52-NOTES

Volume 3 Number 1 48/39 January 1978

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

FRACTURED DIGITS (58/59/PC-100A)

Jared Weinberger (221) is the first (and so far only) 58/59 user to bring to my attention a means of creating the SR-52's fractured digits: ° " ' - and blank (see V1N2p5). Jared's approach is an extension of Rusty Wright's procedure for creating a fractured display (V2N12p3), but requires printer connection. Jared found that the sequence: 1234 GTO 479 LRN GTO TRACE ÷ produced the ° and - symbols in the display of his TI-59 at turn-on. Further investigation reveals the following: 1) Only 2 fractured digits are produced, and they are always in the format: .nnnnfnbnnnf where n=numeral, f=fractured digit, and b=blank; 2) The f on the right is a - or a ° depending upon whether the display is fixed or "full" floating (decapower MSD≠0); 3) The f on the left is determined by the current partition; and 4) Somewhat different effects are caused depending upon what keys are substituted for the second GTO and the ÷ above, and the number and formatting of displayed digits. For alternate partitionings, the first GTO above should be followed by the address of the current last step, and for both the 58 and 59, the following hold: 479.XX produces a °, 399.XX an ', 319.XX a blank, and 239.XX a "; 559.49 with the 59 produces a -; other possible partitions produce redundant results or show a numeral in the left f position.

Jared's discovery confirms what many of us had suspected: that if a way could be found to produce them, the 58/59 fractured digits would be the same as the 52's. What seems strange is the dependency upon printer connection (you still can't print them), partitioning, and why they appear in only 2 display positions, and where they do. Other members are invited to explore further, and as Jared suggests, maybe we will eventually be able to control fractured digit generation and positioning to the extent now known to be possible with the SR-52. Incidently, Jared notes that trying to make the 58/59 believe it is connected to the printer by setting flag 9 won't work!

DESIGNING A PROGRAM/DATA COMPARATOR (59)

Software developers sometimes use code-comparing utility programs as debugging tools when the differences between 2 versions of a program or routine need to be detected and isolated, or when there is reason to believe that the same program or routine stored in 2 separate memory devices is not represented by the same binary configurations.

The same also applies to PPC program development, and the TI-59 with its card reader and large partitionable memory ought to be able to make fast, accurate code comparisons to help the user troubleshoot a variety of practical problems.

Data comparisons are easy enough to mechanize: pointers can be moved through 2 banks, accessing paired registers for x=t comparisons, with mismatches appropriately flagged. But program code can only be compared when it is treated as data, and here the code transfer rules complicate things. Not only do the SR-52 rules apply (V1N2p5), but also if an 8 or 9 is at position B, the overflow number 9.99... D99 is created upon transfer, regardless of what the other 7 step-codes are. So here is a challenge: Write a 59 or 59/PC-100A program that will compare the 240 steps of one cardside with those of another, detecting all possible mismatches, and minimizing false alarms and user intervention. It might be advantageous to make use of the program-controlled List capability (V2N10p3).

ROUTINES

Double Precision Multiplication (56,57,58,59): Taking advantage of guard digits operation, Mike Louder (53) has written a routine about half the length of Norman's (V2N12p1). I've taken the liberty of paring Mike's even further, resulting in having the user bisect each factor with a decimal point for the routine listed below. The key to Mike's approach is forming the 10 MSDs of the product by Int(F1 X F2) where F1 and F2 are the complete input factors. This assures that any carries from the MSD of the fractional products occur properly. Similarly, carries from lower fraction products are properly handled by summing only the fractional parts of each product. LA S1 X R/S S2 = Int R/S R1 INV Int X xXt R2 Int + R1 Int B + xXt LB X R2 INV Int = INV Int rtn. To run: Key multiplicand, press A; key multiplier, press R/S, see 10 MSDs of product; press R/S, see 10 LSDs of product preceded by a decimal point. Inputs must be in the form: ddddd.ddddd .

Efficient Polynomial Evaluator (58/59): Fred Fish (606) has a routine of the form: LA R*ab = X Rcd + Dsz "ab" A R0 = rtn which neatly mechanizes the nested multiplication polynomial evaluation method without using parentheses. Registers ab and cd should be at the end of available data memory, with Reg ab initialized with n (the order of the polynomial) and Reg cd containing the input value for the variable x. Registers 0, 1, ... are initialized with the coefficients for x^o, x^1, ...x^n, within the limit n LT ab.For new case,reinit Rab & cd.

