52-NOTES

Volume 3 Number 10 48/39/38 October 1978

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

THE BUSINESS DECISIONS (9) CROM

Bob Anderson (506) got an early look at this new CROM, and reports it to be a "... super collection of programs." Bob is a business management/administration professional, not given to heaping undeserved praise on TI, so it appears that TI chose some useful business applications to address. At last report, Bob had not yet delved very deeply into program operation or structure, so quality has yet to be determined. Bob did report that a suggested (listed, but not in CROM) number sorting program (p77-78 in the manual) doesn't work. It is terribly slow, destroys the input data, but probably does what was intended: assign consecutive keys (V3N2p3) to input records, and output the keys in an ordering determined by ascending magnitude of corresponding records. It's unfortunate that the documentation doesn't specify what results to expect, and this exercise brings up an important question to ask yourself when writing/using sort routines: What is it you wish/expect to be output: keys or records, or both?

Another problem shows up in this CROM, which needs to be recognized when 2 or more CROMs are needed to solve a single problem: Since you have to turn the power off to change modules, all intermediate results are lost, unless the data are recorded. BD-3 requires ML-19 with the repeated entry of 3 inputs, as well as the BD-3 output. In this case, the user would probably find it easier to put BD-3 in user memory (it's only 60 steps) and not have to change modules, especially if more than one run is anticipated. For extended use, it would be worthwhile to add a post-processor to the BD-3 code, which would initialize the required registers for ML-19, and then call it.

It is too bad that TI still doesn't program the CROM routines to sense machine configuration, and vary I/O processing accordingly. For example, BD-5 would run considerably faster if the pause-displayed outputs were automatically by-passed with printer connection, and there's no reason why the user should have to set partitioning, since it's easy enough for the program to sense whether it is running in a 58 or 59 (... 7 Op17 Op19 ifflg7...) and it is given the max number of rows as input.

Members with appropriate expertise are invited to review any of the CROMS from programming quality and/or technical integrity standpoints. Incidently, recent full page ads (Scientific American and Electronics) identify Math/Utilities as #10 and Electrical Engineering as #11, both to be "Coming this fall...".

The SR-52 Users Club is a non-profit loosely organised group of TI PPC 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, Inc. Membership is open to any interested person: $6.00 includes six future issues of 52-NOTES; back issues start June 1976 @ $1.00 each.

BRIDGE DEAL (V3N2p5)

Bill Skillman (710) has risen to Lou's challenge, and with the program listed below cuts execution time by more than half and saves 170 steps. The key to Bill's approach is that the cards are always put in the correct order in each hand, as they are dealt, eliminating the need for sorting. Here's the algorithm: Form printcode for each of 52 cards symbolized: SA, SK, ...S2, HA, HK, ...C2, one at a time, always in the same order. Following the generation of a card, randomly select one of 4 13-place fields and put the card in the next available position. When a field has been filled, bypass it during subsequent random selections. Continue until all fields have been filled. Print the contents of the 4 fields: 13 lines, the ith line containing the ith element of each field. Fast as it is, there is a possible flaw in this approach, the seriousness of which is left to the user to assess: The more the random selection becomes uneven (one to three fields are filled well before the other(s)) the greater the probability that the remaining field(s) will be filled with consecutive low-rank clubs. Users should try an assortment of RN seeds and many successive deals to see how influential this anomaly is.

In mechanizing this algorithm, Bill does some tricky pointer maneuvering, which users should find helpful in other applications. See how printcode is synthesized via Reg 6 and 86, and try following the double indirect addressing accomplished with Reg 7 and 8. Initializing the last element in each of the 4 "hand" fields (steps 082-090) with a one and testing via the Exc* function (steps 149-153) is a short, fast way to sense when a field has just been filled; steps 159-182 effectively remove filled fields from the random selection process. My only contribution to Bill's program was to add the multiple-hand capability.

