Super Power Macros, Techniques and Secrets for Lotus 1-2-3

Shareware Version 3.4

Introduction:

Part One:

PgDn ↓

Part Two:

Foreword

I discovered Lotus 1-2-3 during my two year sabbatical at Oregon State University, where I finished to develop the expert system for mechanical springs design, The ULTIMATE SPRING DESIGNER. After using the tutorial for a few hours I was hooked, and in two weeks, I finished my first version of The ORGANIZER for LOTUS 1-2-3. It was a menu driven file manager designed to take control over hundreds of worksheets and other DOS applications without leaving Lotus 1-2-3. In no time, I found myself developing macro based applications in 1-2-3, using its macro language before even using it the usual way. Lotus 1-2-3 became a hobby for me. Every day I found more exiting features and I still do. Then I found ***Lotus 1-2-3 for Scientists and Engineers***, written by William J. Orvis and published by SYBEX Inc., which convinced me that Lotus 1-2-3 (or any spreadsheet) is an excellent tool for engineering. Since then, I have been using Lotus as the main development tool for my engineering work, instead of programming in FORTRAN, Quick Basic, etc.

I was always fascinated by Lotus macro language that could reduce tedious tasks. The macros in the literature and in commercial packages were always limited to one type of cell data and the data had to be contiguous. Also, I was disappointed with the macro library concept presented in the literature and the commercial packages, which required combining macros into one file per task or project. Why should I keep dozens of macros in the worksheet and waste memory?

My macro library is menu driven by central Macro Managers allowing the user to activate any macro using "point and shoot" while the macros are stored in the disk. The manager combines the macro, defines the range names and activates the macro. When the work is finished, the manager erases the macro and returns to the main menu. The macros should be as free from limitation as possible. My research has resulted in the SUPER MACRO LIBRARY for LOTUS 1-2-3.

Super Power is not a replacement for Lotus manuals. All of the macro keywords and functions of 1-2-3 are not included. However, ***Super Power*** does include all the techniques that are used in the SUPER MACRO LIBRARY, which are beyond the normal scope of macro books. Some of these tricks and techniques may be found scattered through the literature. In ***Super Power***, they are organized, enhanced, improved and explained.

Super Power describes the techniques used in the macros and how they can be applied, using actual macros from the SUPER MACRO LIBRARY. ***Super Power*** can be used as an unofficial manual to the SUPER MACRO LIBRARY; however it goes far beyond that. It explains, in careful detail, every macro structure and code and presents all the techniques and tricks that were used in the macro. When an issue justifies more explanation, a special chapter is dedicated to it and cross-referenced.

To save you from keying in the macros in ***Super Power***, the SUPER MACRO LIBRARY for LOTUS 1-2-3 is offered at a special low price see the order form of the SUPER MACRO LIBRARY. Occasionally you may wonder why a macro is written in a particular way? There is always more than one way to accomplish a task. In addition, the author wishes to avoid the issue of copyright infringement on other commercial macro libraries. Every effort was made to write the macros and organize so they are unique.

- Every macro is a separate file, which gives greater flexibility and saves memory, especially when the macros are used through the macro managers.
- All range names associated with the macro are numbered, so there is no possibility of two identical range names in different macros.
- The macros are as broad as possible so they can handle ranges, regardless of what they contain, and the data doesn't have to be contiguous. The macro always checks every cell's contents, prefix, and type, and acts accordingly.
- The macros can be used independently, or from the macro manager.
- The macros should work in Automatic and Manual Recalculation mode.
- The macros check for the Lotus release being used and adapt their behavior accordingly. For example: if the macro is used in release 3.0 and up, it will treat 3-D ranges as well as 2-D ranges. However, if the macro senses that it's being used in 2.0/2.01/2.2/2.3/2.4, it will ignore all 3-D functions and macro commands. When it is impossible to use the same macro for all the releases, a special macro is included, usually because of some incompatibility.

Welcome

This is not a standard Lotus macro book, it is a unique, one-of a-kind book. Only a few of the Lotus books include realty-based, powerful unlimited macros; and there only a few are included.

Super Power is totally different. It is based on more than 250 macros included in the SUPER MACRO LIBRARY from K.I.T.A.L. Software. These macros are macros built in so that they do not suffer from the limitations of other commercial macro libraries. For example: they do not demand that the data in a range have to be the same type, contiguous or have other limitations. They handle ranges, not just columns or a single cell, but a range containing any type of data: labels, numbers, formula, blank, @MA and @ERR. If the macro is designed to change the labels to uppercase, it will ignore all the other types of data and will treat only the labels in the range. If the macro is designed to multiply all the numbers by 1.2, it will multiply only and all the numbers in the range and will ignore all the other cells. The macros handle both 2-D and 3-D ranges transparently, and automatically sense what type and version you are using and applies the correct code for every type of Lotus.

We have developed a universal range handling code which is repeatedly used as a whole or with modification in many macros. This routine prompts you to paint the range to be processed and then the routine moves the cell pointer to all the cells in the range and processes them. Even though the range handling routine is repeatedly used in many macros, we still explain the code in every macro.

Super Power is essentially a handbook designed for users of all levels of experience in Lotus 1-2-3 macro language. If you have some experience in macro writing, you can try and jump directly to macros of the fourth [4] level of difficulty. If you are a more advanced user you might start with the study of the eighth [8] level macros. If you have little or no experience, start with the **Macro Tutorial** and then move to macros of zero [0] level of difficulty. ***Super Power*** is flexible, which allows the reader to study macro programming by reading the analysis and explanation of actual macros, not the other way around.

Super Power offers detailed analysis and explanation for every macro code and command, and any technique or trick that is used to produce the correct results. When we analyze a macro and come to such a new or unconventional technique, it is cross-referenced to the chapter dealing with the same technique, or we explain the technique within the macro as needed. All these techniques are in Part One in the **Advanced Macro Techniques** chapter, so the experienced reader can study the techniques without having to read the macros. However, the techniques are simply explained so the principles are easy to understand. In Part Two of the book, the macros are divided into different group categories based on the action they perform: printing, graphics, copy, move, ranges, format etc. Each group is explained in a chapter and each macro in the chapter has a section where it is carefully explained and analyzed.

How to Use ***Super Power***
Book Conventions

How to Use *Super Power*

If you read ***Super Power***, you probably belong to one of two groups:

- The first group of readers purchased ***Super Power*** to get the super power macros and to key them into Lotus 1-2-3 to use them in daily work. If you belong to this group, we suggest using the discount option to purchase the complete SUPER MACRO LIBRARY for LOTUS 1-2-3, a copyrighted package including all the macros in ***Super Power*** plus 220 *.MLB macros to use with the MACROMGR.ADN add-in, and six more library managers in addition to the SMALLMGR.WK1 macro manager featured in ***Super Power***. Keying these complex macros into Lotus 1-2-3 may be a very time consuming process.

  This macro code was written to make these macros as safe as possible with minimal, or no limitations, which demand more code. If you key a macro into Lotus 1-2-3, always key it into an empty worksheet, beginning from the A1 cell according to the given macro text. Then save it. When you need to use it, combine it to your current worksheet. All the cell addresses in the macros are based on the assumption that the first cell is the A1 cell. When you combine it later into your worksheet, Lotus 1-2-3 adjusts the addresses respectively to the location.

- The second group of readers purchased ***Super Power*** to learn the ultimate techniques to create super power macros and applications. If you belong to this group, understand that ***Super Power*** is not trivial to master; but if you want to learn macro programming at its best, you will not find a more complete source. Follow the guidelines for beginners. If ***Super Power*** proves too advanced, we suggest ***1-2-3 COMMAND LANGUAGE*** from the QUE Corporation, 1986, an excellent book written by Darien Fenn. I gained the rest of my knowledge from experience, magazine articles, and analyzing macros that others wrote.

  If after you study a specific macro command or function in the Lotus 1-2-3 manual or on-line help and it is unclear, we recommend that you copy the specific code into Lotus 1-2-3 and create a small macro, including only this code. Then execute it to see what it does. The same is true when you see a complicated string function. Break each function to its sub functions and see what every part does. In ***Super Power***, we do not dedicate the same detailed explanation to functions in the code because Lotus users have to know how to use functions, which are the essence of the spreadsheet. You may not be familiar with a specific function, but you have to know how to write a function in Lotus 1-2-3. To fully understand the function, you have to isolate it, key it separately, and see what it does.

Book Conventions

Super Power uses the following typographic conventions:

Example of conventions	Description
`{BEEP}/RNC, labels101, LOWRCASE`	Macro code listing is written in point eight type with a lighter weight. Ranges such as labels101 are written in lowercase format. Lotus 1-2-3 functions and commands are written in uppercase format. The main macro name like the LOWRCASE macro is also written in uppercase format.
`{BEEP}, /RNC, @INDEX`	Macro codes appearing in the book's text are written in point 10 type with lighter weight and in uppercase format.
[labels101]	When the book's text refers to range names, they are always surrounded by squared brackets.

Macro Tutorial

Beginners

If you are a beginner in the macro field, start with the **What is a Macro?** chapter, which will guide you to write your first macro, and give you insight into the types of macros, macro libraries, etc. After finishing this chapter, start your tour of the macros. To make it easier, each macro title is graded with a difficulty level number starting from zero and going to ten. If you are a beginner, start with the macros marked with [0] or [1] which are very simple macros mimicking the keys of the keyboard. When you gain more confidence, move to the next level and pick a macro to study. Activate every macro and see how it works.

If you see a code and do not understand its purpose, use the STEP mode to activate the macro step by step. If the macro code contains the `{PANELOFF}` or the `{WINDOWSOFF}` commands, change them to `{PANELON}` or `{WINDOWSON}` before you start the macro, otherwise you will be unable to see all the actions of the macro. The key is to EXPERIMENT! EXPERIMENT! EXPERIMENT! Do not be afraid to play with the macro code, which is the only way you can actually see what happens. However, remember to keep a backup copy just in case you manage to destroy the macro code.

Advanced Users

If you have some experience with macro writing, you can still get valuable information from the **What is a Macro?** chapter, because it reviews the types of macros, add-ins and macro library's organization. Then you can pick macros to study. If you are a really advanced user, then start to read and study the techniques in Part One. However, we recommend that Part One, which contains the techniques that have been used in the macros, should be read in order. Many techniques use other techniques as building block; therefore the basic techniques are featured first.

What Is a Macro?

A macro is a series of written instructions consisting of keystrokes, macro commands, or both that the Lotus macro processor can understand and execute to speed up repetitive or complex tasks. A macro can automate procedures normally performed from the keyboard, such as using commands; or a macro can perform complex tasks and programming procedures, such as loops and if-then-else statements. To tell Lotus to start to execute instructions, we usually need to press two keys together, normally these are the ALT key and a one character key, such as ALT and A, in Lotus 1-2-3 for Windows the CTRL key replaces the ALT key .

For example, normally it takes 7 key presses to change the column width to 20. First you press the Slash "/" key, then you select the **W**orksheet **C**olumn **S**et-width menu options, and then you type 20 and press the ENTER key. In macro language, we write it as:

```
/WCS20~
```

Every character in this macro is the first character (capital letter) of the equivalent Lotus menu option: [/] is for the Slash key, [W] for **W**orksheet, [C] for **C**olumn, [S] for **S**et-width, 20 is the column width and tilde [~] stands for the ENTER key in the Lotus macro language. As a Lotus user, you probably know that you can press the same keys (capital letters) from the keyboard to select from the Lotus menu. The macro language uses the same characters. Therefore, if you are familiar with the Lotus menu structure, you can easily write macros that use the same keys you use from the keyboard.

This macro will save 5 key presses every time it is activated. The more you use it, the more key presses and time you save. You can define any set of key presses as a macro and activate it by pressing ALT and an assigned key together. Imagine how time-consuming it is to try to change the width of 50 columns manually instead of using a macro. Try to calculate how much time you are wasting every day by repeating again and again the same key press sequences if you are not using macros.

A macro can just be a set of key presses, but the Lotus macro language includes commands and functions that bring it to a high level programming language almost like BASIC, which many claim is the most popular computer language today. Macro commands and functions add versatility to your macros beyond the keyboard alone. These include:

- Transpose a range with formulas correctly
- Change text appearance in large tables automatically from Lowercase to Uppercase or Proper and vise versa
- Print form letters from an address database
- Print address labels from an address database
- Print a current date stamp like "Monday Dec-15-1991" without typing
- Change a number written in numbers into a number written in words

The SUPER MACRO LIBRARY and other commercially available macro libraries offer extensive samples. Add the Lotus support for custom menus and you can build friendly, menu-driven applications and macros, that work with the Lotus menu bar style, familiar to any Lotus user.

How a Macro Works?

The only way to fully understand what a macro is and how it works is to write one. The minute you manage to write even the simplest macro you have broken the ice; therefore we will start right away:

- Start a new and empty worksheet in Lotus 1-2-3
- Place the cell pointer on an arbitrary cell like the B1 cell
- Start to type exactly as shown:

```
'/WCS20~
```

Type these eight characters and then press the ENTER key. Don't forget to type an apostrophe "'" at the beginning, and a tilde "~" at the end. Now the B1 cell will display:

```
/WCS20~
```

The apostrophe is not shown. It is only a prefix telling Lotus the cell contains text and is aligned left.

- Place the cell pointer on the A1 cell and type:

```
'\X
```

Then press the ENTER key. Again the apostrophe is hidden and the text appears as:

```
\X
```

So far, all we have done is type some text into the A1 and the B1 cells. How can we make Lotus 1-2-3 understand it is a macro and how do we force Lotus to activate it?

Before Lotus can understand that the text in the B1 cell is a set of key presses to execute (a macro), we must assign a RANGE NAME to the B1 cell. For now, let us say that a range name must be composed of the back slash "\" and one of the Alphabet characters, like the "\X" in the A1 cell. But if you just write the "\X" alone and press the ENTER key, the contents of the A1 cell will look like this:

```
XXXXXXXXX
```

Therefore you have to type the apostrophe "'" as the prefix, otherwise Lotus considers the back slash "\" to be the prefix which causes the "X" to fill the column width.

There are two ways to issue a range name for the B1 cell:

The first uses the text already written in the A1 cell, therefore the range name will be [\X]. Place the cell pointer on the A1 cell and type:

```
/RNLR
```

Press the ENTER key. Now the B1 is named [\X]. To verify it, press F5 (the GOTO key), type \X and press ENTER, the cell pointer will land on the B1 cell. Assigning range names

this way is advantageous because A1 serves as a reminder that B1 is also the cell which has the [\X] range name. It is also possible to name a large number of cells in one operation as is done in all the macros in ***Super Power*** (see the built-in instructions in every macro).

The second way to issue a range name isn't highly recommended because it doesn't use the A1 cell; therefore it lacks the reminder of the range name. Place the cell pointer on B1 and type:

```
/RNC\X
```

Press ENTER twice. Now B1 is named [\X]. To verify it, press F5 (the GOTO key), type \X and press ENTER. The cell pointer will land on B1. To start the macro, press the ALT key and hold it down, then press the "X" key while you hold the ALT key. The column width will change to 20 characters width. Why? When Lotus senses the ALT-X key combination is pressed, it looks for a cell named [\X] (the B1 cell) and starts to execute the keys and commands in the cell in the exact order in which they were written. When Lotus reaches the tilde "~" character, it understands it as a command to press the ENTER key.

The macro type that we have just build is called a KEYBOARD macro because it exactly simulates the keys pressed from the keyboard. Most of the macros in ***Super Power*** and other commercially available macro libraries are far more complicated and include macro commands and functions in addition to keyboard key simulations.

The simple macro that we just wrote contains only one cell, but a cell can hold up to 240 characters if you use release 2/2.0/2.2/2.3 and 2.4 (2-D releases) and up to 512 characters in releases 3/3.1/3.1+/3.4 and 123W (3-D releases). Therefore, it seems that we could expand the macro to perform more tasks than just change the column width. But this is unnecessary, we can continue our macro to the next B2 cell and down without filling the whole 240 characters in every cell. Lotus will continue to execute the macro all the way down until it reaches an empty cell or meets the {QUIT} or {RETURN} macro command. A macro command in the Lotus macro language is always surrounded by curled brackets "{}".

Saving a Macro to Disk for Later Use

Macro is a Lotus file just like any other Lotus file, therefore it is saved the same way you save a Lotus file using /FS<macro name> and ENTER. The macro is saved with the ".WK*" extension. To use it again, press /FCCE<macro file name> and press ENTER to combine it to an empty area on your current worksheet and then assign a range name to the first cell of the macro and activate it.

It may seem like we can have up to 26 macros and macro names in one worksheet because there are only 26 letters in the English alphabet. This is partially true since there are some utilities (add-ins) in the market that allow you to use any kind of name for a macro. In ***Super Power***, we included a macro that also does this. The RUNKEY2.WK1 macro allows you to activate macros with any kind of name. For example, a macro can have a name such as [COLWID]. This was the situation until the release of Lotus 1-2-3 version 2.2 and up. The later releases of Lotus 1-2-3 support any macro name including the "*" type. To execute a macro in the new releases, press the ALT-F3 key combination and then point to the macro to execute, press ENTER. You can also use the ALT-* key combination to execute a "*" macro type.

With the introduction of release 2.2/2.3 and 2.4, a new type of macro was introduced. This is a hypertext macro which resides in the memory instead of the worksheet. Therefore, it can be used on many worksheets one after another. You can erase the worksheet or retrieve a new one and the macro will continue to stay in memory ready for execution. This type of macro has the ".MLB" extension added to the macro name. To handle such macros, the MACROMGR.ADN add-in is included with Lotus 1-2-3 2.2/2.3 and 2.4.

What Is an Add-In?

To use the Lotus platform for special needs, the programmers built Lotus 1-2-3 so it can work with external modules that add performance and properties on demand. For example, a programmer with the right development tool (supplied by the Lotus Company) can develop a full blown statistical application that will work in harmony with Lotus 1-2-3 on call. This type of module is called "ADD-IN". The add-in has to be ATTACHED to Lotus and then INVOKED to be used. When it is invoked, it becomes an integral part of Lotus including Menus, Formulas etc. One of the add-ins that comes with Lotus 1-2-3 2.2/2.3 and 2.4 is the MACROMGR.ADN add-in, which allows the *.MLB macros to operate.

Using the MACROMGR.ADN Add-In

Consult the Lotus 1-2-3 manual for instructions on how to use the MACROMGR.ADN add-in manager. The MACROMGR.ADN add-in has to be ATTACHED and then INVOKED to make it usable. To attach it:

- Start Lotus 1-2-3 (release 2.2 or 2.3 or 2.4 only)
- Press the following keys in order

```
/AAMACROMGR
```

Press the ENTER key. Lotus will offer four options to assign two key combinations to invoke the add-in (a combination of ALT and one of the F7, F8, F9 or the F10 function keys). If you choose not to assign the two key combinations, you will need to press in the following order:

```
/AIMACROMGR
```

Press ENTER to invoke the add-in. When the MACROMGR.ADN is invoked, a menu appears and you are offered options to load a macro, to edit a macro, to save the macro on disk etc.

Loading an *.MLB Macro

Invoke the MACROMGR.ADN using the two keys combination (ALT-F*) or use:

```
/AIMACROMGR [ENTER]
```

Choose the **L**oad menu option, then point to the macro name on the screen and press ENTER, or write the macro name and press ENTER.

Executing the Macro in Memory

Press ALT-F3 and point to the macro name on the screen and press ENTER or write the range /routine name to activate and press ENTER.

Macro Listing Conventions

Terminology:

Macro keys, macro code

When we use key sequences in the macro, representing the exact keys on the keyboard e.g. to change the column width, press the `/WCS` keys, we may use the phrase "macro keys", "macro code" freely.

Macro commands, commands, Routine command {routine_name}:

When we use commands which are not presented in the Lotus macro language by one key or may not be available from the keyboard, we may freely use, the "macro command/s" or "command/s". For example: "the macro issues the `{EDIT}{HOME}{RETURN}` macro commands".

Routines, routine names

Super Power freely uses the "routine" term interchangeably with subroutine. For example, "The macro issues the `{BRANCH print121}` macro command which routes the macro control to the [print121] routine".

Range names

When we write range names in ***Super Power***, we always surround then with squared brackets. You can find phrases like: "the cell pointer moves to the B23 cell named [labe121]". In this example, the text refers to the B23 cell which also has the [label121] range name (without the squared brackets).

Section

Section refers to the sub chapter that contains the macro listing and explanation. Every macro in the second part of the book forms a section. There are sections in ***Super Power*** for each macro featured in Part Two.

The macro issues, the macro uses, the macro continues

"the macro issues" means "the macro processor issues". For example, you will see a phrase like "The macro issues the `{GOTO}A1~` macro command"

Macro Structure

Because all the macros in ***Super Power*** are the actual macros from the SUPER MACRO LIBRARY package, and they are designed to work independently or from the Macro Managers, they are built to insure they will work as expected. If you write macros using the same structure, you will also be able to use them with the macro manager. To make it real, we use the LOWRCASE.WK1 macro which turns any label in a selected range into a lower case form. Our example, shows the structure and other conventions used in ***Super Power***. The macro listing for the LOWRCASE.WK1 macro is:

```
     A                B                C                D                E
 1 *---A macro to turn all labels in a 3-D or 2-D range to LOWERCASE form
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 LOWRCASE        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb410,0}~
12 cont410         {LET hereabs410,@CELLPOINTER("address")}~
                   {LET counter410,0}
13 !               {FOR counter410,0,@COLS(Which range ?)-1,1,labels410}
14 !               {LET rel410,@INFO("release")}~{IF @LEFT(rel410,1)<>"@"}
                   {GOTO}{hereabs410}~{LET counterb410,counterb410+1}~
                   {IF counterb410<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs410}~{BRANCH cont410}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter410             3
18 countera410            5
19 labels410       {RIGHT}{LET here410,@CELLPOINTER("address")}~{LEFT}
                   {FOR countera410,0,@ROWS(Which range ?)-1,1,labelsa410}~
                   {IF counter410<@COLS(Which range ?)-1}{GOTO}{here410}~
                   {LET countera410,0}~
20 !
21 here410         $D$1
22 !
23 labelsa410      {PANELON}{RECALC prop410}{IF @CELLPOINTER("type")<>"b"
                   #AND#@CELLPOINTER("type")<>"v"#AND#@CELLPOINTER("prefix")
                   <>"\"#AND#@CELLPOINTER("prefix")<>"|"}{PANELOFF}
                   /RVprop410~~{EDIT}~
24 !               {DOWN}
25 !
26 prop410         \Z
27 !
28 counterb410            4
29 hereabs410      $A$1
30 !
31 rel410
```

- The first 9 cells in the A1..A9 range of the macro are reserved for the description and use instructions.
- The A10 cell holds the [\Z] macro name and the B10 cell holds the `{BREAKON}` macro command. We have chosen this macro name just in case the worksheet is in `{BREAKOFF}` status, but any other DO NOTHING macro name such as `{}`, `{ESC}` will do.
- The A11 cell holds the macro name (LOWRCASE in this example). This

name must be exactly as in the DOS file name of the macro, i.e. when the macro is saved, it should be saved as LOWRCASE.WK* file. This will be explained later when we will learn about the MACRO MANAGER, the last macro in ***Super Power***.

