# 52-NOTES

Volume 2 Number 12 48/39 December 1977

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

---

ROUTINES

Double Precision (56,57,58,59): Norman Herzberg (688) suggests that putting together a library of optimized fully double precision arithmetic routines would be a worthy endeavor for the membership. He leads off with the following integer multiplication routine which runs fast and appears to give correct results to 20 places. Printing the product on a 20-digit line might be a useful enhancement. To use Norman's routine: Key 10-digit multiplicand, press A; key 10-digit multiplier, press B, 10 MSDs of the product displayed; press xXt, see 10 LSDs of product. Multiplicand remains intact for use with a new multiplier. LE' ÷ xXt 1 EE 5 = Int xXt - xXt X xXt 1 EE 5 = rtn LB E' S4 xXt INV SUM4 X R1 S7 = S5 xXt X R2 INV SUM7 = S6 E' Exc6 + xXt + R7 X R4 +R5 = E' X 1 EE 5 = SUM6 xXt SUM5 CLR R6 xXt R5 rtn LA S1 E' S2 xXt Exc1 rtn. Members are invited to try to improve upon Norman's routine and/or address other double precision functions.

The Mathematical Integer Function (56,57,58,59): Joel Pitcairn (514) notes that the built-in Int function does not handle negative reals properly, since for mathematicians the integer part of x is defined as being the greatest integer LE x. Joel suggests using the following routine (slightly revised) to get the integer part of all reals: ...CP + INV Int Op 10 xGEt 1' + L1' 0 = Int... . I invite 56/57 versions (which will have to do without the Op 10 signum function).

More Conditionals (56,57,58,59): Although data can usually be arranged so that the conditionals x=t, xGEt, and their inverses will handle all required comparisons, there are times when it would be handy to have xLEt and its inverse as well. Jared Weinberger (221) suggests: ...xXt xGEt... for an effective xLEt, and ...xXt INV xGEt... for an effective xGTt. Ofcourse, the user needs to take into account the x-t reversal in subsequent processing.

Automatic Number Printer (58,59) Revisited (V2N10p2): It has been brought to my attention that R G Snow's routine might have a flaw, since an input 88888 returns 1111111109 instead of 1111111111. Well, it turns out that RG's routine translates 7s or 8s at either the MSD or LSD position into 8s or 9s, respectively, instead of 11s or 12s. But this is all right since print code 8 prints a 7, and a 9 prints an 8 (see v2n10p4).

MEMBERSHIP ADDRESS CHANGES

203: 9423 Bickford Rd Kernersville, NC 27284; 533: Mink Creek Rd Star Rte Box 431 Pocatello, Idaho 83201.

---

TIPS AND MISCELLANY

TI-58 Users Assistance: Roy Chardon (515) has a TI-59, and has kindly volunteered to broaden my offer to 58 users (V2N7p5): On a "to an extent determined by the circumstances" basis, Roy will 1) attempt to "complete" problems which won't fit on a 58, and 2) will serve as a PPX-59 channel... both "on a best effort basis, with no money (or mag cards) exchanged." Note that Roy's address changed recently (V2N10p5).

A p31 Application (52,58,59): Roy notes that p31 can be put to good use in programs requiring the user to write special applications code. For example, replace the R/S at step 373 of the V2N8p5 program with p31, and delete the first LRN in step 1 of the user instructions.

References To the 58/59 Owners Manual: I plan to continue referring to the original (dash one) version. When 52-NOTES references identify errors or omissions corrected by later versions, there is no harm done, but in cases where apparent contradiction or confusion results, I ask that cognizant members so inform me.

CROM Vs User Code Execution Speed (58,59): Tom Ferguson (421) found that CROM code executed within the module runs faster than the same code downloaded into user memory and then run. For example, the routine C part of getting 69! with ML-16 takes almost twice as long to execute in user memory as it does in the CROM. Members are invited to explore this further, and see if there are certain functions or sequences that show more marked CROM/user-execution speed differences than others.

LRN Mode Lockout (58,59): Fred Fish (606) notes that manual partitioning that leaves the program pointer out of bounds disables the LRN key. An RST restores LRN mode access, although a 0.59 TI-58 partition absolutely disables the LRN key as long as that partition is in effect.

More on Dsz (58,59): Jack Thompson (531) notes an exception to all-register Dsz-ing (V2N8p2): Dsz "40" is interpreted as Dsz Ind. So don't try to Dsz Reg 40 directly.