TI-59/PC Program: Bridge Deal Bill Skillman (710)/Ed

User Instructions: Record banks 1 and 2 with turn-on partition. Key seed (0-199017), press A; key number of hands to be printed (defaults to one), press R/S. Output is printed at the rate of about 3.1 minutes per hand. The ML CROM is required, and tested for.

Program Listing:
```
000:  LD 9 S00 22 S01 35 S02 48 S3 rtn LA S09 1 xXt Pgm1 SBR Write INV
029:  x=t X 9 Op17 CLR R/S S87 R85 Op4 R9 Op6 Adv Op0 R83 Op2 R84 Op3
057:  Op5 Adv D 4 S5 R79 Op1 R80 Op2 R81 Op3 R82 Op4 Op05 1 S22 S35 S48
089:  S61 xXt R75 E' R76 E' R77 E' R78 E' B' Adv Dsz87 060 rtn LE' s86
115:  13 S04 62 S6 Pgm15 SBR D.MS X R5 = S08 1 SUM*8 xXt R*8 S7 R86 +
144:  R*6 Op26 = Exc*7 x=t 159 Dsz4 123 rtn Op35 3 - Exc8 Int = CP x=t
169:  154 S7 R3 Op38 Exc*8 Dsz7 175 GTO 154 LB' fix2 D 13 S4 Op20 Op21
199:  Op22 Op23 R*0 Op1 R*1 Op2 R*2 Op3 R*3 Op4 Op5 Dsz4 195 INV fix rtn
```
Prestored Data:
```
62:  13 26 34 25 201 12 9 8 7 6 5 4 3 360000 230000 160000 150000
79:  3132353723 17133637 3632413723 43173637 143524 1622170000 36171716
```
- - - - - - - - - - - - -

PRINT BUFFER FILLING WITH AN EE OR Eng DISPLAY(58/59/PC)

Bob Petrie (632) notes that filling print buffers via Op1-4 "... with an EE or Eng format yields some interesting results," and suggests that the inherent Int function is not performed. What appears to happen is that as many significant figures (up to 8) as are in the effective integer part of the number represented in EE or Eng format are transferred right-justified to the print buffer mantissa field, along with the exponent as displayed. The sign digit assumes values of 0,2,4,or 6 depending on the length of the transfered mantissa string, and follows different rules for EE and Eng formats. For example, 1.3141516 D07 is the integer 13141516, and in the EE format when subject to an Op 1-4 becomes print-buffer formatted as -000001314516 D-07 (or as an octet of step-code: 76,60,51,41,31,01,00,00 if it could be transfered "intact" to a program register). Displaying the buffer contents causes a left shift of 5, producing -1.3141516 D-12 .

LAST DIGIT VIEWERS

Short routines to extract the hidden guard digits in a 52 have been around for some time now (V1N4p6 and V3N1p3), but comparable ones for the other TI PPCs haven't yet surfaced. The following discussion applies to one or more of the TI PPCs, but not the 52. If a fractured-display approach can't be found, then the mantissa must be maneuvered by some other means, and all which have surfaced so far require use of the EE function, limitations about which are discussed later. As in most program-design situations, it's a good idea to decide exactly what results are desired and how the program is to be invoked, before charging off writing code. In this case it is important to decide first whether the program is to be called by another program, run directly by the user, or even just keyed manually. Steps and execution time can be saved if the user examines a displayed number, and initiates mantissa partitioning according to some simple rules. Also important in this case is the type of stack arithmetic performed. Early 56 machines truncate the 13th digit of the first of the 2 operands used by a dyadic function; later 56s truncate the 13th digit of both operands (V2N2p6). So for either model, register arithmetic is required to preserve 13 places, and one can manually extract the ten least significant digits as described in V1N7p2. But program processing would need to perform the "... subtract from it the first 3 digits (including the indicated decapower)..." part, and this is more cumbersome to get the machine to automate than to do by eye.

