Recalculation Beeps
(PC Magazine Vol 3 No 17, Sept 4, 1984 User-to-User)

The computer can be left unattended for large 123 spreadsheets with numeric calculations which take several minutes and have it BEEP when finished. After starting the recalculation, press the Home key, then PgUp five or six times. The buffered keyboard saves everything subsequent to recalc until that is done and then executes Home and PgUp. Since the cursor is moved to the top of the worksheet by Home, the PgUp causes 123 to beep, thus notifying you that the recalculation is finished.

Buffer Advantage
(PC Magazine Vol 4 No 6 Mar 19, 1985 Spreadsheet Clinic)

The Home/PgUp procedure (see above) takes advantage of the keyboard buffer, but it has the disadvantage of losing the pointer position. You can take advantage of the keyboard buffer and retain your cell location by asking 123 to perform an illegal task. After pressing CALC, issue a "/" to get to the command menu and then press A several times. This illegal command also produces a beep. If you want to have a clear command line when you come back to the computer, also press the ESCape key. This trick is more useful when issued from a macro that would prompt for input after doing the CALC. You will hear the beep and not have to do anything to restart the macro or return to the current cell.

Using the Calc Key
(PC Magazine Vol 3 No 18 Sept 18, 1984 User-to-User)

The Calc (F9) key in 123 is a handy device and has more functions than one might expect. Calc allows full recalculation of a worksheet, but pressing F2 (Edit), then the Calc key will calculate only the single cell where the cell pointer is located. This can be used to check the results of a change on one particular cell without recalculating the entire worksheet. Simply move to the cell you want to check, press F2 and then F9 to see the result in the control panel. Be careful not to press Enter or any cursor-movement keys because the value will replace the formula in the cell. Press the Esc key to return to Ready mode.

Formulas whose values never change can be converted to numbers by using the Edit and Calc keys. As shown on page 93 of the Lotus manual, "to enter the value `ten times pi' (1) Type the formula 10*@PI, (2) Press F9 to convert this formula to the number 31.41592, (3) Press Enter to enter the number."

These helpful keys can also be used in macros where iterative calculations or loops occur. For example, use the following sequence in a macro to increment a counter value: {Edit}+1{Calc}~
This sequence adds 1 to the value of the cell, CALCs the new value, then loads the value into the cell. There is no recalculation of the whole worksheet, and you're left with a value rather than a formula in the cell.

Another useful application of the Calc key is to check on the value of a cell not currently displayed. For example, you might want to check on a value that is way down at the bottom of a very long worksheet. To check on the contents of cell A350, simply type +A350, then hit F9. The value of the cell will appear in the control panel.

Remember to hit the Esc key to clean up, or the value will be left at the position of the cell pointer.

Another trick is to grab a 123 row number and convert it to a label for use in a macro. Say you want to write a macro that will print out a column with an unknown number of rows. The keyboard macro \X in the example below contains a named range called "ROW#?". In a blank cell (C1), calculate the dept of column A, the @COUNT of the whole column (no gaps). The four keystrokes Edit, Calc, Home and apostrophe (') serve to turn the function into a left-aligned label. It is Copied to the cell two rows lower where it provides the crucial "bottom-line" number to the print section of the macro.

```
- - - - -
     A       B       C       D       E       F       G
1      1                          \X
2      2                          {goto}C1~
3      3                          +@COUNT(A1..A2048)
4      4                          {edit}{calc}{home}'~
5      5                          /c~ROW#?~
6      6                          /ppcarA1..A
7      7                 ======>  RANGE NAMED ROW#? HERE
8      8                          ~gq
9
```

Conversion Macro

(PC Magazine Vol 4 No 7 April 2, 1985 Spreadsheet Clinic)

To take the results of a calculation and move them to a different place on the worksheet, replace the data and perform the same calculations again means that you can't simply move the calculated fields to a different section; their values also will change when the worksheet recalculates. The obvious solution is to use the {edit}{calc} function key sequence for each cell after the cells are moved. For many cells this procedure takes a great deal of time, and writing the macro for multi-column processing is onerous. A simple solution for the long edit-calc process is first to name the entire range to be processed and then execute the macro below. This process can be a timesaver for big ranges, especially given the execution time for an {edit} {calc} loop.

