0		
	[LRN]	000 00	Enter learn mode.	
	[2nd] [Lbl]		Enter keystrokes for f(x).	
	[2nd] [A′]			
	[( ] [STO] 11 [x^2]			
	[X] [( ] [RCL] 11			
	[−] 3 [ )] [÷]			
	[( ] [RCL] 11			
	[−] 1 [ )] [ )]			
	[INV] [SBR]	022 00		
	[LRN]	0	Exit learn mode.	
	[2nd] [Pgm] 15	0.	Select program.	
25	[B]	25.	Enter limit.	
.001	[2nd] [B′]	0.001	Enter ϵ.	
.5	[+/−] [C]	−.0007703253	Enter x_0 and find 1st root (asymptote) (with limit)	
2.5	[C]	3.000000028	Enter x_0 and find 2nd root (asymptote) (with limit)	

Method Used

An initial estimate, x_0, of a root of the equation f(x) = 0 is made. An improved estimate of the root can then be made as:[1]

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

In general, the expression:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

may be used to approximate a root of f(x) = 0, so that $x_{n+1} - x_n < \epsilon$.

The following approximations are used by the program to compute f'(x).

$$f'(x) \approx \frac{f[x(1 + 10^{-9})] - f[x(1 - 10^{-9})]}{2 \times 10^{-9}} \qquad x \neq 0$$

$$f'(x) \approx \frac{f(10^{-9}) - f(-10^{-9})}{2 \times 10^{-9}} \qquad x = 0$$

[1] *Handbook of Engineering Fundamentals,* Ovid W. Eshbach, John Wiley & Sons, Inc., New York, 1954, pp. 2-16.

Register Contents

R_{00}	R_{05}		R_{10} Used	R_{15} Used	R_{20}	R_{25}
R_{01}	R_{06}		R_{11}	R_{16}	R_{21}	R_{26}
R_{02}	R_{07}		R_{12} ϵ	R_{17}	R_{22}	R_{27}
R_{03}	R_{08}		R_{13}	R_{18} Used	R_{23}	R_{28}
R_{04}	R_{09}	Limit	R_{14}	R_{19} Used	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
B	9							
C	9-11	0	1	4*				
C′	9-11	0	1	4*				

*Has two open parentheses when user subroutine is called.

MU-16

MINIMAX

Given a function, f(x), this program finds those values of x (called critical values) for which f(x) has a maximum or a minimum. To run the program, you must first enter the function f(x) as a series of keystrokes. Then you must enter a beginning value, x_0, and a positive search increment, Δx.

Searching is always performed in the positive direction and is begun at the point $x_0 + \Delta x$. You should be sure that Δx is less than the minimum distance between any two maximum or minimum points to avoid skipping over one or both points. Maximum and minimum points are found and identified by a change in sign of the value of f'(x). Thus, this program will not find inflection points. Also, note that $x_0 + \Delta x$ must be less than the critical value and Δx must be positive.

You can enter a limit on the number of iterations the program will go through to locate a critical point. This will protect against infinite looping if a critical point is not found. If you enter a limit and a critical point is not located, when the limit number of iterations is reached the program returns a flashing display containing the largest value of x obtained.

Once a critical value has been found, you can also compute the value of the function at that point. You can find other critical values by simply entering a new x_0. The old values of Δx and the limit will be used again unless you enter new values.

Solid State Software TI ©1978				
MINIMAX				MU-16
X → f(x)		(no limit)	(no limit)	
Limit	Δx	$X_0 \rightarrow X_{CRIT}$	→ Next X_{CRIT}	

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Initialize.		[RST]	
2	Select learn mode.		[LRN]	000 00
3	Use A′ as a label.		[2nd] [Lbl] [2nd] [A′]	001 00 002 00
4	Enter f(x) as a series of keystrokes. Do not use [=] or [CLR].			
5	End f(x) with [INV] [SBR].		[INV] [SBR]	xxx 00
6	Leave learn mode.		[LRN]	
7	Select program.		[2nd] [Pgm] 16	
8	Define maximum number of iterations allowed (optional).	LIMIT[1]	[A]	LIMIT
9	Enter search increment.	Δx[2]	[B]	Δx
10a	Enter starting value of x and find next critical value with LIMIT number of iterations.	x_0[3]	[C][5] [2nd] [A′][7] [x≷t]	x_{CRIT} $f(x_{CRIT})$ −1 or 1[4]
	or			
10b	Enter starting value of x and find next critical value.	x_0[3]	[2nd] [C′][6] [2nd] [A′][7] [x≷t]	x_{CRIT} $f(x_{CRIT})$ −1 or 1[4]
11a	Find next larger critical value starting search from previous critical value.		[D][5] [2nd] [A′][7] [x≷t]	x_{CRIT} $f(x_{CRIT})$ −1 or 1[4]
	or			
11b	Find next larger critical value starting search from previous critical value.		[2nd] [D′][6] [2nd] [A′][7] [x≷t]	x_{CRIT} $f(x_{CRIT})$ −1 or 1[4]

NOTES:

1. LIMIT number of iterations are attempted for each critical value search.
2. Δx must be small enough to avoid "skipping over" a critical value of x. Δx must be greater than zero.
3. $x_0 + \Delta x$ must be less than critical value. Search begins at $x_0 + \Delta x$.
4. −1 indicates $f(x_{CRIT})$ is a maximum.
   1 indicates $f(x_{CRIT})$ is a minimum.
5. Calculate with LIMIT.
6. Calculate without LIMIT.
7. x must be in the display to compute f(x).

Example: Find the maxima and minima for the function $f(x) = x^3 - x^2 - x + 2$. Maximum occurs at $x = -1/3$, minimum at $x = 1$.

