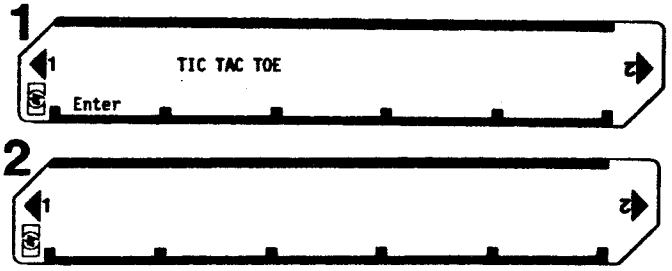
STEP	KEY ENTRY 1	KEY CODE
1	GTO 5	22 05
	SPC	35 84
	8	08
	RTN	35 22
5	1	01
	RTN	35 22
	6	06
	RTN	35 22
	3	03
10	RTN	35 22
	5	05
	RTN	35 22
	7	07
	RTN	35 22
15	4	04
	RTN	35 22
	9	09
	RTN	35 22
	2	02
20	RTN	35 22
	LBL A	31 25 11
	ABS	35 64
	INT	31 83
	9	09
25	x - y	35 52
	x > y	32 81
	GTO 5	22 05
	2	02
	x	71
30	3	03
	3	03
	-	51
	STI	35 33
	GSB i	31 22 24
35	STO A	33 11
	RCL E	34 15
	1	01
	+	61
	STO E	33 15
40	x - y	35 52
	STI	35 33
	RCL i	34 24
	x ≠ 0	31 61
	GTO 5	22 05
45	R↓	35 53
	STO i	33 24
	R↓	35 53
	STI	35 33
49	GTO i	22 24
50	LBL 1	31 25 01
	RCL A	34 11
	5	05
	x = y	32 51
	4	04
55	GTO 0	22 00
	LBL 2	31 25 02
	RCL A	34 11
	GSB B	31 22 12
	x ≥ 0	31 81
60	GTO 0	22 00
	1	01
	0	00
	STO B	33 12
	LBL c	31 25 13
65	RCL B	34 12
	1	01
	-	51
	STO B	33 12
	STI	35 33
70	x ≠ 0	31 51
	GTO d	22 31 14
	RCL i	34 24
	x ≠ 0	31 61
	GTO c	22 31 13
75	RCI	35 34
	CHS	42
	GSB B	31 22 12
	x = 0	31 51
	GTO c	22 31 13
80	RCL 1	34 01
	RCL 3	34 03
	x	71
	x ≠ 0	31 61
	x ≤ 0	31 71
85	GTO 6	22 06
	R↓	35 53
	RCL B	34 08
	x ≠ 0	31 61
	GTO 6	22 06
90	8	08
	GTO 0	22 00
	LBL 6	31 25 06
	R↓	35 53
	GTO 0	22 00
95	LBL d	32 25 14
	ISZ	31 34
	RCL i	34 24
	x ≠ 0	31 61
	GTO d	22 31 14
100	RCI	35 34

STEP	KEY ENTRY 2	KEY CODE
1	LBL 0	31 25 00
	STI	35 33
	CHS	42
	STO C	33 13
5	STO i	33 24
	2	02
	x	71
	9	09
	1	01
10	-	51
	STI	35 33
	GSB i	31 22 24
	CHS	42
	RTN	35 22
15	LBL 4	31 25 04
	LBL 3	31 25 03
	RCL C	34 13
	GSB B	31 22 12
	x = 0	31 51
20	GTO 2	22 02
	GSB 0	31 22 00
	1	01
	CHS	42
	STI	35 33
25	R↓	35 53
	-x-	31 84
	GTO i	22 24
	LBL 5	31 22 05
	CL REG	31 43
30	CLX	44
	R/S	84
	LBL B	31 25 12
	STO 0	33 00
	1	01
35	0	00
	STI	35 33
	LBL b	32 25 12
	RCI	35 34
	1	01
40	-	51
	x = 0	31 51
	RTN	35 22
	STI	35 33
	RCL 0	34 00
45	RCL i	34 24
	x = 0	31 51
	GTO b	22 31 12
	x = y	32 51
49	GTO b	22 31 12
50	x	71
	x ≤ 0	31 71
	GTO b	22 31 12
	RCL 0	34 00
	RCL 1	34 24
55	+	61
	ABS	35 64
	6	06
	x > y	32 81
	GTO b	22 31 12
60	9	09
	+	61
	x ≤ y	32 71
	GTO b	22 31 12
	-	51
65	CHS	42
	x - I	35 24
	RCL i	34 24
	x - y	35 52
	x - I	35 24
70	x - y	35 52
	x ≠ 0	31 61
	GTO b	22 31 12
	R↓	35 53
	RTN	35 22
75		
80		
85		
90		
95		
100		

