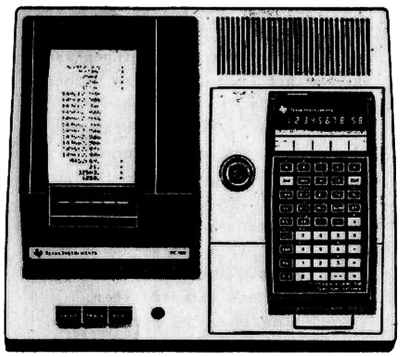
PROGRAMBITEN

Supplement to manual for TI 58, TI 58C, TI 59



Contents

Preface	2
Synthetic Programming	2
Flags	2
Dsz	2
Dsz nn BST	2
"Soft" and "hard" display	2
HIR	2
- Use of HIR as an extra-register	3
- Use of HIR as print register	3
- HIR usage from the keyboard	4
Correspondence Program Step-Data Register	4
Fast Mode	5
Pgm nn R/S	7
Delayed module call	7
The firmware of TI-58/59	7
Graphic mode	8
Practical box	8
Standard for typing key symbols	8

Appendix to Programbiten 82-1
Price for non-members 15 kr

PREFACE

This supplement to the TI-58/58C/59 instruction manual has been added to ensure that new members don't need to be familiar with articles in the "Programbiten" as a preliminary consideration of the contents of previous issues.

However, when we have written it, we have designed it so that can be used as an "encyclopedia" with reference to various more detailed articles in "Programbiten".

The use of this complement to the instruction manual, of course, assumes that the reader is familiar with the contents of the ordinary instruction manual.

Stockholm January 1982

Lars Hedlund
Björn Gustavsson

SYNTHETIC PROGRAMMING

As it is known, an instruction in the program memory is stored as a two-digit code. Tables of these codes can be found in the instruction manual on page V-49 and V-50. It is obvious that the functions 2nd, LRN, SST, Ins, BST and Del are missing codes. If they had one, these would have been 21 or 26, 31, 41, 46, 51 and 56.
Furthermore, code 82 is missing. These codes, which can't be entered into the program memory with a single keystroke, however (and not even as several "merged" keys) can be programmed "synthetically" using a multistep instruction, like STO, and a delete (Del) because the subsequent two digit number is stored in one program step. For example, "21" can be programmed "STO 21 BST BST Del SST". (when listing with printer, the instruction 21 is printed as "2ND.") - Code 31 (LRN) in a program causes the program to stop with the display in programming mode.
Referring to the code 82, see the following section on HIR.

FLAGS

The TI 58/59 has ten flags with numbers 0-9 but not more. Through experiments with synthetic programming, one can think of "discover" more flags but at closer control it turns out to be the same as the ten original ones.

DSZ

In the instruction manual p. V-63, the Dsz function only operates on the registers 00-09. This is not true, but all registers except 40 can be used. ("Dsz 40" is designated as "Dsz Ind"). However, as the register number and the following jump address, they must be programmed synthetically as above, or, in a favorable case, in a somewhat simpler manner. *Example:* "Dsz 29 320" is encoded "Dsz CP (= 29) 3 2nd CLR (= 20). "Dsz 41 021" for example is programmed as "STO 41 STO 21" BST BST BST (go to first STO) Dsz (superimpose the first STO) SST (go to the second STO) 0 (superimpose the second STO) SST (go to the next free step).

DSZ NN BST

A counterpart to "Op 30" - "Op 39" (decrement of register 0-9 contents by one) can be obtained with "Dsz nn BST", where both the register number nn and BST (code 51) are programmed synthetically. Note, however, that Dsz decreases absolute value of the register content down to zero, not past zero!

"SOFT" AND "HARD" DISPLAY

A "soft" display can be deleted with CE (i.e. by entering numbers) and becomes "hardened" by pressing an operation key (cannot be deleted with CE).

HIR

AOS (Algebraic Operations System) means that expressions can be entered as they are written. In order to accommodate this, the calculator must store certain numbers. For example, if 2 + 3 x 4 is entered, 2 and 3 must be saved somewhere for the calculation to be completed. In order to store these operands, 8 special registers are used, called HIR-registers (Hierarchical Internal Registers). These registers are designated H1 to H8 (HIR1 to HIR8).