Regulated Clock (58/59): Dix Fulton (83) has devised a routine that combines a timer with the means to automatically correct it with: 000: R0 + INV D.MS Pause xXt RST LA Fix9 ((D.MS - R1)/(xXt D.MS + R0 - R1)) Prd0 xXt CLR xXt R/S LE D.MS Fix4 S1 + R0 + RST. To run:
1. Enter approximate time correction factor (try .0004), key STO 00
2. Start clock with current time in display: hh.mmss, press E.
3. To regulate: Press R/S during a pause, after clock has been running awhile; key exact time of pressing the R/S, and press A; see delta correction factor displayed; go to step 2.

Fractured Digits Synthesizer (52): Dick Blayney (610) has a nice short routine (slightly revised as listed here) for creating fractured digits in all but positions 0, 10, and 11 (per V1N2p2 numbering), which substitutes arbitrary numerals where no fracturing is specified, producing somewhat the same results as the longer V2N3p4 one: 000: 1/x 1 3' 7' y^x lnx 1 E' LA X 1 EE ± 12 + 1 = S99 INV EE HLT INV log EE fix0 EE ± PROD99 INV fix R99 ÷ SUM60 R70 rtn. To run: 1. Key 10 digits to produce characters at positions 1-10 per the masking rules (V1N2p5), press A; 2. Key 2 digits for the position 12 and 13 characters, press RUN. 3. Press =, and see the synthesized fractured digits display. For new display, press CLR and go to step 1.

Last Digits Viewer (52): Dick has a variation on Jared's V1N4p6 routine, which suggests that there are better sequences than the ... STO + 60 ... that we've been using for display fracturing. Dick's 216: LE ÷ SUM60 = p31 is the same length as Jared's, but doesn't cause wipeout with negative numbers, and appears to properly show the 10th through 13th digits of any machine number. The X operator appears to work as well as ÷ in this routine, but substitution of PROD for SUM incurs the wipeout problem Further investigation of various sequences of the form: ...a b 60 = ... where a is a dyadic operator and b is a register operator may prove to be fruitful.

A Short Closed-Form Fibonacci Number Generator (56,57,58,59): Taking advantage of the fact that the second term in the closed-form formula (V1N4p6) is always small enough to be neglected if results are rounded to the nearest integer, Joel Pitcairn (514) suggests: xXt y^x xXt ÷ 5 $\sqrt{x}$ + .5 = Int, where the T register is initialized with $(1 + \sqrt{5})\div 2$, and the sequence executed with n in the display. Note that the constant stored in the T register is preserved indefinitely... an advantage applicable to other problems requiring repeated use of a constant.

Error State Reversal (52): Following Jared's path (V2N10p2), John Allen (104) points out that ...INV iferr LBL LBL iferr CE... will alternately cause and clear the error indication. This takes one less step than Jared's and saves another step in sensing the condition. Note that CE can't appear as a label for this to work.

TIPS AND MISCELLANY

Friendly Competition: Mort Schwartz (199) has responded to the challenge of Karl Hoppe's 70 digit square program (V2N5p5,6) with one written for the HP-67/97. I have tried a number of examples, and they all appear to work. However, Mort's program takes more than twice as long to run. Interested members may write Mort or me for copies.

Strange CROM Behavior (58/59): Roger Gentry (398) decided to try out sequences of the form Pgm mm R/S (suggested on page V-62 of the Owner's Manual, but acknowledged by TI as a mistake). Results appear to be program-dependent as well as machine-state-dependent. Roger also found strange results from related sequences of the form: Pgm mm SBR N, where N is undefined in Pgm mm. Members are invited to report results. While such efforts may not appear to lead to practical applications, there is the possibility that important aspects of CROM execution will surface.

Magnetic Card Extraction (59): Joel Pitcairn (514) suggests restarting the drive motor with another card when a just-read card is hard to remove. Stop the motor with R/S.

Custom CROMs (58/59): TI is currently accepting orders for custom designed CROMs. For $25,000 it will fabricate up to 1000 copies of up to 99 customer-written programs. Call Dick Cuthbert at TI's Lubbock facility: (806) 747-3737 X470, for details.

