52-NOTES

Volume 2 Number 10 48/39 October 1977

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

- -

FRIENDLY COMPETITION

Dix Fulton (83) has responded to John Ball's challenge (V2N8p4) with an SR-56 satellite predictor program that processes inputs and creates its own constants (neither of which John's does) in addition to producing the same results. So now the challenge is reversed! However, whatever the final outcome, John gets the credit for the efficient use of the polar-rectangular functions that shortens the trig calculations (V1N4p5) as well as a clever use of the x̄ (arithmetic mean) function that shortens two division calculations. Here is Dix' counter challenger:

SR-56 Program: Satellite Predictor — Dix Fulton (83)

User Instructions:

1. Enter Inputs: Key Nodal Longitude (degrees), - Observer Long. =, press RST R/S; key observer latitude, press R/S; key orbital period (minutes), press R/S; key inclination (degrees), press R/S; key height (miles), press R/S; key time from nodal crossing (minutes), press R/S; see Azimuth (degrees) displayed.
2. Get Range (miles): press R/S
3. Get Elevation (degrees): press R/S

Program Listing:

```
00: S1 R/S S2 R/S div 1440 = s7 R/S S4 R/S S6 R/S div 4 = S5 f(n) Mean
28: sin xXt R4 f(n) P/R S0 f(n) Mean cos xXt ± f(n) R/P + R5 + R1 =
50: f(n) P/R ± EXC0 f(n) R/P - R2 = f(n) P/R xXt EXC0 f(n) R/P EXC0
70: f(n) R/P xXt R6 + 3958 = xXt f(n) P/R - 3958 = f(n) R/P EXC0 R/S
95: xXt R/S EXC0 R/S
```

- - - - - - - -

J R Merrill (693) has submitted the following Extended Precision Powers SR-52 Program as an HP-67 challenger:

SR-52 Program: Extended Precision Powers — J R Merrill (693)

User Instructions: For y^x key y (0 LT INT LT 100), press A; key x (INT) press B; see first ten digits of results; press C for next ten digits, and repeat until display flashes. x must be sufficiently small that y^x is less than 10^{220}.

- -

SR-52 Program: Extended Precision Powers J R Merrill (693)

Program Listing:

```
000:  LA S62 rtn LB S67 CMs 0 S99 1 S98 98 S69 97 S66 0 S68 1 Sum66
038:  *R66 X R62 + R68 = *S66 div 1 EE 10 + 1 EE 11 - 1 EE 11 = S68 X
075:  1 ± EE 10 = *SUM66 R66 - R69 = INV ifpos 034 1 ± SUM 67 R68 ±
107:  ifpos 122 1 SUM69 R68 *SUM 69 R67 INV ifzro 025 R69 + 1 = S97
139:  INV EE 1 ± SUM97 *R97 HLT LC R97 - 98 = INV ifzro 141 0 1/x 0 HLT
```

- - - - - - - -

Specific matrices have been found for which Hal Brown's HP-67 program (V2N8p3) gives incorrect results without warning, requiring the user to calculate the inverse of the inverse as a validity check. Hal has been developing improved versions of his program, including one that prints (with the HP-97), but has not yet eliminated the requirement for this validity check nor what can amount to an unacceptable number of manual row/column interchanges. It would appear that he will need to provide for more pivoting and/or treatment of near-zero numbers as zeros to eliminate these shortcomings... and there may not be sufficient HP-67 memory for a one-card program.

- - - - - - - -

ROUTINES

A Short Flag-Reversal (52): Jared Weinberger (221) has devised: ... iffl g 0 LBL LBL INV stflg 0 ... as an efficient way to alternate a flag's state each time this sequence is exercised.

More on Sum of the Digits (V2N7p4): Jared is back in the running with: L1 (S99 EE ÷ EE 00) INV SUM99 1 + R99 LA INV ifpos LBL LBL ± INV ifzro 1' = rtn which in 33 steps handles all 13-place reals with a call to A.