Besides concerning themselves with decapower magnitude and sign, the TI PPCs must take into account display rounding by the EE function, using it in reducing the display to remove MSDs from the full mantissa. For example, on a 58/59 at turn-on, key π - EE = and see that the resulting -4.1 D-10 is the difference between π(13) and π(10) (π(n) means π rounded to n places). Now (with the display still in sci format) key π - EE = again, and see that this time π(13) - π(8) = -4.641 -D8 was performed. For both π(10) and π(8), rounding was "up" one place producing larger numbers than π(13), and the differences produced represent the so-called ten's complement of the desired LSDs of π. Now try all this with e (2.718...) instead of with π, and see that since the rounding is down (truncation) in both cases, resulting differences are equal to the desired LSDs. From these 2 examples it might appear that we have a simple method for extracting the LSDs: Perform - EE = on the display, and if the result is negative, convert the ten's complement result to its inverse (add 10^{n+1} to an n-place ten's complement number). It turns out that this will work for most reals, including those with large decapowers, but not all. The exceptions appear to be numbers whose 11 MSDs are all 9s, but whose decapowers are less than 99, and any number which when its display rounded configuration is subtracted from it produces underflow (i.e. ± π D-99,98,...92; ± e

D-99,98,...92; etc). But there doesn't seem to be a practical way to mechanize this in a program-callable subroutine, such that results are always in a predictable format. The following 58/59 35-step routine (which makes use of Dix Fulton's integer function found elsewhere in this issue) uses a different approach, and appears to work as a program-callable subroutine for all reals, returning the 9 LSDs of the mantissa as a decimal fraction, flashing zero for a zero input: LA abs S11 log + (INV Int - abs) Op10 = Int INV log EE INV Prd11 1 EE 3 Prd11 R11 INV Int INV EE rtn.

If you only want the whole machine number printed, and don't need to use it internally, the simplest (but not neatest) way is to list the octet of 8 steps containing the number as data. The following 58/59/PC routine is subroutine callable, but requires the user to interpret the listed steps (224-231) as data (V1N1p4,5), and prints 16 lines per real: LA xXt 10 Op17 9 S92 9.2 S90 xXt S91 9 Op17 GTO 223.

The non-printer approach can be mechanized for the 57 as: L1 abs S1 log Ct xGEt GTO 2 + INV Int x=t GTO 2 1 ± L2 = Int INV log EE INV Prd1 1 EE 3 Prd 1 R1 INV Int INV EE rtn, which returns with the 7 LSDs of the 11-digit full mantissa. Overflow reals cannot be processed, since the 57 will not execute during an error condition. A 56 version may be written: 00: abs S1 log CP xGEt 16 + INV Int x=t 16 1 ± = Int xXt 10 y^x xXt = EE INV Prd1 1 EE 3 Prd1 R1 INV Int INV EE rtn which returns with the 9 LSDs of the 13-digit full mantissa of non-over-flow reals.

This article was motivated by inputs from Panos Galidas (207), Bob Cruse (889) and John Mickelsen (990). All members are invited to comment, and to suggest better approaches to non-52 last-digit extraction.

SOME 58/59 FIRMWARE REVEALED

Steffen Seitz (1030) has found that an odd CROM calling sequence (V3N3p2) can reveal the code executed by the Σ+, $\overline{x}$, P/R, D.MS, their inverses, and the Op 11-15 functions. With a 58 or 59 at turn-on (ML installed), key: Op9 Pgm24 R/S R/S R/S 99 Op17 GTO 0 R/S D.MS, then LRN and SST through step 487, or key List if the PC is connected. Steffen has identified steps 000-044 with Op12, 047-057 as Op15 , 058-066 as Op14, 067-081 as $\overline{x}$, 084-106 as Op11, 107-148 as INV $\overline{x}$, 149-191 as Op13, 192-249 as Σ±, 250-283 as INV P/R, 284-302 as P/R, 303-340 as D.MS, and 341-379 as INV D.MS. Steps 380-487 make no sense as program instructions, and all but one octet are non-transferable (V1N2p2) if interpreted as data. Continued listing or SSTing past step 487 begins at step 039 with an abs instruction, then 040, 041,... 487 are as before.