ENTER	PRESS	DISPLAY	COMMENTS
	[RST]		Initialize.
	[LRN]	000 00	Select learn mode.
	[2nd] [Lbl] [2nd] [A′]	002 00	Use A′ as label.
	[(] [STO] 20 [x^2]	006 00	Enter f(x) as a series of
	[x] [RCL] 20 [−]	010 00	keystrokes.
	[RCL] 20 [x^2] [−]	014 00	
	[RCL] 20 [+] 2	018 00	
	[)] [INV] [SBR]	020 00	End with [INV] [SBR].
	[LRN]		Exit learn mode.
	[2nd] [Pgm] 16		Select program.
25	[A]	25.	Enter LIMIT.
.5	[B]	0.5	Enter Δx.
1	[+/−] [C]	−.3330078125	Enter x_0 and compute x_{CRIT} (with LIMIT)
	[2nd] [A′]	2.185184973	Compute $f(x_{CRIT})$
	[x≷t]	−1.	x_{CRIT} is maximum
	[2nd] [D′]	1.	Find next critical point (without LIMIT)
	[2nd] [A′]	1.	Compute $f(x_{CRIT})$
	[x≷t]	1.	x_{CRIT} is minimum.

Example: Determine the critical values of $f(x) = (1 + x)/x$. This function has no maxima or minima, but does have a horizontal asymptote. The approximate value of the asymptote can be found using this program and the LIMIT option.

ENTER	PRESS	DISPLAY	COMMENTS
	[RST]		Initialize.
	[LRN]	000 00	Select learn mode.
	[2nd] [Lbl] [2nd] [A′]	002 00	Use A′ as a label.
	[(] [(] [STO] 20	006 00	Enter f(x) as a series of
	[+] 1 [)] [÷]	010 00	keystrokes.
	[RCL] 20 [)]	013 00	
	[INV] [SBR]	014 00	End with [INV] [SBR].
	[LRN]		Exit learn mode.
	[2nd] [Pgm] 16		Select program.
10	[A]	10.	Enter LIMIT.
100	[B]	100.	Enter Δx.
100	[C]	1200. (flashing)	Enter x_0 and compute x_{CRIT} (with LIMIT).
	[CE]	1200.	Stops flashing.
	[2nd] [A′]	1.000833333	Compute f(1200).
	[x≷t]	−1	Indicates approaching a maximum asymptote.

Method Used

Given a function, f(x), this program locates critical values of x by finding those values of x for which f'(x) = 0. The following approximations are used for f'(x):

$$f'(x) \approx \frac{f[x(1 + 10^{-9})] - f[x(1 - 10^{-9})]}{x(2 \times 10^{-9})} \qquad \text{if } x \neq 0$$

$$f'(x) \approx \frac{f(10^{-9}) - f(-10^{-9})}{2 \times 10^{-9}} \qquad \text{if } x = 0$$

Register Contents

R_{00}		R_{05}		R_{10}	$x_0 \rightarrow$critical x	R_{15}	Used	R_{20}	R_{25}
R_{01}		R_{06}		R_{11}	Used	R_{16}	Used	R_{21}	R_{26}
R_{02}		R_{07}		R_{12}	Δx	R_{17}	Used	R_{22}	R_{27}
R_{03}		R_{08}		R_{13}	Used	R_{18}	Used	R_{23}	R_{28}
R_{04}		R_{09}	Used	R_{14}		R_{19}	Used	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm.	Spec. Func. Used	$x \gtrless t$	Fix Decimal Format
A	17							
B	12							
C	9-13,15-19	0	2	4*			x	
D	9-13,15-19	0	2	4*			x	
A'			1					
C'	9-13,15-19	0	2	4*			x	
D'	9-13,15-19	0	2	4*			x	

*Maximum of two open parentheses when user subroutine is called.

MU-17

ROMBERG INTEGRATION

One of the most powerful and efficient techniques of numerical integration is a systematic procedure called Romberg Integration. This program allows you to choose a desired accuracy for approximation of the integral:

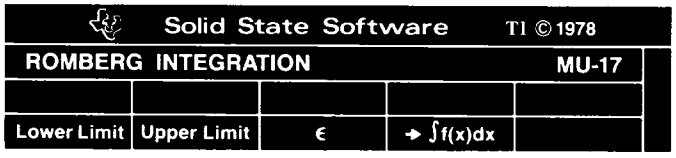
$$\int_a^b f(x)\,dx$$

To run this program, you must enter a subroutine for computing f(x). For details on writing programs for use as subroutines, see *Personal Programming*, pp. IV-46 – IV-51.

Romberg integration applies the trapezoidal rule, then uses a technique called Richardson extrapolation to help eliminate the error terms. The extrapolated values will converge to the correct answer faster than those values computed by the trapezoidal rule only. This program uses the relative convergence criterion for determining when the desired accuracy has been reached. Thus, if we have two successive values, R_{n-1} and R_n, the program stops when

$$\left| \frac{R_n - R_{n-1}}{R_n} \right| < \epsilon.$$

The relative convergence criterion is better than the absolute criterion because it looks for a percentage change in the result rather than for a change in a specified digit. This prevents drifting away from the correct answer due to roundoff error.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Initialize.		[RST]	0.
2	Enter learn mode.		[LRN]	000 00
3	Use A′ as a label.		[2nd] [Lbl]	001 00
4	Enter keystrokes for f(x). Assume x is in the display. Do not use [=] or [CLR] in subroutine.			
5	End with [INV] [SBR].		[INV] [SBR]	xxx 00
6	Exit learn mode.		[LRN]	0.
7	Select program.		[2nd] [Pgm] 17	0.
8	Enter lower limit.	a	[A]	a
9	Enter upper limit.	b	[B]	b
10	Enter accuracy limit.	ϵ	[C]	0.
11	Evaluate integral.		[D]	$\int_a^b f(x)\,dx$