1. TIC - TAC - TOE BY: Mike Richter 2366
2. BY:



67

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	To start or restart	0	A	0.
	Whenever 0 is displayed, restart has occurred. If it catches you by surprise, you probably made a bad entry.			
2	Enter X's move	n	A	-m O's play
	Board numbering: 1 2 3 / 4 5 6 / 7 8 9			
3	Repeat 2 until:			
	a) O's win: continuous flashing of winning digit.			
	b) All taken: restart, display=0.			
	c) You choose to end it (Step 1)			

STEP	KEY ENTRY	KEY CODE
201		
205		
210		
215		
220		
224		

Registers

0	(± = 0	1 X or 12 Value	3 = i	4
5		6 = unused yet		9

0	1	2	3	4
5	6	7	8	9

A X Entry	B Cntrl	C O's	D	E Cntrl	I

Labels

A Entry	B Book/Main	C	D	E
a	b	c Draw	d Draw	e
0 Output	1 First	2 Second	3 Third	4 Fourth
5 Last	6	7	8	9

Flag Set Status

Fix 0

0	1	2	3

the card from step 007 of Block 3 (which is your position immediately after pressing R/S when you have read in side 1 of the Access Card from a Block 0 address) you will be in normal program memory, actually at step 008 but with a program step location number of 001 because the program pointer value has been made 00E. (This program-step sequence on the merged card leaves you with 3 return addresses that place you at the same point once more if you execute a programmed h RTN.) At this location, attempting to key in program steps in W/PRGM mode advances you one step with each attempt but seems to produce no actual insertion until you reach step location number 008. If steps 008-014 already had contents, the ensuing sequence will be neither what has been keyed in nor simply the original steps pushed down by the keystroke entries. I'd be very interested in hearing from someone who has time to explore just what's happening and develop an explanation. It's the only further thing I've noticed that doesn't turn out to fit directly into the general framework already worked out, and that may make it important.

If you try this, don't be mislead by what happens when you back-step. Remember, this normalizes the pointer which sometimes has the apparent effect of making a step disappear but is really a matter of going from, say, a program pointer address o d2E which is addressing step 008 and showing as 001 to program pointer address (d+1)2E, normalized to 02(E+1) or 02F which addresses 02F and displays as step 007.

Lou Cargile, Jr. (753)

R/S

TIC TAC TOE

Ed. Note: The following two program descriptions were omitted from last months issue. The first is TIC TAC TOE by Mike Richter (2356). The program can be found in V4N6P51. The second TIC TAC TOE program by Ron Ryen (205) can be found in V4N6P57.

This program has more academic interest than practical. That interest lies in the use of the Magic Square. (Cf any Number Theory text). The key property of a magic square is that the sum of any row, column, or diagonal is the same as that of any other. Therefore, on a magic square, a Tic Tac Toe win is exactly three digits which sum to 15. Lines 3 through 20 of the program convert from a simpler notation to the magic square; all internal operations are in the converted form. (Note that reading the table bottom up converts the output, where top down handles input.)

The program cannot lose, and rewards cheaters by giving them a restart. Since older children know how to play, they will get bored and quit. Younger ones will get depressed at losing (or, at least, not winning). Note that the program plays the O's, and will respond to any initial move. Labels 1 through 5 are the responses to X's first through fifth plays respectively; Label 5 also provides the restart capability.