- The B10..I** cells are for the macro code and custom menus. The macro is always written in the "B" column. When a custom menu is needed, it can occupy up to eight columns, the B, C ,D, E, F , G, H, and the I columns. Therefore the macro can occupy up to nine columns.
- The cells on column "A" are used only for range names and `{routine}` names. A cell not used for a range name should contain the "`!`" exclamation mark. This is to keep the "A" column contiguous with no blank cells. Any other character can be used instead of the "`!`" character; however we will use it in ***Super Power*** to be compatible with the SUPER MACRO LIBRARY. The [\Z], [LOWRCASE], [cont410], [counter410] and the rest of the labels in the "A" column are the range names of the cells to their right. For example, the [counter410] range name refers to the B17 cell. When you place the cell pointer on the A10 cell, which contains the [\Z] label and carries the instructions in the A2 cell, and press /RNLR END DOWN ENTER, Lotus automatically assigns all the labels in the "A" column beginning with the [\Z] label down to the [rel410] label as the range names for the cells to the right.
- Every range or routine name will be uniquely numbered (in the LOWRCASE macro the number is 410). This way we can use a code from one macro to build another macro and then use the Search/Replace utility to replace the number with a new number.
- If a macro creates a temporary macro/range name during its execution, it must delete it when the macro is finished. A good macro should leave minimum traces.

Note: The "A" column of the macro is dedicated for range names, description and use instructions only. The code starts with the B10 cell. If the macro does not contain a custom menu, all the code is in the "B" column. This is important to remember when you want to key the macro code into Lotus 1-2-3. Therefore, in such macros, the code as it appears here (in the LOWRCASE.WK1 macro) is enough for you to be able to key it into Lotus 1-2-3. It occupies the "A" column and the "B" column only. When the macro uses dynamic string formulas to create its code, we show each formula's code separately and you should key the formula's code into Lotus 1-2-3 and NOT the code as it appears in the main listing (which is the result of the formula that appears in the cell). When the macro uses custom menus, which cannot be displayed on one page without overlapping, we also show every menu code separately. In both of these cases, when the macro includes complicated cell formulas and/or custom menus, we also show a full cell contents list at the end of each section to make it easier for you to key the macro into Lotus 1-2-3.

Writing Macros to Work with the Macro Manager

	A	B	C	D	E
1	*---A macro to invert a column of numbers or text				
2	*---Use the /Range Name Label Right {End} {Down} [ENTER] to define				
3	*---the range names in this column (starts with the \Z macro name)				
4	*---Hold the <ALT> key and press <Z> to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	INVRTCOL	{WINDOWSOFF}{PANELOFF}/RNCInvert column ?~~/RNDinvert...			
12	!	{WINDOWSOFF}{PANELOFF}{GOTO}Invert column ?~/WIC~/DF...			
13	!	/DSRD{BS}.{END}{DOWN}{RIGHT}~P~~G			
14	!	/WDC~			
15	!				

Super Power includes the macro code of the SMALLMGR.WK1 macro manager, which allows you to use all the macros in ***Super Power*** from a menu, without the need to combine them first. The Macro Manager handles all that is needed to activate the macros. The macros in ***Super Power*** were designed to work with the macro manager unless their nature does not allow it (for example, all the toggle macros have to be part of the current worksheet; therefore there is no point in using them with a macro manager). However you can write or modify your macros or any macro that you acquire to make them compatible with the macro manager by following these rules:

- First nine cells in the first column of the macro are for instructions or the exclamation mark "!".
- The 10th cell holds the \Z macro name.
- The 11th cell holds the mnemonic macro name (must be IDENTICAL to the macro file name).
- Downward is for RANGE NAMES, etc.
- There should not be any blank cell in the first column. If the cell is not used for a range name, then it must contain the exclamation mark "!" (this also applies for the first 9 cells); so the first column is contiguous and we can use `{END}{UP}` or `{END}{DOWN}` for naming and erasing.
- Don't use the `{QUIT}` macro keyword because it stops the process and exits to the READY mode when the macro is finished, and the Manager's main menu will not re-appear. Instead, use a branch instruction to an empty routine like `{BRANCH EMPTY}` where EMPTY is an empty cell, with one exception: when an error routine has been activated, the macro loses track. The cell should contain `{BRANCH \Y}`.
- All the macros should start with the \Z macro name and `{BREAKON}`.

If you look at the sample macro INVRTCOL.WK1, you can see A1 to A9 cells hold the use instructions, the A10 cell holds the '\Z label, and the A11 cell holds the INVRTCOL range name (identical to the INVRTCOL<.WK1> file name). All the other cells in the "A" column contain the exclamation mark "`!`" character. You can build your own macros or adapt other macros to be used with the MACRO MANAGER as long as you follow these simple rules.

Custom Menu Listing Convention

When describing menu listings separately from the general macro code (when all the menu codes cannot fit across the screen without overlapping), a code that belongs to the same cell but is too long to be displayed in the screen/page in one line is indented and written in the next line. Example:

```
NENU TITLE ======>  View/change
                   View or change the range area to print
                   Select the range to view and press [ENTER] {GET key1155}
                   {IF key1155="{ESC}"}{DOWN}{UP}{BRANCH end155}{ESC}
                   {IF key1155="~"}{DOWN}{UP}{LET change1155,@CELLPOINTER
INDENTED CODE ========> ("contents")}~{change155}{BRANCH end1155}
                   {key1155}{?}~{LET change1155,@CELLPOINTER("contents")}~
INDENTED CODE ========> {change155}
                   {UP}{DOWN}{MENUBRANCH menu1155}
```

The indented code in this example is the continuation of the code in the line above and should be written in the same cell. In the macro listing, the cells are numbered, a code that belongs to the same cell but is too long to be displayed in the screen in one line is written in the next line, but is not numbered. In the following example we placed the "**Continue ====>**" note to mark these lines:

```
          A              B                C                D              E
 19  crit2142       {WINDOWSON}{PANELON}{GETLABEL "Insert criterion string
     Continue ====> or [F]to Find or [S] to Skip: ",crit3142}~{IF @UPPER
     Continue ====> (crit3142)<>"F"#AND#@UPPER(crit3142)<>"S"}
     Continue ====> {RECALC critform142}{critform142}{WINDOWSOFF}{PANELOFF}
 20  !              {IF @UPPER(crit3142)="F"}/DQF{ESC 4}{QUERY}{?}~
     Continue ====> {WINDOWSOFF}{PANELOFF}{PGDN 2}{HOME}{RIGHT}{END}{DOWN}
     Continue ====> {DOWN}{GOTO}output142~{WINDOWSON}{PANELON}
     Continue ====> {MENUBRANCH extrct142}
 21  !              {IF @UPPER(crit3142)="S"}{RIGHT}{BRANCH crit2142}
 22  !              {IF @LENGTH(crit3142)=0}{RIGHT}{BRANCH crit2142}
```

The code marked with the "**Continue ====>**" note in this example is the continuation of the code in the line above and must be written to the same cell.

Important Note: Sometimes it may seem that there are unnecessary commands or repetitions of the same command in the code of the macros in ***Super Power***; but from our inquiries, they are needed to cover the many versions of Lotus 1-2-3. For the macro to correctly work they are needed. In every Lotus release, there are some quirks and incompatibilities to the previous versions and different releases behave differently in the same situation. There are many examples of this. Sometimes when we look in our macros, we may forget why a specific command is there, but we are very careful not to change it before we can extensively test it in all the versions of Lotus 1-2-3.

Formulas Listing Convention

When we refer to formulas, you will often see the following notation:

```
33 wid207          @STRING(@CELLPOINTER("width")-1,0)
```

Only the right part "`@STRING(@CELLPOINTER("width")-1,0)`" is the formula, "`33`" is the row number where the formula is (in the main macro listing) and "`wid207`" is the description name of the range name that was assigned to the cell containing the formula. We always place the description name to the left of the range with the same range name. Here, the B33 cell is named [wid207], the "`wid207`" text in the A33 cell is only a reminder to clarify the code of the macro. A full cell contents list is attached at the end of each section.

Complete Cell Contents Listing Convention

When macro is complicated, we bring a complete list of all the cell contents of the macro at the end of the section. If you use this list to key the macro into Lotus 1-2-3, you do not have to key the apostrophe. For example, if the list contains these lines:

```
B10: [W9] '{BREAKON}
A11: U [W15] 'FORMLETR
```

it means that the B10 cell contains the `{BREAKON}` macro command and the A11 cell contains the "`FORMLETR`" label. When you key the code into the B10 and the A10 cells of the macro, you do not have to type the apostrophe before the `{BREAKON}` macro command and before the "`FORMLETR`" label. Lotus adds them automatically because they are labels. The [W9] informs you that the "B" column's width is "9" characters. The [W15] informs you that the "A" column's width is "15" characters. The "U" informs you that the A11 cell is an unprotected cell. However, if the list contains lines like these:

```
B10: [W9] '@CODE(...
B10: [W9] '+A20.....
```

You have to type the apostrophe because the macro writes a formula as a label into the panel. If you omit the apostrophe Lotus writes it as an active formula.

Advanced Macro Techniques

Using the {NAME} Macro Command or the F3 Key

When you press the F3 function key when file names or range names are listed on the screen, e.g. after pressing the / **F**ile **R**etrieve, Lotus displays a row of files from the default directory in the panel. If you now press the F3 key, Lotus switches to full screen display of the files in the default directory. The macro equivalent of F3 is the `{NAME}` macro command. To select a range name or a file name, we use the `{NAME}` command to display a full screen list making it easier for you to select the range name or the file. Here is a simple example from the IMPORT.WK1 file.

	A	B	C	D	E
10	\Z	{BREAKON}			
11	IMPORT	/FI{?}~{NAME}{?}~			

here is another example from the GRAPHDEL.WK1 macro.

	A	B	C	D	E
10	\Z	{BREAKON}			
11	GRAPHDEL	{BREAKON}			
12	loop019	/GND{NAME}{?}~{ESC 6}			

here is another example from the RANGEDEL.WK1 macro.

	A	B	C	D	E
10	\Z	{BREAKON}			
11	RANGEDEL	{BREAKON}			
12	loop030	/RND{NAME}{?}~			

Moving the Cell Pointer to the Top Left Cell of the Screen

Many times you need to move the cell pointer to the top left cell of a database or a different kind of table. At the same time you want to place the cell pointer on the upper left cell of the screen. Using just `{GOTO}` to the desired cell address will not be enough if the current cell position is less than 7 columns or/and 19 rows far from the table. Therefore the cell pointer must be moved far enough and then back to the desired cell address. We use this same technique in our macros to move the cell pointer to the last column of the worksheet (the IV column) and then back to the desired cell address. The macro code for such a task may look like:

```
{GOTO}IV1~{GOTO}linkk282~
```

The cell pointer is on the upper left cell of the screen and simultaneously on the cell named [linkk282].

Controlling Screen and Panel Display Activity

When a macro is activated, the screen and the panel show a lot of activity which has some faults when you need to design and develop professional looking applications. This activity greatly slows the macro operation. The classic Lotus macro language has four macro commands to control the panel and screen display activity. The commands are `{PANELON}`, `{PANELOFF}`, `{WINDOWSON}`, and `{WINDOWSOFF}`. These commands do exactly what their names mean. The new releases have the `{CLEARENTRY}` or `{CE}` and the `{PANELOFF CLEAR}` new commands, but because we would like to keep it possible for all the macros to work in release 2.0 and up, we do not use the last three commands in our macros. You will find that we use these macro commands extensively in the macro code to control the macro appearance and to speed operation.

Manipulating the Panel

It is possible to make the macro to type text into the panel exactly as it is typed from the keyboard. This makes a very powerful tool in macro creation and programming. It can be used to display prompts for the users, to manipulate cell contents, to apply a formula on a cell content using the enveloping method, to apply a formula on adjacent cells using the pointing method, etc. This section only deals with the technique for type text into the panel. The next sections deal with more ways to manipulate the panel. To understand the technique, let us look at the following code:

	A	B	C	D	E
1	\Z	Move the cell pointer to the target location and press [ENTER]			
2	!	{GET key102}			
3	!	{ESC}			
4	!				
5	key102				

The macro begins with the text prompt:

```
Move the cell pointer to the target location and press [ENTER]
```

Because Lotus doesn't see any valid command in the text, it understands that it is text to be written to the panel, therefore Lotus just writes it directly to the panel. To make this message useful, we need to pause the macro and allow you to read the message and respond. Therefore the macro issues the:

```
{GET key102}
```

to pause until you press a key. When you press a key, the `{ESC}` command immediately clears the prompt from the panel before Lotus writes it into the current cell. In the new releases of Lotus 1-2-3 (2.2 and up), the `{CE}` or `{CLEARENTRY}` macro command can also be used. Because we want all the macros to work in all the Lotus releases, we used this technique with the `{ESC}` macro command. Below is a more sophisticated example from the LINK.WK1 macro:

	A	B	C	D	E
62	err1@l282	{ONERROR err2@l282,message@l282}			
63	!				
64	err2@l282	{BEEP}{message@l282} press ANY KEY to continue ... {GET key1@l282}{ESC}{BRANCH \Y}			
65	!				
66	message@l282	File does not exist			

When an error occurs, the `{ONERROR err2@l282,message@l282}` writes the error message in the B66 cell, [message@l282], and then activates the [err2@l282] routine. The [err2@l282] routine starts with `{BEEP}` which sounds a beep to warn you that error occurred. Then the macro issues the `{message@l282}` routine command. Because the [message@l282] routine in the B66 cell contains only text (the error message), Lotus types (injects) the text directly into the panel. The macro continues and writes the " `press ANY KEY to continue ...`" text into the panel to create a complete and informative message.

Let us see how this works. First, when an error occurs, the macro writes the Lotus standard error message into the B66 cell, [message@l282]. In this case, the error message is the

"File does not exist" error message. Next, the macro issues the {message@l282} command. Because the [message@l282] routine in B66 contains only text (the "File does not exist" error message), Lotus types (injects) the text directly into the panel which displays:

```
File does not exist
```

Next the macro writes " press ANY KEY to continue ..." into the panel which now displays:

```
File does not exist press ANY KEY to continue ...
```

The macro combines the standard Lotus error message with a customized message to create an informative prompt message in the panel.

Note: In the SUBTOTAL.WK1 macro, there is an example of how to write a formula using the panel.

Injecting Text into the Panel

When we say injecting text into the panel, we refer to a unique and very powerful technique for any macro programmer. Using this technique we can write (inject) any kind of text into the panel instead of manually writing it. Other advanced techniques are based on and use this technique. Look at the following macro:

	A	B	C	D	E
1	\a	{LET here205,@CELLPOINTER("address")}~			
2	!	{GOTO}B100~			
3	!	/M~{here205}~			
4	!				
5	here205	$AF$31			

The macro starts with the `{LET here205,@CELLPOINTER("address")}` macro command which stores the current cell pointer address in the B5 cell named [here205]. Then the macro issues the tilde "~" macro command to force Lotus to update the content of B5 immediately, and then issues `{GOTO}B100~`, which moves the cell pointer to the B100 cell. If we want the macro to move the content of the cell at the current cell pointer position back to the address before it moved to the B100 cell, there is no Lotus function that remembers what was the last cell pointer position. Therefore the macro continues with the `/M~{here205}~` macro command.

When the macro processor reaches the `{here205}` routine command, it starts the [here205] subroutine. However the [here205] subroutine contains only text (the $AF$31 address); therefore Lotus writes the content of the B5 cell, [here205], into the panel. To visualize the process, look what happens in the panel. When the macro issues the `/M~` macro keys the panel (in Lotus 2.4) displays:

```
B100:
Move what? B100..B100                    To where? B100
```

Now when the macro processor processes the `{here205}` routine command, the panel displays:

```
B100:
Move what? B100..B100                    To where? $AF$31
```

The `{here205}` routine command writes the content of the cell named [here205] into the panel as the response to the "To where? " prompt.

Note: The `/M~{here205}~` combination belongs to a group of commands which we call "Indirect macro command" techniques; only the `{here205}` command is the inject technique because the [here205] cell contains only text. Therefore the [here205] routine writes the text into the panel.

Indirect Macro Command ({GOTO} for Example)

	A	B	C	D	E
1	\a	{GOTO}{here410}~			
2	!				
3	here410	X1			

When we combine the previous technique with a Lotus command like {GOTO}, /M, /C, /RND, /RNC and other macro commands, we get a very basic and important technique that is used across many macros in ***Super Power***. Because it is such a vital technique, we named it separately as the "Indirect macro command". When you press the F5 function key and then type the address X1 and press ENTER, the cell pointer jumps to the X1 cell. Look at what Lotus does during this process. Assuming that the cell pointer is on the A1 cell, when you press the F5 key, the panel displays:

```
A1:                                                    POINT
Enter address to go to: A1
```

Now type "X1" and the panel will display:

```
A1:                                                    POINT
Enter address to go to: X1
```

All that's left to execute the movement is to press the ENTER key. When the macro issues the {GOTO} macro command, Lotus acts exactly as it does when you press the F5 key. Therefore the panel displays:

```
A1:                                                    POINT
Enter address to go to: A1
```

When the macro starts, the macro processor processes the {GOTO} command and the panel displays:

```
A1:                                                    POINT
Enter address to go to: A1
```

Now the macro processes the {here410} routine command, writing the content of the B3 cell named [here410] into the panel, which now displays:

```
A1:                                                    POINT
Enter address to go to: X1
```

The {here410} command writes the X1 text into the panel exactly as you type it from the keyboard. Last the macro issues the tilde "~" macro command, the equivalent of the ENTER key, to move the cell pointer to the X1 cell. This trick allows us to issue indirect commands using a text in predefined cells. This is a very powerful technique that highly advanced macro programmers use.

Safely Clear and Insert Data in the Panel

When changing the header or footer or inserting a new setup string in the printing process or saving a file and naming it with a new name or combining a file, there is always the possibility that Lotus will use default data that was previously used in the worksheet. The old data (if any exists) must be cleaned in a safe way, allowing you or the macro to insert the correct data, and also safely handle the process if there is no previous data. Usually the ESC key or the `{ESC}` macro command is an effective way to clear the old data from the panel, but sometimes more than one ESC is needed, especially when the file name includes the directory and sub directories. If there is no old data, the ESC key or the `{ESC}` macro command will take Lotus one step backward, which can be disastrous for the macro process.

Luckily there is a solution to this problem. When you type new data into the panel before the ESC key, it just appends to the old data or replaces part of it, but it does not erase the directories, therefore we must use the ESC key or the `{ESC}` macro command to clear the data. In this case, we need only one ESC, no matter how much text the panel contains.

Insert Print Header or Footer Safely

If you have a range to print that already has been assigned a print header and you want to print the same range with a different header or another range, when you press the /PPOH keys, Lotus displays the last used header in the panel, for example:

```
A1:
Enter header: 1992 Income
```

When you design a printing macro that also changes the header, you have no way of knowing if there is an old header or not. Therefore the macro needs to type a character and then to issue the {ESC} macro command. Let us look at the following code:

```
/PPRA1..C15~OH*{ESC}1993 Gross Income~
```

The code starts with the /PPRA1..C15~ macro keys, which define the A1..C15 as the printed range. Then the macro issues the OH (**O**ption **H**eader) macro keys and adds the astrix "*" character followed by {ESC}. The panel, then, displays the clean prompt:

```
Enter header:
```

The macro writes the "1993 Gross Income" new header and issues the tilde "~" which is the same as the ENTER key. The macro types the astrix "*" to make sure that even if there was no old header, the macro will work correctly. Without the astrix "*" the {ESC} command will return Lotus to the previous menu:

```
Header  Footer  Margins  Borders  Setup  Pg-Length  Other  Quit
```

This destroys the macro because Lotus cannot accept the new header on this menu. The following code will insert a footer instead:

```
/PPRA1..C15~OF*{ESC}1993 Gross Income~
```

Change Printer Setup String Safely

If you need to insert the `\027@\015\027\069\027\071` setup string to print a range and this is the first time you assign a setup string, the following macro code will work well:

```
/PPOS\027@\015\027\069\027\071~
```

However, if there is an old setup string, the macro needs to erase it before it can write the new one. The next code may look sufficient because the macro issues the `{ESC}` command clears the old setup string before it writes the new one.

```
/PPOS{ESC}\027@\015\027\069\027\071~
```

However, this macro code will not work in the case where this is the first time and the panel is empty. In fact , it will return Lotus to the previous menu:

```
Header  Footer  Margins  Borders  Setup  Pg-Length  Other  Quit
```

Now, if the macro tries to insert the setup string, it will cause an error. To solve the problem, the macro must first type at least one character as the new setup string and it can issue the `{ESC}` macro command to clear the panel. Only then the macro can insert the new setup string. Here is the corrected code:

```
/PPOS ESC}\027@\015\027\069\027\071~
```

Notice the SPACE character after the `/PPOS` and before the `{ESC}`. The SPACE character assures that the `{ESC}` macro command will not return the macro to the previous menu. Where an old setup string exists, the Lotus appends the SPACE character to the old setup string and the next `{ESC}` clears them both. Therefore, typing any character as a temporary setup string will insure that the `{ESC}` macro command will clear the setup string, whether a previous one exists or not.

Combine Files Safely

Here is an example from the LINK.WK1 macro for how to safely combine a range or cell from an external worksheet. For example: if you want to combine the content of the B1 cell in the TEST1.WK1 external worksheet in the B: drive to your current worksheet, you need to issue the `/FCCNB1~` keys. However, if the default directory is the C:\TESTMAC\123 directory, at that point the panel will display something like this:

```
Enter name of file to combine: C:\TESTMAC\123\*.wk?
```

To change the C:\TESTMAC\123*.wk? default prompt to the B:, you must press the ESC key a few times depending on the default directory's depth. This is not a problem when you work from the keyboard. All you need to do is to press the ESC key until the panel is clear and displays:

```
Enter name of file to combine:
```

Then type the `B:\TEST1`. How we can do this from a macro? How we can know how many times to issue the `{ESC}` macro command to clear the panel. The answer is simple: we cannot know; however we can work around to solve this problem. It appears that when the panel displays:

```
Enter name of file to combine: C:\TESTMAC\123\*.wk?
```

typing any character, such as the "`D`" character, the panel changes to display:

```
Enter name of file to combine: C:\TESTMAC\123\D
```

If we issue the ESC key, it clears the panel completely and only one ESC is needed. The combination of a character and the following ESC always clears the default directory. Therefore the complete code in a macro should be:

```
/FCCNB1~*{ESC}B:\TEST1~
```

Here we used the astrix "`*`" character before the `{ESC}` macro command.