Example: Evaluate

$$I = \int_0^5 (4x^3 - \frac{3x^2}{2} + 4x - 2)\,dx$$

ENTER	PRESS	DISPLAY	COMMENTS	PRINTOUT
	[RST]	0		
	[LRN]	000 00	Enter learn mode.	
	[2nd] [Lbl] [2nd] [A′]	002 00	Keystrokes for f(x).	
	[STO] 1 [( ] [x^2] [x]			
	[RCL] 1 [x] 4			
	[−] 3 [X] [RCL] 1			
	[x^2] [÷] 2 [+] 4			
	[X] [RCL] 1 [−] 2			
	[ )] [INV] [SBR]	028 00		
	[LRN]	0	Exit learn mode.	
	[2nd] [Pgm] 17	0.	Select program.	
0	[A]	0.	Lower limit, a.	
5	[B]	5.	Upper limit, b.	
1	[C]	0.	ϵ	
	[D]	602.5	I	

MU-17

Method Used

The trapezoidal rule estimates are:

$$R_{n,k} = \frac{\Delta x}{2}\left[f(a) + f(b) + 2\sum_{j=1}^{n} f\,(a + j\,\Delta k)\right]$$

where

$$\Delta x = \frac{b - a}{2^{k-1}} \qquad \text{and} \qquad n = 2^{k-1} - 1$$

Thus,

$$R_{1,1} = \frac{b - a}{2}\,[f(a) + f(b)]$$

$$R_{1,2} = \frac{b - a}{4}\left[f(a) + f(b) + 2f\left(a + \frac{b - a}{2}\right)\right]$$

$$R_{1,3} = \frac{b - a}{8}\left[f(a) + f(b) + 2f\left(a + \frac{b - a}{4}\right) + 2f\left(a + \frac{b - a}{2}\right) + 2f\left(a + \frac{3(b - a)}{4}\right)\right]$$

etc.

Note that

$$R_{1,2} = \frac{R_{1,1}}{2} + \frac{b - a}{2}\left[f\left(a + \frac{b - a}{2}\right)\right]$$

$$R_{1,3} = \frac{R_{1,2}}{2} + \frac{b - a}{4}\left[f\left(a + \frac{b - a}{4}\right) + f\left(a + \frac{3(b - a)}{4}\right)\right]$$

and so on.

Therefore, each approximation by the trapezoidal rule can be obtained from the preceding approximation without recomputing f(x) at any points where it has already been computed.

The Richardson extrapolation is computed as:

$$R_{n,k} = \frac{1}{4^{n-1} - 1}\,(4^{n-1} R_{n-1,k+1} - R_{n-1,k})$$

Thus, when n = 2

$$R_{2,1} = \frac{1}{3}\,(4R_{1,2} - R_{1,1})$$

$$R_{2,2} = \frac{1}{3}\,(4R_{1,3} - R_{1,2}).$$

Reference: *Numerical Methods,* Robert W. Hornbeck, Quantum Publishers, Inc., 1975, pp. 150-155.

Register Contents

R_{00}	R_{05}	R_{10} a	R_{15} Loop counter	R_{20}	R_{25}
R_{01}	R_{06}	R_{11} b	R_{16} Sum	R_{21}	R_{26}
R_{02}	R_{07}	R_{12} Used	R_{17} Counter(j−n)	R_{22}	R_{27}
R_{03}	R_{08} Pointer($R_{n,k}$)	R_{13} Δx	R_{18} Data	R_{23}	R_{28}
R_{04}	R_{09} Counter (j)	R_{14} x	R_{19}	R_{24}	R_{29}

Values for $R_{n,k}$ are stored starting with R_{18}.*

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm.	Spec. Func. Used	x≷t	Fix Decimal Format
A	10							
B	11							
C							x	
D	8-17,18+*		1	2		DMS	x	

*The number of data registers required by this program is dependent upon the function and ϵ. As a rule, 30 data registers should be sufficient; however, if more data registers are needed, the best estimate possible using the allocated space is flashed in the display. The latest ϵ is left in the T-register.

If the partitioning is set at 20 registers, $R_{2,1}$ is flashed in the display. The value in the T-register is meaningless at this point.

DIFFERENTIAL EQUATIONS

Runge-Kutta methods are a particular set of numerical techniques used for solving initial-value problems. This program uses a fourth-order Runge-Kutta method to solve differential equations of the type $y' = f(x,y)$ and $y'' = f(x,y,y')$.

Numerical methods consist of finding approximate solutions at particular points $x_0, x_1, \ldots, x_n$ where the difference between successive x-values is a constant, w ($x_{n+1} - x_n = w$). The value for w is chosen by the user with a smaller w generally yielding a more accurate solution. In this program, w is determined by the number of divisions, n, that the user chooses. Usually, w will be made smaller by increasing n.

To use this program, you must enter a subroutine which defines y' or y'' and the initial values for x_0, y_0 or x_0, y_0, y_0'. You must also enter the number of divisions, n, and the value of the independent variable, $\bar{x}$, for which a solution is desired.

