THE OSI® GAZETTE

OSI C1P Newspaper Route Listing Program Part Two

Charles L. Stanford
Cinnaminson, NJ

Running The Program

With all the preambles and caveats out of the way, let's take a look at the program. Line 0 through whatever contain the customer list. I found it easiest to enter the first time directly as DATA Statements. We devised a form (see Listing 2), and John filled it out from his collection cards and entered the data into the program over a period of several days. Note that Line 0 contains the number of customers. We vary that to match the route, but keep the number of DATA lines at 75. The program doesn't care as long as there are more DATA's than READ's. Be very careful, however, that every line has an identical number of characters.

When printing to the screen or to an external device, Line 425 can be used to select the format of the list. If you want only the Sunday route, use a line such as IF D(X) <> 1 AND D(X) <> 3 AND D(X) <> 6 THEN GOTO 460. Note that AND and not OR is used for <> IFs.

Listing 2

```
 10 INPUT"NUMBER OF CUSTOMERS";N
 20 PRINTTAB(12);"NAME";
 30 PRINTTAB(25);"No.";TAB(35);"ST.";
 40 PRINTTAB(41);"D";TAB(45);"P"
 50 FORX=1TON
 60 PRINTRIGHT$(" "+STR$(X),2);
 70 PRINT"DATA";
 80 PRINT": : : : : : : : :,";
 90 PRINT": : : :,: : :,: :,: :"
 100 NEXT
OK

OK
RUN
NUMBER OF CUSTOMERS? 5
            NAME         No.        ST.   D   P
 1DATA: : : : : : : :,: : : :,: : :,: :,: :
 2DATA: : : : : : : :,: : : :,: : :,: :,: :
 3DATA: : : : : : : :,: : : :,: : :,: :,: :
 4DATA: : : : : : : :,: : : :,: : :,: :,: :
 5DATA: : : : : : : :,: : : :,: : :,: :,: :
OK
```

The BASIC Garbage Collector Bug

There is one more problem involved in getting this program to run successfully. That's the Garbage Collection Bug in BASIC ROM Number 3. Most OSI owners probably already know about this problem, but as an assist for those who are new to it, a short background. When the ROMs were programmed, there were two code errors within the Garbage Collector Routine starting at $B147 in ROM 3. The errors don't stop the routine from running, but they sure keep it from doing anything useful. The GC is necessary to undo the damage to memory caused by the accumulation of strings in upper RAM. When a string is concatenated, or even recreated, all versions are retained. If you say that A$ = A$ + A$, both versions stay in existence. You can see that an active routine such as the one at Line 900 of Listing 1 will soon use up all the RAM. But the GC is automatically called whenever RAM gets short, and all the redundant strings are discarded. On the OSI, not only doesn't this happen, the whole program hangs up, the screen pulses, and only a quick "BREAK" can save source code destruction.

There are several solutions. One is to buy one of the corrected PROMs available from several sources. But Rodger Olsen of Aardvark advised me that even a repaired OSI GC isn't perfect. He suggests the software fix in **The (Real) First Book of OSI**. I just ordered my copy, so I can't comment. What I have done is include a fix I devised, which puts a repaired OSI GC in Page 2 of RAM (the unused part starting at $0222). It is shown in Listing 3. You must remember that a BREAK will require that the Vector at locations 11 and 12 (Dec) be reset. The GC will not fit between $0222 and $02FF, so it

uses part of the original code as subroutines. There are several other fixes available, but one or the other must be used or you'll have to omit the names, and reset the string pointer at 129 and 130 (Dec) each time a loop of the Routine at Line 900 is run.

As is often the case, this program concept can be extended into many other areas. How about a checkbook balancer with the purpose of each check printed out along with its number and amount? A Christmas card checklist? The names can be any length consistent with your RAM; just change the "8" spaces wherever they appear. Routines to add values of numeric variables can be easily added.