Printer "Hiccup" (58/59/PC-100A): Jared Weinberger (221) has found that trace execution of an Ind operation on a register outside the current partition causes the printer to stop momentarily and print a bit of nonsense in place of the out-of-bounds address.

Soft Flash (58/59): Rusty Wright (581) has found that attempted INV sin execution of a number less than -1 produces an error display similar to Mark Stevans' definition of a soft flash (V2N12p4): the faint C flashes along with the displayed number.

A University Association of Calculator Programmers: Bob Moore (488) has been organizing students and faculty at the University of Washington who, among other things, have participated in "...calculator math labs on campus and manufacturer-sponsored programming seminars in the Seattle area." As a fund-raiser, Bob's group is marketing the "...new uncompromising Zlotnian Calculator." Interested members may write Bob for details. Bob is one of the few educators to come to my attention who is examining some of the important pros and cons of calculators and PPCs in the classroom, and has already addressed some of the important considerations that all educators must sooner or later confront. Other members with links to the academic community may find it enlightening to discuss this topic with Bob, and/or share their views via 52-NOTES.

Successful Mag Card Recording Check (59): When at least one spare memory bank is available, Bill Adler (599) suggests using it to verify newly recorded code/data from another bank before losing it at machine turn-off, as there are instances where cards appear to record correctly, but are later found to read improperly.

Agriculture Programs: J E Dunn (791) reports the availability of a collection of SR-52 and TI-59 programs sponsored by the Iowa State University Cooperative Extension Service to aid farmers. Write J E for details.

Interrupt Processing (58/59): Although the new machines don't have the handy manual angle-mode switch for interrupt processing (V1N2p3,4 and V2N2p2), Joel Pitcairn (514) suggests a way to accomplish the same result (albeit requiring some extra manual keying) making use of a flag. A flag test is put in a convenient place inside a loop, and written to cause a branch to interrupt processing code when the flag is set. Then during loop execution, the user keys R/S stflg n R/S to get a "safe" look at intermediate results. The interrupt processing code should, ofcourse, have in it an INV stflg n sequence.

Correction: Reinhold's routine (V2N12p5) should have an = before the last xXt, which takes care of the pending arithmetic, and explains the purpose of the xXt. The number of steps is 30, as stated. Reinhold reports that he and a friend have been trying to devise a practical 59 program to calculate all real and complex roots of polynomials up to about degree 15, and are seeking help from the membership.

CROM HIRs (58/59): Use of the HIR instructions (V2N9) is showing up in some of the applications CROM code. T S Cox (9) reports an HIR 32 at step 240 of the Leisure Library Football program, and Roy Chardon (515) the use of HIR 8 in Pgm 13 of the Real Estate Library.

More on Unintended CROM Code Execution: Rusty Wright (581) found a code 90 (List) at step 063 of ML-11, and reports that the sequence: Pgm 11 SBR 063 causes total wipe-out (except for the CROM!) when the printer is connected.

Strange LRN Behavior (58/59): Jared Weinberger (221) found that certain manual sequences in run mode followed by LRN would create program code. Specifically: 1) STO, SUM, Exc, Prd, or RCL followed by Ind LRN BST show a code 11 (A); 2) xGEt, x=t, SBR, or GTO followed by Ind LRN BST produce code 22 (INV); and 3) CP Ind Lbl or CP Lbl Ind followed by LRN BST produce code 04 (4); code 02 (2) for the 58.

Printer Character Shifting Using the Fix Function (58/59): T S Cox (9) found that he could left-justify 1-3 characters in the Op 4 Op 6 sequence by first putting the display in a fix 6 format. His findings lead to the following generalization for all 4 print buffers: fix n causes an Op 1-4 on the display to be shifted left n places in the print buffer, provided the display is an integer LE 10-n. For example, with turn-on display, key 12 Op 1 Op 5, and see a 9 printed in the 5th position of the first sector. Now, with the 12 in the display, key fix 2 Op 1 Op 5, and see the 9 in the 4th position. Key fix 4...6...8, and see the 9 move successively to the left. Fix n for n odd will, ofcourse, change the code... in this example from the 12 to 0120, which prints 0- .

