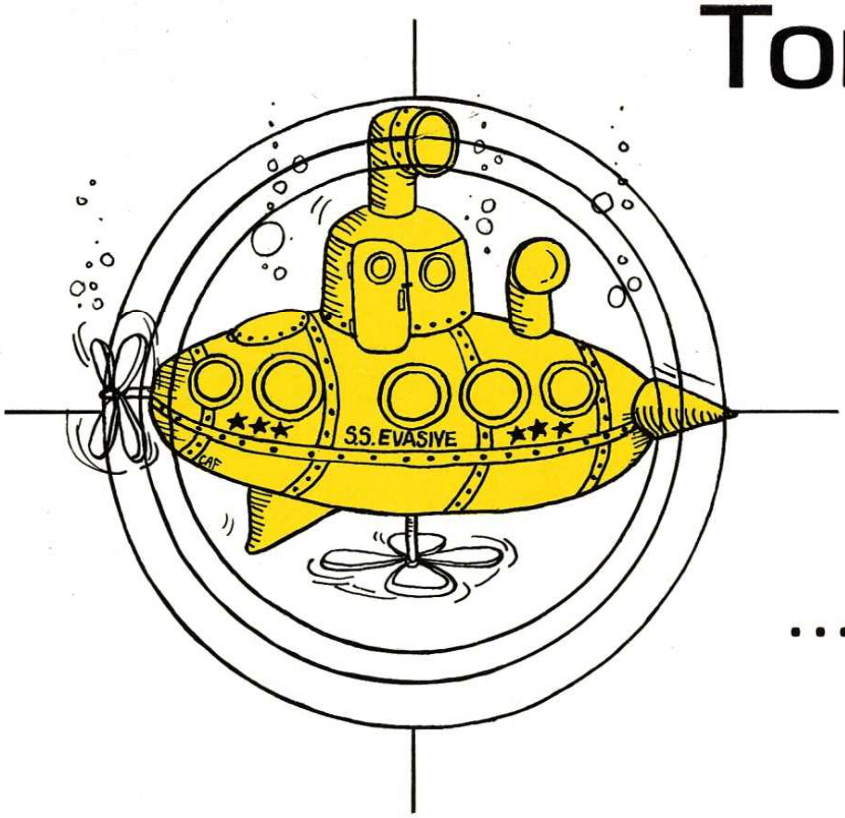

Torpedoes Away!

... submarine game for the SR-52

Charles G. Hanson
Box 612
Carmel IN 46032

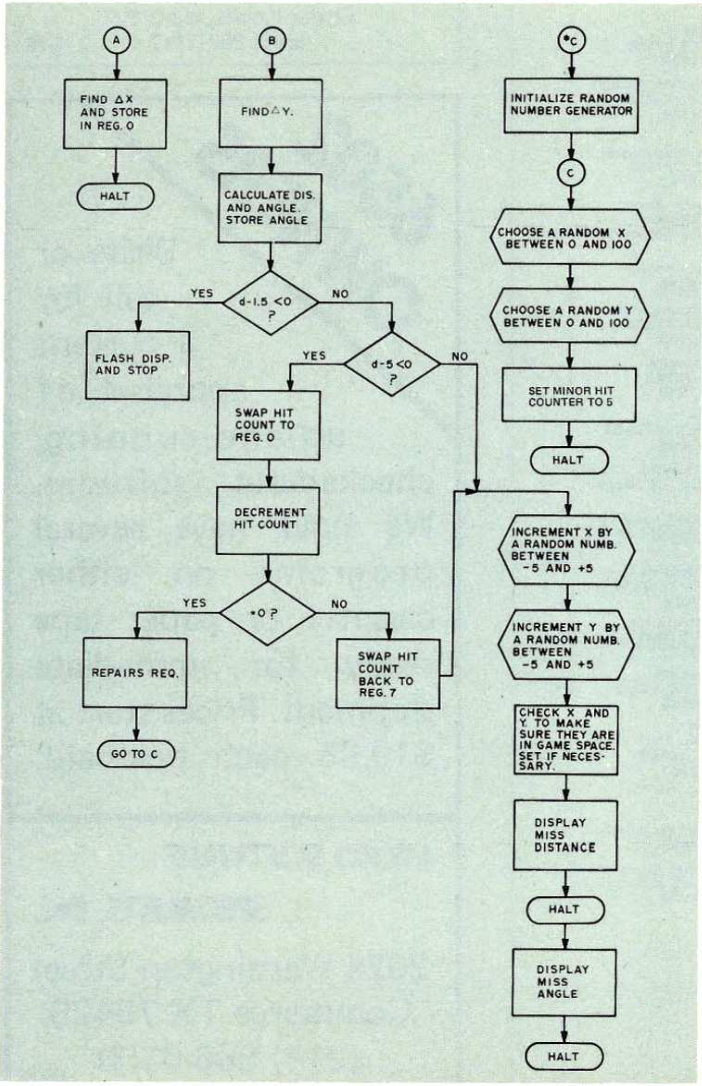

Fig. 1. Flowchart of game.

Peter Stark (*Kilobaud #2*, p. 70) has created an interesting game concept for use on the SR-52. Unfortunately, his implementation has resulted in a game which is virtually unwinnable, and hence, is likely to generate terminal frustration on the part of the player. There are three major difficulties which I have eliminated in the attached program listing.