Ideas For a "Support Software" CROM (58,59): Several members have suggested that a CROM composed of widely used program building blocks would be expecially useful. Send me your candidate routines or ideas, and I'll consolidate them and see if we can interest TI.

More on Printer Head Cleaning: John Allen (104) reports that printer head cleaning with the heavy paper really works (V2N10p5). His PC-100A failed to print certain columns when he first bought it, but performing the cleaning instructions fixed it right up.

Protected Editing (58,59): If you don't want to disturb unrelocatable code near the bottom of program memory while editing code near the top, temporarily repartition to make into data the code to be protected.

INV Viability (58,59): The lack of INV viability noted in V2N6p3 does not apply to INV instructions placed before labels. For example: LA INV LB sin rtn executes as sin when called by B; as INV sin when called by A. Also, the normally merged rtn (INVSBR) can be broken up as: LA INV LB SBR 1' R/S L1' ... which if called by A returns without doing anything; calls SBR 1' with a call to B.

Mag Card Printing (52,59): Horizons Technology Inc is marketing ways to letter mag cards. For details, write Tom Schroeder (663).

SR-51/51A/PC-100A: Several members have called attention to the BYTE (Sept 77 p 176) letter describing SR-51/PC-100A compatibility. Michael DeTraglia (544) found that his SR-51A doesn't have to be "jammed" on to work. For best results, select SR-52 switch position, and trace mode. Printout of non-SR-52 mnemonics suggests that the extra SR-51 functions might at one time have been considered as part of the SR-52 design. Leon Ablon (619) notes that the PC-100 also works in this SR-51/51A application, and that for dollar totals the 51s are better than the 52 since all addends are printed to 2 decimal places with a fix 2 format (the 52 drops trailing zeros, as do the 56, 58 and 59).

SR-52 Extended Memory: Robin Wooding (350) has for several months been using an SR-52 modified with extra memory such that 160 of the normal 224 program steps can be manually switched out and replaced with 160 new steps, or vice versa. Write him for details of some of his applications.

HLT-R/S, rset-RST, and rtn-INVSBR 52-58/59 Differences: While the 52's HLT hardens a soft display, the 58/59's R/S does not. Bob Myers (566) notes that a programmed 52 HLT with one or more pending subroutine returns followed by a manually keyed A-E' that ends in rtn causes control to move back through the original calling path. For the 58/59, following a program-executed R/S and the manual A-E' the next INVSBR encountered executes as an R/S, and effectively clears the subroutine return address register. Upon encountering an rset, the 52 clears a soft display, while the 58/59 RST does not affect the display.

Fractured Display (58/59): Rusty Wright (581) found that in LRN mode at the last step of the current partition, keying a multi-step instruction causes the display to be reformatted in strange ways. For example, with a 59 at turn-on, key 1234 GTO 479 LRN GTO SBR, and see ".00000 1234". The number of digits in the preset display appears to determine the number of zeros appearing to the right of the decimal point. Other rules prevail with a floating point (scientific) display, and Dsz a b or Dsz*cd e at step 479 do even stranger things (the contents of Reg cd shows up in the display).

Corrections: Gerald Donnelly (203) and Roy Chardon (515) note that I got the "mental" and "spiritual" categories interchanged in translating from Heinrich's German (V2N9p3). Interchanging the register pairs 19-20 with 21-22 should set things straight. And, as printed, Bob Myers' printer sensing routine (V2N10p4) sets flag 7 when the printer is not connected.

Decapower Zero Suppression (52): Douglass Darrow (687) notes that the SSTd return from a called subroutine suppresses decapower zeros just as the unmet conditional test does (V2N7p4).

More on INV' and Dummy Operations (56,58,59): Rusty Wright (581) notes that for the 56, INV' supplies a missing operand, but requires 2 successive = or ) strokes for completion of the pending arithmetic. The Owners Manual specifies only CE as a shortcut to supplying missing operands, but Rusty finds that RST also works. The 58 and 59 act like the 56, except that STO, RCL, and SUM also produce missing operands when keyed manually, but as Jared Weinberger (221) found, when executed under program control treat the intended operation as a register address.