This was my first original program for the 67 (rather than a 25 adaptation), and is not esthetically satisfying. However, I'm not cleaning it up since it works. Note that code from 80-94 is a special-case (patch). I ran lots of cases and thought I had it checked out. The first game played by someone else (an 8-year old who didn't know the game) was so bad that she found the special case. And beat the 67. Ergo, a clumsy patch. I think it's unbeatable, but it should be checked out with a 7-year old to be sure. {The code is almost two years old now, and I don't really remember why I did all the clumsy things. So, please don't ask and embarass me for my poor documentation. Thanks.}

This program will entertain most anyone who likes the game of Tic-Tac-Toe for hours & hours due to its many variations in stratagy & intelegence. It is designed to prevent repetition by a special random number generator that makes it vertually impossible to repeat any game in its entirety.

This game is very easy, even for small children, to play since the only thing they have to do is pick their move &/or press "A" or "R/S". Other special features are available for those who desire to change the 97's I. Q. level ("C"), reprint the board ("D"), or alter the program & restart a partialy completed game by using the "B" key.

The 97's I. Q. level, which goes from 0 (dumb) to 10 (smart), decreases when it wins a game & increases when you win. Sometimes it even wins & don't know it. After all, no one wants to play a game with someone who wont play on their level. Whats your Tic-Tac-Toe I. Q. level????? Play 15-20 games & see.

A detailed analysis of this game can be had by sending a S.A.S.E. to R. Ryen (205)

R/S

67 MAGICIAN

The following is based on an old magicians trick. Make yourself a set of five cards as shown:

A			
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

B			
8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31

C			
4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31

D			
2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31

E			
1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31

Secretly choose any number from 1 thru 31; Don't tell the 67! If your number is on card A, then press A - otherwise don't. If your number is on card B, press B, otherwise don't. Examine each card in turn, until you have either pressed E (if your chosen number is on card E) or not (If your number is not on card E). When you are ready for the 67 magician to "read your mind", concentrate on your number (the importance of this step cannot be under-emphasized), and when you feel a certain psychic rapport with your machine, press R/S.

Unlike other magicians, the 67 will repeat a trick. Just choose another number, hit the appropriate keys (A thru E) and then R/S.

Example: you secretly choose the number 23. You then hit A,C,D,E, R/S. Try it on a friend...

```
PROGRAM  001 31 25 11  LBL A    012        02  2
         002           01  1    013     22 00  GTO 0
         003           06  6    014 31 25 15  LBL E
         004     22 00 GTO 0     015        01  1
         005 31 25 12  LBL B    016 31 25 00  LBL 0
         006           08  8    017 33 61 00  STO + 0
         007     22 00 GTO 0     018        44  CLX
         008 31 25 13  LBL C    019        84  R/S
         009           04  4    020     34 00  RCL 0
         010     22 00 GTO 0     021 33 51 00  STO - 0
         011 31 25 14  LBL D    022        84  R/S
```

James Garon (2042)

R/S

P SERIES PROGRAMS

HP-65 Users Club

1 SUPER ALPHA NOTONE — V.K. Heyman, M. Schwartz — P-37 I

New Game

A B SCORE Seed R/S Σ

A craps-type game with 2 players rolling as long as they dare, with cues tied to probabilities.

HP-65 Users Club

1 SUPER ALPHA NOTONE - Data — Heyman/ Schwartz — P-37 II

Σ

HP-65 Users Club

1 THE MAZE — Heyman — P-38

Start

Seed R/S

A B Σ

Two players try to escape from their own mazes guided by cues.