Extract to File with Confidence

When you need to extract part or all of the worksheet into a file from inside a macro, you need to figure how many times to press the ESC key to clear the panel from the default file and the default directory. For example, when you press the /FXF keys from the keyboard, the panel may look like this:

```
Enter name of file to extract to: C:\123\TESTMAC\*.wk1
@SUMRNGS.WK1      AAA.WK1       AAAA.WK1       ADDDATA.WK1
```

This depends on where the default directory is. If the default directory is in a 4th level of a sub directory, you need to press the ESC key more times to change a drive and print a file name. Luckily, there is a way around this. If in this exact position in the panel, you write a character "k" for example into the panel, the panel changes to display:

```
Enter name of file to extract to: C:\123\TESTMAC\k
```

In this example, we used the "k" character, but any other character will do. Now only one ESC is enough to clear the panel. Therefore if you press the ESC key now, the panel will display:

```
Enter name of file to extract to:
```

It is ready to accept the new drive and/or file name. ***Super Power***'s macros use the same technique to safely extract to files and other operations. Let's look at the following code which extracts a range into the B: drive.

```
/FXF*{ESC}B:\~{NAME}{?}~{?}~R{ESC}
```

The macro issues the /FXF macro keys and then writes the astrix "*" character followed by the {ESC} macro command to clear the panel, then issues the tilde "~" macro command which is the same as the ENTER key. The macro continues with the {NAME} macro command which displays a full screen list of all the files in the B: drive, and then issues the {?} macro command to allow you to highlight a file name or type a new file name. When you finish defining the file name and press the ENTER key, the macro issues the tilde "~" again, and then issues a second {?} to allow you to highlight the range to extract. When you press ENTER, the macro issues the tilde "~" again to extract the file.

The end of the code is interesting; you may ask why press the "R" character and then the {ESC} macro command? After the second {?} when you press the ENTER key, two things can happen: If the file name already exists, Lotus displays the following menu:

```
Cancel  Replace  Backup
```

The "R" serves as the answer to the **R**eplace menu option and the macro extracts the range into the file, replaces the old file, and returns to the READY mode. In the READY mode, the {ESC} macro command does nothing. However, if the file name is a new name, then Lotus does not display this menu and the "R" character is printed to the panel; therefore the {ESC} clears the panel before Lotus writes the "R" character to the current cell.

Note: The *{ESC} code we used to clear the panel before the macro writes the B:\ as the

drive to extract the file to has some interesting properties. When we issue the `/FX` keys from the keyboard or a macro, Lotus displays a list of the *.WK* files in the default directory and types something to the panel, it may look like `C:\123\*.WK1`. To clear this prompt from the panel, we usually need to press the ESC key twice. Initially, this was the solution that we used in this macro, however we found that in the 3-D releases, only one ESC was needed. Sometimes it didn't work correctly in the various versions of the 2-D releases. After extensive testing, we found, when the panel display the `C:\123\*.WK1` (for example) and we type any character, only one ESC is needed to clear the panel, no matter how long the prompt in the panel is. This works correctly in all the Lotus releases we have used, including 2-D and 3-D. This is the reason why the macro writes the astrix "`*`" before the `{ESC}` macro command.

When the `{?}` macro command is used in a macro and the macro enters a Lotus menu (not a custom menu), you are expected to use one of the arrow keys and press the ENTER key. If you press the first capital letter of the menu option to choose the menu option, the macro works differently. Lotus may jump to the previous menu and the macro may lose track. It appears that this only happens in the built-in Lotus menus, custom menus are free from this bug. To overcome this limitation and to write a foolproof macro, we have to duplicate the built-in Lotus menu using a macro, so that the menu will look like a standard menu, but actually it is really a custom menu.

Rule: If you are using a macro or macro based application, NEVER USE the first capital letter of the menu option to activate a menu option if the menu is one of the Lotus standard menus. ALWAYS use the arrow keys to highlight the menu option and press ENTER.

Save a Worksheet with Confidence

When we write a macro to save a worksheet, we have to account for two cases. First when a file with the same name exists and, second, when the file name is new. If the file name already exists, Lotus displays:

```
Cancel  Replace  Backup
```

with the option to **R**eplace the file or not. In this case the, "`R`" character will issue the **R**eplace option and the `{ESC}` macro command will have no effect. However, if the file name is new, the file will be saved before the "`R`" is issued and the "`R`" character will be printed to the panel. Therefore the `{ESC}` macro command will clear it from the panel before it destroys the rest of the macro code. Here is a simple macro code which employing this technique:

```
/FS{?}~R{ESC}
```

The code starts with the `/FS` macro keys making Lotus prompt you to type a file name or to choose a name from the file list. Then the macro issues the `{?}` command allowing you to type the file name or move the highlight to a file name in the list. Then press the ENTER key. When you press ENTER, the macro issues the tilde "`~`" which is the same as the ENTER key and issues the `R{ESC}` macro commands to handle these two case described.

Using the F9 or the CALC Key on the Panel

The result of a formula in the panel can be calculated and entered into the panel by pressing the CALC (F9) function key or using the `{CALC}` macro command inside a macro. When the CALC or the F9 key is pressed while a formula is written in the panel, the result of the formula appears in the panel. For example: if you write the `@DATE(92 ,12,25)` date formula into the panel and press ENTER, the current cell will contain this formula, but will display the 33963 date number instead. Sometimes we want only the number and we do not need the formula. If you press the EDIT (F2) key, the panel will display the following:

```
@DATE(92,12,25)
```

If you press the CALC (F9), the formula in the panel will change into its result the panel will display:

```
33963
```

and the function is lost. Here is a more complicated example from the UNDERLIN.WK1 macro.

```
@REPEAT("-",@CELLPOINTER("width")-1){CALC}{HOME}'~
```

Because the code does not start with any Lotus command, Lotus writes the `@REPEAT ("-",@CELLPOINTER("width")-1)` directly into the panel until it reaches the `{CALC}` macro command. The `{CALC}` macro command is the equivalent of the CALC (F9) key, therefore it transforms the `@REPEAT("-",@CELLPOINTER"width")-1)` into a series of (eight) hyphens `--------` (if the column width is nine characters wide). This changes the content of the panel to display the following series of hyphens:

```
--------
```

Next, the macro issues the `{HOME}` macro command, which moves the cursor to the beginning of the panel, and types the apostrophe "`'`" to change it into a label. Lotus will not accept the series of hyphens without the apostrophe because the hyphen is a mathematical operator. The result in the panel is:

```
'--------
```

Then the macro issues the tilde "`~`" macro command which writes the content of the panel into the current cell.

Freezing and Releasing Screen and Panel Activities

Super Power extensively uses the `{WINDOWSON}`, `{WINDOWSOFF}`, `{PANELON}` and the `{PANELOFF}` macro commands to freeze and resume the screen and the panel display activities. These commands are probably the most used commands in ***Super Power*** because:

1. The macro looks more professional.
2. Time, is saved since a macro can work twice as fast when screen and panel activities are suppressed.
3. It controls the panel appearance so we can obtain special effects during some Lotus operations.

Using the Pointing Method

One of the most common way to write formulas is to use the pointing technique, which means pointing to the cell using the direction keys. For example, let us look at the following worksheet:

	A	B	C	D	E	F
1	100	200				
2						

To insert a formula in the C1 cell that adds the content of the A1 and the B1 cell, type the +A1+B1 formula in the C1 cell, or place the cell pointer on C1 and type the plus "+". At that point, the panel displays:

```
+
```

Press the LEFT arrow key twice to move the cell pointer to the A1 cell. The panel displays:

```
+A1
```

Type a second plus "+" and the panel displays:

```
+A1+
```

Again press the LEFT arrow key only once to move the cell pointer to the B1 cell and the panel displays:

```
+A1+B1
```

Press ENTER to finish the formula. This is a very powerful technique when used with the Lotus macro language. Here is an example of a macro using this technique:

	A	B	C	D	E
1	John Brice				
2					

To extract the first name from the name in the A1 cell and place it into the B1 cell use the following macro:

	A	B	C	D	E
1	\Z	@LEFT({LEFT},@FIND(" ",{LEFT},1))~			

Place the cell pointer on B1 and press ALT-Z to start the macro. When the macro starts, it first types the `@LEFT(` text into the panel because the macro processor does not find any command in this text. Then the panel displays:

```
@LEFT(
```

When the macro processor reaches the `{LEFT}` macro command and processes it, the cell pointer moves to the A1 cell and the panel displays:

```
@LEFT(A1
```

Next the macro types `,@FIND(" ",` (including commas and double quotes) and the panel displays:

```
@LEFT(A1,@FIND(" ",
```

Again the macro processor reaches to a second `{LEFT}` macro command and processes it. The cell pointer again moves to the A1 cell and the panel displays:

```
@LEFT(A1,@FIND(" ",A1
```

Last the macro processor types the next text `,1))` to the panel and the panel displays:

```
@LEFT(A1,@FIND(" ",A1,1))
```

The panel contains a formula which calculates the left part of the name in the A1 cell up to the space between the first and last name, which is, of course, the first name. All that remain is to press ENTER to write the formula into the current cell, therefore the macro issues the tilde "~" which is the macro equivalent of the ENTER key.

Here is another example of a macro which underlines a text with right aligned label composed of a series of "=" characters one character shorter then the label above.

```
     A          B          C          D          E
1              First_name
2               =========
```

For example, if the B1 cell contains the "First_name" label which is ten characters long, the resulting underline in the B2 cell will be a series of nine "=" characters as displayed.

```
     A          B          C          D          E
1 \Z       @REPEAT("+",@LENGTH(@STRING({UP},0))-1){CALC}{HOME}"~
```

Place the cell pointer on the B2 cell; the macro types the `@REPEAT("` text to the panel which displays:

```
@REPEAT("+",@LENGTH(@STRING(
```

The macro issues the `{UP}` macro command and the panel displays:

```
@REPEAT("+",@LENGTH(@STRING(B1
```

The macro continues and types the `,0))-1)` text to finish the formula and the panel displays:

```
@REPEAT("+",@LENGTH(@STRING(B1..B1,0))-1)
```

The result of this formula is a string of "=" characters as long a the length of the string in the B1 cell minus one as shown above. The `{CALC}` macro command transforms the formula into the `=========` text and then the macro issues the `{HOME}` macro command, which moves the cell pointer to the beginning of the panel and then types the double quotes to change it into a right aligned label.

The Enveloping Technique

This technique creates an enveloping function around the cell content to produce effective results which are cumbersome to use from the keyboard.

	A	B	C	D	E
1	-100				

For example, to calculate the absolute of the negative number in the A1 cell and replace it back in the A1 cell demands some work. To do it manually, you may insert the `@ABS(A1)` formula in the B1 cell and then move the content of the B1 cell into the A1 cell. A better way is to place the cell pointer on the A1 cell and press the EDIT (F2) key. The panel displays:

```
-100_
```

and the cursor is located at the end of the text in the panel (represented by the underscore). Now, press the closing parenthesis "`)`" without the double quotes, the panel displays:

```
-100)_
```

Issue the HOME key to move the cursor to the beginning of the text in the panel and the panel displays:

```
-100)
^
```

where the "^" under the minus "`-`" represents the cursor. Type the `@ABS(` text and the panel displays:

```
@ABS(-100)
```

which is the formula that calculates the absolute of the value in the current cell. Press the ENTER key. The equivalent macro is:

	A	B	C	D	E
1	\Z	{EDIT}){HOME}@ABS(~			

where the tilde "`~`" command is the macro equivalent of the ENTER key.

Here is a much more sophisticated implementation of the enveloping technique taken from the ERRPROOF.WK1 macro. This sample macro will tamper proof any formula that can produce an ERR and change the formula so that instead of the ERR, the result will be the zero "`0`" value.

	A	B	C	D	E
1	\Z	{BREAKON}			
2	ERRPROOF	{EDIT}{HOME}'~			
3	!	{LET TEMP016,@CELLPOINTER("contents")}~			
4	!	{EDIT}{HOME}{RIGHT}@IF(@ISERR({END}),0,{TEMP016}) {HOME}{DEL}~			
5	!				
6	TEMP016	+C8/D8			
7					
8		ERR			

The B8 cell contains the `+C8/D8` formula which results in ERR because the D8 cell is empty. Place the cell pointer on the B8 cell and start the macro. The result in the B8 cell will be zero "`0`" instead of the ERR. Let's see how the macro works. It starts with the `{EDIT}` macro command which causes the panel to display:

```
+C8/D8_
```

where the underscore represents the cursor position. Next the macro issues the `{HOME}` macro command and writes the apostrophe "`'`" at the beginning of the text in the panel and changes it from a formula into a label and the panel displays:

```
'+C8/D8
^
```

where the "^" represents the cursor position. Next the macro issues the tilde "`~`", the macro equivalent of the ENTER key, and writes the content of the panel into the B8 cell which now displays `+C8/D8`. Next the macro issues the `{LET TEMP016,@CELLPOINTER ("contents")}` macro command to store the content of the B8 cell into the B6 cell named [TEMP016] and force Lotus to update the content of [TEMP016] immediately.

Now the macro uses the content of [TEMP016] to replace the `+C8/D8` function in B8 with the `@IF(@ISERR(+C8/D8),0,+C8/D8)` function. For more details see the **Make Formulas Error Proof** section.

Moving Between Sheets in Straight Lines

In the 3-D releases of Lotus 1-2-3 (3.0/3.1/3.1+/3.4 and 123W), the cell pointer moves from one sheet to another using the CTRL-PGUP and CTRL-PGDN key combination, or in a macro using the {PS}, {PREVIOUSSHEET}, {NEXTSHEET} or the {NS} commands. However, the cell pointer doesn't move in a straight line. It can start in cell A1 in the "A" sheet and jump to the X10 cell in the "B" sheet to Z100 on the "C" sheet and so on. The 3-D releases of Lotus 1-2-3 were built so that the worksheet "remembers" the last location of the cell pointer in every sheet it passed earlier. When you page through the sheets manually or from within a macro, the cell pointer may move in a ZIGZAG path depending on the previous history of its last places in the different sheets.

This may be a bonus for you when you are using the keyboard, but it is a nightmare for macro programmers because they cannot use the {NS}, {NEXTSHEET}, {PREVIOUSSHEET} or the {PS} to move in a straight line in the Z direction. We have found that the cell pointer will move in a "straight path" when the worksheet is in the perspective mode and the syncro is ON. However, changing to this mode before a movement inside a macro can be very noisy (Lotus will beep) because the worksheet may already be in the perspective mode. Therefore, we have developed a safe and elegant way to move between sheets in our macros. We always record the cell address and then use the {NS} or {PS} to move to the other sheets and then use the indirect {GOTO} command to the recorded address. For example:

	A	B	C	D	E
1	\Z	{LET address100,@CELLPOINTER("address")			
2	!	{NS}{GOTO}{address100}~			
3	!				
4	address100	$F$5			
5					

Assuming the cell pointer is located on the F5 cell in the "A" sheet and we want to page to the "B" sheet and we also want the cell pointer to be on the F5 cell. However, the last time the cell pointer was on the "B" sheet, it was placed on the A1 cell. Therefore if you issue the CTRL-PGUP from the keyboard or the {NS} or the {NEXTSHEET} in a macro, the cell pointer will land on the A1 cell in the "B" sheet.

The macro first issues the {LET address100,@CELLPOINTER("address") macro command, which stores the current cell pointer position (the $F$5 address) in the B4 cell named [address100]. Then the macro issues the {NS} macro command moving the cell pointer to the next sheet; the "B" sheet, but the cell pointer goes to the A1 cell. The macro issues the indirect {GOTO}{address100}~ macro command moving the cell pointer to the F5 cell in the "B" sheet. Although there is a separate chapter for the indirect macro command, we will also explain the technique here. To visualize the process, let's see how the command works step by step.

If the cell pointer is now on the A1 cell in the "B" sheet, when the macro issues the {GOTO} macro command, the panel displays:

```
B:A1:
Enter address to go to: B:A1
```

When the macro reaches the {address100} routine command the macro processor

processes this command. However, the cell named [address100] contains only the $F$5 text; therefore Lotus writes (injects) the text into the panel which now displays:

```
B:A1:
Enter address to go to: $F$5
```

Last, the macro issues the tilde "~" macro command which is the macro equivalent of the ENTER key, and sends the cell pointer to the F5 cell in the current sheet which is the "B" sheet. Using this technique you can safely move the cell pointer in straight lines into and out of the worksheet in the Z direction.

Using Lotus Menus Inside a Macro

When a Lotus menu is activated from a macro, don't use the One key option to activate the menu item. Instead use the arrow keys to move the highlight to the desired menu option and press ENTER. When the first capital letter is pressed, Lotus still waits for the ENTER key to be pressed, which can be disastrous. In nested menus, Lotus may not returns to the previous menu; but will wait for the ENTER key. The macro returns to the wrong menu. To demonstrate the problem: when you want to set printing options, you issue the `/PPOO` keys and then Lotus displays the following menu:

```
As-Displayed  Cell-Formulas  Formatted  Unformatted
```

After selecting any one of the four menu options, Lotus displays the next menu:

```
Header  Footer  Margins  Borders  Setup  Pg-Length  Other  Quit
```

To return to the READY mode, select the **Q**uit option twice. Let's look at the following macro:

	A	B	C	D	E
1	\Z	/PPOO{?}~QQ			

The macro starts with the same `/PPOO` macro keys. Then it issues the `{?}` macro command which pauses the macro execution and allows you to select an option from the following menu:

```
As-Displayed  Cell-Formulas  Formatted  Unformatted
```

You can normally select a menu option by pressing the first capital letter or moving the highlight to the menu option and press ENTER. These two options behave the same when we use the keyboard; however in this macro, they behave completely differently. If you press the "U" or the "u" key to select the [**U**nformatted] menu option the macro will stop in the middle and will display:

```
Header  Footer  Margins  Borders  Setup  Pg-Length  Other  Quit
```

But will not return to the READY mode. If you use the arrow keys to move the highlight to the [**U**nformatted] menu option and press ENTER the macro works fine and quits to the ready mode. This bug occurs only in the built-in Lotus menus; but the custom menu is free from this bug. Therefore to overcome this limitation and to write a foolproof macro, we have to duplicate the built-in Lotus menu using a macro, so that the menu will appear to be a standard Lotus menu, but it will be a custom menu.

Rule: If you are using a macro or macro based application, NEVER USE the first capital letter of the menu option to select a menu option if the menu is one of the Lotus standard menus. ALWAYS use the arrow keys to highlight the menu option and press the ENTER key.

Duplicating Lotus 1-2-3 Menus in Macros

As we have seen in the previous chapter, the safest way to use Lotus standard menus from a macro is to duplicate them. Here is a duplicate menu to format dates.

```
     A              B              C              D              E
1 \Z            {MENUBRANCH format008}
2 !
3 format008     1 (DD-MMM-Y 2 (DD-MMM) 3 (MMM-YY) 4 (Long intn'l) 5 (Short
4 !             Lotus standaLotus standLotus standAs configured (MAs conf
5 !             /RFD1~~     /RFD2~~    /RFD3~~    /RFD4~~         /RFD5~~
```

Because we cannot display all the menu options clearly on one page, let's look at the first two:

```
     A              B                          C
3 format008     1 (DD-MMM-YY)              2 (DD-MMM)
4 !             Lotus standard long form   Lotus standard short form
5 !             /RFD1~~                    /RFD2~~
```

When the macro starts, it issues the `{MENUBRANCH format008}` macro command which activates the [format008] custom menu, appears exactly like the Lotus standard men

```
1 (DD-MMM-YY) 2 (DD-MMM) 3 (MMM-YY) 4 (Long Intn'l) 5 (Short Intn'l)  Time
```

When you select the second menu options, the macro issues `/RFD2~~` and formats the current cell to date format "2". However, if you try to save macro code and use the following macro:

```
     A              B              C              D              E
1 \Z            /RFD{?}~~
```

you will find that it works correctly only when you select the menu option using the arrow keys and the ENTER key. If you try to select the 4th menu option by pressing the "4" key, you will find that the macro waits for one extra ENTER to return to the READY mode.

The {MENUBRANCH} and {MENUCALL} Macro Commands

Super Power often uses the {MENUBRANCH} and the {MENUCALL} macro commands and they behave differently; therefore we will explain the difference and provide useful technique to improve and protect applications. To understand these command, let's look at the following simple custom menu macro.

	A	B	C	D
1	\Z	{MENUBRANCH format008}{BRANCH \Z}		
2	!			
3	format008	**H**ome	**E**nd_home	**Q**uit
4	!	Move home	Move to end of worksheet	Quit the macro
5	!	{HOME}	{END}{HOME}	{QUIT}
6	!	{MENUBRANCH format008}	{MENUBRANCH format008}	

This macro allows you to jump between the upper left cell of the worksheet and the lower right cell of the worksheet. When the macro starts it displays the following custom menu:

Home End_home **Q**uit

If you select the [**H**ome] menu option, the cell pointer moves to the upper left cell of the worksheet and then the macro displays the custom menu again. The {MENUBRANCH format008} macro command at the end of the menu option code re-activates the custom menu. Select the [**E**nd_home] menu option and the cell pointer moves to the lower right cell of the worksheet. Then the macro re-displays the custom menu because the {MENUBRANCH format008} command at the end of the menu option code re-activates the custom menu. If you select the [**Q**uit] menu option the macro quits. However, if you press ESC, the macro returns control to the {BRANCH \Z} macro command, which restarts the custom menu. Without the {MENUBRANCH format008} in the B6 and the C6 cells, the macro will end when you select the [**H**ome] or the [**E**nd_home] after the cell pointer moves.

Now let us replace the {MENUBRANCH format008} in the B1 cell and delete the {MENUBRANCH format008} in the B6 and the C6 cells. The macro is:

	A	B	C	D
1	\Z	{MENUCALL format008}{BRANCH \Z}		
2	!			
3	format008	**H**ome	**E**nd_home	**Q**uit
4	!	Move home	Move to end of worksheet	Quit the macro
5	!	{HOME}	{END}{HOME}	{QUIT}

Now when you press ESC, the macro does not quit. Select the [**Q**uit] menu option to quit the macro. However, if you select the [**H**ome] or the [**E**nd_home] menu options, the macro does not end after the cell pointer moves, instead the macro routes back to the {BRANCH \Z} macro command which restarts the macro. The {MENUCALL} command displays a macro menu that you entered in the worksheet. It waits for you to select an item from the menu, and then performs the macro instructions associated with the menu item as a subroutine.

When the instructions are finished, the macro returns control to the command that follows the {MENUCALL} command, in our example to the {BRANCH \Z} command, which re-activates the custom menu, thereby preventing you from quitting the macro. The {MENUBRANCH} command is like the {MENUCALL} command, except that when you select a menu item, 1-2-3 branches to the instructions associated with that item and executes them,

but does not return to the command which follows the {MENUBRANCH} macro command.

Using Custom Menus

To build a custom menu in a macro, see the following sample macro:

	A	B	C
1	\Z	{MENUBRANCH menu032}	
2	!		
3	menu032	**A**-drive	**B**-drive
4	!	Retrieve files from drive A:	Retrieve files from drive B:
5	!	/FR{ESC}{ESC}A:\~{NAME}{?}~	/FR{ESC}{ESC}B:\~{NAME}{?}~

This macro includes only two menu items the [**A**-drive] in the "B" column and the [**B**-drive] menu item in the "C" column. Lotus allows up to eight menu items in consecutive cells in the same row. In this example, we can add six more menu items and the third menu item will start in the C3 cell and downward, the fourth menu item will start in the D3 cell and downward and so on up to the "H" column. The rules to create a custom menu are:

- Select a location for the custom menu. In our example it is the B3 cell.
- Enter up to eight menu items in consecutive cells in the first row of location (the third row in our example) and leave the cell after the final menu item blank (the D3 is empty). Make sure each menu item starts with a different letter or number so you can select an item by pressing the first character.
- Enter the description for each menu item in the cell directly below the menu item. In our example the description for the [**A**-drive] menu item is "Retrieve files from drive A:" and the description for the [**B**-drive] menu item is "Retrieve files from drive B:". 1-2-3 truncates descriptions longer than the width of the window.
- Immediately below the menu-item descriptions (in the third row of the macro menu ange, the fifth row in our sample worksheet), enter the macro instructions that 1-2-3 performs if you select that menu item, the `/FR{ESC}{ESC}A:\~{NAME} {?}~` macro commands if you select the [**A**-drive] menu option, and the `/FR{ESC}{ESC}b:\~{NAME}{?}~` macro commands if you select the [**B**-drive] menu option.
- Assign a range name to the first cell in the custom menu range which is the B3 cell in the B3..C5 range. The name in our sample worksheet is the [menu032] name. To assign the range name, place the cell pointer on the B3 cell, then from the keyboard use the / **R**ange **N**ame **C**reate sequence (`/RNCmenu032~~`). A better way is to write the "menu032" string in the A3 cell as in our sample macro and then place the cell pointer on the A3 cell and use the / **R**ange **N**ame **L**abel **R**ight sequence (`/RNLR~`) to assign the [menu032] range name to the B3 cell.

