52-NOTES

Volume 1 Number 4 48/48 September 1976
Newsletter of the SR-52 Users Club
published at
9459 Taylorsville Road
Dayton, OH 45424

TI ANNOUNCES ITS PROFESSIONAL PROGRAM EXCHANGE (PPX) FOR THE SR-52

$15.00 sent annually to TI entitles the sender to PPX membership, a catalog, and three programs. Additional programs cost $3.00 each. PPX members will receive one free program and several (probably four) mag cards for each program they submit that is accepted by TI. The first catalog will list several hundred (maybe a thousand) programs.

NEW SR-52 OWNERS MANUAL

A new owners manual is now available from TI for $4.95 a copy, which recognizes and discusses previously unannounced features. See the TI advertisement in the September 1976 issue of Scientific American magazine for details about previously unannounced features, and the PPX service.

EDITORIAL

Now that TI has announced its PPX-52 service, it will be my policy not to publish programs that have been (or are planned to be) submitted to TI. I will continue to give priority to clever routines and programs that demonstrate new techniques over those that are mechanized in well known ways. Unpublished programs will be returned to members who so request, and who send SASEs.

MORE ON LBL LBL

Jared Weinberger (221) adds another LBL LBL trick with the sequence: *ifflg 2, *LBL *LBL, *rtx..., which does the same thing as: INV, *ifflg 2, *1, *rtx, *LBL *1',... for a savings of two steps. This use of LBL LBL can be applied to any of the conditional branch statements where a single instruction is to be executed for one condition, and not for the other.

Jared also suggests that the sequence: *ifpos *LBL *LBL A ... will cause SBR A to be called if the display is positive, otherwise "continue". If the sequence following A ends with *rtn, then in the positive case the A sequence is executed twice; in the negative case it is executed once. If the A sequence ends with a HLT it would only be executed once in either case, but the next *rtn encountered would cause a branch to the step following A, unless neutralized by an intervening *rset.

NUMBER RANGE NOTATION

The statement: "2 LT INT LT 10^5" which appears in the Dynamic NIM game (V1N3p3) means that the input number should be an integer greater than 2 and less than 100,000.

The SR-52 Users Club is a non-profit loosely organized group of SR-52/56 owners/users who wish to get more out of their machines by exchanging ideas. Activity centers on a monthly newsletter, 52-NOTES edited and published by Richard C Vanderburgh in Dayton, Ohio. The SR-52 Users Club is neither sponsored nor officially sanctioned by Texas Instruments, Incorporated. Membership is open to any interested person, and a contribution of $6.00 brings the sender six issues of 52-NOTES.

TABLE LOOKUP APPLICATIONS

When a long sequence of functions is to be mechanized, programmers often find it expedient to generate tables, where elements and their ordering determine the functions and their execution ordering, especially if there is no (or no apparent) repetitive pattern to the sequence. In most cases, each table element serves as a pointer to a program-step or register address.

Taking advantage of SR-52 pointer rules (V1N3p1), Alan Trimble (161) was able to apply a table lookup scheme to Dix Fulton's (83) 4 X 4 Determinant program (V1N3p4) to shorten it sufficiently to provide enough room to add a Cramer's Rule solution to four simultaneous equations. The following program is a modification of Alan's program, that speeds up the processing and simplifies inputting. Registers 90-97 contain a lookup table which provides information that specifies the location and sequence of data registers that need to be recalled for determinant calculations. Each of these registers (representing 8 program steps) contains a sequence of 6 pointers to data registers. Note that 6 straight forward recalls would require 18 program steps. Subroutine C sequences through the 8-register lookup table, moving decimal points to configure register contents for desired pointing. Incidently, I wondered at first how Alan's table loading could work, since some of the table registers contain non-transferable code (registers 90, 92, and 93, since the 8th position in each is a numeral). See V1N2p2 for a discussion of transferability. Fortunately, restructuring by the display is just what is required. For example, the keystrokes: rtx, 1, E, D, C, 7, 6, 5 in Reg 90 (which start at step 160) transfer as: *1/x, 1, *stflg, GTO, LRN, SST, *ifflg, *stflg. This sequence could have been specified to be keyed in in the first place, but the two pseudos: LRN and SST are a nuissance to create.

