52-NOTES

Volume 1 Number 1 48/48 June 1976

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

SR-52 Users Club Scope

Club activity centers on contributions to and perusal of the monthly newsletter 52-NOTES. The club does not have legal status or formal structure, and sole responsibility for its operation is taken by the Editor. The club is neither sponsored nor officially sanctioned by Texas Instruments (TI), but is recognized by TI as the first independent group of SR-52 owners/users. 52-NOTES covers SR-52 care, use, and performance, as well as material on how best to program it for various applications. Although some complete programs not earlier submitted to TI for its soon-to-be operating Users Library will be included in the newsletter, clever routines will generally be given priority. Other pocket programmables may be considered as newsletter topics, depending on membership interest, As a start scope will include the SR-52 and SR-56. Material exclusively covering Hewlett-Packard pocket programmables will be referred to the HP-65 Users Club.

Newsletter content covers a broad range of technical disciplines and sophistication, and while each contributor is asked to make a reasonable effort to assure that his material is technically correct, neither the Editor nor contributors assume formal responsibility for any consequences resulting from any use of newsletter material. New discoveries will be credited to the first known discoverers, and those knowingly using another's discovery are asked to give due credit. Newsletter material is not copyrighted, and may be copied and distributed without limitation provided exerpted material is not taken out of context and is given due credit.

ROUTINE/PROGRAM LISTING FORMAT

Your comments are solicited regarding the format used in this first issue for routine and program listings. The idea, ofcourse, is to minimize space and maximize readability and checkability. Step numbers apply to the first instruction in a group. An asterisk signifies 2nd function. Left to right ordering facilitates "one pass" page organization.

UNANNOUNCED FEATURES

Most of you know by now that the SR-52 has capabilities beyond those announced by TI. Although TI is aware of some of these, it is likely to support only only those it announces. This means that future SR-52 production units won't necessarily have all of the current features. Since the club's purpose is to help members to get more out of their machines, topics will not be limited to announced features and their implementation. However, all contributors are asked to include in their shared routines and

---

The SR-52 Users Club is a non-profit loosely organized group of SR-52 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.

programs a statement that tells which (if any) unannounced features are used.

Following is a list of those unannounced features currently known to the Editor, a brief description of what they are, and the names of those who first brought them to his attention. More detailed explanations are begun elsewhere in this issue, and will continue in future issues.

1. Registers 60 through 69: These are used by the machine to hold pending operations, but may be used for data storage with the STO and RCL functions. (Mike Louder)

2. Registers 70 through 97: These are used by the machine to hold program instructions, but may also be used for data storage with the STO and RCL functions. (Mike Louder)

3. Registers 98 and 99: Extra storage registers not affected by either the CLR or *CMs functions. (Mike Louder)

4. Transferable Code: Machine executable code that is unaltered when transfered between data and program registers. (Mike Louder)

5. Split INV Functions: Viability of the INV function as a prefix, even though separated by labels and subroutine calls. (Mike Louder and Sandy Greenfarb)

6. Multiple INV Prefixes: An odd numbered succession of INV prefixes produces the INV function. An even number cancels it. (Ed)

7. Pseudos: Synthetic generation of instruction code not keyable. (Ed)

8. Register 60 Strange Effects: Fractured digits and complete "wipe-out" can be produced by certain register arithmetic operations in Reg 60. (Joel Rice and Sandy Greenfarb)

9. Pseudo 83 and 84 Arithmetic Strange Effects: Same as for Reg 60 arithmetic. (Ed)

10. Use of *D.MS Function for Integer/Fraction Truncation: The sequence: *fix, 0, *D.MS rounds up a real into an integer. (Ed)

11. Use of INV *D.MS to Reduce Reals for Truncation: The sequence: INV, *D.MS, INV, *D.MS reduces the fractional part of a real to less than one half. (Phil Sturmfels)

12. Zero Divide Error Conditions: The sequence: 0, divide, 0, =, CE generates a unique error condition causing the SUM and *PROD functions to execute as INV SUM and INV *PROD, respectively, and causing a change in transferable code rules. (Ed)

13. Program Execution of Card Read: One side of a pre-recorded magnetic card can be read under program control. (Ed)

14. 13-Place Arithmetic: While limited to 12 places for "display" arithmetic, the machine preserves 13 places in register arithmetic. (Ed)

GAMES

Many of you know the fun and pleasure that programming and playing computer games can bring. The SR-52's indirect addressing and comparatively large data storage capacity (for a pocket programmable) give it important advantages to be exploited, especially in handling generalized games. However, the lack of built-in integer/fraction truncation functions is often a disadvantage that needs to be overcome. Also relational tests on two values are slow, and destroy the compared values. My "Bagels One to Ten" program (found elsewhere in this issue) is an example of a generalized Bagels game. I'll publish a "Dynamic NIM" game to play against the machine, which is the toughest NIM I've yet come across. In the meantime, I invite you to send in your best games for possible newsletter publication.