To activate a custom menu, use the `{MENUBRANCH location}` or the `{MENUCALL location}` macro commands as we have seen previously. The `{MENUBRANCH location}` command displays the custom menu that starts in the first cell of location (the B3 cell in our example), and waits for you to select an item from the menu. The menu appears in the first line of the window and the description of the highlighted menu item appears in the second line. After you select an item from a macro menu that `{MENUBRANCH location}` activated, the macro control branches to the associated macro instructions in the third line of the macro menu. Because this is a branch, the macro control does not return to the original macro

location when 1-2-3 completes the macro menu instructions.

The {MENUCALL location} macro command displays the custom menu that starts in the first cell of location, B3 in our example, and waits for you to select an item from the menu. After you select an item from a macro menu that {MENUCALL location} activated, 1-2-3 performs the associated macro instructions as a subroutine, and the macro control returns to the original macro location when 1-2-3 completes the instructions. For more details, consult Lotus documentation on how to assign range names, how to build custom menus and how to use the {MENUBRANCH} and the {MENUCALL} macro commands.

Now let's see how our macro works.

	A	B	C
1	\Z	{MENUBRANCH menu032}	
2	!		
3	menu032	**A**-drive	**B**-drive
4	!	Retrieve files from drive A:	Retrieve files from drive B:
5	!	/FR{ESC}{ESC}A:\~{NAME}{?}~	/FR{ESC}{ESC}B:\~{NAME}{?}~

The macro starts with the {MENUBRANCH menu032} which activates the [menu032] custom menu of two menu items. If you look in the Lotus manual you will find that:

- Custom menus should include every menu item in a different column
- The first cell in every column holds the menu title
- The first character should be capitalized and different for every menu title
- The second cell contains the menu item description line
- The macro code that the menu activates starts at the third cell and downward
- Custom menu can have up to eight menu items

Let's look at the first menu item, the [**A**-drive]:

```
A-drive
Retrieve files from drive A:
/FR{ESC}{ESC}A:\~{NAME}{?}~
```

- The first cell, B3 holds the "**A**-drive" string
- The B4 cell holds the description line "Retrieve files from drive A:"
- The third cell contains the "/FR{ESC}{ESC}A:\~{NAME}{?}~" macro code

The B3 cell must have an assigned range name identifiable by the {MENUBRANCH location} or the {MENUCALL location} macro command. In this example, the location is the [menu032] range name, therefore the B3 cell must be assigned this range name for the macro to be able to operate. The {MENUBRANCH menu032} command activates the menu in the B3..C5 range and displays:

```
A-drive   B-drive
```

When you press the "A" character or highlight the [**A**-drive] menu item and press ENTER, the macro processor executes the code in the B5 cell. When you press the "B" character or highlight the [**B**-drive] menu item and press ENTER, the macro processor executes the code in the C5 cell.

Delete Range Names Safely Inside a Macro

When a macro deletes a range name, there is always the chance that the range name does not exist and an error can occur. To safely delete a range name during macro execution, the macro has to re-assign the range name to insure that such a range name exists before deleting it. If the range name exists, nothing will be changed if the same name is re-assigned to the range, but if the range name does not exist, the macro creates a new one at the current cell pointer location. Now the macro can safely delete the range name without the chance of error. Here is an example of such a macro.

	A	B	C	D	E
1	\Z	/RNCTEMP~~			
2	!	/RNDTEMP~			

Rule: When you plan a macro that needs to delete a range name, first create this range name and only then delete the range name. Thus, there is no chance of an error when the macro tries to delete the range name.

Create Range Names Safely Inside a Macro

One of the most used techniques in ***Super Power*** is the assignment of a range name to the range that you paint. Let's see what happens when you try to assign a range name to a selected range. If you want to assign the [temp] range name to the A1..C24 range, you have to issue / **R**ange **N**ame **C**reate sequence: `/RNCtemp,` press the ENTER key, type A1..C24 and press ENTER again. This is a simple procedure with which Lotus users should be familiar. However, if you try to assign the same range name again, but this time to the F100..I150 range, the steps you use are not the same. First, start with the `/RNCtemp`, then press the ENTER key as before, but now Lotus volunteers and highlight the A1..C24 range and offers it as the default. Continuing as before, will result in assigning the [temp] range name again to the A1..C24 range, while you actually want to assign the range name to the F100..I150 range. Therefore, press the BACKSPACE key to unanchor the cell pointer. Then paint the F100..I150 range. However, this procedure is not safe in a macro.

First we issue the `/RNCtemp~` macro code to re-assign the [temp] range name. If the [temp] range name already exists, this code renames it. If the [temp] range name does not exist, this code assigns it to the current cell. Now we can delete the [temp] range name, because either way it exists, and then we can re-assign it to the desired range. Here is the complete code:

	A	B	C	D	E
1	\Z	/RNCTEMP~~			
2	!	/RNDTEMP~			
3	!	/RNCTEMP~{?}~			

The `{?}` macro command allows you to paint the F100..I150 range. This macro technique is essential to many macros in ***Super Power***, so you will see it any time the macro processes a range.

Rule: When you plan a macro that needs to create a range name during its execution, create this name and delete the range name, then re-create the name again. This way there is no chance of an error that the name will be assigned to a different range.

Range Name as a Prompt

Lotus 1-2-3 allows a range name of up to 15 characters long, which allows you to assign meaningful range names that can serve as helpful prompts. The sophisticated macro programmer can use the range name to display an additional helpful prompt during the process of assigning a range name to a range. Let's look at the following macro code:

	A	B	C	D	E
1	\Z	/RNCWhich range ?~{?}~			

This macro code assigns the [Which range ?] range name and then issues the {?} macro command allowing you to paint the range which will be named the [Which range ?] range. When the macro reaches the {?} macro command the panel displays:

```
Enter name: Which range ?          Enter range: A1..A1
```

Wouldn't it be nice to get rid of the "Enter name:" text before the "Which range ?"

```
Which range ?                      Enter range: A1..A1
```

This prompt looks more professional and the "Which range ?" text acts as a prompt helping you understand what to do next. We could use "Paint range:" or any other meaningful range name to appear as a prompt. But how we can suppress the "Enter name:" text which is part of Lotus code. Here is the solution:

	A	B	C	D	E
1	\Z	{PANELOFF}/RNC{PANELON}Which range ?~{?}~			

The code starts with the {PANELOFF} macro command which freezes the panel display activity, while the macro issues the /RNC macro keys. Just before the macro types the "Which range ?" text, the macro issues the {PANELON} macro command which resumes the panel display activity. Now the panel displays:

```
Which range ?              Enter range: A1..A1
```

Exactly what we want. "Enter name:" is missing and the "Which range ?" string looks to you as a sole prompt. ***Super Power*** uses this technique combined with the previous technique in many macros.

Note: The 3-D releases of 1-2-3 are not compatible with the 2-D releases of 1-2-3. Therefore this will not work and the "Enter name:" will not be suppressed. It works only for Lotus 1-2-3 2/2.01/2.2/2.3 and 2.4. Since this is only an esthetic matter, we decided to use this code in all our macros. In the 3-D releases and the new 2-D releases of 1-2-3, we could use the new {INDICATE} macro command which allows us to create long messages in the first row of the screen, therefore an 80 character long string can be used as a prompt with the {INDICATE "Long ..... string"} macro command but this will force us to write a different code for different versions of Lotus 1-2-3.

To Find if a Range Name Exists

Many times we need to delete range names. We can always use the / **R**ange **N**ame **D**elete command sequence and manually delete the range names. Sometimes, we want to automatically delete all the range names in a range; therefore we need to know if the range contains cells which have range names. This is especially important when erasing part of a worksheet and there are range names in it. Trying to find which cell has a range name is a very time consuming process. In all ***Super Power***'s macros the "A" column contains the names of the cells to the right. Therefore, we can use the content of the cells in "A" column to automatically delete the range names listed in the "A" column using a macro. If you assign range names as we do and use the **/ R**ange **N**ame **L**abel **R**ight sequence, you can use a macro to delete the range names. Here is a macro using the {ONERROR} command to determine whether the range name exists and then delete it. For example, lets us look at the following worksheet:

	A	B	C	D	E	F
1	\a	{LET here283,@CELLPOINTER("address"}~				
2	!	{LET place283,@CELLPOINTER("contents"}~				
3	!	{err1283}{GOTO}{place283}~				
4	!	/RND{place283}~{GOTO}{here283}~				
5	!					
6	err1283	{ONERROR err2283}				
7	!					
8	err2283	{RESTART}{GOTO}{here283}~{QUIT}				
9	!				budget	
10	place283	budget			income	
11	!					
12	here283	$F$9				

This sample worksheet contains a macro in the "A" and the "B" columns and two labels in the F9..F10 range. If you place the cell pointer on the F9 cell and start the macro, it will try to move the cell pointer to the cell with the [budget] range name. If the range name does not exist, an error occurs and the macro returns the cell pointer to the F9 cell. If the name exists, the macro deletes the range name and returns the cell pointer to the F9 cell. Let's see how the macro works.

The macro starts with the {LET here283,@CELLPOINTER("address"} macro command, which stores the current cell pointer address (the $F$9 address) in the B12 cell named [here283], then the macro issues the tilde "~" macro command, which forces Lotus to update the content of the worksheet immediately to reflect the change in the B12 cell. Next the macro issues {LET place283,@CELLPOINTER("contents"} which stores the current cell pointer content (the "budget" string) in the B10 cell, [place283], then the macro issues the tilde "~" to immediately update the content of the worksheet to reflect the change in the B10 cell.

Next the macro uses the {err1283} routine command which activates the {ONERROR err2283} macro command to be prepared to trap and handle errors that might occur if the [budget] range name does not exist. Now the macro issues the {GOTO}{place283}~ indirect macro command. To understand this, look what happens in the panel.

When the macro issues the {GOTO} macro command the panel displays:

```
Enter address to go to: F9
```

Next the {place283} routine command activates the [place283] routine; however this routine contains only text, the "budget" text. Lotus writes the "budget" string into the panel and offers it as the address to go to and the panel displays:

```
Enter address to go to: budget
```

Last, the macro issues the tilde "~" which is the equivalent of the ENTER key to complete the GOTO command. If the [budget] range name does not exist, then Lotus issues an error and the error handling routine issues {RESTART}{GOTO}{here283}~{QUIT} to send the cell pointer back to the F9 cell and quits. However, if the [budget] range name does exist, the cell pointer will move to the cell [budget] wherever it is. Then the macro issues the /RND{place283}~ indirect macro command which deletes the budget range name (the same technique as the previous GOTO command) and last the macro issues the indirect {GOTO}{here283}~ macro command which sends the cell pointer back to the F9 cell.

Another fine example of using the @@() function to find if a range name exists you can find in the UNNAME.WK1 macro:

```
     A            B                C          D          E          F
 1 \Z          {LET rangname166,@CELLPOINTER("contents")}~
 2 !           {LET numcheck166,@@(rangname166)}~
 3 !           {IF #NOT#@ISERR(numcheck166)}{BRANCH delete166}
 4 !
 5 rangname166 budget
 6 !
 7 numcheck166      ERR
 8 !
 9 !                                                  budget
10 !                                                  income
```

Let's again assume the cell pointer is on the F9 cell containing the "budget" string. The macro starts with {LET rangname166,@CELLPOINTER("contents")} which stores the current cell pointer content (the "budget" string) in the B5 cell, [rangname166]. Then the macro issues the tilde "~" command forcing Lotus to immediately update the content of the worksheet to reflect the change in the B5 cell. Next, the macro issues {LET numcheck166,@@(rangname166)}, which stores the content of [budget], the address that the content of [rangname166] refers to. Recall that the cell [rangname166] holds the cell pointer content which is the "budget" string, therefore the @@(rangname166) function expects to return the content of the cell whose name is [budget]. Therefore, if the [budget] is not a range name, the @@() function returns an error. Therefore the cell [numcheck166] will contain the ERR value.

The macro issues the {IF #NOT#@ISERR(numcheck166)} macro command to check if the cell [numcheck166] contains the ERR value or not. If the condition is true and the cell named [numcheck166] contains a label, it means that a [budget] range name exists; therefore the macro issues the {BRANCH delete166} command to initiate the routine which deletes the range name (the routine's code is not listed here). This is a special example using the @@() function to determine if a range name exist or not.

If all we want is to delete the range names, not just to find out if they exist, then there is a much simpler method.

```
     A            B                C          D          E          F
 1 \Z          {LET temp150,@CELLPOINTER("contents")}~
```

```
2 !              {LET tempa150,@CELLPOINTER("contents")}~
3 !              /RNC
4 temp150        budget
5 !              ~~
6 !              /RND
7 tempa150       budget
8 !              ~
9 !                                                  budget
```

This is a brute force technique. It says why bother to find out if the [budget] range name exists? let's assign it to the F9 cell and then delete it. This way we avoid checking if the file name exist. We assume that the cell pointer is on the F9 cell.

The macro starts with the `{LET temp150,@CELLPOINTER("contents")}~` commands which store the contents of the F9 cell (the "`budget`" string) in the B4 cell, [temp150]. Then the macro again issues `{LET tempa150,@CELLPOINTER ("contents")}~`, which store the "`budget`" string the B7 cell, [tempa150]. If you look at the code in the B3..B9 range, it sums up to the `/RNCbudget~~/RNDbudget~` macro code, which creates the [budget] range name and then deletes it. Also notice how the macro stores the "`budget`" string in the correct position so that the "`budget`" string becomes part of the macro code.

Note: This is a classic example of dynamic macro programming, in which the macro changes or inserts the code in the cells before it processes the code in those cells. The Lotus macro language is an interpreter, meaning that it processes the code in sequence, which allows us to change the code in the cells before the macro processor reaches their code.

We have used the last three types of code in the macros listed in ***Super Power***; but until now, we did not have the ultimate solution to the [budget] scenario. During the process of explaining the last three examples, we have found the solution to the most general case. Here's a macro for it.

```
          A              B              C          D          E          F
 1 \Z             {LET here100,@CELLPOINTER("address")}~
 2 !              {GOTO}name100~
 3 !              +{here100}{DOWN}{UP}{EDIT}{ABS}{HOME}{DEL}'~
 4 !              {IF name100<>@RIGHT(here100,@LENGTH(name100))}
                  {namedel100}
 5 !              {GOTO}{here100}~
 6 !
 7 here100        $F$9
 8 !
 9 namedel100     /RND
10 Name100        budget
11 !              ~
```

Let's assume that the [budget] range name exists and it refers to the F9 cell, but there is no text there. The macro starts with the `{LET here100,@CELLPOINTER ("address")}~` macro commands which store the $F$9 string address in the B7 cell, [here100]. Next the macro issues `{GOTO}name100~` which moves the cell pointer to the B7 cell, [here100]. Now the macro writes the plus "+" sign into the panel to start a formula. Therefore the panel displays only the plus sign:

```
+
```

Next the macro issues the {here100} routine command which writes the content of [here100] into the panel following the plus "+" sign. Therefore the panel displays:

```
+$F$9
```

Now the macro issues {DOWN}{UP} to write the content of the panel into the B7 cell, and then issues {EDIT} to enter the EDIT mode. Next the macro issues the {ABS} command which is the same as the F4 key from the keyboard which changes the display in the panel to the following:

```
+budget
```

Next the macro issues the {HOME} command to move the cursor to the plus "+" sign and then issues {DEL} to delete the plus "+" sign. Therefore the panel displays:

```
budget
```

Last the macro issues the tilde "~" command which writes the "budget" label into the B7 cell. This way we have found the range name of the F9 cell. Since the purpose of our example is to show a practical use for this technique, the macro continues with the code to delete the [budget] range name. The manipulation in the panel that we have just seen will result with the range name if it exists; however if it does not exist, the result will be only the address. Therefore the macro continues with the following code:

	A	B	C	D	E	F
4	!	{IF name100<>@RIGHT(here100,@LENGTH(name100))} {namedel100}				

To check if the current cell has a range name, the macro compares the content of cell [name100] with the right part of cell [here100]. The reason is: if the cell has no range name, then the result of the previous manipulation is the address less the first "$" character of the absolute address (F$9 only because of the {ABS} macro command). Therefore this portion of the address is equal to the right part of the content of [here100] which holds the complete $F$9 address. If the condition is true (because the "budget" label is not equal to the "$F$9" label), then the cell has a range name. Therefore the macro issues the {namedel100} routine command which starts the [namedel100] routine deleting the range name and issues the {GOTO}{here100}~ indirect macro command returning the cell pointer to the F9 cell. If the current cell does not have a range name, the macro issues the {GOTO}{here100}~ indirect macro command returning the cell pointer to the F9 cell.

	A	B	C	D	E	F
9	namedel100	/RND				
10	Name100	budget				
11	!	~				

The [namedel100] routine is a simple /RNDbudget~ code which deletes the [budget] range name. However, notice that the content of the B10 cell, [Name100], was entered by the macro during its execution, before the macro processed this code, again a case of dynamic macro coding.

In the 3-D releases of 1-2-3, the @ISRANGE(range) function can test for a defined range name or valid range address, but to keep compatibility with older versions we used the previous techniques that work with all the Lotus releases from 2.0 and up.

Pick a Range Name from the Screen

	A	B	C	D	E	F
1	\Z	{LET here1155,@CELLPOINTER("address")}~				
2	!	{GOTO}{NAME}{NAME}{?}~				
3	!	{LET loc1155,@CELLPOINTER("address")}~				
4	!	{LET new1155,+""&@CELL("contents",loc1155)}~				
5	!	{GOTO}new1155~{EDIT}{HOME}{DEL}{ABS}~				
6	!	{GOTO}{here1155}~/Cnew1155~~				
7	!					
8	here1155					
9	!					
10	loc1155	$D$12				
11	!					
12	new1155					
13	!					
14	test155					

When you issue the F5 (GOTO) key or the {GOTO} macro command or the /RNC or the /RND macro keys in 1-2-3, Lotus displays a list of all the range names in the worksheet. However, Lotus does not offer a macro command or a function to "grab" a range name from the screen display which we could use to add a point and shoot options for other tasks beyond the {GOTO}, /RNC or /RND. This macro demonstrates how we can pick a range name from the screen and record it into the current cell position. This is a very powerful technique that we can use in automated macro driven applications. For example let's assume that the cell pointer is on the D12 cell of this sample worksheet, and we want to capture the [test155] range name and place it in the D12 cell.

First the macro issues the {LET here1155,@CELLPOINTER("address")} macro command, which stores the current cell pointer address (the D12 address) in the B9 cell, [here1155]. Next, the macro issues {GOTO}{NAME}{NAME}, which displays a full screen list of all the range names in the worksheet, if the worksheet contains only what is listed above, the screen will display:

```
Enter address to go to:
    HERE1155            B8
HERE1155        LOC1155         NEW1155     TEST155     \Z
```

The range names are [here1155], [loc1155], [new1155], [test155], [\Z]. If the worksheet contains more range names they would also be displayed. As we previously said, we want to "grab" a range name and place it in the D12 cell. Next the macro issues {?} to pause and wait for you to point to the name of the range to pick. When you point to a range name (the [test155] range name for example) and press the ENTER key, the macro resumes control and finishes the GOTO process, moving the cell pointer to the B14 cell, [test155]. Now the macro issues the {LET loc1155,@CELLPOINTER("address")} command, which stores the $B$14 address in cell [loc1155]. The content of [loc1155] which is the $B$14 address of cell [test155] is now used by the macro to extract the actual name of the B14 cell in a kind of a tricky way.

The macro issues the {LET new1155,+""&@CELL("contents",loc1155)}~ commands, which build the following:

```
'$B$14
```

label in the cell named [new1155]. Next the macro issues {GOTO}new1155~ which puts the

cell pointer on the B12 cell, [new1155], and issues {EDIT} to enter the EDIT mode. Then the panel displays:

```
'$B$14_
```

where the cursor is marked by the underscore. Now the macro issues {HOME} sending the cursor to the very beginning of the panel, which displays:

```
'$B$14
^
```

where the cursor is marked by the "^" under the apostrophe "'" character. Next the macro issues {DEL}, deleting the apostrophe "'" character, and converting the text in cell [new1155] into the

```
$B$14
```

active address. Now the macro issues the {ABS} macro command changing the text in the panel to display:

```
test155
```

Finally, the macro issues the tilde "~", which is the equivalent of the ENTER key. The result is that the content of cell [new1155] was changed to include the "test155" string which is the name of the range you picked from the screen. This is the kind of unconventional trick that drives macro developers to love the Lotus macro language and dig deeper into it. It is highly recommended that you repeat the process manually to see how it evolves.

All that is left to finish the task is to return the cell pointer to its point of origin (the D12 cell) and to copy the range name to that position. To achieve this goal, the macro issues the {GOTO}{here1155}~ indirect command, which puts the cell pointer where it was before we invoked the macro. This is uses what we call the "indirect macro command" which makes use of the {here1155} routine command to write the content of the cell [here1155] into the panel in response to the {GOTO} command. Next the macro issues the /Cnew1155~~ macro code which copies the range name (the content of [new1155]) into the current (the D12 cell).

Using Dynamic Code

The Lotus macro language is an interpreter, the macro processor evaluates and executes the macro instructions in consecutive order, while a compiler first evaluates all the code and then transforms it into a more compact and fixed object. When we use an interpreter, we can design the program to change its code while it is running before it processes this code. This makes the interpreter a very flexible programming language. Because the Lotus macro language is not as rich in commands as other high level languages such as BASIC, PASCAL etc., we need to use this property of the interpreter to achieve performance available in high level computer languages. To demonstrate how we create a dynamic code in a macro let us look at the following simple macro to combine a file.

	A	B	C	D	E	F
1	\Z	{GETLABEL "Enter the name of the file to combine", name100}~				
2	!	{comb100}				
3	!					
4	comb100	/FCCE				
5	name100	budget				
6	!	~				

When you start the macro it issues the

```
{GETLABEL "Enter the name of the file to combine",name100}
```

macro command which displays the

```
Enter the name of the file to combine
```

message in the panel. For example, type the "budget" file name and press ENTER, Lotus stores the name in the B10 cell, [name100], and the macro issues the tilde "~" to force Lotus to update the content of [name100]. Next the macro issues {comb100} activating the [comb100] routine which starts at the B9 cell. The [comb100] routine includes the following /FCCEbudget~ code. Recall that the {GETLABEL} command stored the "budget" label in B10, [comb100], just before the macro issued the {comb100} routine command, thereby dynamically altered the macro code to allow you to combine the BUDGET.WK* file. This form of dynamic macro code is very popular among professional macro programmers.