All the required pointer manipulation, indirect addressing and subroutine calling add significantly to execution time. So, although this program does calculate the determinant of the coefficient matrix (stored in Reg 66), it takes 6 or 7 times as long to run as Dix Fulton's. However, it all fits on one mag card, and there is plenty of room (steps 145-159) to add print instructions. But since there is unrelocatable code at the end, insertions must be made carefully. One safe way is to count them first, then delete a corresponding number of steps in the 145-159 region before making the insertions. Insertions made "above" step 034 will affect the *ifzro absolute branch at steps 069, 070, and 071.

SR-52 Program: Solution To 4 Simultaneous Equations   Trimble/Ed

Press CLR, then key the 20 equation elements by columns: four X1 coefficients, four X2 coefficients, ... four constants. Follow each entry with a keyed A. Elements are numbered 0-19, with the next element to be input shown in each intermediate display. Repeat /:RUN, see Xi :/ i=1,2,3,4

```
000:  *LBL C 1 SUM 99 (*IND RCL 99 x .01 yx RCL 67) STO 98 *IND RCL 98 *rtn
028:  *LBL D 6 STO 67 89 STO 99 (C X C - C X C)X(C X C - C X C) + 1 INV
061:   SUM 67 RCL 67 INV *ifzro 034 0 = div RCL 66 = *rtn
080:  *LBL B *IND RCL 68 *IND *EXC 69
090:  *LBL A *IND STO 68 1 SUM 68 SUM 69 RCL 68 *rtn 1 STO 66 D STO 66 0
116:   STO 69 E D HLT E D +/- HLT E D HLT E D +/- HLT
133:  *LBL E 16 STO 68 B B B B *rtn         (steps 145 through 159 unused)
160:  rtx 1 E D C 7 6 5 rtx 1 8 8 8 0 0 0 rtx 1 B B B 4 4 4
184:  rtx 1 A *E' 9 3 2 1 rtx 1 6 3 7 D A E rtx 1 1 5 2 9 C *E'
208:  rtx 1 2 7 3 *E' E A rtx 1 5 1 6 C 9 D
```

- - - - - - - - - -

A SHOOTING STARS GAME (USING VECTORED PROCESSING)

Since several members have shown an interest in a game (puzzle) called Shooting Stars (see BYTE magazine May 1976), here is a mechanization which makes use of vectored processing (see V1N3p5). A Shooting Stars program needs to present an initial 3 X 3 array, then alter it in accordance with predetermined rules, depending on a succession of player's moves. Since the player is limited to 9 possible inputs (the numerals 1 through 9), and since the most processing required (for the number 5) is 9 steps, each input is scaled by a factor of 9 and displaced (by 136) to make it a pointer to the desired program step to begin the processing. Although it might appear that this program can't be very efficient since 24 program steps are "wasted" (steps 143, 144, 152, 153, 159-162, 170, 171, 177-180, 195-198, 206, 207, 213-216 are unused), the efficiency of the vectored processing more than compensates for the wasted space.

Incidently, Shooting Stars does have solutions. Craig Pearce (18) has a winning sequence of 17 shots that works both forwards and backwards. Has anyone succeeded in using fewer shots to win?

