52-NOTES

Volume 2 Number 7 48/39 July 1977

Newsletter of the SR-52 Users Club
published at
9459 Taylorsville Road
Dayton, OH 45424

A FEW OBSERVATIONS ON THE OCCASION OF OUR FIRST ANNIVERSARY

While most of you seem to be pleased with 52-NOTES subject matter and technical level, some have expressed a sense of difficulty in following a few of the more detailed discussions, and have suggested that a more elementary approach would be better. My aim has been to consolidate and present input material in such a way as to reach PPC users of above average intelligence, inquisitiveness, and attention span, but who may not have pursued formal education past highschool. As I have already suggested to a few, if after careful perusal of specific articles, having worked (actually run) all examples, and carefully referred to cited earlier issues of 52-NOTES you still don't understand something, write me, and I'll do my best to help. While the college professors among you have contributed significantly to 52-NOTES content, so have others with considerably less formal education. In order to make 52-NOTES as attractive to all concerned as I can, and keep the size of each issue within pleasantly managable bounds, I have pursued comprehensiveness with brevity as a general approach.

I expect to continue to give priority to software inventions and discoveries that have potentially broad application. Non-TI-approved hardware modifications are apt to be risky, and will not often be discussed in 52-NOTES. Even when mods appear to work, insidious problems such as power supply overloads and potentially dangerous (to hardware) instruction sequences may cause serious trouble. I intend to continue sharing the best of my own discoveries and inventions covering all the TI PPCs, so long as other members continue to do the same. The ability to "hide" 59 programs on protected cards will motivate some to try to market their best programs. I will publish only the names of members who have such programs to sell; interested members can write to the individuals concerned for further details. But I hope that most will be motivated to share their best ideas openly.

For the 58 and 59, a particularly rich and rewarding field to explore is the efficient use of CROM code. There are apt to be many subroutines, or portions thereof, that can be cleverly used in applications other than those for which they were written. The sequence: *Pgm nn SBR mmm will call a code sequence in Program nn beginning at step mmm, and ending with the first INVSBR encountered (a feature not noted in the owner's manual). Since every 58 or 59 user has the Master Library module, clever use of its code can be taken advantage of by all users. As special applications modules become available, they too can be tapped for clever, efficient use by those who have them.

The program that follows this article was designed to enhance the ML-02 matrix program, facilitating inputs, labeling outputs, and it makes use of the *Pgm nn SBR mmm sequence.

For those of you who would like to explore topics applicable to all (or most) of the TI PPCs, there are still some outstanding invitations to the membership: approaches to teaching computer programming (V1N2p1), specific requests for help (V1N2p4), printer techniques (V1N3p1), rigorous diagnostic programs (V1N4p4), systematic exploration of pseudo behavior (V1N5p3), comprehensive determination of all factors affecting program execution time (V1N7p3), reviews of periodicals of interest to PPC users (V2N2p4), application program generation input (V2N2p5), random number generators (V2N3p2), and program challenges to HP users (V2N4p2).

I'll close this commentary by repeating what I said some months ago: "... it is with pleasant anticipation that I look forward to getting inputs from members who have so far been among the silent majority, as well as to getting more gems from the productive minority."

- - - - - - - - -

TI-58/59/PC-100A Program: Enhanced ML-02 Program — Ed

User Instructions:

1. Key n, press A, see 8 displayed (address of first storage register); n and N printed.
2. Key ith element, with columnwise catenation, press R/S; e_i printed, address of next register displayed; repeat for i=1,2,...n^2; correct entries by direct STO (display-cued); determinant calculated and printed following input of e_{n^2}.
3. For simultaneous equations: If Det $\neq$ 0, press B, "B" printed; key b_i, press R/S; b_i printed; repeat for i=1,2,...n; repeat step 3 for new constant vector; x_i calculated and printed (with labels) following input of b_n.
4. For inverse, press C; inverse printed and labeled by column.

NOTE: For TI-59, max n=8; for TI-58, max n=3, and change all references to Reg 89 to Reg 29.