Solid State Software TI © 1978				
DIFFERENTIAL EQUATIONS				MU-18
				$\bar{X} \rightarrow \bar{Y}$ (2nd)
X_0	Y_0	Y_0'	n	$\bar{X} \rightarrow \bar{Y}$ (1 st)

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Initialize.		[RST]	0.
2	Enter learn mode.		[LRN]	000 00
3	Use [A′] as a label.		[2nd] [Lbl]	001 00
4	Enter f(x,y) or f(x,y,y′) as a series of keystrokes. Do not use [=] or [CLR]. Use register 10 for x, register 11 for y, and register 12 for y′.		[2nd] [A′]	002 00
5	End with [INV] [SBR].		[INV] [SBR]	xxx 00
6	Exit learn mode.		[LRN]	0.
7	Select program.		[2nd] [Pgm] 18	0.
8a	Enter initial x.	x_0	[A]	x_0
8b	Enter initial y.	y_0	[B]	y_0
8c	Enter initial y′, if using a second-order equation.	y_0'	[C]	y_0'
9	Enter number of divisions.	n	[D]	n
10	To solve $y' = f(x,y)$: Enter value of independent variable for which the solution is desired.	$\bar{x}$	[E]	$\bar{y}$
	or			
11a	To solve $y'' = f(x,y,y')$: Enter value of independent variable for which the solution is desired.	$\bar{x}$	[2nd] [E′]	$\bar{y}$
11b	Display slope at $\bar{x}$.		[x≷t]	$\bar{y}'$

MU-18

Example 1: Find the solution for the equation:

$$y' = \frac{1 + y^2}{1 + x^2}$$

at $\bar{x} = 3$ with initial conditions: $x_0 = 2$, $y_0 = 1$.

ENTER	PRESS	DISPLAY	COMMENTS	PRINTOUT
	[RST]	0.	Initialize.	
	[LRN]	000 00	Enter learn mode.	
	[2nd] [Lbl] [2nd] [A′]	002 00	Keystrokes for $y = \frac{(1 + y^2)}{(1 + x^2)}$	
	[( ] [( ] 1 [+] [RCL] 11 [x^2] [ )] [÷] [( ] 1 [+] [RCL] 10 [x^2] [ )] [ )]	019 00		
	[INV] [SBR]	020 00	End with [INV] [SBR]	
	[LRN]	0.	Exit learn mode.	
	[2nd] [Pgm] 18	0.	Select program.	
2	[A]	2.	x_0	
1	[B]	1.	y_0	
16	[D]	16.	n	
3	[E]	1.333333338	Enter $\bar{x}$ and calculate $\bar{y}$. Note: It will take several minutes to complete the calculation.	

Example 2: Find the solution for the equation:

$$y'' = e^{2x}$$

at $\bar{x} = 1$ with the initial conditions: $x_0 = 0$, $y_0 = 1$, $y_0' = 1$.

ENTER	PRESS	DISPLAY	COMMENTS	PRINTOUT
	[RST]	0.	Initialize.	
	[LRN]	000 00	Enter learn mode.	
	[2nd] [Lbl] [2nd] [A′] [( ] [RCL] 10 [X] 2 [ )] [INV] [ln X]	002 00	Keystrokes for $y'' = e^{2x}$	
	[INV] [SBR]	011 00	End with [INV] [SBR]	
	[LRN]	0.	Exit learn mode.	
	[2nd] [Pgm] 18	0.		
0	[A]	0.	x_0	
1	[B]	1.	y_0	
1	[C]	1.	y_0'	
8	[D]	8.	n	
1	[2nd] [E′]	3.097256869	Enter $\bar{x}$ and calculate $\bar{y}$.	
	[x≷t]	4.194532374	$\bar{y}'$	

Method Used

For finding the solution of $y' = f(x,y)$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$k_1 = wf(x_n, y_n)$$

$$k_2 = wf(x_n + \frac{w}{2}, y_n + \frac{k_1}{2})$$

$$k_3 = wf(x_n + \frac{w}{2}, y_n + \frac{k_2}{2})$$

$$k_4 = wf(x_n + w, y_n + k_3)$$

$$w = \frac{\bar{x} - x_0}{n}$$

For finding the solution of $y'' = f(x,y,y')$

$$y_{n+1} = y_n + w[y'_n + \frac{1}{6}(k_1 + k_2 + k_3)]$$

$$y'_{n+1} = y'_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$k_1 = wf(x_n, y_n, y'_n)$$

$$k_2 = wf(x_n + \frac{w}{2}, y_n + \frac{w}{2}y'_n + \frac{w}{8}k_1, y'_n + \frac{k_1}{2})$$

$$k_3 = wf(x_n + \frac{w}{2}, y_n + \frac{w}{2}y'_n + \frac{w}{8}k_1, y'_n + \frac{k_2}{2})$$

$$k_4 = wf(x_n + w, y_n + w\left(y'_n + \frac{k_3}{2}\right), y'_n + k_3).$$

Reference: *Handbook of Mathematical Functions,* Abramowitz and Stegun, National Bureau of Standards, 1964, p. 896.

MU-18

Register Contents

R_{00}	R_{05}	R_{10} x_i	R_{15} w	R_{20}	R_{25}
R_{01}	R_{06}	R_{11} y_i	R_{16} Used	R_{21}	R_{26}
R_{02}	R_{07}	R_{12} y_i'	R_{17} Used	R_{22}	R_{27}
R_{03}	R_{08}	R_{13} Used	R_{18} Used	R_{23}	R_{28}
R_{04}	R_{09} n	R_{14} Used	R_{19} Used	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm.	Spec. Func. Used	x≷t	Fix Decimal Format
A	10							
B	11,13							
C	12,14							
D	19							
E	9-19	0	2	2				
E′	9-19	0	2	2				

DISCRETE FOURIER SERIES

A periodic function f(t) can be expressed as a sum of sines and cosines by the Fourier series.