Editor's Note: To convert a large range of formulas to their values, EDITRANGE is one of the most efficient ways to do it. This macro takes advantage of 123's option, using the /File Xtract command, to save a range to disk as Formulas or Values. The range is extracted in Value form, then immediately brought back into the worksheet with the /File Combine Copy Entire File command. Creating the range CURRCELL as the beginning of the macro serves as a place-marker while the work is being done. An alternative procedure is to use the /Data Query Extract command instead of /File Xtract, as this process also evaluates formulas as they are extracted to the Output range. You have the option of using the same techniques manually rather than within a macro.

```
/rncCURRCELL~{bs}~
{goto}EDITRANGE~
/fxvGLOOPXXX~EDITRANGE~r{esc}
/fcceGLOOPXXX~
{goto}CURRCELL~
```

"Stacking" Results of Repeated Calculations

(PC Magazine Vol 4 No 15 July 23, 1985 Spreadsheet Clinic)

In Vol 4 No 7 (above), a solution was offered to the problem of saving repeated calculations in different areas of the spreadsheet. Another approach uses 123's Columnwise Recalculation mode to store results in a "stack." You can use this method to save the results of succeeding "what-if" calculations.

Figure 1 is an example of how it works. In the range B7..B17 are the results of a calculation. The calculation can be anything but in this very simple example the variable in B5 was reproduced in B7, and increased by 1 in each succeeding cell from B8 to B17.

Figure 1 shows the results (columns F, G and H) of three successive iterations of this calculation using three different variables. Obviously, the three variables were 10, 20 and 30. You can save many different sets of results in the "iteration" columns, in this case, six. Each time the worksheet is recalculated, column H picks up the results from column B, and every other "iteration" column picks up the values in the column to its right.

Figure 2 shows the formulas that make the technique work. Columns C, D and E were omitted; their formulas are identical to those in columns F and G: they point to the values in the column to the right. The @IF logical operator is nothing more than a method to set the stack values back to zero for another round of calculations. Thus, if the zero-flag value in cell C3 (range name ZERO) is set to anything other than 0, a recalculation of the worksheet will reset all values in the iteration columns to 0. Note that this method will work only in manual Columnwise Recalculation mode. In this mode, calculations are made one column at a time, beginning with column A and moving left to right. In the default recalc mode (Natural), all the values in the iteration columns would immediately take on the values in column H.

Use this technique, for example, to store the results for the next 30 years of different possible refinancings of a home mortgage, or for a scientific application such as the results of amino acid composition data from different proteins.

Editor's Note: A flag was added to this model to indicate that the stack is full. For a large worksheet, you may not notice that it was full; succeeding calculations would push the results of the first calculations right out of the stack, and their values would be lost. In cell B21, the added formula is: @IF(D5>0,6,@FALSE) and the cell was given the +,- format. So long as the iteration number for column D remains zero, only a period appears in cell B2. As soon as D5 gets an iteration value, six plus-signs appear in B2. Since the worksheet is doing left-to-right, columnwise recalculations, the flag will not fly until the value in D5 is 2 and column C also contains results. If the flag were to the right of the iteration columns, it could point to C5, instead of D5, and give the same results.

- - - - -

Figure 1: The stack in which calculation results are stored.

```
     A      B        C       D       E       F       G       H
 1
 2                 Zero-Flag
 3                    0        I T E R A T I O N
 4      Variable
 5         30         0       0       0       1       2       3
 6                 ---------------------------------------------
 7         30         0       0       0      10      20      30
 8         31         0       0       0      11      21      31
 9    R    32         0       0       0      12      22      32
10    E    33         0       0       0      13      23      33
```

```
11    S    34         0       0       0      14      24      34
12    U    35         0       0       0      15      25      35
13    L    36         0       0       0      16      26      36
14    T    37         0       0       0      17      27      37
15    S    38         0       0       0      18      28      38
16         39         0       0       0      19      29      39
17         40         0       0       0      20      30      40
```