In the example above, 2 are stored in H1 and 3 in H2. It is actually possible to get these registers directly from program or keyboard and use them as an extra-register.

The HIR yn instruction is used to manage an HIR register, where n is the number of the HIR register and y denotes the operation to be performed according to following table:

y	operation
0	store in Hn (STO)
1	recall Hn (RCL)
(2)	unknown
3	sum to Hn (SUM)
4	multiply to Hn (Prd)
5	subtract in Hn (INV SUM)
6 (7,8,9)	divide in Hn (INV Prd)

The instruction HIR has code 82. Unfortunately, there is no key with that code, so the code must be programmed synthetically.

Example: 18 HIR 04 stores 18 in H4, 3 HIR 64 divide H4 with 3, 7 HIR 34 sums 7 to H4. HIR 14 recalls H4 and gives in this case 13.

(The following doesn't apply to the TI-58C.)

An important limitation on HIR use: if exponent-of-ten or technical notation is set, the absolute value of X-register's exponent-of-10 is taken when using HIR 3n to HIR 9n.

This means that x will be multiplied by a power of 10 if the absolute value of x is less than 1.
Example: 0.035 HIR 35 results in 0.035 = $3.5*10^{-2}$ changes to $3.5*10^{2}$=350, which is then summed up to H5. If the exponent-of-ten notation is set this will not happen.

Use of HIR as an extra-register

The calculator uses HIR register internally when calculating certain functions, as well as a print register. It is important to check that an HIR register is not used for two things at the same time. How the TI-58/58C/59 use HIR registers:

Operation	Uses HIR register
Op 01	H5
Op 02	H6
Op 03	H7
Op 04	H8
Op 11	First 2 available HIR
Op 12	H8 and the first 3 available HIR
Op 13	First available HIR
Op 14	H8 and the first 3 available HIR
Op 15	H8 and the first 3 available HIR
P/R	First available HIR and H7 and H8.
INV P/R	First 2 available HIR and H7 and H8 (H7 is not used on TI-58C)
Σ+	H7 and H8
INV Σ+	H7 and H8
D.MS	First 2 Accessible HIR and H8
INV D.MS	First 2 available HIR and H8
Xm	First available HIR
INV Xm	First 2 available HIR and H8

Also note that HIR registers is used by AOS to store pending operations. This makes H1 and H2 unavailable as extra-registers.

Neither CLR, CMs nor any other instruction zeroes any HIR register. A single HIR register can be set to zero with *0 HIR 0n*. If the printer is connected, all can be cleared with the sequence *CLR D.MS HIR 03 HIR 04 Op 00* (without printer it clears H1 to H4 only; Op 00 can then be excluded).

Use of HIR for printer detection

This can be used for software testing if the printer is connected. "1 P/R Op 00 HIR 18" gives 0 if the printer is connected, otherwise 1. Explanation: P/R uses HIR 8, see below. If you continue with "CP INV EQ B STF 7 Lbl B" you have set flag 7 if the printer is connected and with "... IFF 7 E' R/S Lbl E' printing" the program stops at R/S if no printer is connected. **(The TI-58C has the "Op 40" instruction to do this.)**

Use of HIR as a print register

The print registers can be loaded directly using the HIR instruction. As shown in the table how the HIR registers are used, H5 corresponds to print register 1, H6 to print register 2 etc.

Op 01 can be replaced with HIR 05. But now it appears that the print codes cannot be written normally. The contents of a print register are stored in the form **x.xxxxxxxxxxxx*10^n** when the digit for the decimal sign is separated from zero. The 3 first digits are always ignored when printing. The printing codes are always the last 10 digits.

For example, if we execute 131415 HIR 05 to type "ABC", this is stored as 1.314150000000*10^5 and because the 3 first digits are ignored, the print code will be 41 50 00 00 00 which gives the printout "Ux ". In order to get the correct print, we had to add 3 significant digits to the print code.
If we try 100131415 HIR 05, the writing code will be 13 14 15 00 00 and the print will be "ABC ". The decimal point's location has no meaning; for example, 10.0131415 had been just as good.