Automatic Number Printer (58/59: R G Snow (212) has devised a sequence to convert up to a 5-digit positive integer into the equivalent print code. Following is his routine, revised to be relocatable: LA CP x=t 1' ÷ log Int S8 Op28 INV log L2' + xXt 100 Prd 21 8 xGEt 1' 2 + L1' 1 - Int SUM21 = X 10 Dsz 8 2' CLR EXC21 Op*4 Op5 rtn. With the desired print sector (1-4) stored in Reg 4 and the number to be translated in the display, this routine (A) prints the input number in the desired sector.

Factorials: For the SR-52, Larry Mayhew (145) notes that display-rounding of a positive real presents a true integer to the X! function (there is no error condition set).

It may be worth some effort to find the most efficient ways to get $X!$ on the 58/59, just as with Int/frac routines for the SR-52 since there is no built-in $X!$ function for the 58/59. ML-16 will do factorials, but user program access takes 9 steps, and 69! takes 20 seconds to execute. A closed solution using Stirling's formula which approximates n! by: $(2npi)^{\frac{1}{2}} \times (n/e)^n$ might be better for large n. I invite efficient mechanizations of this formula and/or better approaches from the membership.

Special Case Routines: Larry Mayhew (145) points out the value of using efficient special-case routines when there is assurance that data involved are consistent with applicable restrictions. For example, he has found that for the SR-52, just plain D.MS will properly round the inexact results produced by y^x, log, etc on integers. This also works for the 58/59. A fix 0 prefix is only required when a

fractional part shows up in the display. Larry has also found that the inexact results of raising 10 to integer powers less than 11 via INV log can be made exact with just D.MS. For the 58/59 this applies to the full range from 0 to 99, since the shorter display mantissa does all the necessary display rounding. All the machines produce exact results with INV log n when n is 20, 30, 40, or 50. For other n between 10 and 100, fix 0 is required to produce exact SR-52 results.

Eigenvalue Calculations (58/59: Bob Thacker (30) discovered that ML-02 could be used to calculate real eigenvalues (in some cases) from real square matrices by the so-called Rutishauser L-R Transformation (iterative) method. The key to Bob's approach is a subroutine call to step 682; but as he notes, in some cases in order to get convergence, pivoting must be suppressed, or the matrix rearranged before being input. Successful convergence reduces the original matrix to upper triangular form, with the eigenvalues along the diagonal. The following routine may be used with ML-02 to perform a specified number of iterations automatically: LA S0 L1' Pgm 2 C Pgm 2 SBR 682 Dsz 0 1' R/S. 1. Enter matrix elements per ML-02 steps 1-3. 2. Key number of iterations, press RST A; 1 displayed. 3. Examine elements: press Pgm 2 C', R/S, see ith element; repeat for i=1,2,...n^2. 4. Examine row indices: press R/S, repeat n times: If not 1,2,...n, pivoting has occurred. 5. For more iterations, go to step 2; for new problem, go to step 1. Printouts of each successive iteration via the PC-100A would enhance the detection of convergence, and precision growth.

I/O IDEAS

An SR-52 air navigation program from Ernst Viehweger (696) provides examples of some handy I/O techniques that may be helpful with other machines, and in other applications involving many I/O parameters. Ernst organizes inputs such that they may be entered in any order and/or combination. Similarly, outputs may be individually specified, with computation time minimized when there has been no change in any of the inputs. Each input is processed via a user-defined key (A-E'), but keys are economized by having the user connect related pairs with + or - operators, later separated by the input routines with 0 =: each input routine ends with either rset or INV stflg. The flag is set by the main processing routine, and tested by a single output routine, determining whether or not a new computation is required. Keyboard integer inputs to the output routine correspond to the register addresses where the desired outputs are located and which are retrieved by indirect addressing.