Using Formulas to Create Dynamic Code

In the previous chapter, we showed one technique to create dynamic code in Lotus macros. Here is a second way to create dynamic code using string formulas, a much more powerful technique. To demonstrate the technique, let us alter the previous example to the following:

	A	B	C	D	E	F
1	\Z	{GETLABEL "Enter the name of the file to combine", name100}~				
2	!	{RECALC comb100}				
3	!	{comb100}				
4	!					
5	comb100	/FCCEbudget~/FEbudget~				
6	!					
7	name100	budget				

This macro allows you to combine a file and then delete the file from the disk. The macro starts with the same

```
{GETLABEL "Enter the name of the file to combine",name100}
```

macro command as in the previous chapter which displays:

```
Enter the name of the file to combine
```

in the panel. When you type the "budget" file name and press ENTER, Lotus stores the name in the B7 cell, [name100], and then the macro issues the tilde "~" command to force Lotus to update the content of B7, [name100]. Next the macro issues {RECALC comb100} which updates the dynamic string formula in cell [comb100]. The /FCCEbudget~/FE budget~ code you see in the B5 cell named [comb100] is the result of the following string formula:

```
+"/FCCE"&B7&"~"&"/FE"&B7&"~"
```

The formula combines the "/FCCE" text with the content of the B7 cell (containing the file name), the "~" character, the "/FE" text, repeats the content of the B7 cell and the "~" character. When the content of B7 is the "budget" label The result of this formula is the

```
/FCCEbudget~/FEbudget~
```

macro code in the B7 cell of the macro listing.

If you type another file name such as "sales92" the result of the formula will be:

```
/FCCEsales92~/FEsales92~
```

Using the file name two or more times in the same code line is easily accomplished just by adding it twice to the same formula. ***Super Power*** often uses this type of formula to create sophisticated dynamic codes.

Printing to File or Printer with Confidence

When you try to print a worksheet into a file or the printer using the keyboard, you can press the first character "R" to select the [**R**ange] menu option in the Lotus print menu or to highlight the [**R**ange] menu option and press ENTER. If you want to print into a file you have to assign a file name. However, if the file name already exists, then you need to choose the [**R**eplace] menu option, pressing the "R" character the second time. If the file name is new, the [**R**eplace] menu option does not appear so the "R" character will be pressed only once. It is not a problem when you use the keyboard but how can we make the macro know when to issue the "R" command once or twice?

To handle both cases, we issue two range names to the same range that we want to print. The first is the [area] range name, and the second is the [Rarea] range name. When the file is new, the macro thinks the range you want to print is the [Rarea], but when the file name already exists, Lotus understand the "R" character before the [area] name as the [**R**eplace] menu option.

	A	B	C	D	E	F
1	\Z	/PF{?}~RRarea~GQ				

To summarize: you need to assign two range names to the range you want to print. The second range name has an additional "R" character before the first range name. In this example, the first name is the [area] name and the second name is the [Rarea] range name. When the file name is new, the first "R" is for the [**R**ange] menu option and the macro prints the [Rarea] range. When the file name is old the first "R" is for the [**R**ange] menu option and the second "R" for the [**R**eplace] menu option and then the macro prints the [area] range.

Embedding a Printing Setup String in the Worksheet

Lotus allows the use of embedded printing commands in a worksheet. The printing command string uses the "||" as a prefix. It has to be included within the printed range, but Lotus does not print the setup string, instead it sends the setup string to the printer. Let's assume that:

	A	B	C	D	E	F
1	|\027\015					
2	This text will be printed as a compressed text					
3	|\027					
4	This text will be printed as a normal text					

The A1 cell contains the "||\027\015" text. The first bar character is not shown in the cell because Lotus uses it as a prefix to know that the following text in the same cell is not to be printed. To insert a local setup string for printing, you need to enter the second bar followed by the setup string. In the A1 cell, the setup string is the "\027\015" string which instructs an Epson compatible printer to print the text in the cells below the A1 cell at the compressed mode. The setup string in the A3 cell sets the text back to normal; therefore the text below the A3 cell will be printed at the default mode.

The fact that Lotus does not print a text preceded by "|" prefix can be used to insert notes into the worksheet that you do not want Lotus to print with the rest of the text.

Recording Cell Pointer Position Smartly

For a long time life was simple, There was only Lotus 2.0/2.01, Then came Lotus 2.2, but still was it a 2-D worksheet, Then came Lotus 3-D. To identify a cell in the 3-D releases, it is not enough to state the address, you need to state the sheet where the address is, for example C:A1..A1 is the A1..A1 address in the "C" sheet. Lotus 3.0 and up introduced the "coord" attribute so we can use a formula like `@CELLPOINTER("coord")` which returns the sheet and the address, such as A:B21..B21 meaning the address B21..B21 in the "A" sheet.

When we design a macro that should work in the 3-D releases of Lotus 1-2-3, we need to use the `@CELLPOINTER("coord")` function. In the 2-D worksheet, we need to use the `@CELLPOINTER("address")` function. When we want the macro to work transparently in both types to Lotus 1-2-3, the macro has to check the type of worksheet using the `@INFO("release")` function. The `@INFO("release")` function is a valid function only in the 3-D releases of Lotus 1-2-3. It will produce different results in a 2-D release or a 3-D release. Let's look at the following macro.

	A	B	C	D	E	F
1	\Z	{LET release123,@INFO("release")}~				
2	!	{RECALC position123}{RECALC store123}				
3	store123	{LET here123,@CELLPOINTER("coord")}~				
4	!					
5	position123	coord				
6	!					
7	release123	'3.00.00				
8	!					
9	here123	'C:H20..H20				

The content of the B5 and the B3 cells are the result of the following string formulas:

```
3 store123        +"{LET here123,@CELLPOINTER("""&B5&""")}~"
5 position123     @IF(@LEFT(B7,1)<>"@","coord","address")
```

The macro starts with the `{LET release123,@INFO("release")}` macro command which writes the result of the `@INFO("release")` function in the B7 cell named [release123]. If you use a 3-D release of Lotus 1-2-3, then this command stores the Lotus release in the A1 cell. For example, if you use version 3.0, the result in the B7 cell is the '3.00.00 label. However, if you use a 2-D release the result is the '`@INFO("release")` label.

Note: When Lotus recognizes an unknown function in the `{LET}` macro command, it stores it as text. For example if you mistakenly write `{LET A1,@SUN(B1..H20)}` the result in the A1 cell will be the `'@SUN(B1..H20)` label.

Next the macro issues the tilde "~" command which is the macro equivalent for the ENTER key to force Lotus to update the content of the B7 cell, and then issues `{RECALC position123}{RECALC store123}` to update the dynamic string formulas in the B5 and B3 cells named [position123] and [store123] respectively. Before we continue let us see the formula in the B5 cell:

```
@IF(@LEFT(B7,1)<>"@","coord","address")
```

This formula returns the "coord" label if the first character of the label in the B7 cell is not the "@" character which means that you are using a 3-D Lotus release. If "@" is the first character of the label in the B7 cell, than you are using a 2-D Lotus release and the result of this formula is the "address" label. The second formula in the B3 cell is:

```
+"{LET here123,@CELLPOINTER("""&B5&""")}~"
```

This uses the result of the formula in the B5 cell to create the correct code for the Lotus release that you are using. If you are using a 3-D release, the resultant code of this formula is:

```
{LET here123,@CELLPOINTER("coord")}~
```

If you are using a 2-D release, the resultant code is:

```
{LET here123,@CELLPOINTER("address")}~
```

Now that all the formulas are updated, the macro issues `{LET here123,@CELLPOINTER ("coord")}` which stores the result of the current cell pointer position in the B9 cell, [here123]. In our sample macro, the B5 cell contains the "coord" label, therefore the relevant code in the B3 cell stores the coordinate of the cell pointer in the B9 cell. If the cell pointer is on the H20 cell of the "C" sheet, the result is the `'C:H20..H20` label in the B9 cell. ***Super Power*** often uses this macro technique where the macro needs to record the cell pointer position to return to later.

Handling Manual and Automatic Recalculation Smartly

When we build a macro, we have to account for working in **A**utomatic recalculation mode or **M**anual recalculation mode. The macro should update every formula that the macro directly affects. After every command that changes a cell content, the macro must force Lotus to update the cell or the formula. There are three ways to update the cell:

- First, we can use the `{CALC}` macro command, the macro equivalent of the F9 key. This is not always the best way, because if the worksheet is large and contains many formulas, this command will also update them. This may take a long time and considerably slow the macro.

- The second, and preferred way, is to use the `{RECALC range}` macro command which has a local effect. It recalculates only the cells in the range specified inside the `{RECALC}` command.

- The third way is to issue the tilde "`~`" macro command after the `{LET}` macro command, which forces Lotus to update the cell affected by the `{LET}` macro command.

Note: ***Super Power*** often uses the tilde "`~`" macro command, especially after the `{LET}` macro command. This is necessary even in the **A**utomatic **R**ecalculation mode. For example, if the macro issues the `{LET A1,@CELLPOINTER("address")}` macro command, although this command changes the contents of the A1 cell, 1-2-3 does not automatically recalculate the `@CELLPOINTER("address")` formula after executing the `{LET}` command when the worksheet recalculation is set to Automatic. To force recalculation after the `{LET}` command, you need to follow the command with the tilde "`~`" or the `{CALC}` macro command.

The sample macro in the previous chapter uses the second and third technique.

Copy a Cell Using Adjacent Column Length

	A	B	C	D	E	F
1	+B1+C1+D1	100	200	300		
2		200	500	100		
3		250	400	150		
4		150	300	200		
5		290	550	180		
6		200	100	600		
7		300	700	800		
8		400	200	400		
9		600	900	300		

If you have that above sample worksheet with the +B1+C1+D1 formula or any other formula and you want to copy the formula in the A1 cell into the A2..B9 range, you issues / **C**opy, then you press ENTER, and then you press the period [.] key to anchor the cell pointer. Paint the A1..A9 range and press ENTER. This is simple when the data is in the limited B1..D9 range, But what will you do if the data is included in the B1..D2000? Will you use the direction keys to paint the A1..A2000 range?

There is a much better and faster way. Use the length of the data in the "B" column as long as it is contiguous. You can issue: / **C**opy and press ENTER, next press the period [.] key to anchor the cell pointer to the A1 cell and press the RIGHT arrow key. Then Lotus highlights the A1..B1 range. Now press the END key followed by the DOWN key. Now Lotus highlights the A1..B2000 range. Next press the LEFT key limiting the highlight to the A1..A200 and press ENTER. Lotus will copy the formula in the A1 cell to the A2..A2000 range.

We have used the length of the "B" column to copy the formula in the A1 cell to the same length in the "A" column. This technique is necessary when we use a macro to automatically copy formulas or cells to match the length of the adjacent column or row.

	A	B	C	D	E	F
1	\Z	/C~.{RIGHT}{END}{DOWN}{LEFT}~				

If you place the cell pointer on the A1 cell in the above sample worksheet and then start this macro, it will do the same thing. The macro starts with the /C~ macro keys, issues the period [.] key, then the RIGHT END DOWN LEFT keys and then the tilde "~", which is the equivalent of the ENTER key. ***Super Power*** uses this technique to copy formulas or erase data or fill data.

Monitoring and Controlling User's Input

A good macro design or a macro based application design should limit the user only to the keys that the programmer plans to allow him to use. Therefore the macro has to monitor and control every key that the user uses and respond respectively. It is possible to limit the user to use only the keys we want, or to control macro flow based on the user key presses. Let's look at the following simple macro:

	A	B	C	D	E	F
1	\Z	{GET key102}~				
2	!	{key102}				
3	!	{\Z}				
4	!					
5	key102	{DOWN}				

This macro demonstrates how we can capture every key you press. The macro begins with the `{GET key102}` macro command. This command causes Lotus to pause and wait until you press a key. When a key is pressed, Lotus stores the key's macro code in the B5 cell named [key102]. If you press the DOWN arrow key, Lotus stores the `{DOWN}` macro command in B5, [key102]. Next the macro activates the `{key102}` routine command, which contains only the macro code of the key you just pressed (the `{DOWN}` macro code), therefore the macro executes this command and the cell pointer moves one cell down. Next the macro issues `{BRANCH \Z}` which routes macro control to the first cell of the macro and starts all over again.

If you press another key, Lotus stores it in the B5 cell and the `{key102}` routine command executes this key again. This looks like the normal response when using the keyboard; but this time there is a delay, the macro first records the key and then executes it. If we add a code to the macro that checks the keys the macro stores, then we can write a code that executes only the keys we want. For example:

	A	B	C	D	E	F
1	\Z	{GET key102}~				
2	!	{IF key102="{DOWN}"#OR#key102="{ UP}"}{key102}				
3	!	{IF key102="{ESC}"}{QUIT}				
4	!	{BRANCH \Z}				
5	!					
6	key102	{DOWN}				

This macro has no practical value, but it demonstrates the idea. The macro accepts only UP and DOWN arrow keys. If you press any other key, the macro issues `{BRANCH \Z}` and starts again. Only when you press the ESC key does the macro issue the `{QUIT}` command and quits.

Here is a nice example from the ERASENDS.WK1 macro:

	A	B	C	D	E
1	move504	Use the arrow keys to move the cell pointer and press RETURN:			
2		{GET key504}{ESC}~			
3	!	{IF key504="~"}{BRANCH routine504}			
4	!	{IF key504="{PGDN}"#OR#key504="{PGUP}"#OR#key504="{DOWN}"#OR#key504="{UP}"#OR#key504="{LEFT}"#OR#key504="{RIGHT}"}{key504}			
5	!	{BRANCH move504}			
6					

```
 7 routine504    ........
```

This macro combines a prompt in the panel with control of the keys you use. The macro writes:

```
Use the arrow keys to move the cell pointer and press RETURN:
```

and immediately follows it with the {GET key504} macro command which pauses the macro execution and waits until you press a key. When you press a key, the macro issues {ESC} clearing the message from the panel before Lotus writes it into the current cell and damages the worksheet. Next, the macro issues {IF key504="~"} to check if you pressed the ENTER key. Then the macro issues {BRANCH routine504} which routes macro control to the [routine504] routine. If ENTER was not pressed the macro issues the

```
{IF key504="{PGDN}"#OR#key504="{PGUP}"#OR#key504="{DOWN}"#OR#
    key504="{UP}"#OR#key504="{LEFT}"#OR#key504="{RIGHT}"}
```

compound IF command. If you press one of the PGDN, PGUP, DOWN, UP, LEFT or RIGHT arrow keys, the macro issues the {key504} routine command. However this routine contains only the macro code of the key that you just pressed. This macro command is executed and the cell pointer moves accordingly. If you press any other key, the macro issues {BRANCH move504} and starts again. This macro accepts only the direction keys and the ENTER key.

Here is a another example from the FILEMNGR.WK1 macro.

```
          A              B                C              D              E
 1 loop522      Use arrow keys to move and press [RETURN] to execute
                or [ESC] to quit..{GET key522}{ESC}
 2 !            {IF key522="{UP}"#OR#key522="{RIGHT}"}{UP}{BRANCH loop522}
 3 !            {IF key522="{DOWN}"#OR#key522="{LEFT}"}{DOWN}{BRANCH loop522}
 4 !            {IF key522="{PGUP}"#OR#key522="{HOME}"}{PGUP}{BRANCH loop522}
 5 !            {IF key522="{PGDN}"#OR#key522="{END}"}{PGDN}{BRANCH loop522}
 6 !            {IF key522="{ESC}"}{QUIT}
 7 !            {IF key522="~"}{BRANCH continue522}
 8 !            {BRANCH loop522}
 8 !
 9 key522
10 !
11 continue522 ........
```

In this macro, the key you press is stored in the B10 cell named [key522]. If you press the UP or the RIGHT keys, the macro issues the {UP} macro command and moves the cell pointer one cell up. If you press the DOWN or the LEFT keys the macro issues the {DOWN} macro command and moves the cell pointer one cell down. If you pressed the PGUP key or the HOME key, the macro issues the {PGUP} macro command and moves the cell pointer one page up. If you pressed the PGDN key or the END key, the macro issues the {PGDN} macro command and moves the cell pointer one page down. If you press the ESC key, the macro issues the {QUIT} macro command and quits. If you press the ENTER key, the macro issues the {BRANCH continue522} macro command and routes to the [continue522] routine. If you press any other key, the macro issues the {BRANCH loop522} macro command and starts all over again.

Cleaning After Macros

It is good practice to leave as few traces as possible after using a macro. Usually a macro comes with range names that must be defined before the macro can be used. When the macro is finished, the range names stay with the worksheet even if you erase the macro. As you use more macros, more range names are added to the worksheet making it bigger and slower. Therefore, we have tried to clean after every macro. The macros include the code to erase the temporary range names that the macro itself creates during execution. The range names that come with the macro can be deleted using the UNNAME.WK1 macro. When the macros are used from the macro manager, the macro manager does the cleaning job and even cleans itself.

Quitting Without the {QUIT} Macro Command

Lotus includes a macro command that can force a quit from a macro during macro operation. This command is the `{QUIT}` macro command. However this quitting technique is not always suitable. If another macro (like a macro manager) activates this macro and this macro uses the `{QUIT}` macro command, Lotus will quit the macro and the macro processor will not branch back to the calling macro. ***Super Power*** uses a replacement technique. This is the branch to an empty routine/cell quit technique. When we force a macro to branch into an empty cell or empty routine, the macro quits. However, if another macro called this macro, the macro processor returns control to the calling macro. If another macro (like a macro manager) activated this macro, the macro processor will route back to the macro manager. A second option is to issue the `{RETURN}` macro command which quits the macro and returns to the calling macro.

Advanced Use of the {LET} Macro Command

Often we can use the {LET} macro command to accomplish many tasks beyond the simple {LET location,entry} type of command where the entry can be a label or a formula, as shown in Lotus documentation. Where we use the {LET} macro command is equally as important as the way we use it. Let's look at the following macro:

```
     A              B                C          D          E          F
1 \Z            /PPOS*{ESC}{LET string140,@CELLPOINTER("contents")}
                {string140}~
2 !
3 string140     \027\015
```

This macro will use the setup string text in the current cell to assign a new setup string without the need to manually type it into the panel. It allows you to use a table of common setup strings to assign a new setup string. All you have to do is place the cell pointer on a cell containing a setup string that you want to assign and start the macro.

This macro starts the setup string process using the /PPOS*{ESC} macro code. Notice the astrix "*" character between the /PPOS key commands and the {ESC} macro command. This astrix "*" character clears the previous setup string if it exists and types the astrix instead or just types the astrix if there was no previous setup string. Now the {ESC} macro command safely clears the panel and a new setup string can be printed to the panel. Without the astrix character, the {ESC} macro command can destroy the macro. If this is the first time in the work session that the **/ P**rinter **P**rint **O**ption **S**etup is issued, the {ESC} command without the preceding astrix "*" returns Lotus to the previous menu, where Lotus cannot accept the setup string.

To insert the content of the current cell (the setup string from a table for example) into the panel, the macro uses the following trick: it first issues the {LET string140, @CELLPOINTER("contents")} command which stores the content of the current cell in the cell [string140] and then issues the {string140} routine command which injects the content of [string140] into the panel as the response to the prompt to insert the setup string into the panel. Next the macro issues the tilde "~" command, which is the macro equivalent of ENTER. Notice that {LET string140,@CELLPOINTER("contents")} and {string140} were issued while the macro was in the middle of the process of defining a new setup string. This is a fine example of the advanced use of the {LET} macro command. Another way to use the {LET} macro command is to record the current cell pointer location for later use. Here is a simple example:

```
     A              B                C          D          E          F
1 \Z            {LET here140,@CELLPOINTER("address")}~
2 !             {GOTO}B7~{EDIT}{HOME}'~
3 !             {GOTO}{here140}~
4 !
5 here140       $H$5
6 !
7 !                      100
```

The {LET here140,@CELLPOINTER("address")} macro command stores the current cell pointer address in the B5 cell, [here140]. Assuming that the cell pointer is on the H5 cell, the command writes the $H$5 address label in the B5 cell. Next the macro issues

{GOTO}B7~, which moves the cell pointer to the B7 cell. Then the macro issues {EDIT} to enter the EDIT mode and then issues {HOME}, which moves the cursor to the beginning of the 100 number. Now the macro writes the apostrophe "'" and changes the 100 number into a label and issues the tilde "~" to write the new label into the B7 cell.

To return to the H5 cell, the macro issues the indirect {GOTO}{here140}~ macro command using the content of B5, [here140], to send the cell pointer back to the H5 cell.

One of the finest examples of the use of {LET} macro command is in the **Find Painted Range Address from Within a Macro**. In this technique, the macro combines cell pointer movement with the {LET} command during the middle of a / **R**ange **N**ame **C**reate process to find the addresses of the four corners of the range you paint and also constructs the range address.

Using the {INDICATE} Macro Command for Prompts

The {INDICATE string} macro command was changed in the new releases of 1-2-3 (2.2 and up) to allow the use of long strings. Up to release 2.01, the string was limited to five characters. Beginning with release 2.2, the string can be 80 characters long and more. This feature can be used to display long prompt messages to improve utility and clarity. ***Super Power*** does not use this command to display long messages, because we wanted to keep the macros compatible with all the releases of 1-2-3. However, to use this command to display neat and long prompts in the first row of the screen, Let's look at two macros that use the {INDICATE} macro command:

	A	B	C	D	E	F
1	\A	{INDICATE "Paint the range to be processed"}				
2						
3	\B	{INDICATE}				

If you start the \A macro the first row of the screen will show:

```
B12:                              Paint the range to be processed
```

where the "Paint the range to be processed" prompt is aligned to the right because the prompt is less than 80 characters long. To clear the prompt from the panel, start the \B macro, the {INDICATE} macro command without parameters returns Lotus to the standard mode where the indicate string is five characters long.

	A	B	C	D	E
1	\A	{INDICATE " processed	"	Paint the range to be	
2					
3	\B	{INDICATE}			

If the prompt message has enough leading and trailing spaces to construct an at least 80 character long prompt such as:

```
"                    Paint the range to be processed                   "
```

then when you start the \A macro, the first row of the screen will show:

```
                    Paint the range to be processed
```

where the "Paint the range to be processed" prompt is nicely centered.

Changing a Date String into a Valid Date

If the current cell contains a string with the MM/DD/YY date format, the following macro can transform it into an active date. This is very handy when data is transferred from different systems to the MS-DOS system. Often dates are transferred as text strings. Using this technique, you can transform dates from string format to valid date numbers.