When using Op 01-04, the numbers in the print registers are stored in a non-normalized form: 0.00XXXXXXXXXXX $*10^0$, i.e. the digit for the decimal character is zero. When printing, the 3 first zeroes are ignored. (Note: There is no possibility to save a similar storage with the HIR instruction, but 3 significant digits must enter the print codes).

Example: If you press 131415 Op 01, the print code is stored in H5 in the following way: $0.000000131415*10^0$, the first zeroes are ignored when printed and give " ABC".

What are the reasons for using the HIR instruction instead of Op 01-04 to store print codes?
First, Op-loading considers both fixed decimal and exponent-of-ten notation. HIR-loading has no such disadvantages.
Second, parts of a write register can be changed if HIR-loading is used.
Example: Assume that X1, X2, X3 ... should be printed in the upper edge. The print codes can be loaded with the sequence 4402 + 1 EE 12 = HIR 08 (possibly CLR for deleting exponent-of-ten notation). This give printing X1. To get X2 execute 1 HIR 38; similarly done to get X3 etc. (The first sequence can be shorter if the print code is stored in a register. The sequence will be RCL nn HIR 08.)

Use of HIR from the keyboard
Sometimes it may be good to be able to perform the HIR instruction from the keyboard. The easiest way to place a series of code 82 in the program memory when there are steps over. If the step counter is then set to a program step, then there is a code, just pressing SST, to execute the HIR instruction. This is then followed by two digits from the keyboard, to enter the operation code.

Example: To execute HIR 11, press SST 11.

Articles on HIR in Programbiten:
PB 78-1 p 18: Stig Petersson, HIR
PB 79-2 p 12: Lars Hedlund, HIR - The print registers
PB 80-2 p 13: Lars Hedlund, HIR - Good but dangerous
PB 80-3 p 13: Lars Hedlund, More about HIR - among other things from the keyboard
PB 80-4 p 30-33, 37: Gösta Blume, HIR - Better everyday life
PB 80-4 p 40: Lars Hedlund, For HIR-winner
PB 81-1 p 5: Björn Gustavsson, Input of HIR.

CORRESPONDENCE PROGRAM STEP-DATA REGISTER
On TI-58/58C/59, the same memory is used for programs and data (non-simultaneous memory). It is therefore possible to store data in a register and change the partition, after that the register becomes program code.

For example, try the following: Enter 9208.821176, press STO 59 (STO 29 on TI-58/58C). Then set partition with 5 Op 17 (2 Op 17). At step 483 (243), the following program is stored: *Lbl A HIR 08 RTN*.

R59 corresponds to steps 480-487 on TI-59 and steps 000-007 on TI-58/58C. The tables below show the correspondence between program step and data register.