- - - - -
Figure 2: The formulas that make the stack work.

```
     A     B            F                   G                   H
 1
 2
 3
 4      Variable
 5         10   @IF($ZERO,0,+G5)   @IF($ZERO,0,+H5)   @IF($ZERO,0,+H5+1)
 6              --------------------------------------------------------
 7      1*B5    @IF($ZERO,0,+G7)   @IF($ZERO,0,+H7)   @IF($ZERO,0,+B7)
 8      +B7+1   @IF($ZERO,0,+G8)   @IF($ZERO,0,+H8)   @IF($ZERO,0,+B8)
 9   R  +B8+1   @IF($ZERO,0,+G9)   @IF($ZERO,0,+H9)   @IF($ZERO,0,+B9)
10   E  +B9+1   @IF($ZERO,0,+G10)  @IF($ZERO,0,+H10)  @IF($ZERO,0,+B10)
11   S  +B10+1  @IF($ZERO,0,+G11)  @IF($ZERO,0,+H11)  @IF($ZERO,0,+B11)
12   U  +B11+1  @IF($ZERO,0,+G12)  @IF($ZERO,0,+H12)  @IF($ZERO,0,+B12)
13   L  +B12+1  @IF($ZERO,0,+G13)  @IF($ZERO,0,+H13)  @IF($ZERO,0,+B13)
14   T  +B13+1  @IF($ZERO,0,+G14)  @IF($ZERO,0,+H14)  @IF($ZERO,0,+B14)
15   S  +B14+1  @IF($ZERO,0,+G15)  @IF($ZERO,0,+H15)  @IF($ZERO,0,+B15)
16      +B15+1  @IF($ZERO,0,+G16)  @IF($ZERO,0,+H16)  @IF($ZERO,0,+B16)
17      +B16+1  @IF($ZERO,0,+G17)  @IF($ZERO,0,+H17)  @IF($ZERO,0,+B17)
```

```
DIR=A:\
A>
```

Custom Rounding

(PC Magazine Vol 4 No 24 Nov 26, 1985 Spreadsheet Clinic)

123's @ROUND function is fine for decimal rounding, but what if you want to round a value to the nearest eighth, sixteenth, or even four hundredth? In row 4 of the figure below is a general formula that will round off a number by the rounding factor or your choice.

The table in rows 6 through 15 demonstrates this formula. Column D contains the original numbers, and column E contains the results of the rounding. The rounding factor in this particular case is 0.0625 (one sixteenth) in cell G7, which has the range name FACTOR.

A continuous-loop macro that makes it easy to enter numbers for rounding is included in rows 17 through 19. Call the macro with the Alt-R key combination. After you enter a number, the macro will format the cell to four decimal places, move to the next row, and then start over again. When you have finished entering numbers, get out of the loop by hitting Ctrl-Break.

Later, if you want to round the same numbers by a different factor just change the value in cell G7, and you will get an instant recalculation. If you want your results to be values instead of formulas, just add {calc} before the tilde at the end of the first line of the macro.

```
     C        D             E           F       G
 3         GENERAL ROUNDING FORMULA
 4        @INT(Number/Factor+.5)*Factor
 5
 6        ORIGINAL       ROUNDED TO            FACTOR
 7         NUMBER       ONE SIXTEENTH          0.0625
 8        ---------------------------
 9          0.8888          0.8750
10             0.8          0.8125
11          0.0777          0.0625
12        234.4567        234.4375
13          0.0313          0.0625
14          0.0312          0.0000
15              14         14.0000
16
17   \R   @int({?}/FACTOR+.5)*FACTOR~     Enter number with {?}.
18        /rff4~~                         Format cell to 4 decimals.
19        {down}/xg\r~                    Move down, loop to start.
```

Recalculating Values in Individual Cells

(PC Magazine Vol 4 No 26 Dec 10, 1985 Spreadsheet Clinic)

When you work with large spreadsheets, it sometimes takes 123 a long time to recalculate. You can set the program to do manual recalculation, so that it updates values only on command. However, you may sometimes need to recalculate only a single cell or a small range of cells, in which case it's a bother to have to wait for an update of the entire spreadsheet.

The solution is to copy a cell, or range of cells, onto itself. If you want to update the value of a single cell, move the cursor to it and hit /C followed by two Returns. For a range of cells, define both the source and target ranges as you would normally, but just be sure they are in the same range. You'll get a quick recalc of only the cells you need.

Editor's Note: This is a good solution to an annoying problem. 123 doesn't recognize it as a solution, however. Even if you copy an entire spreadsheet onto itself and get a complete recalculation, you will still have a CALC message at the bottom of your screen, meaning that 123 thinks you still need to recalculate.

###
###
###########