	A	B	C	D	E	F
1	\Z	{EDIT}{HOME}{DEL}@DATEVALUE("{END}")~				

To understand how the macro works, let's assume that the current cell contains the "10/01/92" string. The macro begins with the {EDIT} macro command which enters the EDIT mode. Therefore the panel displays:

```
'10/01/92_
```

The cursor position is marked by the underscore at the end of the string. The next macro command is {HOME} which sends the cursor to the very beginning of the panel which now displays:

```
'10/01/92
^
```

The cursor position is marked by the "^" at the beginning of the panel. The {DEL} macro command deletes the apostrophe "'" and the @DATEVALUE(" characters are typed into the panel, because Lotus does not find any command in the @DATEVALUE(" text it types it directly to the panel. Now the panel displays:

```
@DATEVALUE("10/01/92
```

To type the rest of the formula, the macro issues the {END} macro command which sends the cursor to the end of the panel which displays:

```
@DATEVALUE("10/01/92_
```

Now the macro types the double quote and the closing parenthesis ") to finish the formula in the panel which displays:

```
@DATEVALUE("10/01/92")
```

All that left is to press the ENTER key which is exactly what the tilde "~" macro command does. The result is a formula which envelops the date string to create a valid active date in the current cell.

Using the {FOR Loop} Macro Command

The {FOR} macro command deserves some remarks since it is in the heart of many super power macros. The {FOR Counter,Start,Stop,Step,Subroutine} creates a for loop; it repeatedly performs a subroutine call to subroutine.

- Counter is the address or name of a cell that keeps track of the subroutine execution during the {FOR} loop.
- Start is the initial value for the counter.
- Stop is the value that tells 1-2-3 when to terminate the for loop.
- Step is the value added to counter each time 1-2-3 executes the subroutine.
- Subroutine is the range address or the name of the subroutine that 1-2-3 executes.

The Start, Stop, and Step can be numbers, numeric formulas, or the addresses or names of cells that contains numbers or formulas where Counter is a name of a cell which serves as a counter. When 1-2-3 encounters a {FOR} loop command, it enters the Start value in the counter and then compares the number in the Counter with the Stop value. If the number in the Counter cell is less than or equal to the Stop value, 1-2-3 calls and executes the Subroutine.

When the subroutine is finished, Lotus routes the control back to the {FOR} loop command and increases the value in the counter cell by the Step value. When the number in the Counter cell is greater than the Stop value, the {FOR} loop ends. Before looking at an example, let's see other looping examples and limitations. The following macro is a loop macro using the classic looping technique usually used in a high language such as BASIC.

	A	B	C	D	E	F
1	\Z	{LET count125,0}~{LET total125,0}~				
2	loop125	{IF count125=100}{QUIT}				
3	!	{LET count125,count125+1}~				
4	!	{LET total125,+count125^2+total125}~				
5	!	{loop125}				
6	!					
7	count125	32				
8	total125	11440				

This macro calculates the sum of the 1x1+2x2+3x3+...+nxn series. In this example, the "n" equals 100. The macro starts with the {LET count125,0}~{LET total125,0}~ macro commands which set the values in the B7 and B8 cells named [count125] and [total125] respectively to zero. Next, the macro issues {IF count125=100} to check if the value in the counter cell [count125] is equal to 100. If so, the macro issues the {QUIT} command and quits. If not and the value in the B7 cell (the counter) is less than 100, the macro issues {LET count125,count125+1} which increases the counter by one. Next the macro issues {LET total125,+count125^2+total125} adding the next item in the series to the summation cell [total125]. Last the macro issues the {loop125} routine command which starts the next loop from the start. If you run this macro, you will be surprised to find that after 32 loops, Lotus stops and display:

```
Too many nesting levels in macro calls
```

This error message is easy to fix, just change the {loop123} command to the {BRANCH

loop125} command and everything will work well. The corrected macro is:

	A	B	C	D	E	F
1	\Z	{LET count125,0}~{LET total125,0}~				
2	loop125	{IF count125=100}{QUIT}				
3	!	{LET count125,count125+1}~				
4	!	{LET total125,+count125^2+total125}~				
5	!	{BRANCH loop125}				
6	!					
7	count125	100				
8	total125	338350				

However this is a simple macro and sometimes you cannot just replace the command. Now that everything seems to work, we still have a little problem: speed. If you will run the corrected macro and will compare its speed to the speed of the next macro which uses the {FOR} loop command, you will find that the next macro is more than twice as fast.

	A	B	C	D	E	F
1	\Z	{LET total125,0}~				
2	loop125	{FOR count125,1,100,1,sum125}				
3	!					
4	count125	100				
5	totatl125	338350				
6	!					
7	sum125	{LET total125,+count125^2+total125}~				

This macro starts with {LET total125,0} which sets the value in the summation cell named [totatl125] to zero. Next the macro issues {FOR count125,1,100,1,sum125} which executes the [sum125] routine one hundred times and quits.

Note: Notice the tilde "~" macro command after all the {LET} macro commands in the macros. It forces 1-2-3 to update the cells affected by the {LET} command.

Using Matrices in Lotus 1-2-3

Matrices are mathematical forms that belong to college math, which we are not going to cover; however one particular matrix is interesting because we can use it to simulate the @SUMPRODUCT function, available only in the 3-D releases of 1-2-3. To understand the function, let's look at the following worksheet:

	A	B	C	D	E	F	G	F
1	ITEM	QUANTITY	PRICE					
2	Cameras	5	300	300	150	500	250	250
3	Lenses	6	150					
4	refrigerators	4	500					
5	washing machines	6	250					
6	dryers	2	250					
7								
8								
9	TOTAL TODAY	6400						

The @SUMPRODUCT of the B2..B6 range and the C2..C6 range gives the sum of the products of the cells in these two ranges (+C2*B2+C3*B3+C4*B4+C5*B5+C6*B6). The @SUMPRODUCT function multiplies the values in corresponding cells in multiple ranges and then sums the products. For example, @SUMPRODUCT(B2..B6,D2..D6) is the same as the +C2*B2+C3*B3+C4*B4+C5*B5+C6*B6 function in our sample worksheet. However it only works in the 3-D releases of 1-2-3. In the 2-D releases of 1-2-3, we can use the / **D**ata **M**atrix operation to get the same results when we want the sum product of two equal length columns.

When you define the B2..B6 as the first data matrix and the transposed C2..C6 as the second data matrix and then select the Multiply menu option, you get the same result. Therefore we have to transpose the C2..C6 range and put it in the D2..H2 range. Place the cell pointer on the B9 cell and issue: / **D**ata **M**atrix. Lotus 1-2-3 displays the following menu:

```
Invert  Multiply
```

select the [**M**ultiply] menu option and press ENTER. Now the panel displays:

```
Enter first range to multiply: B9
```

Type D2..H2 and press the ENTER key (always type the row range first). Now the panel displays:

```
Enter second range to multiply: B9
```

Then type B2..B6 and press the ENTER key. Next the panel displays:

```
Enter output range: B9
```

Press the ENTER key and the 6400 result will appear in the B9 cell.

Using System Commands in the New Releases

Beginning with the 2.2 version and up, the `{SYSTEM}` macro command is no longer limited only to shell out to DOS but can include a DOS command, for example the `{SYSTEM "command"}` such as `{SYSTEM "DIR/P"}` to display the directory list, or `{SYSTEM "C:\BASIC\BASICA"}` to start the BASICA program. This allows Lotus 1-2-3 to act as a front end to any DOS application. You can also write stand alone DOS programs written in any language, which perform much faster than it would take to run them in the Lotus macro language. Then write a macro which uses the `{SYSTEM}` macro command to start this DOS program from inside Lotus and import the results back into Lotus automatically.

Note: To activate an executable DOS file from a Lotus 1-2-3 2.0/2.01 macro we needed special tricks. See the SETTIME.WK1.

Find Painted Range Address from Within a Macro

When you design a macro which prompts you to paint a range, you can take advantage of it and, at the same time, you can record the address of the four corners of the painted range and construct the range address of the same painted range. To achieve this task, you need to use the {LET} macro command as demonstrated in the following macro:

	A	B	C	D	E	F
1	\Z	/RNCTable_range~{BS}{?}				
2	!	{LET rightbot,@CELLPOINTER("address")}.				
3	!	{LET leftbot,@CELLPOINTER("address")}.				
4	!	{LET lefttop,@CELLPOINTER("address")}.				
5	!	{LET righttop,@CELLPOINTER("address")}~				
6	!	{LET rangeadd,+lefttop&".."&rightbot}~				
7	!					
8	rightbot	$K$10				
9	leftbot	$H$10				
10	lefttop	$H$5				
11	righttop	$K$5				
12	rangeadd	$H$5..$H$10				

The macro starts with the /RNCTable_range~ macro command which assigns the [Table_range] name and then issues {BS} to free the cell pointer and then issue {?} halting the macro execution and waiting for you to paint the range and press the ENTER key. When you press ENTER, the macro does not immediately issue the tilde "~" (the macro equivalent of the ENTER key) to finish the name create command, instead it issues {LET rightbot ,@CELLPOINTER("address")} which records the current cell pointer location. However at that moment, the cell pointer is on the lower right cell of the painted range. Therefore this command stores the current cell pointer address in the cell [rightbot].

To make it easier to understand the technique, let's assume that you paint the H5..K10 range. Therefore the B8 cell, [rightbot], should contain the $K$10 cell address. Now the macro issues the period [.] key which moves the cell pointer to the next range corner. However, we cannot be sure which corner it is. It can be the lower left corner or the upper right corner depending on the history of the worksheet. When this is the first time that the [Table_range] range name is assigned, the cell pointer will move to the H10 cell at the lower left corner of the painted range.

Next the macro issues {LET leftbot,@CELLPOINTER("address")} which records the current cell pointer location in the B9 cell, [leftbot]. To move the cell pointer to the next corner (upper left corner), the macro issues the period [.] key again and then issues {LET lefttop,@CELLPOINTER("address")} which records the current cell pointer location in the B10 cell, [lefttop].

To find the last corner, the macro issues again the period [.] key which moves the cell pointer to the upper right corner of the painted range and then issues {LET righttop ,@CELLPOINTER("address")} which records the current cell pointer location in the B11 cell, [rightop]. Only now the macro issues the tilde "~", which is the macro equivalent of the ENTER key, to finish the name create procedure. Now that all four corners are known, the macro issues {LET rangeadd,+lefttop&".."&rightbot}~ which combine the content of [lefttop] with the double period ".." string and the content of [rightbot] to create the painted range address as shown in B12, [rangeadd].

Creating Multiple Copies of a Range

You may already know how to create multiple copies of a range with a selected number of empty rows or columns apart in one session; however for the reader whom is unfamiliar with this technique, here it is:

	A	B	C	D	E	F
1	10	20				
2	30	15				

Let's look at the following hypothetical worksheet: if you want to create four more copies of the A1..B2 range downward with three rows apart, here is the way to do it. First you need to calculate the following: number of copies X (number of rows in the range + number of rows apart), which in our example is 4*(2+3)=20. Now place the cell pointer on the A1 cell and from the keyboard issue: / **C**opy and paint the A1..B20 range (20 rows) as the source range and press the ENTER key. Then point to the A6 cell (the location of the first copy) as the target range and press ENTER. The result is the following worksheet:

	A	B	C	D	E	F
1	10	20				
2	30	15				
3						
4						
5						
6	10	20				
7	30	15				
8						
9						
10						
11	10	20				
12	30	15				
13						
14						
15						
16	10	20				
17	30	15				
18						
19						
20						
21	10	20				
22	30	15				

Notice how Lotus placed the A21..B22 range even though you painted only the A1..B20 range. This technique is nice, but you need to do some calculations to know the size of the range to paint. To see a macro which uses this technique and frees you from the calculation process see the COPYMULT.WK1 macro.

2-D and 3-D Lotus Releases

Super Power often uses the terms 3-D Lotus release or 2-D Lotus release. When we use 3-D Lotus release, we mean Lotus 3.0/3.1/3.1+/3.4 and 123W for Windows 1.0/1.0a/1.1 which have multiple sheets and three-dimension capabilities, while 2-D release refers to the 2.0/2.01/2.2/2.3 and the 2.4 Lotus releases which have one sheet only. Many of ***Super Power***'s macros automatically check what kind of spreadsheet you use and adapts themselves accordingly.

Using Functions Allowed in a 3-D Release Inside a 2-D Release

The 3-D releases of 1-2-3 include functions which are not available in the 2-D releases, therefore, if you build an application containing such an alive function, you cannot even save it as a *.WK1 file. Lotus will display a warning message and will notify you that the file cannot be saved with the *.WK1 extension without damage to the file. However you can use the `{LET location,function}` macro command to calculate the function and put the result in the "location" address. Because the "function" in the `{LET}` command is pure text, Lotus will not recognize it as a function until it executes the `{LET}` macro command. Therefore you can save such a command with a *.WK1 file.

When a 2-D release of 1-2-3 processes the `{LET}` command, the result will be the function text in the "location" address. For example, if the command is `{LET A1,@INFO("release")}`, the result in the A1 cell will be the `'@INFO("release")` label. However, if you use Lotus 1.1 for windows, the A1 cell will display the `'1.10.00` label which is the version number of the program. To summarize, if the current 1-2-3 recognizes the function, it calculates it correctly, otherwise it returns the function text as the answer. How can you benefit from this? The answer is simple: use the results to know which version you use (2-D or a 3-D), or which add-in is attached or which 2-D version you use if we purposely use functions which only the new 2.2/2.3/2.4 versions recognize to differentiate from the 2.0/2.01 versions.

Using the {CE} Macro Command in the 3-D Releases

	A	B	C	D	E
1	\Z	This is a neat message ...{GET key126}{ESC}			
2	!				
3	key126				

This macro displays a custom prompt message in the panel. When the macro starts, the macro processor cannot find Lotus commands in the text, therefore it writes the text into the panel which displays:

```
This is a neat message ...
```

To keep the message in the panel so you can read it, the macro issues the {GET key126} macro command which pauses the macro until you press a key. When you press a key, the macro continues and issues {ESC} to clear the message from the panel. Using the {ESC} command works for all the Lotus releases, which is why we used it. To give you a complete picture, the new 3-D releases of 1-2-3 offer the {CE} or the {CLEARENTRY} macro commands to clear the panel, for example:

	A	B	C	D	E
1	\Z	This is a neat message ...{GET key126}{CE}			
2	!				
3	key126				

However after the {CE} macro command, the worksheet stays in the EDIT mode. While the {ESC} macro command returns to the READY mode. The {CE} has a great advantage in macros such as:

	A	B	C	D	E
1	\Z	/FR{CE}{?}~			

It clears everything from the panel no matter how long it is, and allows you to insert a new file name to retrieve. If you are looking for the {CE} command in Lotus 1-2-3 help, you will not find it. It is documented only in the manual.

Using Macro Commands of a 3-D Release in a 2-D Release

Why would we want to use commands that are allowed only in a 3-D release in a 2-D release? The answer is simple: we don't, but we want a macro to work in all the releases transparently; therefore, if we are using 3-D commands in the macro, we must make sure that we won't get an error when we are using the macro in a 2-D release. We have to write the macro in such a way that when you use a 3-D release the commands will be executed. However, they must be ignored if you use a 2-D release. This is also true when we use commands that are available only in new 2-D releases such as 1-2-3 2.2/2.3 and 2.4 that are not available in Lotus 2.0/2.01.

An example of such a command is the {BORDEROFF}. If we are going to use this command in a macro to hide the row and column headings while we use a new release of Lotus 1-2-3, we have to make sure that when you use Lotus 2.0/2.01, the macro will not stop with an error message. The trick will fool Lotus 1-2-3. For every command that is unique to some releases, we need to create a do-nothing routine with the same name. For example:

	A	B	C	D	E
1	\a	{BORDEROFF}			
2	!				
3	borderoff	{}			

If Lotus recognizes this routine as a reserved command, it will give preference to the reserved command and hide the borders; but if Lotus doesn't support the {BORDEROFF} macro command, it will branch to the [borderoff] routine that holds the do-nothing command {} and will continue to the next command. This technique is good for any foreign command that comes with add-ins or new releases. This is an important technique and it can also be used in macros that use commands from Add-ins. If the add-in is attached, then Lotus will execute the command because the add-in supports it. However, if the add-in is detached, then Lotus will "ignore" these commands.

Insert Numbers as Labels

	A	B	C	D	E
1	\Z	{GET key092}{IF @CODE(key092)>=48#AND#@CODE(key092)<=57}{EDIT}'			
2	!	{key092}{?}~{BRANCH \Z}			
3	!				
4	key092				

Sometimes we need to insert data that uses numeric characters, but as labels, when we type a street address "`1000 Bell Avenue`" or a zip code. This macro automatically detects the keys that you press and types the apostrophe "`'`" if the first character is a number. If you activate the macro and then you want to type the 97331 zip code, the macro detects the "9" character and precedes it with the apostrophe "`'`" to show `'97330`.

The macro starts with the `{GET key092}` macro command, which halts the macro execution and waits until you press a key. Lotus stores the key in the B4 cell, [key092]. Next the macro issues `{IF @CODE(key092)>=48#AND#@CODE(key092)<=57}`, which checks if the ASCII code of the character in the B4 cell is between the ASCII code 48 and 57 included. If so, the character is a number, therefore the macro issues the `{EDIT}` macro command to enter the EDIT mode and then writes the apostrophe "`'`" character directly into the panel. Now the macro issues the `{key092}` routine command which injects the content of the [key092] routine into the panel. To make it easier to understand, let's assume that you want to type "`1000 Bell Avenue`".

When you press the "`1`" key, the macro detects this as a number, stores it in cell [key092], enters the EDIT mode, and types the apostrophe into the panel which displays:

```
'
```

Next, the macro issues the `{key092}` routine command, which injects the content of the [key092] routine into the panel, which displays:

```
'1
```

Now, the macro issues `{?}` which allows you to use the keyboard to type the rest of the address. When you are finished and just before you press ENTER, the panel displays:

```
'1000 Bell Avenue
```

When you press ENTER, the macro processor understands that you have finished with the `{?}` macro command and then issues the tilde "`~`", the macro equivalent of ENTER, and writes the content of the panel into the current cell as a label with the apostrophe prefix. Last, the macro issues `{BRANCH \Z}` routing the macro control back to the start to allow you to continue to enter data. To finish the macro, you need to issue the Ctrl- Break.

Finding the Release of Lotus Being Used

When macro programmers write a macro for others to use, they have to remember that there are users still using old releases of Lotus 1-2-3. Often you own and keep previous releases for many reasons. Therefore, a good macro should be able to operate on all or many of the releases with few problems. The way to do it from within the macro is to look for the known differences between the different releases. Those differences can be functions or commands which are only available in some of the release or are "bugs" or peculiarities that have been found in different versions. These differences can be used as a fingerprint of the release.

Starting with the introduction of 3-D spreadsheets of Lotus 1-2-3 3.0/3.1/3.1+/3.4 and Lotus for Windows 1.0/1.0a/1.1, a new function has been added, the `@INFO("release")` function, which returns the current version number as a label. For example, if the current version is 3.1 then the `@INFO("release")` function returns the 3.10.00 result. The `@INFO("release")` function differentiates between 3-D releases. From our experience with hundreds of macro, we could not find any difference in operation between 3.0/3.1/3.1+ and 3.4 therefore ***Super Power***'s macros do not try to check for the exact version. When you issue the `@INFO ("release")` function in Lotus for windows, the result is 1.10.00 for version 1.1 or 1.00.00 for version 1.0. Again, we could not find a difference when we use our macros. Therefore ***Super Power***'s macros do not specifically look for the version number of a 3-D release.

However we do check if you use a 2-D release or a 3-D release. We use the `@INFO ("release")` function but in a different way. The macro issues the `{LET location ,@INFO("release"}` macro command to insert the result of the function in the "location" address. However, if you use a 2-D release which does not support this function, the result in the location address will be the function text (the `'@INFO("release"}` label). Therefore, we can check for the "@" character. If it is there, than you use a 2-D release. The ***Super Power***'s macros do not check further than determining whether you use a 2-D or a 3-D release.

For the curious reader eager to find the exact version, here the results of research efforts to detect fingerprints for every Lotus version from 2.0 and above. We already know that we can use the `@INFO("release"}` function to differentiate between the 3-D and 2-D releases, therefore we will concentrate on how to detect the exact 2-D release.

In release 2.2/2.3 and 2.4, a new attribute has been added to the `@CELLPOINTER` and the `@CELL` functions, which is "filename", therefore we can use the `@CELLPOINTER ("filename")` function. If the `@CELLPOINTER("filename")` function returns a label, it means that you use one of the 2.3/2.3 or 2.4 releases. If the function returns ERR, then you use either version 2.0 or 2.01. Use the `@ISERR(@CELLPOINTER("filename"))` function. If the result is "`1`" the release is 2.0 or 2.01 if the result is "`0`" the release is 2.2 or 2.3 or 2.4.

Let us assume that the result indicates to the 2.0/2.01, how can we find which one it is? Here is a potentially new solution. After extensive research, we found the result of the `@IF(@CELLPOINTER("contents")="",0,1)` formula is not the same in 2.0 and 2.01. In 2.0 the result is ERR, but the same formula works correctly in 2.01 and returns "`0`" if the current cell is empty or "`1`" if it is occupied. It appears that 2.0 doesn't allow us to check if a cell is empty by comparing its contents to the NULL string. This was fixed in release 2.01 and up.

What is left to find out is how to check if you use 2.2 or 2.3 or 2.4, after extensive research we have found that we can use the DATE functions to differentiate between 2.2 and 2.3. For example, the `@DAY("David")` formula returns ERR in 2.3 but returns zero "`0`" in 2.2. At this particular moment, we have not found a direct and simple way to differentiate between 2.3 and 2.4. Now we can build an automatic general purpose macro that can help application builders and macro developers to route their macros based on the release being used.

Here is a sample macro summarizing all these techniques.

	A	B	C	D	E
1	\Z	{LET rel1410,@INFO("release")}~			
2	!	{IF @LEFT(rel1410,4)="3.00"}{GOTO}rel3.00.00~{QUIT}			
3	!	{IF @LEFT(rel1410,4)="3.10"}{GOTO}rel3.10.00~{QUIT}			
4	!	{IF @LEFT(rel1410,4)="3.40"}{GOTO}rel3.40.00~{QUIT}			
5	!	{IF @LEFT(rel1410,4)="1.10"}{GOTO}123w1.10~{QUIT}			
6	!	{IF @LEFT(rel1410,4)="1.00"}{GOTO}123w1.00~{QUIT}			
7	!	{RECALC rel2.01or2.2.3}{RECALC rel2.0or2.01}			
		{RECALC rel2.22.3}{RECALC aiserr2.0}			
8	!	{IF rel2.01or2.2.3=0}{IF rel2.2or2.3=0}{GOTO}rel2.2~			
		{QUIT}			
9	!	{IF rel2.01or2.2.3=0#AND#rel2.22.3=1}{GOTO}rel2.3~			
		{QUIT}			
10	!	{IF aiserr2.0=1}{GOTO}rel2.0~{QUIT}			
11	!	{GOTO}rel2.01~{QUIT}			
12	!				
13	!				
14	rel2.01or2.2.3	0			
15	rel2.0or2.01	1			
16	rel2.2or2.3	ERR			
17	aiserr2.0	0			
18	rel1410	@INFO("RELEASE")			
19	rel2.22.3	0			
20	!				
21	rel3.00.00				
22	rel3.10.00				
23	rel3.40.00				
24	123w1.10				
25	123w1.00				
26	rel2.0				
27	rel2.01				
28	rel2.2				
29	rel2.3				

Before we start to analyze the macro, notice that the code in the B14, B15, B16, B17 and the B19 cells is the result of the following formulas:

14	rel2.01or2.2.3	@ISERR(@CELLPOINTER("filename"))
15	rel2.0or2.01	@IF(@CELLPOINTER("contents")="",0,1)
16	rel2.2or2.3	@DAY("DAVID")
17	aiserr2.0	@ISERR(B14)
19	rel2.22.3	@ISERR(B16)

The macro starts with the `{LET rel1410,@INFO("release")}` macro command which stores the result of the `@INFO("release")` function in the B18 cell named [rel1410]. Then the macro continues with the tilde "`~`" macro command to force 1-2-3 to immediately update the content of the B18 cell to reflect the last command. Next the macro issues `{IF @LEFT(rel1410,4)="3.00"}`, which checks if the first four characters of the content of B18 are equal to "`3.00`". If this is true, it means you use 3.0, therefore the macro issues `{GOTO}rel3.00.00~`, which moves the cell pointer to the B21 cell named [rel3.00.00] and quits.