LIST/TRACE OPTIONS UNDER PROGRAM CONTROL and MORE ON PRINTER CONNECTION SENSING (58/59/PC100A)

In the course of trying to find ways for a running program to determine printer connection (V2N9p2), A B Winston (707) examined some of the listing options both with and without the printer, when executed under program control. His results lead to the following observations: Contrary to the last statement on page VI-4 of the owner's manual, termination of INV List executed under program control does not return control to the keyboard, and both program and label listing can be made under program control without relinquishing control to the keyboard upon completion. The latter are accomplished by having a List or Op 8 instruction at the beginning of a called subroutine which ends with an INVSBR as the last step in the current partition.

For example, if program partitioning ends with step 479, the sequence: ... SBR 476... 476: List stflg 5 rtn executes under program control beginning at SBR by listing steps 477-479, then resumes with the steps following the SBR 476 call. The sequence: ...SBR 475... 475: Op 8 stflg 5 rtn executes as a label search from step 477 to step 479, and returns to the calling program having effectively done nothing. In both cases, the SBR call without printer connection causes flag 5 to be set. Thus at a cost of only 8 steps, a flag can be automatically set when a program is running without printer connection... better than the 12 steps required by the HIR method (V2N9p2). For this purpose, OP 8 is "cleaner" than List. Then, as A B suggests, it only takes 8 steps to do: ...SBR 475... 475: Prt Op 8 R/S rtn which either prints and continues, or halts, depending upon printer connection. R U Myers (566) suggests: ...20 Op 7 Op 18 CE... which in only 7 steps sets flag 7 if the printer is connected, but precludes the use of flag 7 for sensing real errors.

While program call-execution of the List and Op 8 functions may find occasional practical application apart from printer sensing, call-execution of INV List will probably find greater use: It's a cheap way to output tagged results, and do this without a forced halt.

Bob Myers brought to my attention the special trace-control feature of flag 9, which is rather obscurely mentioned on page IV-65 of the owner's manual (instead of on page V-67). Selective trace-printing under program control is a nice feature, but ignorance of this special flag 9 behavior could cause considerable debugging grief!

TIPS and MISCELLANY

No-Hassle Partitioning (59): Roy Chardon (515) suggests recording any/all programs with the 479.59 default partition, then let the program repartition as required, saving the user from having to manually repartition before card read.

Clerical Aids: Mack Maloney (246) suggests writing VxNxpx opposite a membership list entry corresponding to the 52-NOTES applicable correction, obviating the need to squeeze corrections in on the list itself. He also suggests that correspondence to me requiring a reply be written or typed with sufficient left hand margin to give me room to make my reply on a copy. This will save me time, and remind you what I'm replying to. If you don't type, use a soft lead pencil, or black ink, or other writing medium that will work with a thermal copier.

Statistics Keys Tricks: Sandy Greenfarb (200) suggests that clever manipulation of the statistics built-in functions (and/or the statistics Ops) may yield shortcuts to mechanizing unrelated problems (see Dix Fulton's SR-56 challenger program elsewhere in this issue). The idea is to take advantage of the various arithmetic combinations of inputs distributed among the registers used by the statistics functions. I invite the membership to explore the possibilities and to report fruitful results.

Extended Print Code (58/59): Jack Thompson (531) notes that all 100 2-digit integers produce print code. There are no unannounced characters, but even numbered character matrix rows except 8 may be extended 2 characters into the next row by adding columns 8 and 9. For example, 08 prints a 7 and 09 prints an 8 (handy for incremented printout); 28 prints an M, and 29 an N. But 18 is the same as 10, and 19 the same as 11. This print-code behavior is also noted by Carl Seel (328

More on Joel Pitcairn's Trig Algorithms (V2N9p6): What was incorrectly referred to as a BYTE magazine error is an approach that can be significantly improved. Joel has since found that setting: Tan Ao=(xi+Aiyi)/(yi-AiXi), Arctan xo= zi+xi/yi, Tanh Ao=(xi+Aiyi)/(yi+Aixi), Arctanh xo=zi+xi/yi yields 13 place results with Max=7 (about half the number of iterations originally required). Joel's findings should be of interest to software/firmware designers for full scale computers as well as for the PPCs.