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t)$$

where:

$$a_n = \frac{2}{T}\int_0^T f(t) \cos n\omega t \, dt \qquad n = 0,1,2,\ldots$$

$$b_n = \frac{2}{T}\int_0^T f(t) \sin n\omega t \, dt \qquad n = 1,2,\ldots$$

$\omega = 2\pi f$ = radian frequency

$T = 1/f$ = period of f(t)

This program uses discrete versions of the above formulas to calculate the coefficients a_n, b_n of the Fourier series.

The discrete formulas are

$$a_j = \frac{2}{N}\sum_{k=1}^{N} Y_K \cos\left(\frac{2\pi Kj}{N}\right)$$

$$b_j = \frac{2}{N}\sum_{k=1}^{N} Y_K \sin\left(\frac{2\pi Kj}{N}\right)$$

where:

J = order of the coefficient to be calculated

$Y_K = f(t_K)$

N = 1,2,...

$N \geqslant J > 0$

NOTE: N should be at least twice as large as the expected harmonic of the fundamental frequency of f(t)

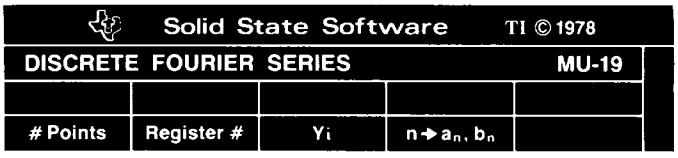
This program calculates one set of a_n, b_n for each execution.

MU-19

Use as a Subroutine

If this program is used as a subroutine, the user must store the appropriate values in registers 10, 11, and 12. The user must also store the input functional values in contiguous registers starting with the register whose address is in R_{11}. The value of a_n will be in the display register and b_n will be in the T-register upon return from the subroutine.

Reference: *Higher Mathematics for Engineers and Physicists,* Sokolnikoff, McGraw-Hill, 1941.



USER INSTRUCTIONS

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		**[2nd] [Pgm] 19**	
2	Enter number of equally spaced observations from f(t).	# Points	**[A]**	# Points
3	Enter Y_1 address.[1]	Register #	**[B]**	Register #
4	Enter Y_K, start with Y_1 and repeat Step 4 until the number of values entered in Step 2 have been entered.	Y_K	**[C]**	Y_K
5	Enter the order of coefficients to be calculated.	n	**[D]** **[x≷t]**	a_n b_n

NOTES: 1. The value entered represents the first register alloted to data storage, Register 16 or above.

Example: Given the following 12 functional values, calculate the Fourier series coefficients a_n and b_n for n = 0 through n = 6. It should be noted that n should never be greater than the number of functional values entered.

K	Y_K	K	Y_K
1	2.0	7	2.7
2	3.2	8	2.4
3	3.7	9	0
4	2.5	10	–3.3
5	1.2	11	–3.3
6	1.5	12	0

ENTER	PRESS	DISPLAY	COMMENTS
	[2nd] [Pgm] 19		Select program.
12	[A]	12.	Number of functional values.
16	[B]	16.	Y_1 address
2	[C]	2.	Y_1
3.2	[C]	3.2	Y_2
3.7	[C]	3.7	Y_3
2.5	[C]	2.5	Y_4
1.2	[C]	1.2	Y_5
1.5	[C]	1.5	Y_6
2.7	[C]	2.7	Y_7
2.4	[C]	2.4	Y_8
0	[C]	0.	Y_9
3.3	[+/–] [C]	–3.3	Y_{10}
3.3	[+/–] [C]	–3.3	Y_{11}
0	[C]	0.	Y_{12}
0	[D]	2.1	a_0
	[x≷t]	0.	b_0 (always 0)
1	[D]	–1.417222017	a_1
	[x≷t]	1.885961277	b_1
2	[D]	–0.55	a_2
	[x≷t]	1.905255888	b_2
3	[D]	.5833333333	a_3
	[x≷t]	.0166666667	b_3
4	[D]	0.25	a_4
	[x≷t]	.0577350269	b_4
5	[D]	.0838886833	a_5
	[x≷t]	.0192946108	b_5
6	[D]	0.	a_6
	[x≷t]	0.	b_6

MU-19

Register Contents

R_{00}	R_{05}	R_{10} N	R_{15} Used	R_{20}	R_{25}
R_{01}	R_{06}	R_{11} Y_1 address	R_{16}	R_{21}	R_{26}
R_{02}	R_{07}	R_{12} 2π n/N	R_{17}	R_{22}	R_{27}
R_{03}	R_{08} Pointer	R_{13} a_n accum.	R_{18}	R_{23}	R_{28}
R_{04}	R_{09} Used	R_{14} b_n accum.	R_{19}	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
A	10							
B	8,11							
C	8							
D	8-15			2			x	

CALCULATOR STATUS

This program contains an assortment of subroutines which allow you to determine the status of the following calculator operations:

1. Flags set
2. Partitioning set
3. Angular mode setting
4. Fix-decimal setting
5. Number of parentheses opened
6. Printer connected

In addition to being able to individually determine the status of the above operations, you can store the coded value of the first four status operations in data register R_{00} by pressing [A]. By storing the contents of R_{00} in another register, you can save a particular calculator status. Then, at a later time you can store the coded status value back in R_{00} and press [2nd] [A'] to force the calculator to assume the status set by the status code.

Conditions and Limitations

Since the status determining subroutines must interact with the operations of the calculator, care must be taken to prevent accidental change or loss of information. The following description of each status subroutine should be reviewed carefully before using it. Also, be sure to review the Program Details at the end of this program description.