Program Listing:

```
000: *Lbl D xEt 1 SUM 89 RCL 89 INV *x=t *5' + 2 = STO 89 *Lbl *5'
018: *Op 04 INVSBR *Lbl E R/S *Op 06 STO*Ind 01 *Lbl *1' *Op 21 RCL 01
034: R/S STO*Ind 01 *Prt RCL 01 INV *x=t *1' *Adv *Adv INVSBR *Lbl A
048: STO 07 STO 00 x² + 7 = xEt 31 *Op 04 RCL 07 *Op 06 *Adv 13 *Op 04
070: 8 STO 01 E *Op 00 161737 *Op 03 *Op 05 *Pgm 02 C *Adv *Adv R/S
092: *Lbl B 1*Pgm 02 D RCL 07 STO 06 *Op 00 14 *Op 02 *Op 05 CLR
111: *Lbl *7' R/S *Pgm 02 SBR 355 *Dsz 6 *7' CLR *Pgm 02 E *Pgm 02 *A'
129: 4402 STO 89 *Op 04 RCL 07 STO 06 *Lbl *6' RCL*Ind 01 *Op 06
147: *Op 21 4408 D *Dsz 6 *6' *Adv *Adv *Adv CLR R/S *Lbl C CLR *Pgm
166: 02 *B' *Op 00 243142 *Op 03 *Op 05 *Adv 1502 STO 89 *Op 04 1
190: *Lbl *2' *Pgm 02 *C' RCL 07 STO 06 *Lbl *3' *Op 21 *Op 24
205: RCL*Ind 01 *Op 06 *Op 36 *Lbl *4' *Pgm 02 SBR 860 *Dsz 6 *4'
221: *Adv 1508 D 1 + RCL 03 = *Dsz 0 *2' *Adv *Adv *Adv CLR R/S
```

- - - - - - - - -

0^x Definitions

G E Wilkins (25) notes that the various PPCs produce varying results upon execution of 0^x with the y^x function, as the following table shows:

Machine	0^0	0^{+x}	0^{-x}
SR-52	1	0	0
SR-56	1	0	0
TI-58/59	1	0	error
HP-25	error	error	error
HP-55	error	0	error
HP-65	error	error	error
HP-67	error	0	error

Perhaps this is not too surprising, since there appears to be no clear-cut mathematical approach to defining 0^x. G E suggests that one can argue that since log y^x = xlogy, and log 0 is mathematically undefined, then 0^x should also be undefined. Or on the other hand, one can examine the limit of y^x as x and y approach zero, and find that 0^0 ought to be 1, and 0^x ($x \neq 0$) ought to be zero. In any case, G E is looking for an efficient SR-52 routine that treats y^x such that an error is produced only when y is negative and x is not an integer, or when y is zero. He wants a correct result with correct sign when y is a negative real, with x any integer or 0, or y is a positive real with x any real, all in less than the 65 steps it has taken him. Members are invited to help out, and/or shed more light on the 0^x definition problem.

HYPERBOLIC FUNCTIONS SHORTCUT

Roy Grubb (483) passes along the following approach to producing hyperbolic trig functions, which he came across in a Sinclair Electronics manual for its small PPC: *LBL E (((INV lnx + 1) 1/x X 2 - 1) +/- INV tan X 2) *rtn. Using this basic routine, the desired functions are produced by: sinh: E tan; cosh: E cos 1/x; tanh: E sin; sech: E cos; cosech: E tan 1/x; coth: E sin 1/x. Inputs should be in radians; any angular mode will do.

MORE ON REG 60, 61, ...69 BEHAVIOR (52)

In attempting to find a pointer-filling shortcut, Phil Sturmfels (49) discovered a means of sensing which values held in the pending arithmetic stack are "non-normalized". (Following a convention adopted by the HP-65 Users Club, I will use the term "normalized" to apply to numbers composed only of binary-converted-to-decimal (BCD) bytes; "non-normalized to numbers with one or more of their 16 bytes representing hexadecimal (base sixteen) numbers.) The only way to move the contents of Reg 60 to a program register for detailed examination is through the display register, which always normalizes a non-normalized number (see V2N1p4). Register arithmetic on a non-normalized value in Reg 60 also appears to produce normalized results. However, as Phil discovered, a non-normalized number in Reg 60 is not affected by its use for indirect addressing, and its pointer behavior can identify it as non-normalized. For example, key: 1 + 2 X 3 y^x in RUN mode. Find that Reg 60, 61, and 62 appear to contain 1, 2, and 3 respectively. Now use these registers as pointers, and find that only the 2 behaves as a 2; the "1" and the "3" behave as 0, and it appears that they are non-normalized. It may be that there are some non-normalized numbers that point normally, and this phenomenon probably deserves further exploration.