ROUTINES

The more a routine (short progrm or function) is used, the more its execution and memory efficiency matter. The sharing of routines in this space will help all of us write better programs. I'll kick off with a few that have come to my attention, or that Ihave evolved, and look forward to seeing many member contributions.

Integer/Fraction Truncation:

I expect that so far more users have spent more time trying to perfect integer/fraction truncation routines than any other. And for good reason: programs that need these functions are apt to need all the memory space they can get, and are also apt to cycle through the int/frac routines often enough that execution inefficiencies are magnified noticeably. It appears that the routines that execute the fastest also take up the most progrm memory. Length and complexity of these routines depend upon the range of situations prevalent when they are invoked. The simplest case is that where only positive reals and a non-"scientific" display prevail during its execution. For this case the sequence: *LBL A, (STO - .5), *fix 0, EE, INV EE, *rtn is the fastest to execute, but takes 14 program steps. *LBL B, (STO - .5), *fix 0, *D.MS, *rtn saves two steps, but is a bit slower. *LBL C, INV *D.MS, INV *D.MS, *fix 0, *D.MS, *rtn saves two more steps, but is slower yet to execute. However, Routine C will handle negative and zero reals besides positive ones. If zero must be accounted for as a possible input argument, then Routines A and B would require a zero test, and would lose to C. All three routines would require INV EE to start with if the display format is "scientific" when they are invoked, and need to end with *fix 9 (or INV *fix) if display format must be returned to "initial mantissa". All things considered, Routine C looks like the best, unless speed is at a premium. But don't assume these are the best there are... by all means send in your better ones for publication!

Automatic Fill of Reg 60 - Reg 69 with a Single Number:

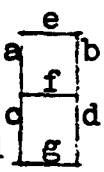
Key: *LBL E, STO, X, (, INV *if err, E, CE, *rtn in LRN mode. In run mode: Key n, E; result: there are ten "copies" of n in Reg 60 through Reg 69 plus the "original" in the display. Note: Each Reg 6X n is "attached" to a "X" operator.

Short Fibonacci Sequence:

In LRN mode starting at loc 000 key: HLT, +, *EXC 69, *rset. In run mode key: *rset, SST, CLR 1, I:RUN:I Note: I: A :I means keep repeating A until "done". The first press of RUN produces Fzero=0; each successive RUN produces the next Fibonacci Number. Ffifty= 12586269025. This routine could be made subroutine callable by writing it: *LBL E, +, *EXC 69, *rtn; in which CLR, 1 should precede the call.

DISPLAY FUN

Several SR-52 users have already discovered that the display can be made to present "fractured" digits. If digit composition is labeled as follows:



generated either by Reg 60 manipulation: able to create some can be randomly produced the following four patterns can be pseudo 83 or 84 arithmetic, or by ab, a, abef, f. So far, I've been symmetrical pattern groupings which by the following program:

SR-52 Program: Dynamic Display Fun*

User Instructions

Step	Procedure	Units	Press	Display
1	Enter Program	card	2nd read twice	
2	Key Random Number Seed (r)	10 GT r GT 0	A	0
3	Get symmetric fractured digit Display		=	??
4	Start next display		RUN	0
	Repeat steps 3 and 4 as often as desired			

*This program requires access to Reg 60 through 99.