1. **Save Calculator Status.** Pressing [A] assembles the status of all flags, the partitioning, angular mode and fix-decimal setting, and codes the status into data register R_{00}. Since other subroutines and programs also use R_{00}, it is important to move the contents of R_{00} to another data register for longer term storage. IMPORTANT: Partitioning with N = 0 (no data registers) is not permitted since R_{00} must save the status code. The status code stored in R_{00} uses all 13 digits of the register. Be sure the guard digits are maintained when moving the status code data. See pages V5 and C-1 of your Personal Programming manual for additional information about guard digits.

2. **Initialize Calculator Status.** Pressing [2nd] [A'] uses the status code in R_{00} to set flags, partitioning, angular mode and fix-decimal as they existed when [A] was pressed to produce the status code. IMPORTANT: Pressing [2nd] [A'] without a valid status code in R_{00} may cause loss of program or memory data. This initialization subroutine does not reset flags already set. Therefore, you must use [RST] or otherwise reset any flags already set prior to using this subroutine to ensure exact duplication of the status code in R_{00}.

3. **Flags Set Status.** Pressing [B] causes the numbers of the flags currently set (or high) to be displayed. A display of "0." indicates that no flags are set and "0.0" indicates that only flag 0 is set. Flag numbers 1 through 9 are indicated by number to the left of the decimal. For example, if flags 7, 4, 2 and 0 are set when [B] is pressed, 742.0 will be

displayed. IMPORTANT: The flag 0 "set" indication is produced by a fix-decimal setting of one. Since checking flag status may affect the fix-decimal setting, check fix-decimal status before using [B]. Be sure to restore the original fix-decimal setting or use [2nd] [Fix] 9 to return normal floating point display when a zero appears to the right of the decimal after pressing [B].

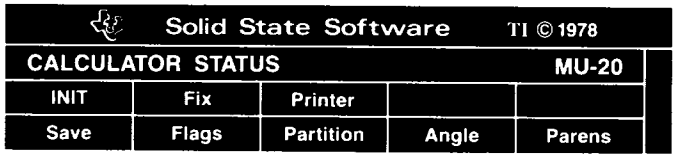
4. **Fix-Decimal Status.** Pressing [2nd] [B'] displays the current number of decimal places selected for display. The display of "9." indicates floating decimal selection.

5. **Partitioning Status.** Pressing [C] displays partitioning status as a number (N) between "0." and "6." for the TI-58 or between "0." and "10." for the TI-59. The number (N) represents the number of 10-register sets devoted to data registers. This number corresponds with the number (N) entered into the display to set partitioning with [2nd] [Op] 17.

6. **Printer Connect Status.** Pressing 0 [2nd] [C'] while the calculator is connected to the PC-100A or PC-100C printer causes flag 0 to be set. Pressing 0 [2nd] [C'] when the printer is NOT connected resets flag 0. IMPORTANT: This subroutine uses the internal register for the 8th pending math operation. You should not have more than seven pending math operations or a stored alphanumeric with [2nd] [Op] 04 when using this subroutine.

7. **Angular Mode Status.** Pressing [D] displays the code for the current angular mode where "0." is for degrees, "−1." for radians, or "1." for grads. A simple convention for remembering this code is that radian is the second function of the [−] key and grad is the second function of the [+] key.

8. **Open Parentheses Status.** Pressing [E] displays the current number of open parentheses without affecting the pending parentheses or operations. Remember that the number of open parentheses does not necessarily correlate with the number of pending operations.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select program.		[2nd] [Pgm] 20	
2	Save calculator status.[1]		[A]	0.
3	Initialize calculator status.[1]		[2nd] [A′]	0.
4	Flags set status.[1]		[B]	Flags set.
5	Fix-decimal status.[1]		[2nd] [B′]	No. fixed.
6	Partitioning status.[1]		[C]	10-Reg. sets
7	Printer connect status.[1]	0	[2nd] [C′]	0.(flag 0)
8	Angular mode status.[1]		[D]	0 = D, −1 = R, 1 = G
9	Open parentheses status.[1]		[E]	No. parentheses open

NOTE: 1. See Conditions and Limitations on the previous page for details about status subroutines.

Example: Perform the following key sequence, then check calculator status:

[RST] [CLR] [2nd] [St flg] 4 7 [2nd] [Op] 17 [2nd] [St flg] 0

[2nd] [Fix] 3 [( ] 2 [+] [( ] [2nd] [Rad]

PRESS	DISPLAY	COMMENTS
[2nd] [Pgm] 20	2.000	Select program.
[2nd] [B′]	3.000	Fix 3 decimal.
[B]	4.0	Flags 0 and 4 set.
[2nd] [Fix] 3	4.000	Restore fix 3.
[C]	7.000	7 X 10 data registers.
[D]	−1.000	Radians mode.
[E]	2.000	2 open parentheses.
[A]	0.000	Save status in R_{00}.

Now change the actual calculator status by pressing [CLR] [RST] [2nd] [Fix] 9 6 [2nd] [Op] 17 [2nd] [Deg]

[RCL] 0 [STO] 10	−0.7130001	Save status code in R_{10}.
[2nd] [Pgm] 20	−0.7130001	Select program.
[2nd] [B′]	9.	Fix 9.
[B]	0.	No flags set.
[C]	6.	6 X 10 data registers.
[D]	0.	Degrees mode.
[E]	0.	No parentheses open.

To initialize the calculator back to the previous status, return the coded status back to R_{00}.

[RCL] 10 [STO] 00	−0.7130001	Return status code to R_{00}.
[2nd] [A′]	0.000	Initialize original status.