If you do not use 3.0 the macro issues `{IF @LEFT(rel1410,4)="3.10"}` which checks if you use 3.1. If this is true, the macro issues `{GOTO}rel3.10.00~`, which moves the cell pointer to the B22 cell named [rel3.10.00] and then quits. The next command is `{IF @LEFT(rel1410,4)="3.40"}`, which checks if you use 3.4. If so, the macro issues `{GOTO}rel3.40.00~`, which moves the cell pointer to the B23 cell named [rel3.40.00] and quits.

Now the macro issues `{IF @LEFT(rel1410,4)="1.10"}` which checks if you use 123W 1.1. If so, the macro issues `{GOTO}123w1.10~`, which moves the cell pointer to the B24 cell named [123w1.10] and then quits. Next the macro issues `{IF @LEFT(rel1410,4)="1.00"}` macro command which checks if you use 1.0 or 1.0a, if so, the macro issues `{GOTO}123w1.00~`, which moves the cell pointer to the B25 cell named [123w1.00] and then quits. The macro has checked for the 3-D releases of Lotus 1-2-3; from now on, the macro looks for the 2-D releases of 1-2-3.

The macro continues with `{RECALC rel2.01or2.2.3}{RECALC rel2.0or2.01}{RECALC rel2.22.3}{RECALC aiserr2.0}` to calculate and update the formulas in B14, B15, B16 and B19. Next the macro issues `{IF rel2.01or2.2.3=0}` which checks if you use 2.01 or 2.2/2.3/2.4. If this is true, then you use 2.2 and up. Therefore the macro continues with the `{IF rel2.2or2.3=0}` macro command to check if it is 2.2 or one of the 2.3 or 2.4 versions. If this is also true, then you use 1-2-3 version 2.2, therefore the macro issues `{GOTO}rel2.2~`, which moves the cell pointer to the B28 cell named [rel2.2] and quits.

If the previous condition was false, then you use either 2.0 or 2.01 or 2.3/2.4, therefore the macro issues `{IF rel2.01or2.2.3=0#AND#rel2.22.3=1}` which checks for version 2.3/2.4. If the combined condition is true, then you use either 2.3 or 2.4 and the macro issues `{GOTO}rel2.3~` and moves the cell pointer to the B29 cell, [rel2.3]. Next is `{IF aiserr2.0=1}` to determine if you use 2.0. If this is true, the macro issues `{GOTO}rel2.0~` which moves the cell pointer to the B26 cell, [rel2.0].

If all the previous conditions were false, then you use 2.01; therefore the macro issues `{GOTO}rel2.01~`, which moves the cell pointer to cell [rel2.01] and quits.

Summary: This macro demonstrates how we can use the differences in the versions of Lotus 1-2-3 to use a macro to determine the version you currently use. This allows the macro developer to create a foolproof macro code, which can work transparently across many versions of 1-2-3.

Advanced Use of the Response to the {GET Key} Command

When the macro processor encounters the {GET keyname} macro command, it pauses and waits until you press a key. When you press a key, Lotus stores it in the cell with the [keyname] range name. This allows the macro developer to know which key you press. The question is what to do with this key? The following macro demonstrates the use of this macro command and an advanced use of the stored key.

	A	B	C	D	E
1	\Z	{ESC}Use the direction keys and press ENTER to hide. Press ESC to quit.			
2	!	{GET key096}{ESC}~{IF key096="{ESC}"}{ESC 6} {BRANCH ret096}			
3	!	{IF key096="~"}/WCH~{BRANCH \Z}			
4	!	{key096}{BRANCH \Z}			
5	!				
6	key096	{ESC}			
7	!				
8	ret096				

This macro allows you to continuously hide columns in the worksheet. The macro starts with the {ESC} macro command (to be explained later) and then continues with pure text which Lotus writes into the panel as:

```
Use the direction keys and press [ENTER] to hide. Press ESC to quit.
```

To hold this prompt text in the panel, the macro issues {GET key096} which pauses the macro and waits until you press a key. The macro stores the key in the B6 cell named [key096] and then issues {ESC} to clear the message from the panel. Otherwise Lotus will write the prompt message into the current cell. Next the macro issues {IF key096= "{ESC}"} to check if you pressed the ESC key. If so, the macro issues {ESC 6} to return to the READY mode and then issues {BRANCH ret096} which routes macro control to the empty [ret096] routine and quits.

Next the macro issues {IF key096="~"} to check if you pressed the ENTER key (the tilde "~" is the macro equivalent of the ENTER key). If so, the macro issues the /WCH~ macro keys and hides the current column and then issues {BRANCH \Z} to route macro control back to the beginning of the macro to allow you to hide more columns. If you did not press ENTER or ESC, the macro issues the {key096} routine command which Lotus executes. However, because [key096] contains only the key that you just pressed, Lotus executes the same key. For example if the key is the slash "/" key, then the main menu should appear. However, remember the {ESC} at the beginning? It returns Lotus to the READY mode. If you pressed the "A" character or any other character, the {key096} command injects this key into the panel. Again the {ESC} command at the beginning of the macro comes to the rescue and clears the character from the panel before the macro writes the prompt message into the panel.

Super Power uses the {GET keyname} macro command to create special effects or to control input.

Managing Macro Libraries

Grouping the Macros in One *.WK1 File

Keep a group of macros in one spreadsheet file (*.WK1) and combine it to the current worksheet every time you need the macros. A spreadsheet of macros is called a library. This is a simple, often used method. However it is limited, old fashioned, and memory and time consuming. There are always some macros in the library that you do not use at your working session. Therefore they use valuable memory because they belong to the worksheet. Trying to erase these macros before you start to work demands precise preparation, and a solid knowledge of the exact actions you are going to do during the working session, which are not so simple to predict.

Grouping the Macros in One *.MLB File

This method is identical to the previous one and therefore suffers from the same limitations.

Using Menu-Driven Macro Managers

In this method, every macro is saved as an independent file. The macros are located in a sub directory. Using the menu driven manager you can:

- See the list of all the macros in the directory
- Activate the macros using point and shoot
- Read the macro's operation instructions

The advantages are clear (see Appendix-A):

- Only one macro resides in the memory at the same time
- The manager loads the macro, defines the range names and activates the macro without your interference. The result is memory space and preparation time saving, because there is no need to plan the work ahead.
- The work is menu driven using a Lotus style menu bar.

Using the Macros Manually

You can combine the macros of ***Super Power*** to an empty part of the worksheet, use the /**F**ile **C**ombine **C**opy **E**ntire-file Lotus commands sequence. Then follow the built-in use instructions in the first nine rows of the macro, dedicated to macro description and use instructions.

Using the Macro Manager

The last macro in ***Super Power*** is a central macro manager which allows you to start the macros of ***Super Power*** from the disk. Combine only the macro manager to your worksheet. When you start the macro manager, it displays a custom menu, which you can use to start or view the macros in the disk.

Even if you are an inexperienced user, you can take full advantage of the ***Super Power***'s macros. All you need to know is how to key these macros into Lotus or purchase the SUPER MACRO LIBRARY . To use the macro manager, move the cell pointer to an empty area outside your work area to insure that all cells down to the manager are empty. It is best to combine the macro manager's code to an area above and left or right and down to the working area. When you insert or delete rows it will not affect the macro manager. To get there quickly, use the following keys sequence F5 (GOTO) END HOME RIGHT PGDN. The cell pointer will move to a cell down and right to the work area.

To combine the macro manager, issue: / **F**ile **C**ombine **C**opy **E**ntire_file SMALLMGR and press the ENTER key, then read the built-in instructions in the first nine rows of the manager and start it. From here on, the work is menu-driven and automatic. The macro manager combines the macros, activates them and erases them after use.

To combine the macro manager issue:

```
/ File Combine Copy  Entire-file SMALLMGR
```

and press the ENTER key. Place the cell pointer on the cell containing the \Y label use:

```
/ Range Name Label Right
```

and press the END and the DOWN keys, then press the ENTER key. To start the manager, press and hold down the [MACRO] key ([ALT] for most computers or [CTRL] for 1-2-3 for Windows), then also press the "Y" key. The macro will start and the next menu will appear:

```
Default-dir  Macro_run  View  List  Erase  Quit
```

[**D**efault-dir]

Select this menu option to set the default directory to the same drive\directory where the macros are. Trying to run a macro which is not in the default directory will result in an error message.

[**M**acro_run]

Select this menu option to start a macro. When you select this menu option, the macro displays the directory list. Highlight the macro to run and press ENTER. After a short delay the macro is operative. When the macro ends, the manager deletes all macro's range names, and erases the macro and re-displays the main menu.

Note: If the directory holds files other than the macros, the macro displays all the *.WK* files in the default directory. Therefore we recommend that you place the macros in a separate directory.

[**V**iew]

Select this option to display the directory list. Highlight the macro to VIEW and press ENTER. After a short while, the screen displays the macro's code and using instruction, use the direction keys (ONLY) to scroll through the macro. When you press ENTER, the manager erases the macro and the main menu appears. To execute the macro when you are in the **V**iew mode, press the **E**xecute key.

[**L**ist]

Select this option to list all the files in the current directory. This option lets you verify the existence of a macro in the current directory or disk. This is especially useful when you keep the macros on several 360K floppy diskettes. Press ENTER to quit the session.

[**E**rase]

Select this option to remove the macro manager from the worksheet. This option deletes the range names associated with the manager and erases the manager's text and code, leaving no trace of the macro manager. Because the manager erases itself, the process will end with an error message. Therefore, when you see the error message, press ENTER to return to READY mode.

Note: If you press [CTRL BREAK] during the MANAGER SESSION, Lotus activates the error handling routine and prompts you to press ENTER to return to the main menu. However, the manager will not erase the macro's code and the macro's range names, and they will stay in the worksheet. To erase them, QUIT the manager and then place the cell pointer on an empty range, re-activate the manager and re-activate the same macro again. When the macro ends the manager will erase the macro and all the range names.

Change All the Labels in a Range to Lowercase Form

Normally this macro should be in Part Two of ***Super Power*** in the **Modifying Macros** chapter. However, after we have covered all the advanced techniques in the previous pages, it is time to analyze at least one macro and show some of the basic techniques in action. Therefore we will take a deep look in the LOWRCASE.WK1 macro which changes all the labels in a selected range into a lowercase form. When you read the analysis, the notes refer you to the specific technique we saw earlier.

```
     A                B                C                D                E
 1 *---A macro to turn all the labels in a 3-D or a 2-D range to LOWERCASE
       form
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6          THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                     IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 LOWRCASE        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb410,0}~
12 cont410         {LET hereabs410,@CELLPOINTER("address")}~
                   {LET counter410,0}
13 !               {FOR counter410,0,@COLS(Which range ?)-1,1,labels410}
14 !               {LET rel410,@INFO("release")}~{IF @LEFT(rel410,1)<>"@"}
                   {GOTO}{hereabs410}~{LET counterb410,counterb410+1}~
                   {IF counterb410<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs410}~{BRANCH cont410}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter410             3
18 countera410            5
19 labels410       {RIGHT}{LET here410,@CELLPOINTER("address")}~{LEFT}
                   {FOR countera410,0,@ROWS(Which range ?)-1,1,labelsa410}~
                   {IF counter410<@COLS(Which range ?)-1}{GOTO}{here410}~
                   {LET countera410,0}~
20 !
21 here410         $D$1
22 !
23 labelsa410      {PANELON}{RECALC prop410}{IF @CELLPOINTER("type")<>"b"
                   #AND#@CELLPOINTER("type")<>"v"#AND#@CELLPOINTER("prefix")
                   <>"\"#AND#@CELLPOINTER("prefix")<>"|"}{PANELOFF}
                   /RVprop410~~{EDIT}~
24 !               {DOWN}
25 !
26 prop410         \Z
27 !
28 counterb410            4
29 hereabs410      $A$1
30 !
31 rel410
```

Notice that the B26 cell named [prop410] contains the following formula:

```
26 prop410       @LOWER(@CELLPOINTER("contents"))
```

If you intend to key this macro into Lotus, you have to key the formula and not the code as it appears in the main listing.

The text in the A1 cell describes what the macro does. The text in A2 contains the instructions for how to assign the labels in the "A" column as range names to the cells to the right in the "B" column. The text in the A3 cell explains how to start the macro. The text in the A6..A7 range explains in which Lotus versions you can use the macro.

	A	B	C	D	E
10	\Z	{BREAKON}			
11	LOWRCASE	{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~ {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~ {LET counterb410,0}~			

The macro starts in the B10 cell with the do nothing {BREAKON} macro command. The reason is connected to the way the macro manager works. It always looks for the [\Z] range name to assign the range name for the combined macro. Often the macro loops back to the beginning of the macro code. If the loop is explicitly to the [\Z] range name, you will not be able to change this name to [\A] for example. Therefore the loop is always to the macro name itself such as {BRANCH LOWRCASE} instead of {BRANCH \Z}. Sometimes you may see the {BREAKON} macro command twice and it concern the way the macro is used as an *.MLB macro in the SUPER MACRO LIBRARY.

The [LOWRCASE] name is the range name of the B11 cell, and it is also identical to the DOS name of the LOWRCASE.WK1 file. The code in B11 starts with {WINDOWSOFF} {PANELOFF} which freeze the screen and panel display activities as explained earlier.

The next piece of code is:

```
/RNCWhich range ?~~/RNDWhich range ?~
    /RNC{WINDOWSON}{PANELON}Which range ?~{BS}{BS}{?}~
```

The macro now prompts you to paint the range to modify, while it assigns a range name to the range that you paint. The macro uses this range name later when it modifies the cells in the range. Therefore the macro starts a process which we call the "safe technique" to assign a range name. This time the range name that the macro assigns is the [Which range ?] range name. First the macro creates this name then it deletes it and then creates it again. Why we use such an odd range name as [Which range ?]? Lotus allows us to use range names of up to 15 characters long, so we can use this property to assign range names with a meaning. Not only does the macro use the "safe technique" to assign a range name, it combines it with a second technique which displays the [Which range ?] range name in the panel so it looks to you as a helping prompt.

Notice the exact location where the macro issues the {WINDOWSON}{PANELON} macro commands which resume panel and screen display activities just before the macro types the [Which range ?] range name. If the cell pointer is on the C10 cell for example, this results in the following display in the panel:

```
Which range ?                          Enter range: C10..C10
```

Without this technique the panel would display:

```
Enter name: Which range ?              Enter range: C10..C10
```

Because the macro issues the {WINDOWSON}{PANELON} macro commands after the /RNC

macro keys, it suppresses the "Enter name:" text and displays the "Which range ?" text which looks like a helping prompt. The macro continues with the {BS} macro command which frees the cell pointer in the cell and the panel displays:

```
Which range ?                          Enter range: C10
```

Note: Beginning with Lotus version 2.2 and up, the {INDICATE "string"} macro command can accept a long string and is no longer limited to only five characters; therefore an 80 character long string can be used as a prompt with the {INDICATE "Long string"}. However, because we want the macros to work with all the releases of Lotus 1-2-3, we used the previous technique instead.

Updating the LOWRCASE macro to include the {INDICATE} prompt will give:

```
{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RNDWhich range ?~
    INDICATE "                Paint the range to be
    processed                 "}/RNC{WINDOWSON}Which range ?
    {PANELON}~{BS}{BS}{?}~{INDICATE}{WINDOWSOFF}
```

Where the

```
"             Paint the range to be processed                "
```

prompt string should be at least 80 characters long to suppress any Lotus prompt and display only the custom prompt string. The {INDICATE} command without the string clears the panel from the prompt. For now, while a large number of users still use 2.0/2.01, we will continue to use the previous prompt technique. Later, when most users will switch to the newer releases, we may prefer the {INDICATE} prompt.

Now the macro issues {?} which pauses the macro and allows you to move the cell pointer and to paint the range to modify. When you press ENTER, the macro resumes control and issues the tilde "~", the macro equivalent of the ENTER key, and assigns the [Which range ?] name to the painted range. Let's assume that you painted the C20..E22 range.

When the macro finished assigning the range name, the cell pointer moved back to the C10 cell. To move the cell pointer to the C20 cell, the macro continues with {GOTO}Which range ?~. Next it issues {LET counterb410,0}~, which sets the value in the B28 cell, [counterb410], to zero. The macro uses the B28 cell named [counterb410] as a counter. With that we conclude the definition of the [Which range ?] range. Now the macro "knows" it is going to process the [Which range ?] range.

For simplicity's sake, let's assume that you painted the C20..E22 range which looks like this:

	C	D	E	F
20	DAVID	KING	45	
21	ALLEN	DOYLE	47	
22	ROBERT	DUVAL	42	

To calculate the number of rows in the [Which range ?] range, we can use the @ROWS(Which range ?) formula. To find the number of columns in the [Which range ?] range, we can use the @COLS(Which range ?) formulas, so the macro "knows" how to handle the whole range.

Now let's look on the second part of the LOWRCASE macro:

	A	B	C	D	E
12	cont410	{LET hereabs410,@CELLPOINTER("address")}~			
		{LET counter410,0}			
13	!	{FOR counter410,0,@COLS(Which range ?)-1,1,labels410}			
14	!	{LET rel410,@INFO("release")}~{IF @LEFT(rel410,1)<>"@"}			
		{GOTO}{hereabs410}~{LET counterb410,counterb410+1}~			
		{IF counterb410<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs410}~{BRANCH cont410}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

The macro continues with `{LET hereabs410,@CELLPOINTER("address")}` which stores the current cell pointer location in the B29 cell, [hereabs410], and issues the tilde "~" to force Lotus to update the content of [hereabs410]. When we finished the definition of the [Which range ?] range name, the cell pointer was on the C20 cell (the upper left cell of [Which range ?]), so why do we "record" the cell address? When we want to return to the C20 cell, we can always use the `{GOTO}Which range ?~` command. The answer is quite simple, we need the C20 address for 3-D releases where [Which range ?] may contain more than one sheet, as be explained later. Next the macro issues `{LET counter410,0}` which sets the value in the B17 cell, [counter410], to zero. The macro continues with:

```
{FOR counter410,0,@COLS(Which range ?)-1,1,labels410}
```

which executes the [labels410] routine as many times as the number of columns in the [Which range ?] range. The [counter410] cell holds the count of the columns that the macro processed during it's execution. Before we continue with the code, let us look at the [labels410] routine:

	A	B	C	D	E
19	labels410	{RIGHT}{LET here410,@CELLPOINTER("address")}~{LEFT}			
		{FOR countera410,0,@ROWS(Which range ?)-1,1,labelsa410}~			
		{IF counter410<@COLS(Which range ?)-1}{GOTO}{here410}~			
		{LET countera410,0}~			

This routine starts with the `{RIGHT}` command which moves the cell pointer to the first cell of the next column in [Which range ?]. Then it issues `{LET here410, @CELLPOINTER("address")}`, which stores the current cell pointer location in the B21 cell, [here410], and then `{LEFT}` to return. When the macro finishes processing the first column, it uses this address to return directly to the upper cell of the next column to save the movement steps all the way back to the top of the next column. Imagine how long it will take to move the cell pointer from the last cell in the current column to the first cell of the next column. If the range has 1000 rows, it will take a very long time to use the `{UP @ROWS(Which range ?)}{RIGHT}` commands to put the cell pointer on the first cell of the next column. Next the macro issues:

```
{FOR countera410,0,@ROWS(Which range ?)-1,1,labelsa410}
```

which executes the [labelsa410] routine as many times as the number of rows in the [Which range ?] range. This `{FOR}` command processes the whole column. Before we continue let's look at the [labelsa410] routine.

	A	B	C	D	E
23	labelsa410	{PANELON}{RECALC prop410}{IF @CELLPOINTER("type")<>"b"			

```
                  #AND#@CELLPOINTER("type")<>"v"#AND#@CELLPOINTER("prefix")
                  <>"\"#AND#@CELLPOINTER("prefix")<>"|"}{PANELOFF}
                  /RVprop410~~{EDIT}~
24 !              {DOWN}
25 !
26 prop410        \Z
```

Recall that the B26 cell, [prop410], contains this formula:

```
26 prop410        @LOWER(@CELLPOINTER("contents"))
```

The [labelsa410] routine processes the whole column, cell after cell. First the routine starts with {PANELON} to allow you see some action in the panel while the routine transforms the labels into lowercase form. The routine issues {RECALC prop410}, a macro command which recalculates and updates the @LOWER(@CELLPOINTER("contents")) formula and returns the lowercase form of the current cell pointer content. For example, if the cell pointer is on the C20 cell in the sample worksheet, the formula will return the "david" string. Remember that the macro should work when the worksheet is also in manual recalculation mode. For a foolproof operation, the macro must make sure that it processes only labels. Therefore the macro issues the

```
{IF @CELLPOINTER("type")<>"b"#AND#@CELLPOINTER("type")<>"v"#AND#
    @CELLPOINTER("prefix")<>"\"#AND#@CELLPOINTER("prefix")<>"|"}
```

compound {IF} commands, which check that the cell is not blank, that the cell does not contain a value, that the prefix is not the back slash "\" and that the prefix is not the bar "|". The back slash and the bar type prefixes have special role in Lotus 1-2-3; therefore we do not want the macro to mess up the worksheet. When all the conditions are met the macro issues /RVprop410~~, which copies the result of the formula in the B26 cell, prop410, as a value into the current cell (in our example, the "david" label).

The next {EDIT}~ commands seem to be unnecessary, however we have found that without these two commands, the macro does not work correctly in some versions of Lotus, when we use this macro from the macro manager. We added these commands to force Lotus to update the content of the current cell after the /RVprop410~~ command. Finally, the macro issues {DOWN} which moves the cell pointer to the next cell in the current column and returns the control to the {FOR} loop in the [labels410] routine.

```
     A            B                C                D            E
19 labels410      {RIGHT}{LET here410,@CELLPOINTER("address")}~{LEFT}
                  {FOR countera410,0,@ROWS(Which range ?)-1,1,labelsa410}~
                  {IF counter410<@COLS(Which range ?)-1}{GOTO}{here410}~
                  {LET countera410,0}~
```

When the {FOR} command finishes processing all the cells in the column, the macro issues {IF counter410<@COLS(Which range ?)-1} which checks if the macro did not finish processing all the columns in the range. If so, the macro issues the {GOTO}{here410}~ indirect macro command sending the cell pointer directly to the first cell in the next column. Finally, the routine resets the value in the [countera410] counter to zero and returns the control to the {FOR} command in the [cont410] routine.

```
     A            B                C                D            E
12 cont410        {LET hereabs410,@CELLPOINTER("address")}~
                  {LET counter410,0}
13 !              {FOR counter410,0,@COLS(Which range ?)-1,1,labels410}
```

```
14 !              {LET rel410,@INFO("release")}~{IF @LEFT(rel410,1)<>"@"}
                  {GOTO}{hereabs410}~{LET counterb410,counterb410+1}~
                  {IF counterb410<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs410}~{BRANCH cont410}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
```

When the {FOR} command finishes processing all the columns in [Which range ?], the macro issues {LET rel410,@INFO("release")}~ storing the result of the @INFO ("release") 3-D function in the B31 cell, [rel410]. Lotus uses the result of this function to find if you are using a 2-D or a 3-D Lotus release. The macro issues {IF @LEFT(rel410,1)<>"@"} to check whether the first character in [rel410] is not equal to the "@" character. If so, you are using a 3-D lotus release, therefore the [which range ?] range may contain more than one sheet to process.