TI-59		352-359	R75	720-727	R29	112-119	R45
		360-367	R74	728-735	R28	120-127	R44
000-007	R119	368-375	R73	736-743	R27	128-135	R43
008-015	R118	376-383	R72	744-751	R26	136-143	R42
016-023	R117	384-391	R71	752-759	R25	144-151	R41
024-031	R116	392-399	R70	760-767	R24	152-159	R40
032-039	R115	400-407	R69	768-775	R23	160-167	R39
040-047	R114	408-415	R68	776-783	R22	168-175	R38
048-055	R113	416-423	R67	784-771	R21	176-183	R37
056-063	R112	424-431	R66	792-799	R20	184-191	R36
064-071	R111	432-439	R65	800-807	R19	192-199	R35
072-079	R110	440-447	R64	808-815	R18	200-207	R34
080-087	R109	448-454	R63	816-823	R17	208-215	R33
088-095	R108	456-463	R62	824-831	R16	224-231	R32
096-103	R107	464-471	R61	832-839	R15	216-223	R31
104-111	R104	472-479	R60	840-847	R14	232-239	R30
112-119	R105	480-497	R59	848-855	R13	240-247	R29
120-127	R104	488-495	R58	856-863	R12	248-255	R28
128-135	R103	496-503	R57	864-871	R11	256-263	R27
136-144	R102	504-511	R56	872-879	R10	264-271	R26
145-151	R101	512-519	R55	880-887	R09	272-279	R25
152-159	R100	520-527	R54	888-895	R08	280-287	R24
160-167	R99	528-535	R53	896-903	R07	288-295	R23
168-175	R98	536-543	R52	904-911	R06	296-303	R22
176-183	R97	544-551	R51	912-919	R05	304-311	R21
184-191	R96	552-559	R50	920-927	R04	312-319	R20
192-199	R95	560-567	R49	928-935	R03	320-327	R19
200-207	R94	568:575	R48	936-943	R02	328-335	R18
208-215	R93	576-583	R47	944-951	R01	336-343	R17
216-223	R92	584-591	R46	952-959	R00	344-351	R16
224-231	R91	592-599	R45			352-359	R15
232-239	R90	600-607	R44	**TI-58**		360-367	R14
240-217	R89	608-615	R43	000-007	R59	368-375	R13
248-255	R88	616-623	R42	008-015	R58	376-383	R12
256-263	R87	624-631	R41	016-023	R57	384-392	R11
264-271	R86	632-639	R40	024-031	R56	393-399	R10
272-279	R85	640-647	R39	032-039	R55	400-407	R09
280-287	R84	648-655	R38	040-047	R54	408-415	R08
288-295	R83	656-667	R37	048-055	R53	416-423	R07
296-303	R82	664-671	R36	056-063	R52	424-431	R06
304-311	R81	672-619	R35	064-071	R51	432-439	R05
312-319	R80	680-687	R34	072-079	R50	440-447	R04
320-327	R79	688-695	R33	080-087	R49	448-455	R03
328-335	R78	696-703	R32	088-095	R48	456-463	R02
336-343	R77	704-711	R31	096-103	R47	464-471	R01
344-351	R76	712-719	R30	104-111	R46	472-479	R00

In order to understand how this correspondence can be utilized practically, we must look at how the calculator stores numbers internally.

All numbers are stored in exponent-of-ten notation, irrespective of how they appear in the display, i.e. as $m*10^n$, where m is a 13 digit number ($1 \leq m < 10$) and n the exponent. By storing 2 digits in each program step, the number enters 8 program steps: 6 1/2 program steps are used for m, 2 times 1/2 program step is used for the exponent and 1/2 program steps are used for the signs of m and n. These signs are stored according to:

Number	Meaning
0	m*10^n
2	-m*10^n
4	m*10^-n
6	-m*10^-n
8	Overflow (i.e. 9.9999999 99 flashes upon recall)

Example: The number -2471.7176 (equal to $-2.4717176*10^3$) is stored in R0 of TI-59 in the following manner:

```
959 24
958 71
957 71
956 76
955 00
954 00
953 00
952 32
```

all of these are stored in steps 952-959. (There is a virtual decimal point between the tens and ones digit in step 959.) The exponent (= 03) is stored in the ones digit of step 953 and in the tens digit of step 952. The signs are stored as second digit of step 952.

This contents simultaneously create a program (from step 956): *Lbl SBR SBR CE*.

It is therefore possible to allow a program to create new programs, which can then be used as subroutines.

Here we shall only look at one of the possibilities with this technique. The program below arranges an indirect jump to a label whose code is in the display at the moment of the call.

TI-58/58C			TI-59		
000	76	LBL	000	76	LBL
001	11	A	001	11	A
002	32	X:T	002	32	X:T
003	03	3	003	06	6
004	69	OP	004	69	OP
005	17	17	005	17	17
006	32	X:T	006	32	X:T
007	85	+	007	85	+
008	93	.	008	93	.
009	06	6	009	06	6
010	01	1	010	01	1
011	95	=	011	95	=
012	42	STO	012	42	STO
013	29	29	013	59	59
014	02	2	014	05	5
015	69	OP	015	69	OP
016	17	17	016	17	17
017	61	GTO	017	61	GTO
018	02	02	018	04	04
019	46	46	019	86	86

When it's executed a GTO N (where N is the code of the input label) is stored in step 486 (246 on TI-58/58C).