PSEUDO 73 LIMITATIONS (52)

Larry Mayhew (145) points out that p73 "... cannot be trusted as the equivalent of rset without the flag modifications, because under certain important conditions it will simply be ignored during program execution. Example: Beginning at step 000, write: HLT INV *ifflg 1 *1' *LBL *1' 12 p73 45 HLT. In RUN mode key *rset RUN RUN and see 12 as expected. Now key *rset *stflg 1 RUN RUN and see 1245, which shows that the p73 executed as a no op. In general, if any conditional test has been made without a transfer occurring, and if after that test p73 occurs and there has been no intervening STO, RCL, SUM, EXC, PROD, CLR or fix, the p73 will be ignored."

STEP 223 ODD BEHAVIOR (52)

Larry also notes that "If step 223 contains part of a numerical address (program or register) that is left incomplete at 223, and if the program runs through step 223, and a digit is then keyed manually, a 'semi-crash' will occur: Every register is cleared, but flags remain unchanged." Or, if a conditional instruction is positioned at step 223, and executed with RUN n RUN where n is a digit, the machine goes into a read state. Larry has also gotten a step 224 to appear in the display in a sort of through-the-lookingglass Alice-In-Wonderlandish hocus pocus by way of a run-down battery. Write him for details.

INFORMATION REFERRALS

Largely to save time, I have adopted a policy of suggesting via 52-NOTES that members contact a contributor directly, without my having obtained prior approval from him in cases where I consider topics inappropriate for 52-NOTES coverage. I don't believe this has caused contributors to be overwhelmed with solicitations, but it always helps to include at least a SASE with each request. Members not wishing such announcements to be made should so indicate when contributing material.

MORE ON DON ELLIS' PUBLICATIONS (V2N3p3)

D K Patterson, Jr (272) has examined some of Don's statistics programs and reports an error in one of them. Members using Don's publications may wish to contact D K and/or Don for details. But send D K some change (rather than a SASE) since US stamps won't be of much use to him in Canada.

MORE ON SUM-OF-THE-DIGITS (V2N5p6)

I've received a number of 13-digit sum routines now, most of which won't handle all numbers, which makes them hard to compare. So here is a new routine from Howard Cook that appears to handle all reals, but requires 56 steps (Karl Hoppe (507) has one with 61 steps). Shorten it if you can, but "better" routines must be able to handle all reals. *LBL E INV *fix *ifpos +/- +/- *LBL +/- STO 01 EE div EE 00 = INV *Prod 01 *fix 0 *LBL *LBL CLR RCL 01 *ifzro *EXC - .5 = EE SUM 99 INV SUM 01 10 *PROD 01 GTO *LBL *EXC *EXC 99 *rtn. Comparable routines for the 56/57/58/59 ought be be shorter.

DECAPOWER ZERO SUPPRESSION FOLLOWING NEGATIVE TESTS (52)

Jared Weinberger (221) notes that any decapower zero is suppressed following an unmet test using a defined label for its transfer instruction. The suppressed-zero display can only be seen when a test is executed manually or SST'd, since a HLT or rtn restores the suppressed zero.

SR-56 PROGRAM EXCHANGE UPDATE (V1N6p6)

Dave Johnston's 1 July 77 Catalog lists 86 programs covering math, statistics, physics, games, finance, operations research, and misc. Dave plans to get a TI-57 and will broaden his exchange service to include 57 programs. And since TI does not plan to provide a program exchange service for the 58 either, there will be a need to be filled for that machine too. Write to Dave if you plan to buy either the 57 or 58. If response looks like more than he wants to handle, we'll look for additional help. Those members with 58s who join TI's PPX-59 and wish to contribute programs to it may send me copies of programs on TI forms with sample problems, all checked out on the 58, but scaled to the 59, and I will put them on mag cards (until such time as there are too many such requests). Include blank mag cards and a large SASE.

MACHINE COVERAGE IN 52-NOTES

Since the nature of contributed material influences how I prioritize newsletter topics, , if you want to see more coverage of a particular machine, write about it; clever routines, inventions, and discoveries are what I look for most, and appear to be what most of you prefer too.