Except for the number of open parentheses, the calculator now has the same status it had after the first key sequence of this example. You can repeat the first eight lines of the "Press" column to check status again.

Register Contents

R_{00} Used	R_{05}	R_{10}	R_{15}	R_{20}	R_{25}
R_{01} Used	R_{06}	R_{11}	R_{16}	R_{21}	R_{26}
R_{02}	R_{07}	R_{12}	R_{17}	R_{22}	R_{27}
R_{03}	R_{08}	R_{13}	R_{18}	R_{23}	R_{28}
R_{04}	R_{09}	R_{14}	R_{19}	R_{24}	R_{29}

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
A	0,1	0-9	1	1			x	
B	1	0-9		2				1,9
C				1				
D								
E	0,1	7		1			x	
A′	0,1	0-9					x	0-9
B′				1			x	
C′	HIR_8	0					x	

VARIABLE ARITHMETIC

This program simplifies keyboard operations by making the user-defined keys [A] through [E] directly accessible for storage and recall of variables. For example, the simple key sequence n [2nd] [E'] [A] defines key [A] as n. The variable n can then be recalled for use at any point in a calculation by pressing [A].

To illustrate how the program works, assume a rectangle of length L and width W. The values of L and W can be entered using L [2nd] [E'] [A] and W [2nd] [E'] [B]. Then, pressing the following key sequences will yield the parameters indicated.

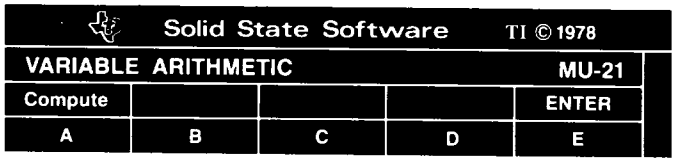
[A] → length L
[B] → width W
[A] [×] [B] [=] → area
[2] [×] [A] [+] [2] [×] [B] [=] → perimeter
[A] [x^2] [+] [B] [x^2] [=] [$\sqrt{x}$] → diagonal

Variables can be calculated using subroutines that you enter in program memory. For example, after A and B have been entered, the area of the rectangle can be calculated as variable C using the key sequence [2nd] [A'] [C] if the following subroutine is entered into program memory.

[2nd] [Lbl] [C] [RCL] 21 [×] [RCL] 22 [=] [STO] 23 [INV] [SBR]

Direct storing and recalling of the variables must be used in the subroutine. Variables A through E reside in data registers R_{21} through R_{25}, respectively.

These examples may suggest specific applications in your particular field of interest. The program can be a useful tool any time you perform calculations which involve repeated use of the same numbers.



STEP	PROCEDURE	ENTER	PRESS	DISPLAY
1	Select Program.		[2nd] [Pgm] 21	
2	Store Variable A	A	[2nd] [E′] [A]	A
3	Store Variable B	B	[2nd] [E′] [B]	B
4	Store Variable C	C	[2nd] [E′] [C]	C
5	Store Variable D	D	[2nd] [E′] [D]	D
6	Store Variable E	E	[2nd] [E′] [E]	E
7	Recall Variable A (see note 1)		[A]	A
8	Recall Variable B (see note 1)		[B]	B
9	Recall Variable C (see note 1)		[C]	C
10	Recall Variable D (see note 1)		[D]	D
11	Recall Variable E (see note 1)		[E]	E
12	Compute Variable A (see note 2)		[2nd] [A′] [A]	A
13	Compute Variable B (see note 2)		[2nd] [A′] [B]	B
14	Compute Variable C (see note 2)		[2nd] [A′] [C]	C
15	Compute Variable D (see note 2)		[2nd] [A′] [D]	D
16	Compute Variable E (see note 2)		[2nd] [A′] [E]	E

NOTES: 1. If before pressing keys [A] – [E], a numeric entry (other than zero) is made that may be cleared with the [CE] key, the display will flash when [A] – [E] is pressed and the operation must be repeated.

2. In order to compute a variable, a subroutine for performing the computation must be stored in program memory under the appropriate label. (For example, a user-defined subroutine for computing variable A must be stored in program memory under label A.) Direct storing and recalling of variables in the subroutine may be desirable. The registers associated with the variables are $A - R_{21}$, $B - R_{22}$, $C - R_{23}$, $D - R_{24}$, $E - R_{25}$.

Example: In a laboratory experiment you wish to find the acceleration and final velocity of a car in a model transportation under various conditions. Using this program and the equations:

$$\text{acceleration} = (2 \times \text{distance})/\text{time}^2$$

and

$$\text{velocity} = \text{acceleration} \times \text{time}$$

Compute the results of the following runs.

Run 1: Distance = 25 feet Time = 1.7 seconds

Run 2: Distance = 25 feet Time = 1.5 seconds

Run 3: Distance = 25 feet Time = 1.3 seconds

ENTER	PRESS	DISPLAY	COMMENTS
25	[2nd] [E′] [A]	25.	Enter distance.
1.7	[2nd] [E′] [B]	1.7	Enter $time_1$
2	[×] [A] [÷] [B] [x^2] [=]	17.30103806	Compute $acceleration_1$
	[×] [B] [=]	29.41176471	Compute $velocity_1$
1.5	[2nd] [E′] [B]	1.5	Enter $time_2$
2	[×] [A] [÷] [B] [x^2] [=]	22.22222222	Compute $acceleration_2$
	[×] [B] [=]	33.33333333	Compute $velocity_2$
1.3	[2nd] [E′] [B]	1.3	Enter $time_3$
2	[×] [A] [÷] [B] [x^2] [=]	29.58579882	Compute $acceleration_3$
	[×] [B] [=]	38.46153846	Compute $velocity_3$