Writing Good Diagnostic Programs: Reinhold Patzer (689) experienced a special-case fault with his 59 that suggests another consideration in writing rigorous diagnostic routines (V1N4p4): Provide for both manual keying and program execution of functions. The case Reinhold reports may be rare, but it is subtle, and was probably difficult to isolate: On his 59, "...Reg 65 appears to behave well if used only by the program. If any key is pressed the second bit of the first mantissa digit is cleared"!. Reinhold didn't say how long it took him to isolate this fault... but suppose it had concerned the mantissa's LSD instead of the MSD... it might well have gone unnoticed indefinitely.

Membership Address Changes: 545: O'Brien Box 187 Somerset, NJ 08873; 658: 20 Confucius Plaza #7C New York, NY 10002; 680: RD 1 Box 286 Neward Valley, NY 13811.

More on Out-of-Bounds Pointer Behavior (58/59): Roger Gentry (398) has pursued the out-of-bounds pointer phenomenon a bit further (V2N12p2) and his results lead to the following observations: If the pointer is positioned within the first octet of steps beyond the current partition, repeated keying of LRN will eventually put the machine into learn mode. The initial position of the pointer within the octet produces different results: For end-of-partition (EOP) + 1, 2 LRNs show step EOP; for EOP + 2, 2 LRNs show step EOP + 1; for EOP + 3, 2 LRNs show EOP + 2; EOP + 4, 5, 6 repeat this pattern, except that 3 LRNs are required, and EOP + 7 and 8 take 4 LRNs. In each case, if a displayed step is out of bounds, its value is always 00, regardless of what may have been written there. An SST switches to RUN mode, but BST appears to operate normally from EOP + 2 to EOP + 1 to EOP. With EOP + 1 or 2 in the display, LRN SST effectively executes the 00 (or appears to). "Real" code at steps EOP + 1 or 2 is not affected by any of these maneuverings. All of this may have something to do with the code execution buffer (V2N10p5). If so, it would seem that repartitioning so as to just barely place out of bounds the register whose contents is currently in the code execution buffer does not completely cut off access to this buffer.

FRACTURED DIGITS REVISITED (58/59)

Fred Fish (606) came in second to Jared in the fractured digits race (by letter), but has a technique that doesn't require printer connection, and makes possible the generation of at least 3 fractured digits in the display. Like Jared's, Fred's approach builds on Rusty's, but the key element is use of the SST function. Following a display position notation of 0-11 analogous to the 0-13 one adopted for the SR-52 (V1N2p5), it is possible to fracture positions 1, 2, and 7 with one sequence, and position 5 with another. Partitioning determines the position 7 character in the same way as explained on V3N1p1; similar partitioning dependency, but different rules determine position 5; positions 1 and 2 can be made ", blank, or unfractured numeral by specific combinations of flags 0, 1, 5, and 6. Fred's sequence for fracturing position 7 is: Stflg SST, with a full display, starting in learn mode at the last step of the current partition. The same sequence for fracturing position 1 with a " includes the setting of flag 6; setting both flags 1 and 6 changes the " to a blank. With only flag 5 set, the " goes in position 2, and setting both flags 0 and 5 changes the " to a blank. Following the SST with = produces a soft 6. Setting only flags 5 and 6 produces the " at positions 1 and 2. Replacing Stflg SST with fix SST produces the same fracturing, but after the =, a soft 8 is produced; Dsz SST produces ms 0.0 ds 90 where ms is the mantissa sign and ds is the decapower sign of the initial display; and Ifflg SST produces ms 0.0 ds 80. GTO, Lbl, or SBR SST fracture a position 5 digit, depending on the following partitioning: 239.XX produces a blank, 319.XX a ', 399.XX a °, 479.XX a -, and (for the 59) 959.00 a " . The execution of the SST in Fred's approach appears to move the pointer "somewhat" out of bounds; it takes 2 successive LRNs to get into learn mode, but the step displayed is the "proper" last one (see V3N1p5).

I'll wind this up with a collective hats off to Rusty, Jared, and Fred for getting us started at 58/59 fractured digits, and look forward to more inputs from others on this lively topic.

<u>Correction:</u> The city for member #680 on the previous page should be <u>Newark Valley</u>.