Program Listing

```
000  *LBL A       002  STO 99      005  *LBL *1'     007 *D' B C E
011  E E E C      015  B E C E     019  E C B E      023 E C *C' HLT
027  *D' B C E    031  C E C E     035  C E C *C'    039  HLT *D'B C E
044  E E E C      048  B E C E     052  C E C *C'    056  HLT *D'B C
060  E C E E      064  C E C B     068  E E C *C'    072  HLT GTO *1'
075  *LBL B       077  RCL 99      080  X *pi =      083  *rt x  - D =
087  STO 99       090  E D         092  STO 98       095  - 7 =
098  *ifpos B     100  1 -         102  RCL 98       105  =
106  *ifpos B     108  RCL 98      111 *rtn          112  *LBL C
114  SUM 92       117  *rtn        118  *LBL D       120  *Ifzro
121  *ifzro       122 (STO - .5)   128  *fix 0       130  *D.MS
131  *LBL *ifzro  133  *rtn        134  *LBL *C'     136  RCL 92
139  +            140  SUM 60      143  + 0 *rtn     146  *LBL *D'
148  RCL 91       151  STO 92      154  *rtn         155  *LBL E
157  X 10 =       161  *rtn        162  *LBL *E'     164  *IND
165  *EXC 98      168  *E' A       170  0 0 0        173  C A A
```

REGISTER BEHAVIOR FROM A SOFTWARE VIEWPOINT (Part I)

Judging from many of your questions already, I sense that register behavior is of considerable concern to many SR-52 users. The discussion that follows is based on input from Mike Louder and Carlisle Phillips, as well as on my own experience.

Even without detailed knowledge of the hardware design, the user can discover new things about register behavior just by observing results under carefully contrived circumstances. Take the manual sequence: 1, INV lnx, -, STO, =. Where does the resulting -9 -12 come from? Now try *pi, -, STO, =. Why do you get zero this time, but not for e? (Note: n-STO= is equivalent to n-n=). Now try 1, INV lnx, STO 01, INV SUM 01, RCL 01. This time there is no residual. WHY? In order to find the answers, we need to make use of the display in both the LRN and calculate (run) modes. By performing STO and RCL of data and instruction codes through the potentially addressable 00-99 range, many of you have arrived at the following five classes of registers: General Data: 0-19, Read-Only Zeros:20-59, Pending Arithmetic 60-69, Program Memory: 70-97, and non-clearing Data: 98 and 99. You soon discover that Reg 70-97 in LRN mode display the most information, and it becomes evident that each register (in any category) may be thought of as a 64-bit binary word, divided into 16 4-bit bytes. As a convention that should be easy for all to follow, write the sequence starting at location 000 in LRN mode: B, tan, *rtn, *5', *ifzro, B, tan, *rtn. The resulting code should look like: 000 12, 001 34, 002 56, 003 78, 004 90, 005 12, 006 34, 007 56. Label the

16 instruction code digits as follows: 12=AB, 34=CD 56=EF, 78=GH 90=IJ, 12 (at loc 005)=KL, 34=MN, and the last 56=OP. Now switch to run mode and RCL 70. -5.634129079 41 is displayed, which bears some resemblance to the 8 op codes, but not much! Now STO 71, and examine steps 008-015 in LRN mode. They should represent the same "program" as in steps 000-007. Apparently all 16 digits got transfered, even though only 12 showed in the display in run mode. Here's the key: By the above convention, OP are the highest order displayed mantissa digits, followed by MN, KL, IJ, and GH. EF and C are the 11, 12 and 13th digits, respectively. D is the higher order exponent digit, A the lower order one, and B contains information determining mantissa and exponent signs as follows: 0=++, 2=+-, 4=-+, and 6=--. Now RCL 70 again, and see if the displayed number makes sense, keeping in mind that the 11, 12, and 13th places round up in the display to the 10th. Now key: -, STO, RCL 60, STO 70, and look at steps 000-007. Can you see what Reg 60 did to the displayed number in "attaching" it to the "-" operator? Part II in a later issue will continue register behavior discussion.

PROGRAMMING TIPS

Here are a few programming tips that may be new to many of you. If you have others you'd like to share, send them in for possible publication. This first group was brought to my attention by Carlisle Phillips:

22 Sequential Data Registers: Reg 98, 99, 00, 01, ...19 may be addressed indirectly consecutively as 98, 99, 100, 101, ...119, extending the usual 01-19 sequence by two.

Fast Execution By Absolute Addressing: Conditional or unconditional branches performed many times will execute in noticeably less time if they are in absolute form. The time saved is greatest for branch-to points near the end of a program. Best approach is to debug a program using relative (label) addressing. Then when it runs satisfactorily, remove the labels and replace their symbolic references by 3-digit absolute addresses, keeping in mind that a single symbolic reference to a label takes the same number of program steps as an absolute branch, i.e. 3. (see the program "Bagels One to Ten" found elsewhere in this issue). Incidently, those of you who plan to send in games for publication should optimize execution speed by maximizing absolute addressing.