Table 1

```
10 DIMN$(2):DATA"A","BBB"
20 X=0:Y1=3:Z=487
30 READN$(1):READN$(2)
```

0300 00	START SOURCE CODE	
01 17	ADDRESS OF NEXT LINE	
03 0A	LINE NUMBER	
04 00		
05 85	TOKEN - DIM	
06 4E	ASCII - N	
07 2A	- $	
08 28	- (	
09 32	- 2	
0A 29	-)	
0B 3A	- :	
0C 83	TOKEN - DATA	
0D 22	ASCII - "	
0E 41	- A	
0F 22	- "	
10 2C	- ,	
11 22	- "	
12 42	- B	
13 42	- B	
14 42	- B	
15 22	- "	
16 00	END OF LINE	
17 2A	ADDRESS OF NEXT LINE	
18 03		
19 14	LINE NUMBER	
1A 00		
1B 58	ASCII - X	
1C AB	TOKEN - =	
1D 30	ASCII - 0	
1E 3A	- :	
1F 59	- Y	
20 31	- 1	
21 AB	TOKEN - =	
22 33	ASCII - 3	
23 3A	:	
24 5A	Z	
25 AB	TOKEN - =	
26 34	ASCII - 4	
27 38	- 8	
28 37	- 7	
29 00	END OF LINE	
032A	THROUGH 033B HOLDS LINE #30	
033C 00	END OF LINE	
3D 00	CODE FOR	
3E 00	"LAST LINE"	
033E 58	ASCII - X	
3F 00	2ND CHAR	
40 00	VALUE OF X	
41 00	IN 32-BIT	
42 00	FLOATING POINT	
43 00	BINARY	
44 59	ASCII - Y	
45 31	- 1	
46 82	VALUE OF Y1	
47 40		
48 00		
49 00		
4A 5A	ASCII - Z	
4B 00	2ND CHAR	
4C 89	VALUE OF Z	
4D 73		
4E 80		
4F 00		
50 4E	ASCII - N	
51 80	TOKEN - $STRING	
52 13	7+(DIM+1)*4	
53 00	ALWAYS 00	
54 01	ALWAYS 01	
55 00	ALWAYS 00	
56 03	DIM + 1	
57 00	FOUR BYTES TO	
58 00	EACH ELEMENT OF	
59 00	AN ARRAY; START	
5A 00	WITH ELEMENT 0	
5B 01	LEN OF $STRING	
5C 0E	ADDRESS OF	
5D 03	$STRING	
5E 00	ALWAYS 00	
5F 03	LEN OF $STRING	
60 12	ADDRESS OF	
61 03	$STRING	
62 00	ALWAYS 00	
0363	UNUSED RAM	

Listing 1A

```
0 DATA75
1 DATA"STANFORD",2903, 1,1,1
2 DATA"JONES   ",   2,12,2,3
3 DATA"SMITH   ", 321, 2,1,6
4 REM-LINES 4 THRU 73 OMITTED
74 DATA"       ",   0, 0,0,0
75 DATA"       ",   0, 0,0,0
```

Sample Run-Entire List

```
JOHN'S INQUIRER ROUTE
------ -------- -----

1. STANFORD  2903 GEORGETOWN RD   DAILY & SUN  GARAGE
2. JONES        2 BARTON CT       DAILY ONLY  REAR DOOR
3. SMITH      321 BRANCH PIKE     DAILY & SUN  UNDER RUG
4.              0
5.              0
```

Sample Run-Sunday Customers Only

```
JOHN'S INQUIRER ROUTE
------ -------- -----

1. STANFORD  2903 GEORGETOWN RD   DAILY & SUN  GARAGE
3. SMITH      321 BRANCH PIKE     DAILY & SUN  UNDER RUG
```