R/S

FURTHER READING

ASTRONOMY. The numbered circulars of The Maksutov Club has covered a wide range of topics related to Maksutov telescope originally, early 5 years, and Neo-Brachyt, Solano, and Compound Schmidts during the last 15 years. In circular 188, dated July 14, 1976, Editor Allan Mackintosh started publishing two or three PPC programs in each issue. Recently the 1800 pages of the Mak circulars have been edited into a more formal Quarterly "Telescope Making Techniques". In circular 189 Allan commented on the use of PPC's. "In the original manuscript, there was an appendix containing computer programs for various aspects of telescope making. These programs were mostly written in Fortran for large computers. It is proposed to abandon these programs altogether, and replace them with our rapidly growing collections of programs for handheld calculators...". TMT for winter V1N1 and spring V1N2 have been published. So far, no programs. The Mac circulars, however, still have 2-3 programs per issue (RR13 Box 312, Bloomington, IN 47401, Allan Mackintosh, Editor.) alternating between the HP-25 and SR-52. The latest news is that

est desert (and they can!). This requires a logically complicated structure - and besides their charming smallness this is their most interesting feature.
Michael Deckers (# 2382)

"Quot capita, tot sensus"
Horatius
"As many calculator wish-lists as members in the Club"

SOME SHORTCOMINGS OF THE ABOVE CONCEPTS

"I neither learned wisdom nor have the knowledge of the holy."
Prov 30.3

Despite the appealing description of some concepts avoiding explicit jumps these are not at all free from problems. Here are some.

- When nested conditions have the form
  f x=y ⌘1 ELSE f X=0 ⌘2 ELSE ⌘3 FI with only one FI for both conditions then the second condition may occur only after the ELSE of the first one. A more complicated semantics is required for the method
  f $^{-1}$x≠y f x=0 ⌘2 ELSE ⌘3 FI ELSE ⌘1 FI which allows for a second branching in both alternatives but a stack is needed for the level number of nest. Searching for the ELSE belonging to a certain condition is more complicated - nevertheless an analogous procedure is required for nested loops, and the method looks more logical. Note that at any rate the semantics of ELSE implies that the sequence ELSE (any keys) FI acts as NOP when not preceded by a condition. I t is understood that RTN also acts as FI.

- Similar problems arise for nested loops which are necessary for matrix algebra. I n the program piece

  f DSZ1 f DSZ2 ⌘1 OD ⌘2 OD the calculator must find out which OD belongs to which DSZ and must have a stack of addresses for the jumps back. So the number of nests is limited. If some ⌘ contains a further conditional instruction or a CASE instruction, the same procedure (find out which FI belongs to which condition or CASE) has to be applied; another address stack is unnecessary.

- A nesting of CASE instructions does not seem desirable,which makes the hardware only a little bit simpler. I propose not to include computed GTOs to absolute addresses (HP67's GTO(i) when RI is a computed negative number) and to use CASE instructions or indirect addressing instead. This will avoid complications in indirect addressing methods because there are many more programsteps than registers. Most GTO(i) with RI negative may be replaced by absolute jumps, since the jump address is already known at compile time.

- The suggested implicit jumps are relatively fast because they jump only a few steps forward or backward but of course they are not absolute. In order to make them faster I have not yet found anything less utopic than a machine which upon switching to FAST-mode runs through the program, automatically collects all addresses of conditions, CASEs, ELSEs, FIs and ODs and inserts them in the right places as GTOs. This, however, means sort of compiling run and will appear only on the next but many machines, if ever. Everybody has his REM-phase.

- The most unproblematic feature is the address list at the beginning (or somewhere else) in the program. These addresses are not affected by file-shifts. The feature seems to be quite natural: If, say, some 10 instructions of a program all use the same address then it is wasted memory space to write this address 10 times. I t is better to refer to it by a letter (A eg) out of 20 letters instead of by a number out of 100. What has to be done carefully then is the choice of merged codes. For example, one may assume that a given program does not use more than two ISZ or DSZ instructionsat a time (ie affecting each other). So we need two merged DSZ instructions, say DSZ P and DSZ U which can act on any register we want. Different lists for GTO and GSB, and for STO, RCL, ... seem appropiate.

Michael Deckers (2382) R/S

T I C - T A C - T O E

In response to an urgent inquiry, this note explains the details and principles of the TicTacToe program published in the September 65 Notes. The program architecture is elaborate, and uses the features of the 67 extensively; I doubt that it could be coded for the TI machines.