SR-52 Program: Shooting Stars (for PC-100 printer) Ed

1. Initialize "Universe": press E; see printed:

```
111
181
111
```

2. Shoot a star (an 8) by keying a number (1-9) corresponding to the position of a star from the locator:

```
123
456
789
```

At the start, the only star is at position 5. See printed a verification of the shot, followed by a modified Universe which shows that the shot star has been turned into a black hole (a 1), and that surrounding stars and black holes have been "complemented" (stars have become black holes and vice versa) in accordance with the following rules, where * is the shot star, #'s indicate affected stars or black holes, and o's indicate the unaffected objects:

```
* # o   # * #   o # *   # o o   o # o   o o #   o o o   o o o   o o o
# # o   o o o   o # #   * o o   # * #   o o *   # # o   o o o   o # #
o o o   o o o   o o o   # o o   o # o   o o #   * # o   # * #   o # *
```

3. Repeat step 2 until you win: obtain a central black hole surrounded by stars, in which case the winning printed universe is followed by a negative integer indicating how many shots were required. If an attempt is made to shoot a black hole, -1 will be flashed. Recover by pressing CE, and try again (go to step 2).

Program Listing

```
000:  CLR B B B - 2593 - RCL 05 = *ifzro 050 CLR HLT
020:  *LBL B + (C X 100 + C X 10 + C) *prt *rtn
039:  *LBL C 1 SUM 69 *IND RCL 69 *rtn RCL 19 +/- *prt HLT
056:  *LBL *A' *IND RCL 98 - 8 = *rtn
066:  *LBL D STO 98 *A' *ifzro 078 7 + 1 = *IND STO 98 *rtn
085:  *LBL E *CMs 9 STO 00 1 *IND STO 00 *dsz 093 8 STO 05 *rset
106:  *LBL A *pap *prt D *A' *ifzro 134 9 *PROD 98 136 SUM 98 1 SUM 19
       *IND GTO 98
134:  RCL 98 D 1 +/- + = HLT
145:  2 D 4 D 5 D *rset     154:  1 D 3 D *rset     163: 2 D 5 D 6 D *rset
172:  1 D 7 D *rset         181:  2 D 4 D 6 D 8 D *rset 190: 3 D 9 D *rset
199:  4 D 5 D 8 D *rset     208:  7 D 9 D *rset     217: 5 D 6 D 8 D *rset
```

- - - - - - - -

DIAGNOSTIC PROGRAMS

Mark Stevans (216) points out that TI's Diagnostic Program I (BA 1-18 in the Basic Library) will not produce the intended error codes when malfunctions are detected. The intended results do not occur because no matter what branches are executed, the program always terminates with the same "all is well" sequence. This anomaly can be remedied by replacing the *rtn instructions at steps 035, 050, 065, 084, and 089 with *rset instructions, deleting step 018, and inserting a HLT at step 000. But this program, even as corrected, doesn't really make many significant or critical tests. It would be nice to have a set of comprehensive diagnostic routines that covers all known features of SR-52, SR-56, and PC-100 combinations. So send me your diagnostic routines, and I'll publish the best ones, judged on efficiency and on how well they cover and critically test all the important functions. Fault isolation should be hierarchical, i.e. high level routines identify problems occurring in one or more functions in a large category of functions; low level routines single out specific functions. Blocks of 112 steps (one side of a card) should contain routines of approximately the same hierarchical level.

PROGRAMMED CARD WRITE

Several members have reported success at getting cards written under program control. My runaway motor problem (V1N3p4) appears to be intermittent.

Dix Fulton (83) offers the following routine that automatically records 8 data words and retrieves them, with all required code contained on one side of a card: 000: *LBL A 8 STO 00 83 STO 01 CLR HLT *IND STO 01 X 1 INV SUM 01 = *dsz 012 INV *read *LBL B 83 STO 00 *IND RCL 00 HLT *dsz 037. Affix write tab to a card and insert in read/write slot. Initialize: press A. Enter 8 numbers, each followed by RUN. Card will be automatically recorded. To demonstrate results, turn calculator off then on, manually read programmed side of card, and press B to retrieve the first recorded number. Press RUN for each of the remaining 7 numbers.

TIPS

EFFICIENT HANDLING OF CONSTANTS: If a constant is used only once in a program, don't bother to create it and store it, only to have to recall it later. Create it at the needed point. If a constant is used more than once, balance its length against how often it is used. A 3-digit (3 step) constant should be created each time it is needed. A constant requiring more than 8 steps can be more efficiently pre-stored in program memory (provided there is program memory available).

