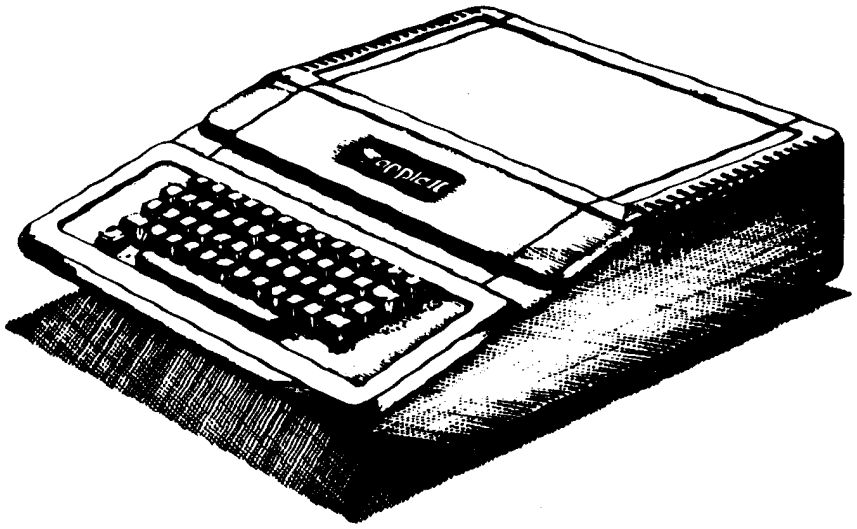
Apple 2 Computer Technical Information



Apple][Computer Family Information

AppleSoft BASIC Info:
Missing AppleSoft Random Number

Isaacson – Apple Orchard – Sep 1983

Document # 48

Ex Libris David T. Craig

"DTCA2DOC-048-00.PICT" 144 KB 2001-04-03 dpi: 300h x 300v pix: 2070h x 2772v

The Missing Applesoft Random Number

by Mark Isaacson

The Applesoft manual states that a random number may be generated for the Apple by using the Applesoft command PRINT RND(1). This command has many uses in games and in simulations. However, the Apple does not generate *true* random numbers with the RND command. Let's look at the problem Applesoft has with generating random numbers, and attempt to remedy the situation.

The Random Number

First, a random number is a number between certain limits, selected by chance. The ideal is that any number within the defined range has an equal probability of being selected. The classic illustration is 20 equal-sized pieces of paper put in a hat, and one number is picked. Random numbers are useful in programs where one of many things may happen, or a program in which chance has a role. A game might use a random number to determine whether a ball moves left or right, or if something occurs or does not, or which one of a number of conditions is set up.

The Apple contains a function that picks random numbers electronically. A random number is generated in Applesoft by the commands PRINT RND(1), or A=RND(1). In the first example, the Apple will print a randomly selected number that is in the range between zero and one. In the second example, the computer will assign to variable A a random value between zero and one. This is the only type of random number Applesoft can print. For example, typing PRINT RND(100) would not give an error in Applesoft, but the number generated would still be a number between zero and one.

To get the computer to generate larger random numbers, the number from the random command must be multiplied. For example, to get some number between zero and ninety, type PRINT RND(1) * 90. To get whole numbers, use the INT command, e.g. PRINT INT(RND(1) * 90). The INT command truncates numbers, so if A=3.3, INT(A) would produce a value of 3.

A program might use the RND function this way:

```
100 S = RND(1) * 10
110 IF S = 2 THEN PRINT "THE MONSTER HAS DIED"
```

We have defined a range of ten possible outcomes, and assigned a consequence to one of these possibilities. So there is a one-in-ten chance of the monster's demise at this time. Other possible outcomes can be covered by other IF statements, or program control can be effected using ON... GOTO.

The Problem With Applesoft Random Numbers

Although random numbers are very useful for Applesoft programs, they suffer from a significant "bug". Applesoft random numbers are not truly random at all. I noticed this first when I had been using a professional quiz program written in Applesoft. I noticed that when I turned off the Apple and later re-ran the program, the program asked the same questions with the same choices, in exactly the same order! The same problem occurred when I wrote a program to simulate a roulette wheel. The same supposedly random numbers were generated when I ran the program a few hours later. The RND bug was especially bothersome now, in a program where the random numbers played such a vital role.

The true nature of random numbers can be seen by turning off the computer, turning it back on, and typing "PRINT RND(1)". The computer will always respond with the same "random" number, .973136996. Also, since the Applesoft random number generator uses a seed for finding random numbers (one number is used in a complex formula to find the next), and the Apple's seed number is always the same initial value, the *pattern* of random numbers Applesoft produces will continue to be the same. In other words, the Apple will generate the *same set* of random numbers *every time* it is turned on.

The problem arises from the place where Applesoft gets its random numbers. On Page 141 of the Applesoft (Apple II) manual is a listing of important memory locations to Applesoft. Included in this list are the locations that Applesoft uses to get its random numbers. These locations are $C9 to $CD (in decimal, from 201 to 205). The problem with these locations is that they hold the exact same values every time the Apple is turned on.

But there is a solution to this problem. The Apple does have a real random number generator, which is located from $4E to

Sep 1983

$4F (in decimal, that's locations 78 and 79). These two locations are set randomly by the Apple whenever it is waiting for a key to be pressed. One way to make Applesoft have random numbers is to "feed" the real random numbers from the machine language generator to the not-so-random Applesoft random number generator. This would set RND(1) to a truly random value the first time it was printed, and it would also randomize the Applesoft random number seed. That would create a "set" of totally different random numbers. Here is a very short machine language routime to give Applesoft the real random numbers from the machine language generator.

```
AD 10 C0    LDA C010      ; Clear keyboard
20 0C FD    JSR FD0C      ; Randomize 4E and 4F, wait for
                            keypress
A6 4E       LDX $4E       ; Load X from randomized 4E
86 CA       STX $CA       ; Store X at byte CA of Applesoft RND
                            generator
A6 4F       LDX $4F       ; Load X from other randomized loca-
                            tion
86 CB       STX $CB       ; Store X in another byte of Apple-
                            soft RND generator
60          RTS           ; End subroutine
```

The program is totally relocatable. To type it in, for example, at $300 (decimal 768), enter machine language by typing CALL -151, then type:

```
300:AD 10 C0 20 0C FD A6 4E 86 CA A6 4F 86 CB 60
```

-then press RETURN.

Reenter BASIC. To save this subroutine to disk, type `BSAVE RND,A$300,L$10`

How The Program Works

The program first clears the keyboard strobe. This insures that if a key has just been pressed, it will not stop the READKEY subroutine from waiting for a keypress. The program then jumps to READKEY (JSR FD0C). This merely waits for a keypress. The important thing is, READKEY randomizes the machine language random number generator at $4E and $4F. The program then loads the newly-randomized $4E into the X register, and saves it in $CA, a part of the Applesoft random number storage area. It does the same thing for the memory location $4F, which was also randomized. The subroutine then ends (RTS).

Running The Program

To run the program, BLOAD it into memory, then CALL it. It will wait for a keypress. For example, type

```
BLOAD RND
CALL 768
PRINT RND(1)
```

Note that the call to 768 assumes that the program was saved at $300, which is the same place. The number returned will be a truly random number, which corrects Applesoft's random number deficiency. I recommend that you employ the above subroutine into all your Applesoft games and other programs that rely upon the RND statement. Who knows what new surprises await you the next time you run them?