Listing 1B

```
 200 READX:N=X:DIMN$(X),A(X),S(X),D(X)
 201 DIMP(X),S1$(15)
 202 GOSUB700
 205 FORX=1TO9:PRINT:NEXT
 210 PRINT"NEWSPAPER ROUTE":PRINT
 215 PRINT"1.  PRINT THE ROUTE TO PRINTE
R":PRINT
 220 PRINT"2.  PRINT THE ROUTE TO SCREEN
":PRINT
 225 PRINT"3.  ADD A CUSTOMER":PRINT
 230 PRINT"4.  DELETE A CUSTOMER":PRINT
 240 INPUT"ENTER YOUR PREFERENCE";X2
 245 ONX2GOSUB300,400,500,600
 250 GOTO205
 300 REM-PRINT TO PRINTER
 310 POKE517,1
 320 GOSUB420
 330 POKE517,0
 340 RETURN
 400 REM-PRINT ROUTE TO SCREEN
 405 PRINT:PRINT:PRINT:PRINT
 410 PRINTTAB(15);"JOHN'S INQUIRER ROUTE
 415 PRINTTAB(15);"------ -------- -----
 420 PRINT:PRINT:FORX=1TON
 425 REM-SELECT LIST ON THIS LINE
 430 PRINTRIGHT$("  "+STR$(X),2);".";
 435 PRINTTAB(4);N$(X);TAB(14);RIGHT$("
  "+STR$(A(X)),4);
 440 PRINTTAB(19);S1$(S(X));
 445 PRINTTAB(35);D1$(D(X));
 450 PRINTTAB(48);P1$(P(X))
 460 X=USR(X)
 470 NEXTX
 480 RETURN
 500 REM
 510 PRINT"          ADD A CUSTOMER":PRINT
 515 INPUT"ENTER CUSTOMER'S CODE NUMBER"
;X3:PRINT
 520 FORX=NTOX3+1STEP-1
 525 N$(X)=N$(X-1):A(X)=A(X-1):S(X)=S(X-
1):D(X)=D(X-1):P(X)=P(X-1)
 530 X=USR(X):NEXTX
 535 INPUT"ENTER CUSTOMER'S NAME";N$(X3)
:PRINT
 540 INPUT"ENTER HOUSE NUMBER";A(X3):PRI
NT
 545 INPUT"ENTER STREET NAME CODE NUMBER
";S(X3):PRINT
 550 INPUT"ENTER DELIVERY CODE NUMBER";D
(X3):PRINT
 555 INPUT"ENTER SPECIAL LOCATION CODE N
UMBER";P(X3):PRINT
 560 N$(X3)=LEFT$(N$(X3)+"          ",8)
 565 PRINTN$(X3);A(X3);S1$(S(X3));D1$(D(
X3));P1$(P(X3))
 575 X=USR(X)
 580 INPUT"ADD ANOTHER";I$
 585 IFLEFT$(I$,1)="Y"THEN500
 590 GOTO800
 600 REM-DELETE
 610 PRINT:PRINT:PRINT:PRINT
 620 INPUT"ENTER CUSTOMER'S CODE NUMBER"
;X4
 630 FORX=X4TON-1
 640 N$(X)=N$(X+1):A(X)=A(X+1):S(X)=S(X+
1):D(X)=D(X+1):P(X)=P(X+1)
 645 X=USR(X)
 650 NEXTX
 660 N$(N)="          "
 670 A(N)=0:S(N)=0:D(N)=0:P(N)=0
 680 INPUT"DELETE ANOTHER";I$
 685 IFLEFT$(I$,1)="Y"THEN600
 690 GOTO850
 700 REM-STARTUP SEQUENCE
 710 FORX=1TON
 720 READN$(X):READA(X):READS(X):READD(X
):READP(X)
 730 NEXTX
 750 S1$(1)="GEORGETOWN RD":S1$(2)="BRAN
CH PIKE":S1$(3)="ESSEX CT"
 755 S1$(4)="SOMERSET DR":S1$(5)="BERGEN
 DR":S1$(6)="SALEM DR"
 760 S1$(7)="BRIGHAM CT":S1$(8)="SALEM C
T":S1$(9)="COOPER CT"
 765 S1$(10)="HUNTERDON DR":S1$(11)="RIV
ERTON RD":S1$(12)="BARTON CT"
 768 S1$(13)="CARRIAGE WAY":S1$(14)="MID
DLESEX DR"
 775 D1$(1)="DAILY & SUN":D1$(2)="DAILY
ONLY":D1$(3)="SUN ONLY"
 780 D1$(4)="SAT ONLY":D1$(5)="M-F ONLY"
:D1$(6)="SAT-SUN ONLY"
 790 P1$(1)="GARAGE":P1$(2)="CARPORT":P1
$(3)="REAR DOOR"
 795 P1$(4)="IN DOOR":P1$(5)="MAIL BOX":
P1$(6)="UNDER RUG"
 799 RETURN
 800 REM-SAVE DATA - ADD CUSTOMERS
 805 B=782
 810 FORX=NTO1STEP-1
 815 B=782+(X-1)*28
 820 GOSUB900:X=USR(X)
 825 NEXTX
 830 CLEAR:GOTO200
 850 REM-SAVE DATA - DELETE CUSTOMERS
 855 B=782
 860 FORX=1TON
 865 GOSUB900
 870 X=USR(X)
 875 B=B+6:NEXTX
 880 CLEAR:GOTO200
 900 REM-SAVE DATA
 905 L$=LEFT$(N$(X)+"          ",8)
 910 Q=8:POKEB,34:B=B+1:GOSUB995:POKEB,3
4:B=B+1
 915 B=B+1:L$=RIGHT$("    "+STR$(A(X)),4
)
 920 Q=4:GOSUB995
 925 B=B+1:L$=RIGHT$("  "+STR$(S(X)),2)
 930 Q=2:GOSUB995
 935 B=B+1:L$=RIGHT$(" "+STR$(D(X)),1)
 940 Q=1:GOSUB995
 945 B=B+1:L$=RIGHT$(" "+STR$(P(X)),1)
 950 Q=1:GOSUB995
 955 RETURN
 995 FORR=1TOQ:POKEB,ASC(MID$(L$,R,1)):B
=B+1:NEXTR:RETURN
 999 END
OK
```