Articles about correspondence program steps-data Register in Programbiten:
PB 79-1 p 20-21: Claes Schibler, Program step - program memory
PB 79-3/4 p 26-27: Sven Ostberg, Something about TI-59's computer-like features
PB 80-2 p 6-7: Sven Ostberg, Something about TI-59's computer-like features, part 2
PB 81-4 p 11: Björn Gustavsson, Program Loader

FAST MODE

Fast mode is a state where program runs at double speed. It was discovered in the spring of 1980 by West German Martin Neef. Now the method is used for all programs that need to be executed quickly.

Fast mode is achieved by placing the sequence **Pgm 02 SBR 239 9 0** in the program memory at step 005 and executed (the ML module must be inserted). This results in the entire memory being erased. It is possible to enter a program directly from the keyboard or insert a magnetic card.

A program to be written in fast mode may look like that on the next page. Steps 000-015 as the initialization routine look similar to most fast mode programs. After that our own program will run, as in the case of the routine for the factorial.

```
000  00   0      012  22  INV    024  00   00
001  00   0      013  58  FIX    025  00   00
002  00   0      014  22  INV    026  20   20
003  76  LBL     015  57  ENG    027  01    1
004  11   A      016  25  CLR    028  95    =
005  36  PGM     017  91  R/S    029  99  PRT
006  02   02     018  42  STO    030  91  R/S
007  71  SBR     019  00   00    031  61  GTO
008  02   02     020  43  RCL    032  00   00
009  39   39     021  00   00    033  18   18
010  09   9      022  65   x
011  00   0      023  97  DSZ
```

We are coming back to the program, but first of all, a reminder of the rules that are associated with fast mode.

1. When you switch to fast mode, the entire program memory and data records are deleted (not the HIR registers).
2. The partition is always set to 479.59 (6 Op 17) on TI-59 (to 239.29 (3 Op 17) on TI-58) when moving to fast mode. It is not possible to enter fast mode on the TI-58C (see note at the end in the description).
3. When switched to fast mode, it will be stopped with decimal fixed to Fix 0.
4. Programs can be entered or changed directly from the keyboard. SST, BST, Ins and Del can be used unobtrusively. Merged codes cannot be entered because the calculator is in fast mode (for example, 72=ST*, 92=RTN).
5. Data can be input as usual with keys 0-9, +/-, decimal point, CE, CLR, EE.
6. Using all keys that leave behind a hard display will start running the program in the program memory. This may cause the caller to be taken out of fast- mode, for example, if there is no program. An exception to this rule of Prt, which can be used unobtrusively.
7. RST used by the keyboard when programmed does not remove fast mode. RST used in a program removes fast mode and may cause a "crash", after which it is only possible to turn off the calculator.
8. R/S can be used to start a program in fast mode. SBR nnn is used from the keyboard to start a program at step nnn.
9. The only way to stop a program running in fast mode is to turn off the calculator.
10. An R/S in a program will not work unless the display is soft or R/S is not immediately after a Prt or Pause (Prt goes well without printer.)
11. RTN never used.
12. No subroutine calls can be used, like functions such as D.MS, P/R or statistical functions (that use the firmware) or library programs.
13. No labels can be used. All jump addresses must be absolute.

Let us now look at the program above. It is fed in on a regular basis. It should then be recorded on a magnetic card for proofing, because the program is erased by moving to fast mode.
The program is executed by pressing A. The program memory is erased while the ringer is switched to fast mode. Now the magnetic card can be reinserted. Immediately after the reading, program execution starts. The following sequence is executed: 9 0 INV Fix INV Eng CLR R/S, and stays with a zero in the display. Now an integer can be input and R/S is pressed. The factorial for this number is calculated.
The following are important to note when doing fast mode programs:
Program steps 005-011 should look like in this program. The steps 000-004 may contain anything, though we recommend that step 003-004 contains Lbl A, so that initialization to fast mode can only be done by pressing A.
Steps 012-015 may change, but note that they will be performed twice. R/S on either of these steps is completely inappropriate. In addition, Fix 0 will automatically resume when you go to fast mode - if Fix 0 is not required, then either INV Fix should remain.
If a program requires with a different partition other than the normal, the program must set the correct partition.
If a program does not fit on one side, the part of the program contained in block 1 must read the remaining cards. This is conveniently done as in this program:

```
000  00   0      008  02   02     016  02   2
001  00   0      009  39   39     017  91  R/S
002  00   0      010  09   9      018  03   3
003  76  LBL     011  00   0      019  91  R/S
004  11   A      012  22  INV     020  04   4
005  36  PGM     013  58  FIX     021  91  R/S
006  02   02     014  22  INV
007  71  SBR     015  57  ENG
```