More on Printer Head Cleaning V2N9p3): As many of you wrote, there are only 100 print elements, not 700, and I stand corrected. However, it seems that there might be more heat buildup in an element that is fired 7 times in the fraction of a second that a line is formed than if it is only fired once. It also appears that if only one character row is printed per line, available heating power is more concentrated. But in any case, the proof should be in the pudding... has anyone actually unclogged a print element using a head cleaning routine with the heavy paper?

CROM Library Notes (58/59): For the Applied Statistics (2) Library, Gerald Donnelly (203) has an improved data entry method for program ST-03. Write him for details. Gene Werner (120) warns that a 3 X 8 inch addendum sheet packed loosely in the box is easy to miss.

PC-100 Modification for TI-58/59 Use: Steve Marum (188) claims to know how to modify a PC-100 printer so it acts like a PC-100A (without battery charging). Write him for details.

SR-52 Mechanizations of Analytic Models: Ron Zussman (88) has written 2 more programs in conjunction with recently published articles: "How to Anticipate Performance of Multipoint Lines" (DATA COMMUNICATIONS July 77 pp51-54) and "Predict System Dependability With a Pocket Calculator" (ELECTRONIC DESIGN 13 Sep 77 pp100-104). Write Ron at 2456 Ocean Parkway Brooklyn, NY 11235 for details.

Membership Address Changes: 161: 6615 Kentland Ave Canoga Park, CA 91307; 450: 17 Pinewood Dr West Boylston, MA 01583; 538: 5019 Calhoun #232 Houston, TX 77004; 120: 11006 Jean Rd Huntsville, AL 35803; 188: 520 Talley Sherman, TX 75090;373: 1241 Amherst W Los Angeles, CA 90025; 515: 2527B 25th Loop KAFB, NM 87116; 49: 510 Yeatman St Louis, MO 63119; 365: 60 San Milano Goleta, CA 93017.

More on Card Read Under Program Control (59): A reliable source has revealed a program-execution-architecture feature that renders the parallel processing inference (V2N9p5) incorrect. The observed behavior is due to the way the machine processes instructions: Before a step is executed, the entire contents (8 steps) of the register containing this step must be in a code-execution buffer. When this step is a Write preceded by INV, a prepositioned card is read, then the rest of the code in the code-execution buffer is executed, provided none of it causes a transfer out. Execution then resumes with the first step in the next octet of stored code, which now is new, having just been read in. What appeared to be a bank position dependency was coincidentally a register-position dependency. At most, 7 steps of old code execute following a Write in the first step of an octet.

Ops 20-39 With a 959.0 Partition (58/59): James Merrill (693) found that attempts to increment or decrement Reg 0-9 when they are not within the data partition produce unique flashing displays. Results are different for the 58 with a 479.0 partition than for the 59 with 959.0 I invite explanations for displayed results from the membership.

TIC TAC TOE (59/PC-100A)

Although Tic Tac Toe is more an easy puzzle than a game, its computer/calculator mechanization can be challenging. Fred Fitzgerald (252) leads off with a challenger that follows; press A and follow instructions. Other members are invited to try to out-do Fred with more efficient algorithms, better I/O, etc.