Listing 3

```
41000 REM-GARBAGE COLLECTION FIX
```

```
41ØØ POKE11,34:POKE12,2
41Ø2Ø FORX=ØTO139
41Ø3Ø Y=PEEK(45383+X):POKE546+X,Y
41Ø4Ø NEXTX
41Ø5Ø FORX=ØTO46
41Ø6Ø Y=PEEK(45596+X):POKE696+X,Y
41Ø7Ø NEXTX
41Ø8Ø POKE613,4:POKE699,2:POKE7ØØ,24
41Ø9Ø POKE629,177:POKE63Ø,2
411ØØ POKE686,76:POKE687,211:POKE688,177
4111Ø POKE689,166:POKE69Ø,157
4112Ø POKE691,2Ø8:POKE692,3
4113Ø POKE693,76:POKE694,19:POKE695,178
4114Ø POKE743,38:POKE744,2
OK
```


Voracious Butterfly

John Wright
Ottawa, Canada

The name came after seeing the program run. Voracious Butterfly was originally just a first exercise in using peeks, pokes and graphics, and as a visual check on how random is RND.

Display

A 24 x 24 section of the screen is filled with G187, the mini chequerboard, and G43, +, moves around one square at a time in a random direction. Each time it lands on a new square it 'eats' the G187 and replaces it with a G32 (Blank), G42 (*) or a character. When all the characters are displayed, the end routine pokes in another word and strips out the remaining G187s. A counter at the bottom of the screen increments by 100 every 100 steps.

Program

The program has 6 modules and a main line. The subroutines at 300, 500 and 1100 are called once each and could have been written in the main line. Conversely L70 to L150 could have been another module.

SUB 250 converts from X, Y coordinates to a POKE address.

SUB 300 to 480 reads in the word which is used in the end routine, puts 32 in all locations of the MA matrix to POKE blanks, replaces some of those 32s by 42s to sprinkle stars in the top and bottom thirds of the screen, and zeros counters.

SUB 500 to 560 reads character data into MA and puts a 1 in MB corresponding to each character in MA.

L70 to 155 picks the start point for the Butterfly and POKEs two zeros for the counter.

SUB 800 to 960 takes 100 steps. On each step the contents of MA are poked to the screen location, the contents of MB are added to TT (MB is 0 unless there is a display character in which case MB (X,Y) is 1. It is then reset to 0).

L840 checks conditions for a normal exit.

L860, 870 give the next step in the walk, with equal probability of staying still or moving to any of the eight adjacent squares.

L900, 910 stop the Butterfly from going off screen. Using SGN allows it to be done with one statement instead of separate IFs for 0 and 25. If the Butterfly goes off left, it reappears right as though there is a wrap-around. Similarly for top and bottom.

SUB 1000 to 1090 adjusts the base of the random number by incrementing the original input. This