PHYSICAL EFFECTS OF THE PRINTER ON THE SR-52: Mike Marquis (205) has found that a mag card left in the top slide when the SR-52 is connected to a powered-on PC-100 can be magnetically altered, especially if left for long periods of time (many hours). Also, printer heat can affect SR-52 performance, notably card-read operation: a problem encountered by at least one member (Bill Kennedy (166)).

MAG CARD CARE: Phil Sturmfels (49) suspects that sliding mag cards in and out of the viewing window can produce scratches serious enough to cause read or write problems. Does anyone else share this suspicion? (I don't use the viewing window). TI points out in the old SR-52 Owner's Manual (page 176) that scratches and/or oily deposits can cause mag card problems.

EFFICIENT COORDINATE CONVERSION PROCESSING

Many navigation and astronomical problems require spherical trigonometry solutions which in classical form amount to successions of sine, cosine, and tangent manipulations. Since these tend to consume considerable program memory and execution time, it is often productive to try to optimize the algorithms vis-a-vis the machine to be used. Some time ago John Ball (Smithsonian/Harvard Center for Astrophysics) noted that the number of keystrokes required to solve spherical trig problems on the early "scientific" calculators could be significantly reduced by clever use of built-in rectangular/polar conversion functions. I have revised some of his algorithms (intended for RPN machines) for SR-52 mechanization. The following program handles most of the required spherical trig with the *P/R function. I chose this program as an example, since it is also one in which several members have expressed interest. NASA catalogs and bulletins are mailed free of charge to anyone who writes to: Code 512 GSFC, Greenbelt, MD 20771 stating which satellites (up to 20 or so) he would like predictions for.

SR-52 Program: UT/AZ/ALT/Range from NASA Bulletins Ed

1. Key Node Time as HHmm.mm, press *A'
2. Key Node Longitude as Degrees West, press RUN (flashing pi indicates radian mode, in which case switch to degree mode, and continue)
3. Key satellite latitude as degrees, press A
4. Key time increment as mm.mm, press B, see event time as HH.mmss
5. Key Longitude correction as positive degrees, press C
6. Key satellite height as kilometers, press D
7. Get azimuth: press E, see azimuth in degrees
8. Get altitude: press RUN, see altitude in degrees
9. Get Slant Range: press RUN, see slant range in kilometers
   For new look angle to to step 4; for new node, go to step 2

After writing program into memory, but before recording on mag card, in run mode key observer longitude (degrees west), STO 93; key observer latitude (degrees), STO 94. Caution: Insertions or deletions after observer coordinates have been stored should be paired to maintain unrelocatable code at proper steps.

Program Listing

```
000:  *LBL E *fix 2 1 STO 00 RCL 01 *P/R STO 04 RCL 93 - RCL 02 = *P/R
024:   *EXC 04 INV *P/R - RCL 94 = *P/R *EXC 00 *EXC 04 INV *P/R *ifpos
044:   +/- + 360 = *LBL +/- HLT RCL 04 *EXC 00 INV *P/R STO 04 sin div
066:   (RCL 03 div 6370 - RCL 04 cos + 1 = INV tan STO 07 +/- + 90 -
093:   RCL 04 = HLT RCL 04 sin X 6370 div RCL 07 sin = *rtn
114:  *LBL A STO 01 *rtn
120:  *LBL B *fix 4 div 60 + RCL 05 = INV *D.MS *rtn
135:  *LBL C + RCL 06 = STO 02 0 *rtn
147:  *LBL D STO 03 *rtn
153:  *LBL *A' INV *D.MS div 100 = *D.MS STO 05 0 HLT STO 06 *pi sin
173:   *ifzro *pi 0 HLT *LBL *pi *pi INV sin HLT
```

- - - - - - - - - - -

CODES FOR MANTISSA AND EXPONENT SIGNS

C A Matz (282) points out an error (by implication) in my discussion of sign codes (V1N1p5). Position B (the units place of step 000) determines mantissa and exponent signs as follows: 0= both positive; 2=mantissa negative, exponent positive; 4=mantissa positive, exponent negative; and 6= both negative.