Program/Routine Listings for TI-56,57,58,59 Use: 52-NOTES articles whose titles are followed by: (56,57,58,59) or some subset thereof are likely to contain listings written for TI-59 use. Generally no changes are necessary for the 58, although in some cases, register addresses and partitioning need to be scaled down. For the 56, labels and their associated relative addressing must be replaced with absolute addressing, and for the 57, available labels should be substituted for specified A-E' ones. For both the 56 and 57, register assignments should be carefully assessed vis-a-vis existence and special use (such as by statistics functions). By and large, if a PPC's identifier doesn't appear in the title, application of associated routines may require a non-trivial translation effort (see V2N8p6).

SR-52A Zero Divide: Paul Blair (526) notes that the special zero divide error state (V2N11p3) does not occur with his SR-52A (V2N5p3).

Flashing Display Variations: Mark Stevans (216) notes that there are 2 types of flashing display for the SR-52: 1) Display is completely blank between flashed positive numbers GT 1, and 2) The 2 minus signs appear dimly between flashed numbers. Mark refers to the first as a hard flash, and the second as a soft flash. Dividing by zero or taking the INV sin of a number GT 1 produces a hard flash; keying an undefined label produces a soft one. An SR-56 error condition caused by a manual or SSTd STO INV' (discovered by Rusty Wright) is soft. I haven't yet found a way to produce a soft flash on the 58 or 59.

More on Execution of Unintended CROM Code (58,59): Rusty Wright found a code 81 at step 139 of Pgm 3 of the Statistics CROM, and reports that it executes as RST: control returns to step 000 of user memory (see V2N8p6). Rusty also found a couple of code 31s (steps 074 and 077) in the same CROM program, and found that they are ignored ... like p31 in protected code (V2N9p4).

Another SR-52 Analytic Model (V2N10p5): Ron has another program, titled: Computer Queuing Analysis on a Handheld Calculator (COMPUTER DESIGN Nov 77 pp85-94). Write Ron for details.

Quaternion Arithmetic (59): Joel Rice (3) has written programs to perform quaternion (spinor) arithmetic for both real and complex inputs, and is prepared to share his listings with members who send him a SASE with 2 stamps.

FRIENDLY COMPETITION

At least one HP-25 user has conceded that 100 unmerged SR-56 steps amount to more effective memory than 49 merged HP-25 steps (comment by Jim Davidson 65-NOTES V4N8p21)... and Jim has done a lot of HP-25 programming. Both machines are still powerful PPCs for the money, but the SR-56 has always cost less, and so far, no HP-25 program has come to my attention that couldn't be duplicated or bettered on the SR-56.

A year ago in a 65-NOTES article on Taylor Series Summations on Programmable Calculators (V3N9p17), Jim challenged users of AOS machines with: can you "...do as well as RPN for Taylor's series? Or even come close?" citing some HP-25 routines he had written. Well, Reinhold Patzer (689) has risen to the challenge with some SR-56 routines, all of which take less effective memory than the corresponding HP-25 routines, and produce results more accurate by a factor of 100. As a specific SR-56 response to Jim's benchmark program for $e^x$ (65-NOTES

V3N9p42 - Program 3), Reinhold offers: 00: S0 1 S1 X xXt R0 ÷ R1 X (CE + xXt X xXt 1 SUM 1) INV x=t 06 xXt R/S, which in 30 steps takes less effective program memory than Jim's HP-25 routine: 01: S0 0 S1 1 S2 R2 R1 1 + S1 ÷ R0 X S2 + x≠y GTO 06 GTO 00, where 18 HP-25 steps use 36.7% of memory compared with Reinhold's 30 % (29% if one omits the last xXt, which appears to be unnecessary). On a test case of $e^{25}$ I find that Jim's routine gets 6-place accuracy in 64 iterations, taking 46.1 seconds. Reinhold's gets 9 places in 71 iterations, taking 67.3 seconds. Thus it would appear that Jim's runs significantly faster, at 1.39 iterations/second vs Reinhold's at 1.05, but it would take Reinhold's only 50.4 seconds to get 6-place accuracy. In his article, Jim points out the HP RPN features that facilitate his approach, but it is worth noting that one of the key features: datum replication from T to Z when the stack is dropped is nicely matched in this exercise by the SR-56 T register feature that its contents is not altered by AOS stack movement, or arithmetic operations. It should also be noted that AOS machines without the T register (notably the SR-52) would be at a significant disadvantage in this competition. Another point: When HP-25/SR-56 routines are being compared, if their application is enhanced by being subroutine-callable, as might well be the case in this exercise, the HP-25 loses. However, if Reinhold's routine is called by a main program, there is a pending arithmetic operation that must be cleared following disposition of the results if further use of the arithmetic stack is to be made.