DATA ENTRY SENSING

Izzy Nelken (576) asks if there is a way during SR-52 program execution to sense whether data have been entered from the keyboard. There is no built-in function in the TI machines comparable to the HP-67 Flag 3 (which is automatically set and post-test cleared when data are entered during a program halt or pauze), so it appears that the display would need to be examined, assuming that keyed data would fall in a given numerical range outside of the possible display values produced by the program. If anyone has a better idea, Izzy would be pleased to hear from you. But it's probably more practicable to design programs to always expect some data to be keyed at specific break points. A keyed zero or one provides easily tested data for a two-way branch: a handy way to respond yes or no to printed questions (see the TI-59/PC-100A program in V2N6p3).

MEMBERSHIP LIST CORRECTIONS

44: Box 233; 97: 18 Carty Ave Ft Monmouth NJ 07703; 307: Collins Radio Group MS 424-101 Dallas, TX 75207; 368: 2631 Navarre Ave #209 Oregon OH 43616; 376: Buchenweg 24 D-5000 Koeln 40 Germany; 518: Math Dept SUNY Geneseo, NY 14454.

ASSESSMENT OF THE 58/59 MASTER LIBRARY CROM PROGRAMS

Members with appropriate expertise can contribute significantly by assessing/analysing specific ML programs, All can be down-loaded into 59 user memory; all but ML02 and ML19 into a 58's. If you have a 58 but not a 59, and want listings of these two programs, send me a SASE. A cursory look at ML-02 (the longest CROM program) shows a repeated use of in-line code sequences, which may be justified by increased execution speed. But it also appears that more data registers than necessary are used. I invite more analysis of this and other ML programs, and will air pertinent discussion via 52-NOTES. While we can't change the CROM code, we can use it more effectively if we understand its structure and limitations.

58/59 TIPS

Short Form Addressing: Although the omission of leading zeros when keying register or step addresses can save manual keystrokes, it is easy to forget that a non-digit keystroke must follow. So it can be worth the extra manual steps to avoid address mistakes that can be time consuming to find; the number of required program steps is the same either way.

ML-01 Print Routine Use: The last sentence in the user instructions should read "... except that the program must not be called." If a program other than ML-1 is accessed, automatic printing won't occur. Also, only manually keyed user defined keys (A-E') will initiate the automatic prints (calls to other labels, absolute addresses, and/or R/S sequences will not). This routine does make use of indirect program accessing, and is worth examining (via Op 9 download) to see how it works.

Taking Advantage of Control Operations: From time to time, a glance at page V-27 of the owner's manual will remind you of all the things the 39 special control operations can do. Keep in mind the use of Ops 20-39 in place of programmed 1 SUM n or 1 INV SUM n (where 0 LT n LT 10); they save steps, don't alter the display, and run faster.

INV and Ind Combinations: There are a lot, and some can be complicated. Page V-68 sorts them out, showing proper sequences, and which Ind combinations are merged. One misprint that caught my eye is in the 19th entry: replace Dsz with ifflg. Remember that in all cases, Ind follows the associated direct instruction (unlike the 52 or 56). Note that for both flag and Dsz commands both the flag number or Dsz register and transfer address can be indirectly specified. A double indirect sequence such as: *Dsz *Ind 25 *Ind 26 must be completely rewritten if either indirect address is to be changed, otherwise the paired digits would not be properly merged. Incidently, *Dsz *Ind XX will dsz the contents of any addressable data register within the current partition as specified by the contents of Reg XX (not just Reg 0-9 as is the case for direct Dsz). For direct Dsz it is important to remember to key only a single-digit register address. For example, *Dsz 0 4 keyed in LRN mode is interpreted as *Dsz 0 004, which means decrement Reg 0 and go to step 004 if the contents of Reg 0 is not zero.

The Nop Instruction: Contrary to what the manual says on page V-51, there are quite a few code sequences which an interposed Nop will alter, notably between a command and an address, and between merged commands.

Getting Merged Code into the Last Partitioned Step: The mechanism by which some instructions are merged won't work at the last partitioned step. This appears to be because the first keystroke(s) of some merged instructions increment(s) the program pointer, and when this occurs at the last step, there is the automatic switch to RUN mode. The most likely such merged instruction to be at the last step is INVSBR (code 92), and here are two ways to get it there: 1) starting at the next to the last step, key RCL 92, then overwrite the RCL with the intended next-to-last instruction, or 2) if the last step is less than 959 (479 for the 58) temporarily partition to a larger program field, write INVSBR at the desired step, then repartition as desired.