November 1, 1979 Vol. I, No. 1

OSI-tems

SOME FINE POINTS OF OSI BASIC
(by D. Schwartz)

1) INPUT strings

A string containing a comma or a colon cannot usually be input in response to an INPUT statement. However, it can be if it is enclosed in quotation marks when INPUT.

Example:

```
10 INPUT A$
RUN
?SIMON,CARLY
?EXTRA IGNORED
OK
PRINT A$
SIMON
OK
```

BUT:

```
10 INPUT A$
RUN
?"SIMON,CARLY"
OK
PRINT A$
SIMON,CARLY
OK
```

2) DATA strings

Strings in data statements follow the same rules as strings entered in response to INPUT statements. That is, they need not be enclosed in quotation marks unless they contain commas or colons. By using both the quotated and non-quotated forms, any string may be read from data or input except:

1) A string that contains both an embedded quotation mark and a comma or colon.

2) A string whose first character is a quotation mark.

Strings with embedded quotatin marks can only be read from data or input (or assembled using CHR$)--they can't be stated explicitly in program text.

3) Limits of FOR-NEXT loops

The limits of a FOR-NEXT loop are evaluated and fixed the first time the loop is executed. If these limits contain expressions the values of which change during the course of the loop, this will not affect the execution of the loop.

Example:

```
10 X=5
20 FOR J=1 TO X
30 X=0
40 PRINT J;
50 NEXT J
RUN
1 2 3 4 5
OK
```

The upper limit is evaluated as 5 the first time the loop is executed. It doesn't matter that X then becomes equal to 0.

4) Formatting of printout of numbers

Each time a numerical value is printed, a space is printed both before the number and after. (The "before" space is omitted with negative numbers, because the minus sign takes its place.)

The printout from the above program should thus really look like this:

-1--2--3--4--5 (Where each "-" represents a space)

The "before" space is included in STR$(X), but the "after" space isn't. Thus:

LEN(STR$(25))=3

Numbers up to 999999 are printed in full while numbers one million or greater are printed in exponential notation (ex., 999999+1=1E+06). All numerical values are held to 24 bits internal accuracy (six digits are displayed). The largest integer that can be stored unambiguously is, therefore, 2**24, or 16,777,216 (displayed as 1.67772E+07).

5) RND function

The RND function is not described qccurately in the manual.

a) Positive arguments:

RND(X), when X>0, will always give the next random number in the sequence that is going. The particular value of X used has no effect at all on what number is generated.

Example: The first RND number after a cold start is always .500482, regardless of the argument X.

b) 0 argument:

RND(0) does not advance the sequence of numbers. RND(0) is always equal to the last number generated by RND(X) with positive X, whatever that was.

c) Negative arguments:

RND(-X), when X>0, is a RANDOMIZE command. It causes the computer to leave the sequence of Random numbers that has been going and start on a new one. The value of X in this case does matter--there is a differenet "sequence" of random numbers for each number X (integer or fractional). RND(-A) with the same A will always generate the same sequence of numbers generated by succeeding calls to RND(X), a X>0, but RND(-B), with A≠B, will generate a different sequence.

6) String comparisons

Although it's not mentioned in any OSI "documentation", the OSI BASIC has the capability of comparing strings by ">" and "<" as to their alphabetical order".

For example, "A"<"B", "HELLO"<"HI", and "HI">"HI THERE". This ability can be used to alphabetize a list of names, titles, etc., without having to take ASCII values of the strings and compare these numbers in a more or less complicated way.

7) Logicals

Logical expressions have numerical values in all computer systems. In OSI BASIC, the value of a TRUE statement is -1 (that's minus one, not one); the value of a FALSE statement is 0 (zero). This can be used to simplify some coding.

Example:

```
XA = X + 48 - 7 * (X>9)
```

sets XA equal to the ASCII value of the hexadecimal character corresponding to the number X(0-15).

8) Changing functions

Also not mentioned is the fact that the definition of a user-defined function may be changed during a program.

Example:

```
10 DEFFNA(X) = 3 * X + 10
.
.
.
220 IF X > 10000 THEN DEFFNA(X)= 3 * X/100 * 10
```

Etc. The last DEF executed always controls what the function will actually mean.

UNLOCKING OSI "END-USER" PROGRAM DISKS
(by M. Bolles)

Users of OSI floppy disc based software have no doubt discovered that OSI software has not necessarily been fully debugged before sale to you. (Examples: PD-1 contains reference to an "Effective Rate of Return" program, which, when run, is a "Depreciation" program. "Annuity I" will not produce an accumulated total when you try to determine an amount over a period of years; in addition, the graphic display is lousy.)

Fortunately, it is possible to "UNLOCK" these "unlockable" end-use discs in order to modify the programs, in much the same manner that you "UNLOCK" OS65D:

1) When the menu is booted in at start-up, type in "PASS" to the INPUT response "YOUR SELECTION". BEXEC* will respond "System open." and will reinstate an UNLOCKed state.

2) At this point, remove the system disc and insert the standard OS65D Ver.3.0 disc. Type in:

DISK!"LOAD DIR(cr)".

3) Reinsert the original system disc. Type in RUN (you have in memory the program listing for directory, remember). You will receive on CRT a directory listing of the system listing for the system disc.

EXAMPLE:

OS6503	0-12
BEXEC*	14-14
SAVING	16-20
ANN I	21-23
ANN II	24-26
EARRFV	27-28
BIO	29-31
CAL	32-34
S-DATA	36-36
DATA1	37-38

4) Now that we have both an UNLOCKed system and the file name for the program we wish to modify (or get to run correctly in some cases), we may proceed in exactly the same manner as we would for our own development disks, i.e., LOAD in the program, LIST, and make modifications, then PUT it back onto disk.