Challenges aside, I recommend Jim's article to PPC users who need help in formulating problems and synthesizing algorithms. Jim takes the reader from brute force, inefficient, but easy-to-understand approaches to trickier more efficient ones, which as Reinhold shows can apply to AOS as well as to RPN machines. Reinhold acknowledges that Jim gets the credit for tricky exiting, and efficient $x^i$ and $x!$ computations.

Hal Brown has succeeded in getting a new version (65-NOTES V4N8p10) of his 5 X 5 matrix program (52-NOTES V2N8p3 and V2N10p2) to solve more special-case problems without manual intervention, and appears to have eliminated the error-without-warning cases. Interested members may send a SASE to Richard Nelson (2) for copies of this latest version, but are advised to insert 2 missing steps: 50: GTO 0 and 150: x≠0. I invite comments. The only other HP-67/97 5 X 5 program to come to my attention is in the "High Level Math" volume of HP's recently published Users Library Solutions. This is a 1-card program by John L Gustafson Ruddock House Caltech Pasadena, CA 91126 that calculates the determinant and inverse by a method similar to Barbara Osofsky's (V2N5p5), requiring no manual permutation of rows or columns, but which is less compact, slower (takes about half an hour to run), and does not handle ill-conditioned matrices as well as either Barbara's or Hal's. For the ill-conditioned matrix A whose elements are $a_{ij}=1/(i+j-1)$, John's program doesn't quite get one-place accuracy, Barbara's gets 4, but Hal's wins with 5 (TI-58/59 ML-02 gets 8). Here is a situation where for a particular approach the greater arithmetic precision of the TI machines is significant. If Barbara's program were to be run with 10-digit precision, results would probably be similar to John's. Hal's and ML-02 both employ pivoting, and I would expect ML-02 results to be similar to Hal's if produced with 10-digit precision.

KEYING PRINTER CHARACTER CODE (58/59)

While the Rausch Overlay method (V1N3p2) works well to number-code the alphabet for non-alphanumeric machines, it does not handle all 64 of the 58/59 print characters, and requires code-translation-machine-execution time. Lou Cargile (625) has tried a different approach, which in one form or another looks promising: He has typed/written out the 64 characters on strips of paper which he fastens conveniently on the printer. Characters are grouped according to the first digit of their print codes, and are annotated with the second digit. For example, associated with the numeral 2 key are the characters FGHIJKL which have as their respective second digits 1234567. Some might find it easier just to string out all 64 characters with the corresponding 2-digit print codes beneath each; or perhaps Lou's method could be sufficiently miniaturized to make feasible a direct keyboard overlay. Whatever the approach, the goal is to optimize the speed and accuracy with which the user converts the characters to 2-digit keystrokes, until such time as he has memorized them. Other ideas are invited from the membership.

EDITORIAL: 52-NOTES STYLE/CLUB PHILOSOPHY REVISITED

A majority of members who comment on 52-NOTES style and content continue to like things as they are. Complaints haven't changed much since we began, and generally reduce to: 1) material is too technical and/or terse, 2) important items are buried in long paragraphs, 3) abbreviations/mnemonics are not defined often enough, and 4) membership expiration notices should be issued. The nature of these complaints suggests that the few dissatisfied members regard their membership as a one-way magazine subscription, and I recommend that they reexamine the Club's purpose (V1N1p1), and my approach to writing 52-NOTES (V1N5p1, V2N7p1). People teaching technical disciplines often find it helpful to point out to their students that they should not consider such courses to be spectator sports. Learning to use modern computing machinery is no exception, and I reemphasize the importance of actually working examples on your PPC, and carefully and thoroughly covering all references. Don't skip over something you don't understand at first glance; dig into it, and write for help if need be. One advantage of a 6-page newsletter is that it is not impractical to re-read many back issues from time to time... a good way to put lost or overlooked tools within reach, and to refocus your attention on topics whose value to you has waxed since first acquaintance.

I'm pleased that the ranks of the productive minority are growing, but suspect that there are still many in the silent majority who are hiding their lamps under the proverbial bushel. Members who participate the most seem to have the least difficulty getting all the information they want from each issue of 52-NOTES, and they rarely miss an issue due to forgetting membership expiration dates! I think it benefits all concerned to keep 52-NOTES' technical level reasonably high, otherwise the top contributors will lose interest.

As the end of 1977 approaches, I take this opportunity to wish you and yours a happy holiday season, and another challenging year ahead.