LBL LBL Tricks: The ability of the LBL function to serve simultaneously as both a label identifier and as a label itself may be exploited to space-saving advantage. The sequence LBL LBL ----GTO LBL A --- defines both the loop LBL ----GTO LBL and a label called "A". The sequence: LBL LBL func ----*rtn defines two different subroutines: 1) called by SBR LBL that does func followed by ----, and 2) called by SBR func that does only ----.

Run-Time D-R Switch Sensing: For any program that is sensitive to the D-R switch position, and which has sufficient unused program memory, the following routines will alert the user to improper switch position: For desired degree mode: *pi, sin, *ifzro, *pi ---- *LBL *pi (84), 0, HLT will flash zero if switch is in radian mode. For desired radian mode: *pi, sin, INV *ifzro, *pi ---- *LBL *pi, (84), 0, HLT will flash zero if switch is in degree mode. Note: use of pseudo 84 creates an error condition in one step.

SR-52 Program: Bagels One To Ten* R C Vanderburgh

User Instructions

Step	Procedure	Units	Press	Display
1	Enter Program	card	2nd read twice	
2	Key number size (n*)	0 LT Int LT 11	D	
3	Key Random Number seed	Positive Real	RUN	
4	Generate Mystery Number		E	0.
5	Guess Number	Integer*	A	Score*

For new guess, repeat step 5.
For new mystery number, go to step 4.
For new number size, do step 2, then skip to step 4.

*Object of this game is to guess an n-digit positive integer by scanning with n-digit positive integers. Player chooses the size of n at step 2. Mystery integers contain no repeated digits, and thus the step 4 execution time will vary as a random number generator does or does not produce repeating digits, as well as upon integer size. Integer part of score gives the number of correct digits in correct places (fermi). Fractional part of score gives the number of correct digits in wrong places (picos).

Program Listing

```
000  *LBL *D'     002  RCL 14       005  X *pi =      008  *rt x -*E' =
012  STO 14       015  X 10 =       019  *E'          020 STO 12 *rtn
024  *LBL *E'     026  *ifzro       027  039          030  (STO - .5)
036  *fix 0       038  *D.MS *rtn   040  *LBL E       042  2 STO 11
046  *D' RCL 12   050  STO 01       053  1 STO 13     057  *D' RCL 12
061  - *IND RCL13 066   = *ifzro    068  053          071  1 SUM 13
075  RCL 13 -     079  RCL 11 =     083  INV *ifzro   085  058
088  RCL 12       091  *IND STO 11  095  1 SUM 11     099  RCL 15 -
103  RCL 11 =     107  *ifpos       108  053          111  CLR *rtn
113  *LBL D       115  STO 15 HLT   119  STO 14       122  CLR *rtn
124  *LBL A       126  STO 16 CLR   130  STO 17       133  RCL 15
136  STO 00       139  RCL 15       142  STO 11       145  RCL 16
148  div 10 =     152  - *E'        154  STO 16 =     158  X 10 =
162  STO 12       165  RCL 12 -     169  *IND RCL 11  173  = *ifzro
175  201          178  1 INV        180  SUM 11       183  RCL 11
186  INV *ifzro   188  165          191  *dsz         192  139
195  RCL 17       198  *fix 1 *rtn  201  RCL 00 -     205  RCL 11 =
209  *ifzro       210  214          213  .1           215  SUM 17
218  GTO 191
```

MISCELLANEOUS

TI_Notes: Extra copies of the SR-52 Owner's Manual and Basic Library are available from TI at $4.95 and $3.50, respectively. (For Carl French: TI is looking into your unfilled order problem). Four applications program packages are currently available at $29.95 each: Math, Statistics, Finance, and Electrical Engineering. Sets of 40 blank mag cards with carrying case and black tabs are available at $15.95 each. Call TI Customer Relations at toll free 800-527-4980 or write to P O Box 22283 Dallas, TX 75222. The idea of a TI sponsored SR-52 Users Library is currently in limbo. The PC-100 printer is being shipped to distributors in small quantities.

Membership List: Members who do not wich to have their names, addresses, etc. to appear on a distributed membership list should so indicate to the Editor by the end of July. Letters to the Editor that require individual reply should include a SASE.