The mathematical interest stems from the concept of a Magic Square(see any text on Number Theory). The order m Magic Square has the property that each row, column and diagonal sums to a constant, m(m SQ +1)/2. For order 3, that is 15. This is the order-3 Magic Square:

8 1 6
3 5 7
4 9 2

(and its obvious permutations).

A corollary (not always developed in the texts) is that every set of m numbers summing to that value is a row, column, or diagonal of such a square; this can be proved by counting the possibilities and showing that there are 2m+2 of them, the same as the number of row, columns and diagonals. Therefore, if the TicTacToe board were numbered as an order-3 Magic Square, a win would be exactly any three cells belonging to one player which sum to 15.

The program converts the inputs to Magic Square space through the conversion table in lines 3 through 20. (Lines 1 and 2 are for initialization; in fact, 2 cannot be reached, and is a NOP.) It validates the entry, then converts it with a relative GSB (line 34). It then increments the pass count (i.e., the number of moves made) and confirms that the entry has not already been used. It then goes to the code for that pass by GTO 1 (indirect GTO) at line 55.

The first play by the machine will be the center square if available, the corner if the player has selected the center. Since the machine has no win on pass 2, it looks only for a potential block (GSB 8). On plays 3 and 4, it looks for a win first, then if none for a block. If neither a win nor a block is found, then it goes for an arbitrary play - trying first for one which requires the player to block. Output is converted.

This was my first program explicitly for the -67, and is inefficiently coded as a result. After I programmed it and checked it out, I took it to a neighbor's house to try it out on a 7-year-old. Her first try beat the machine! That is the only bug I found, and the special code in lines 80-94 is an out-and-out patch to handle the case. [Axiom: If you think your code is foolproof, try it on a fool or a child.]

The explanation provided for and omitted from the issue with the program was a lot shorter and less detailed than this one, and it will be interesting to see which gets published (if either).

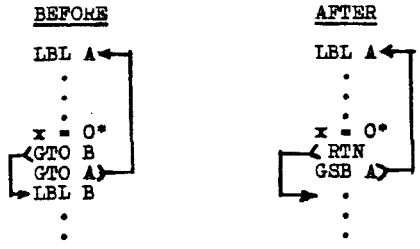
P.S.: There are other board numberings that work:

7 8 9
4 5 6 is one.
1 2 3

Mike Richter (2356) R/S

L O O P I N G W I T H O U T A N "E X I T L A B E L

Looping within a program has always been a problem because it uses too many labels. This is particularly bothersome if you have several looping routines. I have found a way to exit a loop without using a label, though the use of the GSB and RTN keys. This process also saves one step, as shown below:



Note: (*) Can be any conditional branching inst.
Caution: Do not use this within another sub-routine because of the limits (see P 206/67 Manual)

J. Bradley Flippin (712)

ED. NOTE: The use of the RTN and GSB is a clever solution to the problem. Another, and more direct solution is;

```
f LBL A
  .      to test place 1, - in place of the dots and have a
  .      number like 10 in the display
  .
X ≠ 0
GTO A
  .      Place h, π, R/S in place of the dots. Displays π
  .      when exiting the loop.
  .
```

This approach uses inverse logic to 'do if true' to continue the loop. R/S

C O L O R E D C A R D S

****** COLORED CARDS ***** Richard: Thanks for the excellant meeting 12/3. Dave Kemper, Tom Schroeder, John Kennedy and you put on a good show. A $7.50 check is enclosed for the Club on behalf of the following 10 members who have ordered and received cards: 639, 1269, 1552, 2243, 2345(#2), 2388, 2406, 2433, Wachel, Simmons. We now have 97 cards in colors: White, Black, Orange, Blue, Pink, Green, Yellow. Price: 6cards/$1.00 Also HP65 (grain is wrong for 67/97) Cards 10/$1.00. If some of the club chapters wish to get an order for 1000 cards we will pass the bulk buy savings on for $150/1000. Leonard Prince (34)