Steffen suggests that the HIR 20 instructions at steps 045-046 and 081-082 serve as rtns. If so, HIR 20 behaves differently when executed in the firmware than in user memory, where it appears to do nothing but harden a soft display. Some variations on Steffen's sequence seem to work: ML Pgms 2, 3, 4, 8, 11, 12, 13, 18 and 25 may be substituted for Pgm 24, and 9 Op17 for 99 Op17; no repartitioning is required with a 58. No other CROM module appears to work, so it seems that the linkage mechanism to get to this firmware is peculiar to the Master Library. Although the code looks like it is in user memory, it probably isn't: It can't be executed or edited, and a BST causes an infinite hangup. Also, the 488 steps exceed the current partition, and the circular stepping (039, 040, ...487, 039, 040, ... etc) is inconsistent with user memory behavior.

Other members are invited to build upon Steffen's discovery, and to try to gain access to other sections of firmware and/or internal registers (IAR, subroutine return address stack, arithmetic stack pointer, etc).

REPEATING DECIMALS

As a follow-on to George Hartwig's all-digits generator (V3N6p5), Jared Weinberger (221) notes that 9800 ÷ 9801 = .99 98 97 ...0. These are rounded single cycles of repeating decimals, whose unrounded cycle patterns are 987654320, and 99 98 97 ... 03 02 00, respectively. These 2 examples suggest a more general relationship: Starting with the integers 80 and 81, and calling these K_o and K_o +1, form K_n by preceding K_o with n nines and following it with n zeros, n=1,2,... . Then the fraction K_n÷(K_n+1) is a repeating decimal having a cycle length of (n+1)(9X10^n+9X10^{n-1}+ ...9) digits, having the form: 9...9 9...8 ... 0...2 0...0, each p...q group being n+1 digits long. For example, for n=1 (Jared's example) K ÷(K +1)=.99 98 ... 02 00, 99 98 etc with a cycle length of 198 digits. For n=2, we have 998000÷998001=.999 998 ... 002 000, 999 998 etc, which has a cycle length of 2997 digits. Note that in each case 0...1 is skipped. As n increases, the cycle length grows exponentially, so it is impractical to check this relationship for very large n, even with a large fast computer. Anyone have **a** mathematical proof?

There are, ofcourse, other interesting (and many more not so interesting) repeating decimals (fractions whose numerators and denominators are integers, but for which when the indicated division is performed, the remainder cannot be expressed exactly as a decimal fraction). Members may find the following 58/59/PC infinite division program helpful in trying out various repeating decimal generators. To save it, record banks 1 and 4; to use it, key the dividend, press A; key the divisor, press R/S. The input problem is print-confirmed, and the quotient printed: the integer part first, followed by 20-digit lines of decimal fraction until stopped with R/S.

```
000:  LA S12 72 Op4 R12 Op6 ÷ R/S S11 64 Op4 R11 Op6 = INT S10 Prt 1
030:  S13 4 S15 1 EE 8 S16 INV EE 0 S14 5 S17 R10 X R11 = INV SUM12
057:  10 Prd12 R12 ÷ R11 = INT S10 R*10 X R16 = SUM14 .01 Prd16 Dsz17
085:  048 R14 Op*13 1 SUM13 Dsz15 035 Op5 GTO 029
Prestore:  00:  1 2 3 4 5 6 7 10 11 12.
```

Mechanizing the inverse: Given a repeating decimal, find the equivalent numerator and denominator integers looks like a considerably tougher problem to solve if other than a brute-force trial and error approach is taken. Members are invited to address this challenge.