The same problem can be worked using subroutines in program memory to perform the calculations. C will be used for acceleration and D for velocity. The solution using this method is as follows:

	[RST] [LRN]	000 00	Enter learn mode
	[2nd] [Lbl] [C] [( ]		
	[RCL] 21 [×] 2 [÷] [RCL] 22		
	[x^2] [ )] [STO] 23 [INV] [SBR]	015 00	Subroutine for computing acceleration.
	[2nd] [Lbl] [D] [( ]		
	[RCL] 23 [×] [RCL] 22 [ )]		
	[STO] 24 [INV] [SBR]	027 00	Subroutine for computing velocity.
	[LRN]	0	Exit learn mode.
	[2nd] [Pgm] 21	0	Select program
25	[2nd] [E′] [A]	25.	Enter distance
1.7	[2nd] [E′] [B]	1.7	Enter $time_1$
	[2nd] [A′] [C]	17.30103806	Compute $acceleration_1$
	[2nd] [A′] [D]	29.41176471	Compute $velocity_1$

Acceleration and velocity must be computed in sequence after distance and time have been entered. The other two runs are calculated similarly.

Register Contents

R_{00}	R_{05}	R_{10}	R_{15}	R_{20}	Pointer	R_{25}	E
R_{01}	R_{06}	R_{11}	R_{16}	R_{21}	A	R_{26}	
R_{02}	R_{07}	R_{12}	R_{17}	R_{22}	B	R_{27}	
R_{03}	R_{08}	R_{13}	R_{18}	R_{23}	C	R_{28}	
R_{04}	R_{09}	R_{14}	R_{19}	R_{24}	D	R_{29}	

Program Details

Label	Data Reg. Used	Flags Used	SBR Levels	Paren. Levels	Calls Pgm	Spec. Func. Used	x≷t	Fix Decimal Format
A	20,21	0,1						
B	20,22	0,1						
C	20,23	0,1						
D	20,24	0,1						
E	20,25	0,1						
A′		1						
E′	20	0						

ONE YEAR LIMITED WARRANTY FOR CALCULATOR AND/OR LIBRARY MODULE

WARRANTEE: This Texas Instruments Electronic Calculator Warranty extends only to the original purchaser of the calculator or module.

WARRANTY DURATION: This Texas Instruments Electronic Calculator and/or Library Module is warranted to the original purchaser for a period of one (1) year from the original purchase date.

WARRANTY COVERAGE: This Texas Instruments Electronic Calculator and/or Library Module is warranted against defective materials and workmanship. **THIS WARRANTY IS VOID IF: (I) THE CALCULATOR OR MODULE HAS BEEN DAMAGED BY ACCIDENT OR UNREASONABLE USE, NEGLECT, IMPROPER SERVICE OR OTHER CAUSES NOT ARISING OUT OF DEFECTS IN MATERIAL OR WORKMANSHIP, (II) THE SERIAL NUMBER HAS BEEN ALTERED OR DEFACED.**

WARRANTY PERFORMANCE: During the above one (1) year warranty period your defective calculator or module will either be repaired or replaced with a reconditioned model of an equivalent quality (at TI's option) when the calculator or module is returned, postage prepaid and insured, to a Texas Instruments Service Facility listed below. In the event of replacement with a reconditioned model, the replacement unit will continue the warranty of the original unit or 90 days, whichever is longer. Other than the postage and insurance requirement, no charge will be made for such repair, adjustment, and/or replacement.

WARRANTY DISCLAIMERS: ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE ONE (1) YEAR. TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR LOSS OF USE OF THE CALCULATOR OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES OR DAMAGES INCURRED BY THE PURCHASER.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you.

LEGAL REMEDIES: This warranty gives you specific legal rights, and you may also have other rights that vary from state to state.

TEXAS INSTRUMENTS CONSUMER SERVICE FACILITIES

Texas Instruments Service Facility
P.O. Box 2500
Lubbock, Texas 79408

Texas Instruments Service Facility
41 Shelley Road
Richmond Hill, Ontario, Canada

Consumers in California and Oregon may contact the following Texas Instruments offices for additional assistance or information:

Texas Instruments Consumer Service
3186 Airway Drive Bldg. K
Costa Mesa, California 92626
(714) 540-7190

Texas Instruments Consumer Service
10700 Southwest Beaverton Highway
Park Plaza West, Suite 11
Beaverton, Oregon 97005 (503) 643-6758

IMPORTANT NOTICE REGARDING PROGRAMS AND BOOK MATERIALS

TEXAS INSTRUMENTS MAKES NO WARRANTY, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR PARTICULAR PURPOSE, REGARDING THESE PROGRAMS OR BOOK MATERIALS OR ANY PROGRAMS DERIVED THEREFROM AND MAKES SUCH MATERIALS AVAILABLE SOLELY ON AN "AS-IS" BASIS.

IN NO EVENT SHALL TEXAS INSTRUMENTS BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE PURCHASE OR USE OF THESE BOOK MATERIALS OR PROGRAMS AND THE SOLE AND EXCLUSIVE LIABILITY TO TEXAS INSTRUMENTS, REGARDLESS OF THE FORM OF ACTION, SHALL NOT EXCEED THE PURCHASE PRICE OF THIS BOOK. MOREOVER, TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR ANY CLAIM OF ANY KIND WHATSOEVER AGAINST THE USER OF THESE PROGRAMS OR BOOK MATERIALS BY ANY OTHER PARTY.

TEXAS INSTRUMENTS
INCORPORATED
DALLAS, TEXAS

Printed in U.S.A.

1014984-10