The actual program should start at step 022. Procedure for review is as follows: Read block 1. Press A. Re-enter block 1 again, 2 is displayed - enter block 2, 3 is displayed - enter block 3, 4 is displayed - load block 4.
If you have less than 4 blocks, simply remove unnecessary instructions.

Lastly, be aware of the above limitations - because subroutines cannot be used.

Recently (autumn 1981) two new methods for getting into fast mode have been discovered. These can be used on the TI-58/58C also, and the program will not be erased when switching to fast mode.

Articles on fast mode in Programbiten:
PB 80-3 p 14: Lars Hedlund, Deep diving into the TI-59
PB 80-4 p 38: Björn Gustavsson, Factorial program in fast mode
PS 81-1 p 11: Björn Gustavsson. More about fast mode
PB 81-3 p 29-32: Björn Gustavsson, More about fast mode -2

PGM nn R/S

Contrary to what is stated in the manual (p IV-52, V-61), the call Pgm nn R/S can be used, which is actually mentioned in the table on page V-62 in the manual! The call starts the module program on the step that the module's program counter is pointing to.

Example: Suppose a module program looks like this and has program number 06:

000	76	LBL	007	20	20
001	15	E	008	72	ST*
002	00	0	009	00	00
003	42	STO	010	92	RTN
004	00	00	011	61	GTO
005	92	RTN	012	00	00
006	69	OP	013	06	06

If the call is first done with Pgm 06 E, the module program counter will point to step 006. After typing Pgm 06 R/S, steps 006-010 are executed and the module program counter then points to step 011. Pgm 06 R/S can now be repeated as many times as you wish.

In general, if R/S can be used from the keyboard when using a particular module program, then Pgm nn R/S can be used in programs.

DELAYED MODULE CALL

It is possible to distinguish Pgm nn from R/S. Pgm nn can be executed in a program, and when the user presses R/S from the keyboard, the program continues in the module!
This can be used when you want to tune in a running program without stopping it.

The methods is as follows: ensure that the module's instruction pointer points to a convenient program section by making a normal subroutine call to the module program. Then execute Pgm nn BST. When the user presses R/S, the program section will be called.

Example: The following program increases R1 by 1 and when the user presses R/S the result (content of R1) is shown.

000	92	RTN	009	01	01	018	61	GTO
001	76	LBL	010	71	SBR	019	00	00
002	15	E	011	00	00	020	16	16
003	25	CLR	012	98	98	021	76	LBL
004	42	STO	013	36	PGM	022	11	A
005	00	00	014	01	01	023	43	RCL
006	42	STO	015	51	BST	024	01	01
007	01	01	016	69	OP	025	81	RST
008	36	PGM	017	21	21			

This is what the current part of Pgm 01 (standard module) looks like:

098	92	RTN
099	76	LBL
100	11	A
101	98	ADV
102	99	PRT
103	62	PG*
104	00	00
105	11	A
106	99	PRT
107	92	RTN

When the user presses R/S, the module executes the program that immediately calls Lbl A into the regular program memory. There the content of R1 is recalled and RST is performed to clear the subroutine call register. One problem, however, is that the user won't be able to press R/S key for a long time since this operation will stop when the queue is resumed in the normal program.
Please note:

1. If, after Pgm nn BST, you want to call a subroutine in the normal program line, the corresponding routine in the module will be called. The same applies to R/S.
2. RTN removes the effect of Pgm nn BST.
3. There are 6 subroutine levels known. A call of one module program, which in turn calls the common program memory, takes up two subroutine levels. These can only be reset with RST.
4. When returning to the usual program memory, stops execution if R/S is depressed.