TIPS AND MISCELLANY

<u>The Math Integer Function (V2N12p1, V3N6p6, V3N7p5)</u>: Dix Fulton (83) beats the others with: ... + (INV Int - abs) Op10 = Int ... which is both shortest and fastest, and appears to handle all cases.

<u>TI's Sourcebook For Programmable Calculators:</u> Kirk Gregg (748) brought this new TI publication to my attention. 58 or 59 owners who bought/buy their machines between 15 August 1978 and 31 October 1978 are eligible for a free copy. Others may buy the book at $12.95 each. This appears to be a 58/59 version of the Programming Workbook written for the 52 (V1N5p2), but with more applications examples. It may be useful to users of other machines. I will report further after I've seen this book.

Flag Status Routines (52,58,59): Several members have suggested multi-step routines to display the status of flags. Assuming that the purpose of these routines is to help debug other programs, I find it difficult to find an occasion for which the considerably shorter manually keyed: ifflg n 888 (V1N4p6), or ifflg n N (V3N3p6) wouldn't be better. Comments?

The Newest 59s: Dave Leising (890) reports that the newest 59s bear a code of the form: ATA nnnn (V3N4p1), and "... contain a different mag head which will not allow reading cards ..." made on older machines. TI acknowledges that some calculators are assembled in Abilene (Texas), which probably accounts for the first A in the ID code. Mag card compatibility may be unit-dependent: I find that some cards recorded on older LTA machines will read (sometimes) on an ATA machine made the 23rd week of 1978. Hardware visible in the battery well appears to be the same as for LTAs of late 1977 vintage, except that 4 resistors are larger in the newer machine, and the PC board, while having the same layout, is more like the first machines (June 1977) in the appearance of the surface material. No differences in functional behavior have come to my attention yet.

Membership Address Changes: 207: 150 Monroe St #302 Rockville, MD 20850; 398: 35 Oakfields, Broad St, Guildford, Surrey GU3 3AS ENGLAND; 616: 761 Via Flaminia Vecchia 00191 Roma ITALY; 689: bei Hartmann Schnewittchenweg 5 D7500 Karlsruhe GERMANY; 998: 9 Centre Park Dr Ottawa, Ont K1B 3C2 CANADA.

Machine Power Consumption (Blank Display V3N9p6): Sven notes that power (milliwatt) drain is a better way to compare machine states than current (milliamp) drain, since for any fixed state, the machine draws more current as the battery becomes more discharged (output voltage drops). For a displayed 1, Sven measures 515-540 mw for one pack, 530-560 mw for another, and an average 1.45% power consumption difference between blank and 1 displays.

Writing HIR Sequences (58,59): Tony Tschanz (1034) suggests sequences of the form: ...R82 BST BST i SST j ... where i is the instruction preceding the HIR and j is the HIR operand. j is most easily formed by finding a function key with the required op code. For example, to write: ... + H12 ..., key R82 BST BST + SST B. This method saves the Del step required by the usual method, but is no better if i is an op code which by itself would be a pseudo. For example, ... S31 H3 ... written S31 R82 BST BST Del SST 3 has no more keystrokes than SST R82 BST BST BST S31 SST 3.

Flashing Display Variations (V2N12p4): Tony finds that the 59 (58 too) does have an error condition display format analogous to the 52's soft flash: The sequence: CLR 1/x 0 . alternates the C symbol with the displayed zero. It appears that the soft flashing display prevails through number buildup in the display, but turns hard when the display itself is hard, unless either minus sign is present. For example, CLR 1/x 25 initiates a soft flash, which turns hard following the keying of STO. But CLR 1/x 25 ± maintains the soft flash following STO.

Mag Card Cues (52,59): In cases where successive mag cards are to be read, but where their selection is determined by the outcome of on-going processing, Bob Petrie (632) suggests having the last processing of each card provide a cue identifying the next card to be read. The first processing of the next card compares the cue with its identity, and alerts the user to mis-matches.