TI-59/PC-100A Program: TIC TAC TOE Fred Fitzgerald

Program Listing:

```
000:  LD' R*11 Op02 1 SUM11 R*11 Op03 1 SUM11 R*11 Op4 Op05 1 SUM11 rtn
026:  LE' R49 Op2 Op3 Op4 Op5 rtn LB' 1 SUM12 R*11 + R*9 SUM11 R*11 +
054:  R*9 SUM11 R*11 = S*12 x=t 5' - R41 = x=t 073 rtn stflg2 R11 S19
079:  R9 S18 rtn LC' R45 SUM1 SUM4 SUM07 0 S11 D' E' D' E' D' R45 ± SUM1
107:  SUM4 SUM7 CLR Op0 EXC29 x=t 132 Adv Op4 CLR EXC30 Op3 Op5 GTO 428
132:  Adv Adv Adv CLR R/S Prt Adv - 1 = S10 R*10 x=t 160 R44 S29 R43
155:  S30 GTO 111 1 SUM40 R46 S*10 R9 INV x=t 187 4 S11 R4 x=t 9' 0 S11
183:  stflg1 GTO 9' R58 xXt 20 S12 8 S11 B' Dsz "11" 197 Op29 B' 1 S11
208:  B' 2 S11 B' Op29 B' 2 S11 Op29 B' INV iflg2 237 R19 S11 R18 S9
235:  GTO 5' R59 xXt R28 xGEt 8' R27 xGEt 8' CP 0 S11 R0 INV x=t 269
258:  R59 - R23 - R24 = x=t 9' 2 S11 R2 INV x=t 289 R59 - R23 - R26 =
287:  x=t 9' 6 S11 R6 INV x=t 309 R59 - R24 - R21 = x=t 9' 8 S11 R8
314:  x=t 9' L8' CP INV iflg1 327 3 S10 1 INV SUM10 R10 ABS S11 R*11
338:  x=t 9' GTO 327 L9' CP 51 S9 R42 S*11 R*12 - R58 = x=t 377 R40 - 4
367:  = INV x=t C' R50 S29 gto C' R56 S30 R57 S29 GTO C' L5' INV stflg2
392:  CP R*11 x=t 9' R11 - R*9 = INV xGEt 371 S11 GTO 5' LA R39 Op2 R38
419:  Op3 R37 Op4 Op5 E' CP Adv CLR 3 Op17 CMs 6 Op 17 INV stflg1 R40
443:  x=t 451 0 S40 GTO C' Op0 R36 Op3 R35 Op4 Op5 R34 Op3 R33 Op4 Op5 Adv
474:  GTO C'
```

Prestored Data:

```
33:  3563360000 3341362300 2201200 1731371735 3732170000 37131500
39:  372415 0 360000 320000 4532410015 2317133773 200000002 500000 0 0
49:  2020202020 1635134340 -1 3 -4 2 0 45324100 2732361773 640000
59:  1000000
```

- - - - - - - -

MORE ON HIR OPERATIONS (58/59):

R G Snow (212) notes that Ops 1-4 copy the display into the 10 LSDs of HIRs 5-8. This is a non-normalized format for data representation (see V1N1p4,5). When such a "number" is displayed, or acted upon by register arithmetic, it is normalized: the digit string is shifted left until the most significant place of the mantissa is non-zero, and the effective multiplications by factors of ten are cancelled by corresponding decrements of the decapower (see V2N1p3,4). In this normalized format, the original print code would not be correctly interpreted, but by adding a 1-digit integer to it, it would be. For example 1314151617 Op1 Op5 prints ABCDE. 1 HIR 35 normalizes with correct printer interpretation the contents of HIR 5 (Op 5 again produces ABCDE). 1.5 EE ± 7 HIR 55 effectively erases the C, and a subsequent Op 5 produces AB DE. As R G points out, since the printer only looks at the mantissa's ten LSDs, multiplications (or divisions) of the normalized contents of HIRs 5-8 by factors of ten do not change printer interpretation. R G suggests use of the P/R function to transfer the contents of the X and T registers to HIRs 7 and 8. The sequence: a xXt b P/R stores a in HIR 7 and b in HIR 8. He has also found that an SST'd HIR followed by a manually keyed Ind ab performs the HIR operation specified by the contents of (ordinary) Reg ab.