First, Pete's version provides the player no information concerning the location of the submarine which is directly usable in improving his next shot. The information consists only of a distance by which you missed. It is necessary to plot this information on a piece of graph paper in order to establish a new target point, but note that the sub could be anywhere on the circle. Firing a second shot will allow you to draw a second circle which will allow you to zero in on two possible areas in which the sub might be hiding. A third shot should now get you close enough to a defi-

nite locale that you have a fair chance for a minor hit on the fourth shot.

But before you get your hopes too high, note that after every shot the sub moves to a new location somewhere within a grid of 100 points centered on its previous position. However, you only have that previous position within a probable grid of maybe 16 points. This means that your fourth shot is aimed at a grid of approximately 196 points, including uncertainty. Now consider that there are only 69 points within the minor hit circle and that only one of them is a kill, and you will understand why I suggest that this game could easily lead to a severe case of frustration.

However, there is still one more gremlin lurking within Pete's game. You are playing the game with the impression that the sub is lying within a grid defined as $0 \leqslant X \leqslant 100$, and $0 \leqslant Y \leqslant 100$. It is not, and in fact you will probably be outside the allowed range during your first game. This is because Pete allows his random number generator to start with a seed of zero, which means that the first number generated is 100, which is precisely on the border. In addition, there is no routine included which will prevent the position jogging part of the program from generating a subsequent position between 100 and 105. The highest number that I saw before discovering this aspect of the problem was 108.

Therefore, in the interest of preserving the sanity of all SR-52 owners who would like to play Submarine, I submit the attached program which does the following:

1. Restricts the sub to the 0 to 100 grid.
2. Places 9 points within the kill zone.
3. Provides direction as well as range for the player who wants it.
4. Assures that the sub begins the game well inside the grid.

A comparison of the attached flowchart with Pete's will show the important differences. I have also used the Texas Instrument conventions in indicating keystrokes.

The new rules of play are as follows:

1. Start the first game by pressing *C
2. Enter your target x Press A
3. Enter your target y Press B
4. Flashing display indicates a kill. Begin a new game by pressing CLR, C
5. Steady display shows the distance by which you missed (same as Pete's version).
6. Press RUN to display the angle to the target in polar notation. Your target point is considered the origin. Skipping this step will not affect program operation.
7. Enter new target locations as above.

Good hunting! ■

Location	Keystrokes	Comments
000	*LBL A	Enter x for depth charge
002	+/- + RCL 0 1 =	Calculate Δx
008	*EXC 0 0 HLT	Store in register 0, wait for y value
012	*LBL B	Enter y value for depth charge
014	+/- + RCL 0 2 =	Calculate Δy
020	INV *P/R	Calculate distance and angle
022	STO 0 4	Store angle in register 4
025	RCL 0 0 - 1.5 =	compare distance to 1.5 for kill
033	*if pos *4	If greater, skip to *4
035	0 *1/x HLT	If less, Flash display and stop
038	*LBL *4	
040	RCL 0 0 - 5 =	Compare distance to 5 for minor hit
046	*if pos *6	If greater, skip to *6
048	*EXC 0 7	If less, swap hit counter from register 7
051	*EXC 0 0	to register 0
054	INV *dsz C	Decrement hit counter, skip to C on zero
057	*EXC 0 0	Put distance back in register 0, swap hit
060	*EXC 0 7	counter back to register 7
063	*LBL *6	
065	SBR *1	Get a small random number
067	SUM 0 1	Move sub sideways
070	SBR *1	Get another small random number
072	SUM 0 2	Move sub vertically
075	RCL 0 1	
078	INV *if pos *7	If new x position is less than 0, go to *7
081	- 1 0 1 =	
086	*if pos *9	If new x is greater than 100, go to *9
088	RCL 0 2	
091	INV *if pos *3	If new y is less than 0, go to *3
094	- 1 0 1 =	
099	*if pos *5	If new y is greater than 100, go to *5
101	RCL 0 0 HLT	Display miss distance and stop
105	RCL 0 4 HLT	Display miss angle and stop
109	*LBL *C	
111	*CMs	
112	* π *log	Initialize seed of random number generator
114	STO 0 3	to be nonzero and store in register 3
117	*LBL C	Reset
119	SBR *8	Get a big random number
121	STO 0 1	Store as subs x position
124	SBR *8	Get another big random number
126	STO 0 2	Store as subs y position
129	5 STO 0 7 HLT	Set hit counter to 5 and stop
134	*LBL *1	Start of subroutine to get a small
136	5 +/-	random number
138	+ .1 X	
142	*LBL *8	Start of subroutine to get a big
144	1 0 0 X ((7 y^x 9	random number
153	X RCL 0 3	
157	X 5 +/- INV *log)	Shuffle numbers to make them
163	- (RCL - .5)	seem random
170	*fix 0 *D.MS INV *fix)	
176	STO 0 3 =	
180	*fix 0 *D.MS INV *fix	
185	*rtn	End of random number subroutines
186	*LBL *7	
188	0 STO 0 1	Set x position to 0
192	GTO 0 8 8	Go test y position
196	*LBL *9	
198	1 0 0 STO 0 1	Set x position to 100
204	*rtn	Go test y position
205	*LBL *3	
207	0 STO 0 2	Set y position to 0
211	GTO 1 0 1	Display miss distance
215	*LBL *5	
217	1 0 0 STO 0 2	Set y position to 100
223	*rtn	Display miss distance

Program.