Articles about delayed module calls in Programbiten:
PB 81-1 p 24-25: Björn Gustavsson, To influence a program while running without stopping the program
PB 81-1 p 15: Bjorn Gustavsson, Random numbers
PB 81-4 p 28-31: Anders Persson, Cost for telephone conversation

THE FIRMWARE OF TI-58/59

The TI-58/59 has a firmware (regardless of the module installed) used to calculate statistical functions and conversions. It can be accessed in the following way (for this example, the standard module must be inserted): press (ignore flashing, R/S shouldn't be pressed too fast) Op 09 Pgm 02 R/S R/S R/S Op 17 GTO 000 Op 17 R/S P/R LRN.

If you do not have a printer, you can single with SST to step 487; if you have a printer, press LRN again, and then List will print the program. After step 487 happens a jump to step 039; likewise, the list will be canceled at step 503. To get to the following sections before use the formula above, again with "GTO 489" or "GTO 505".
Steps 000-379 contain programs for statistical functions and conversions. Steps 380-511 contain the numerical value for calculating ln x, etc. (see also the section on correspondence between program steps and data registers; the same rules don't exactly apply here).

Articles about the firmware in Programbiten:
PB 80-3 p 14-15: Lars Hedlund, Deep diving into the TI-59
PB 80-4 p 23: Lars Hedlund, Deep Diving into the TI-59 (2)

GRAPHIC MODE
According to the instruction manual, shapes (e.g. "plotting" of curves) can only be written with whole characters. West German Michael Sperber has realized that by pressing a key sequence from the keyboard, you can program an instruction that interrupts the printout so that only the first lines of a character is written without line feed so that the next row continues directly after the preceding character. By selecting the appropriate letters, you can draw figures with about three times better resolution in both directions.

At step 024-025, the program must have "SUM Ind 80" and then 6 "Nop" or zero. From the keyboard you must press, with the standard module inserted and in partition 9 or 10 Op 17 (4 Op 17 on TI-58), "GTO 024 CLR Pgm 19 SBR 045 P/R LRN Ins LRN RST CLR". Flashing must be ignored! This sequence implies that a special code is entered at step 024, so that an "Op 05" on, for example, steps 021-022, will be canceled during execution. Since labels are not available after step 024, it must be noted that direct addresses within this area are moved one step.

Article about Graphic Modes in Programbiten:
PB 81-2 p 10-13: Lars Hedlund, Graphics Mode and Plot 60

PRACTICAL BOX

"Op 20" - "Op 39"
Use these instructions instead of the longer "1 SUM 00" etc.

Automatic program partition
We recommend that programs that are not written in normal partition are recorded in normal partition. The required partition (e.g. 1 Op 17) is then put into the program at the appropriate place. The use of the program is thus much simpler. (Do not use spared magnetic cards!)

Alpha code in data registers
Try to allocate the calculator's partition so that the print code (preferably in HIR form) is stored in data registers whose contents are recorded on magnetic cards. In this way, the program becomes faster than if the print codes were to be created in the program

Exclusion of right parentheses before "="
All right parentheses before = can be excluded, because = closes all open parentheses. Example: RCL 01 x (RCL 02 + RCL 03 =.

ADDENDUM
The programs at the top of the second column on page 5 in the Complement to the instruction manual can be shortened by 2 program steps, as the programs below show. We chose to use the longer variants in the Complement, because we think they are somewhat easier to understand.

TI-58/58C			TI-59		
000	76	LBL	000	76	LBL
001	11	A	001	11	A
002	85	+	002	85	+
003	03	3	003	06	6
004	69	OP	004	69	OP
005	17	17	005	17	17
006	93	.	006	93	.
007	06	6	007	06	6
008	01	1	008	01	1
009	95	=	009	95	=
010	42	STO	010	42	STO
011	29	29	011	59	59
012	02	2	012	05	5
013	69	OP	013	69	OP
014	17	17	014	17	17
015	61	GTO	015	61	GTO
016	02	02	016	04	04
017	46	46	017	86	86