STREAMLINED DYNAMIC NIM

Rick Wenger (235) has done a nice job reorganizing my Dynamic NIM program (V1N3p3) so that it runs considerably faster, and leaves room for added features (including a possible printer version). I suspect that Rick is not alone in wondering why this program works, and asks if there is an algebraic expression for the nth term of the series: 0, 1, 1, 2, 3, 5, ... that the program generates. The series is a Fibonacci sequence (see V1N1p3), and there is indeed a closed-form expression for the nth term: FN=(((1+SQRT(5))/2)**N-((1-SQRT(5))/2)**N)/SQRT (5) which I have written as a quasi-FORTRAN statement for typing convenience. For those unfamiliar with FORTRAN, it reads: set FN equal to an expression that is formed as follows: divide the quantity one plus the square root of five by 2, raise this to the Nth power, subtract from this the quantity one minus the square root of five divided by two and raised to the Nth power and divide the result by the square root of five.

SR-52 Program: Dynamic NIM (revised) Ed/Wenger

1. Key starting number, press A
2. Key your move, press RUN
3. Repeat step 2 until there's a winner

Program Listing

```
000:  *LBL A STO 01 - 1 = STO 04 RCL 01 *ifzro 164 + RCL 04 div 5 INV *LOG
026:  = INV EE *fix 5 HLT STO 02 +/- *ifpos 049 + RCL 04 = *ifpos 059
049:  pseudo 84 RCL 04 *fix 0 GTO 031 RCL 02 INV SUM 01 RCL 01 *ifzro 174
073:  STO 08 +/- + RCL 02 X 2 = *ifpos 164 0 STO 05 1 STO 06 RCL 06 +
100:  *EXC 05 = STO 06 - RCL 08 = INV * ifpos 096 *ifzro 132 RCL 05
124: INV SUM 08 GTO 088RCL 02 X 2 - RCL 08 = *ifpos 157 1 INV SUM 01
[illegible]  X 2 GTO 007 RCL 08 GTO 147 55178.4 *fix 2 HLT pseudo 84 3507.1
[illegible]  *fix 1 HLT
```

- - - - - - - - - - - -

ROUTINES

FRACTURED DISPLAY UNDER PROGRAM CONTROL: Jared Weinberger (221) has discovered that putting pseudo 31 at program step 223 can get program execution to halt with no decimal point in the display. The sequence 216: *LBL A + STO 60 = pseudo 31 when executed with 1.11 in the display halts with all zeros in the display. Perhaps someone can figure out how to get other display format variations without a manually keyed = (V1N2p5).

EXPONENT EXTRACTOR

Jared has also devised a routine that displays the exponent (power of ten) as an integer of any non-zero number: *LBL A (EE div EE 00) *LOG INV EE *rtn.

CALCULATE MODE FLAG TEST: Mark Stevans (216) points out that the sequence: *ifflg n 888 (where n=0,1,2,3,4) keyed in RUN mode will result in a flashed display if the designated flag is set. The 888 could, ofcourse, be replaced by any number greater than 223. This method of flag testing would be especially helpful when a large progrm using flags is being debugged, since it doesn't require any program memory.

MEMBERSHIP ADDRESS CHANGES/CORRECTIONS

Make the following substitutions to your membership lists: 3: 890 West End Ave Apt 4C New York, NY 10025. 13: 13723 Sancho Ct Tampa FL 33612. 70: GUDZ, W 88 Baird St Rochester, NY 14621. 75: #312 Willowick Apts College Sta, TX 77840. 81: Box 974 Wrightwood, CA 92397. 189: Box 364 Gardner, Kansas 66030.