The macro continues with the indirect {GOTO}{hereabs410}~ command, which moves the cell pointer to the address of the upper left cell of the current sheet of [Which range ?]. Then the macro issues {LET counterb410,counterb410+1}~ which increase the counter value in the B28 cell, [counterb410], which counts the number of sheets that the macro processes. Now it issues {IF counterb410<@SHEETS(Which range ?)} to check if all the sheets were processed. If the macro has to process more sheets, it issues the {NS} command moving the cell pointer to the next sheet to process, and then issues {GOTO} {hereabs410}~ which moves the cell pointer to the address of the upper left cell of the current sheet of [Which range ?].

To process the new sheet the macro issues {BRANCH cont410} routing the macro control to the start. When the macro finishes processing all the sheets, the macro issues {GOTO} Which range ?~, which moves the cell pointer to the upper left cell of the first sheet. Then issues /RNDWhich range ?~ to delete the [Which range ?] temporary range name to leave a clean worksheet.

Note: In this and many other ***Super Power***'s macros, we have used the two nested {FOR} commands to move the cell pointer along the column and from column to column to process all the cells in the range. The experienced user may say that we can use the @INDEX function to process the cells of the range without movement. It is true, but it has a drawback. The {PUT} command, usually associated with the @INDEX function, inserts fixed results into the cells but not formulas. In this macro, even though we used the /RV to enter fixed data into the cells, ***Super Power*** contains many macros that process the cells and leave the results as formulas. Only when we use the technique we have used here, can we edit the cell and envelop it with a formula. Therefore we have chosen to use one range handing routine for all our macros in ***Super Power***. When we tested the two methods, there was no distinct speed advantage to the @INDEX methods, because moving one cell every time is very fast and the movement to the top of the next column is instantaneous using the indirect {GOTO} command.

We can create UPPERCASE and PROPER macros just by changing the @LOWER function to @UPPER or to @PROPER.

Till now we have seen that the macro consists of the following stages:

- It prompts you to paint the range to be changed to lowercase format, and at the same time the macro assigns the [Which range ?] name to the range which is also used as a supporting prompt.

- The macro resets the counters to zero [counter410] for the number of columns, [countera410] for the number of rows in [Which range ?], and [counterb410] for the number of sheets in case that [Which range ?] is a 3-D range (for release 3.0/3.1/3.1+/3.4 and 123W).
- The macro starts the outer [labels410] loop routine on the number of columns activating the inner [labelsa410] loop routine to process the rows. When an inner loop is finished, the column counter increased by 1 and the next inner loop starts. When the column counter [counter410] reaches the number of columns in the [Which range ?] range, the macro stops; but not before it deletes the [Which range ?] range name. When the [Which range ?] is a 3-D range, the macro does not stop, it steps to the next sheet in the [Which range ?] range. Now we can see the reason for recording the cell address in the [hereabs410] range.

SUMMARY

In this chapter we covered the following issues:

- Naming a range safely using /RNC and /RND
- Using a meaningful range name to display a prompt
- Recalculating a formula to update the result even in manual recalculation mode
- Testing for the release that you use
- Using {FOR} loop macro command, and nesting {FOR} loop commands
- Recording cell pointer address for later use
- Using indirect macro command
- Paging safely in a "straight path" between sheets in 3-D spreadsheet
- Ending a routine using {RETURN} or blank cell
- Using the {INDICATE} macro command to display long prompts
- We used {PANELON}, {PANELOFF}, {WINDOWSON}, {WINDOWSOFF}, {NS}, {BS}, {FOR}, {EDIT}, {RECALC}, {GOTO}, {INDICATE}, {RIGHT}, {LEFT}, {IF}, {DOWN}, {RETURN}, {LET} and {PUT} macro commands
- We used the @COLS, @ROWS, @CELLPOINTER, @SHEETS, @UPPER, @LOWER, @PROPER, @LEFT @INFO("release") and the @INDEX functions and the #AND# logical function
- We used "address", "type", "prefix", "contents" attributes in the @CELLPOINTER ("attribute") function

Not bad for one macro. For more information on these functions and macro commands, consult your Lotus manual. ***Super Power*** contains hundreds of other macros employing and using these techniques and many others. We believe reading the analysis of these macros is the best way to learn the Lotus macro language.

Macros Affecting Rows, Columns and Ranges

[5] Transpose a Range

```
          A                B                C                D
E
  1 *---A macro to transpose a 3-D or 2-D column wised range to row wised
  2     range or wise versa. This macro is for a range contains labels or
  3     values but NOT formulas. To transpose ranges containing formulas
  4     see the TRANSPO2.WK1 and TRANSPO3.WK3 macros.
  5 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
  6     range names in this column (starts with the \Z macro name)
  7 *---Highlight the cell to be calculated
  8 *---Hold the [ALT] key and press [Z] to activate the macro
  9 !
 10 \Z              {BREAKON}
 11 COLTOROW        {WINDOWSOFF}{PANELOFF}/RNCSource range ?~~
                    /RNDSource range ?~/RNC{PANELON}Source range ?~
                    {WINDOWSON}{BS}{BS}{?}~
 12 !               {WINDOWSOFF}{PANELOFF}/RNCTarget range ?~~
                    /RNDTarget range ?~/RNC{PANELON}Target range ?~
                    {WINDOWSON}{BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}
 13 !               /RTSource range ?~Target range ?~~
 14 !               /RESource range ?~/RNDSource range ?~
                    /RNDTarget range ?~
```

This macro does a standard Lotus transpose; however it makes the task easier. First it prompts you to define the source range to transpose, and then the target range (the upper left cell of the target range is enough). Then the macro transposes the range. Because the macro uses the standard Lotus transpose, it cannot correctly handle ranges with formulas. If the range to be transposed contains formulas that relate to cells inside the range, use the TRANSPO2.WK1 or the TRANSPO3.WK3.

The macro starts with `{WINDOWSOFF}{PANELOFF}` which freeze screen and panel display activities. Next the macro uses the "safe technique" to temporarily assign the [Source range ?] name to the range to be processed and simultaneously uses the name as a prompt. Then the macro repeats the same technique to assign the [Target range ?] name to the target range while it uses the name as a prompt. Now that both ranges are known to the macro, the macro issues `/RTSource range ?~Target range ?~~` to transpose the source range and put the result in the target range. Last, the macro issues `/RESource range ?~` which erases the source range, and then issues `/RNDSource range ?~/RNDTarget range ?~` to delete the temporary [Source range ?] and [Target range ?] range names to leave a clean worksheet.

[7] Absolute Transpose

```
     A                B                C                D                E
 1 *---A macro to TRANSPOSE a 3-D or 2-D range that includes formulas,
 2     numbers and labels without changing the address references.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 4     the range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                      IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z             {BREAKON}
11 TRANSPOS       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{PANELON}Which range ?~{WINDOWSON}
                  {BS}{BS}{?}~{GOTO}Which range ?~
12 !              {WINDOWSOFF}{PANELOFF}/RNCWhere to ?~~/RNDWhere to ?~
                  /RNC{PANELON}Where to ?~{BS}{BS}{WINDOWSON}{?}~
                  {WINDOWSOFF}{label051}
13 !              /RTWhich range ?~Where to ?~~
14 !              {GOTO}Where to ?~/RNCWhere to ?~{BS}{BS}
15 !              .{DOWN @COLS(Which range ?)-1}
16 !              .{RIGHT @ROWS(Which range ?)-1}~
17 !              {numbera051}{number051}
18 !
19 label051       {GOTO}Which range ?~{LET counterb051,0}~
20 cont051        {LET counter1051,0}{LET hereabs051,@CELLPOINTER
                  ("address")}~
21 !              {FOR counter1051,0,@COLS(Which range ?)-1,1,labels1051}~
22 !              {LET rel051,@INFO("release")}~{IF @LEFT(rel051,1)<>"@"}
                  {GOTO}{hereabs051}~{LET counterb051,counterb051+1}~
                  {IF counterb051<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs051}~{BRANCH cont051}
23 !              {RETURN}
24 !
25 counter1051          2
26 counter1a051        13
27 labels1051     {RIGHT}{LET here1051,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a051,0,@ROWS(Which range ?)-1,1,
                  labels1a051}~{IF counter1051<@COLS(Which range ?)-1}
                  {GOTO}{here1051}~{LET counter1a051,0}~
28 !
29 here1051       $C$1
30 !
31 labels1a051    {EDIT}{HOME}'{DOWN}
32 !
33 !
34 number051      {GOTO}Which range ?~{LET counterb051,0}~
35 cont1051       {LET counter2051,0}{LET hereabs051,@CELLPOINTER
                  ("address")}~
36 !              {FOR counter2051,0,@COLS(Which range ?)-1,1,numbers3051}~
37 !              {LET rel051,@INFO("release")}~{IF @LEFT(rel051,1)<>"@"}
                  {GOTO}{hereabs051}~{LET counterb051,counterb051+1}~
                  {IF counterb051<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs051}~{BRANCH cont1051}
38 !              {GOTO}Which range ?~/RNDWhich range ?~/RNDWhere to ?~
39 !
40 counter2051          2
41 counter3051          0
42 numbers3051    {RIGHT}{LET here2051,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter3051,0,@ROWS(Which range ?)-1,1,numbers1051}~
                  {GOTO}{here2051}~{LET counter3051,0}~
43 !
44 here2051       $C$1
45 !
46 numbers1051    {EDIT}{HOME}{DEL}{DOWN}
47 !
48 numbera051     {GOTO}Where to ?~{LET counterb051,0}~
49 cont1a051      {LET counter2a051,0}{LET hereabs051,@CELLPOINTER
```

```
                ("address")}~
50 !            {FOR counter2a051,0,@COLS(Where to ?)-1,1,numbers3a051}~
51 !            {LET rel051,@INFO("release")}~{IF @LEFT(rel051,1)<>"@"}
                {GOTO}{hereabs051}~{LET counterb051,counterb051+1}~
                {IF counterb051<@SHEETS(Which range ?)}{NS}{GOTO}
                {hereabs051}~{BRANCH cont1a051}
52 !
53 counter3a051        0
54 counter2a051       13
55 !
56 numbers3a051 {RIGHT}{LET here3051,@CELLPOINTER("address")}~{LEFT}
                {FOR counter3051,0,@ROWS(Where to ?)-1,1,numbers1a051}~
                {GOTO}{here3051}~{LET counter3a051,0}~
57 !
58 here3051     $Q$1
59 !
60 numbers1a051 {EDIT}{HOME}{DEL}{DOWN}
61 !
62 counterb051        1
63 hereabs051   $A$1
64 !
65 rel051
```

Please refer to both the TRANSPO2.WK1 and the TRANSPO3.WK3. This macro is for the cases where the range to be transposed contains data and formulas that refer to cells and ranges OUTSIDE of the range to be transposed. The macro turns all the cells into labels and then transposes the range and un-labels all the cells. The result is that all the formulas are unchanged and refer to the same cells and ranges as prior to the transpose action. When you transpose a range in the worksheet and the range contains formulas, the new formulas in the transposed range are changed depends how much the rows and the columns are shifted.

This is one of the beautiful and essence properties of an electronic spreadsheet. What happens if we do not want the formulas to be updated? We can use absolute addresses, but what if we want to transpose a whole range and keep the exact formulas with reference to data outside the range? The solution is to turn all the formulas to labels, and than transpose the range. But turning a 100X100 range to labels is real punishment. The TRANSPOS.WK1 macro does the job automatically. First, it turns all the values in the range into labels, then it transposes the range to the target location and last the macro UN-LABELs the two ranges.

```
     A          B          C          D          E
11 TRANSPOS     {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                Which range ?~/RNC{PANELON}Which range ?~{WINDOWSON}
                {BS}{BS}{?}~{GOTO}Which range ?~
12 !            {WINDOWSOFF}{PANELOFF}/RNCWhere to ?~~/RNDWhere to ?~
                /RNC{PANELON}Where to ?~{BS}{BS}{WINDOWSON}{?}~
                {WINDOWSOFF}{label051}
13 !            /RTWhich range ?~Where to ?~~
14 !            {GOTO}Where to ?~/RNCWhere to ?~{BS}{BS}
15 !            .{DOWN @COLS(Which range ?)-1}
16 !            .{RIGHT @ROWS(Which range ?)-1}~
17 !            {numbera051}{number051}
```

The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the origin range to process, and simultaneously assigns the [Which range ?] range name to the same range, which acts as a prompt. Next the macro repeats the same procedure and prompts you to paint the target range to process, and simultaneously assigns the [Where to ?] range name to the target range, which also acts as a prompt. Next the macro issues the {label051} routine command which turns all the values in the origin range, the [Which range ?] into labels.

	A	B	C	D	E
19	label051	{GOTO}Which range ?~{LET counterb051,0}~			
20	cont051	{LET counter1051,0}{LET hereabs051,@CELLPOINTER ("address")}~			
21	!	{FOR counter1051,0,@COLS(Which range ?)-1,1,labels1051}~			
22	!	{LET rel051,@INFO("release")}~{IF @LEFT(rel051,1)<>"@"} {GOTO}{hereabs051}~{LET counterb051,counterb051+1}~ {IF counterb051<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs051}~{BRANCH cont051}			
23	!	{RETURN}			

This routine starts with {GOTO}Which range ?~ which moves the cell pointer to the upper left cell of the origin range. Next the macro issues {LET counterb051,0}~ to set the value in cell [counterb051], which serves as a counter, to zero. Next the macro issues {LET counter1051,0}~ to set the value in cell [counter1051], which also serves as a counter, to zero. Now the macro issues {LET hereabs051,@CELLPOINTER("address")}~ to store the address of the upper left cell of the [Which range ?] range in cell [hereabs051] for later use. The macro will use it to process all the sheets of [Which range ?] if it contains more than one sheet. To change all the values in the origin range, the macro issues {FOR counter1051 ,0,@COLS(Which range ?),1,1,labels1051}~ which activates the [labels1051] routine as many times as the number of columns in the origin range.

	A	B	C	D	E
27	labels1051	{RIGHT}{LET here1051,@CELLPOINTER("address")}~{LEFT} {FOR counter1a051,0,@ROWS(Which range ?)-1,1, labels1a051}~{IF counter1051<@COLS(Which range ?)-1} {GOTO}{here1051}~{LET counter1a051,0}~			

The [labels1051] routine uses the {RIGHT}{LET here1051,@CELLPOINTER ("address") }~{LEFT} commands to record the address of the first cell of the next column in cell [here1051]. When the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here1051]. The {FOR counter1a051,0,@ROWS(Which range ?)-1,1,labels1a051}~ macro commands activate the [labels1a051] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
31	labels1a051	{EDIT}{HOME}'{DOWN}			

The [labels1a051] routine issues {EDIT} to enter into the EDIT mode, and then {HOME} moves the cursor to the beginning of the text in the panel. The macro prefaces the content of the panel with an apostrophe "'" which changes it into a label if the cell content is a value, and adds it to an existing label if the content is a label. The {DOWN} moves the cell pointer to the next cell in the current column and simultaneously writes the content of the panel into the cell. Let us assume the cell contains the formula @ABS(-10) which appears in the cell as the number 10. After {EDIT}{HOME}', the cell shows the formula as the string of text "@ABS(-10)". If the cell contains a label, the routine adds an extra apostrophe "'" which causes the text in the cell to appear with a leading apostrophe "'". The {DOWN} moves the cell pointer to the next cell in the current column, and simultaneously writes the content of the panel into the current cell. When the [labels1a051] routine is finished, the macro returns control to the {FOR} loop in the [labels1051] routine.

	A	B	C	D	E
27	labels1051	{RIGHT}{LET here1051,@CELLPOINTER("address")}~{LEFT}			

```
{FOR counter1a051,0,@ROWS(Which range ?)-1,1,
labels1a051}~{IF counter1051<@COLS(Which range ?)-1}
{GOTO}{here1051}~{LET counter1a051,0}~
```

When the value in cell [counter1a051] reaches the number of rows in the [Which range ?] range minus one, the macro issues `{IF counter1051<@COLS(Which range ?)-1}` to check how many columns were processed. If the value in [counter1051] is less than the number of columns in [Which range ?], the macro issues the indirect `{GOTO}{here1051}~` command which moves the cell pointer to the first cell of the next column. Next the macro issues `{LET counter1a051,0}~` to reset the value in [counter1a051] to zero. When the [labels1051] routine is finished, the macro returns control to the [cont051] routine to process the next column.

	A	B	C	D	E
19	label051	{GOTO}Which range ?~{LET counterb051,0}~			
20	cont051	{LET counter1051,0}{LET hereabs051,@CELLPOINTER ("address")}~			
21	!	{FOR counter1051,0,@COLS(Which range ?)-1,1,labels1051}~			
22	!	{LET rel051,@INFO("release")}~{IF @LEFT(rel051,1)<>"@"} {GOTO}{hereabs051}~{LET counterb051,counterb051+1}~ {IF counterb051<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs051}~{BRANCH cont051}			
23	!	{RETURN}			

When the macro finishes processing all the columns in the first sheet of [Which range ?], (in a 2-D release this is the only sheet) it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues `{LET rel051,@INFO("release")}~`, which store the result of the `@INFO("release")` 3-D function in cell [rel051]. Then it issues `{IF @LEFT(rel051,1)<>"@"}`, which checks the first character of the content of [rel051]. If "@" is the character, you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the `{GOTO}hereabs051}~` indirect command moving the cell pointer to the address of the upper left cell of the first sheet of [Which range ?].

The macro continues with `{LET counterb051,counterb051+1}~` which increase the value in [counterb051] by one. Now the macro issues `{IF counterb051<@SHEETS (Which range ?)}` to check if the value in [counterb051] is less than the number of sheets in [Which range ?]. If so, there are more sheets to process, therefore the macro issues `{NS}` to move the cell pointer to the next sheet. Next the macro issues `{GOTO} {hereabs051}~` moving the cell pointer to the same address as the address of the upper left cell of the first sheet of the [Which range ?] range, but in the new sheet. Last, the macro issues `{BRANCH cont051}` to process the cells of the [Which range ?] range in the new sheet. When all the sheets of the [Which range ?] range are processed the [cont051] routine returns control to the main macro.

	A	B	C	D	E
11	TRANSPOS	{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND Which range ?~/RNC{PANELON}Which range ?~{WINDOWSON} {BS}{BS}{?}~{GOTO}Which range ?~			
12	!	{WINDOWSOFF}{PANELOFF}/RNCWhere to ?~~/RNDWhere to ?~ /RNC{PANELON}Where to ?~{BS}{BS}{WINDOWSON}{?}~ {WINDOWSOFF}{label051}			
13	!	/RTWhich range ?~Where to ?~~			
14	!	{GOTO}Where to ?~/RNCWhere to ?~{BS}{BS}			
15	!	.{DOWN @COLS(Which range ?)-1}			
16	!	.{RIGHT @ROWS(Which range ?)-1}~			
17	!	{numbera051}{number051}			

Now that all the cells in the [Which range ?] range are labels, the macro issues `/RTWhich range ?~Where to ?~` which transposes the content of the [Which range ?] range into the [Where to ?] range. Next the macro issues `{GOTO}Where to ?~` which moves the cell pointer to the upper left cell of [Where to ?]. Now the macro issues:

```
/RNCWhere to ?~{BS}{BS}.{DOWN @COLS(Which range ?)-1}.
    {RIGHT @ROWS(Which range ?)-1}~
```

to re-adjust the size of the [Where to ?] target range to the same size of the first sheet of the [Which range ?] origin range. In this range, the number of rows is the number of columns of the [Which range ?] range, and the number of columns is the number of rows in the [Which range ?] range. Notice how the periods [`.`] in the command move the cursor inside the painted range, and how the macro uses the result of the `@COLS` and the `@ROWS` function to determine how many rows to expand the target range down and how many columns to expand the target range to the right. Now we have two ranges of labels, therefore the macro issues the `{numbera051}` routine command to un-label the [Where to ?] target range, and the `{number051}` routine command to un-label the [Which range ?] origin range .

```
     A              B                C              D              E
48 numbera051     {GOTO}Where to ?~{LET counterb051,0}~
49 cont1a051      {LET counter2a051,0}{LET hereabs051,@CELLPOINTER
                  ("address")}~
50 !              {FOR counter2a051,0,@COLS(Where to ?)-1,1,numbers3a051}~
51 !              {LET rel051,@INFO("release")}~{IF @LEFT(rel051,1)<>"@"}
                  {GOTO}{hereabs051}~{LET counterb051,counterb051+1}~
                  {IF counterb051<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs051}~{BRANCH cont1a051}
```

The [numbera051] routine is built the same way as [label051]. We could probably use [label051] to do what [numbera051] does using dynamic string formulas to change the code respectively in order to make the macro shorter but more complex. However we prefer to assign a separate routine for each task. The code of this routine is identical to the code of the [label051] routine, some of the counter names are different and the routine which is activated by `{FOR counter2a051,0,@COLS(Where to ?)-1,1,numbers3a051}~` is [numbers3a051], which has the same function as the [labels1051] routine.

```
     A              B                C              D              E
56 numbers3a051   {RIGHT}{LET here3051,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter3051,0,@ROWS(Where to ?)-1,1,numbers1a051}~
                  {GOTO}{here3051}~{LET counter3a051,0}~
```

Again the macro issues `{RIGHT}{LET here3051,@CELLPOINTER("address")}~ {LEFT}` to store the address of the upper cell of the next column in [here3051]. The `{FOR counter3051,0,@ROWS(Where to ?)-1,1,numbers1a051}~` commands activate the [numbers1a051] routine as many times as the number of rows in the [Where to ?] target range.

```
     A              B                C              D              E
60 numbers1a051   {EDIT}{HOME}{DEL}{DOWN}
```

The routine issues `{EDIT}` to enter the EDIT mode and then `{HOME}` to move the cursor to the beginning of the text in the panel, and then `{DEL}` to delete the apostrophe "`'`" to turn the cell content back to value if it was a value at first, or delete the extra apostrophe "`'`" which precedes the previous label. Therefore the [numbers1a051] routine is the opposite of the [labels1a051] routine. Let us return to the [numbera051] routine.

	A	B	C	D	E
48	numbera051	{GOTO}Where to ?~{LET counterb051,0}~			
49	cont1a051	{LET counter2a051,0}{LET hereabs051,@CELLPOINTER ("address")}~			
50	!	{FOR counter2a051,0,@COLS(Where to ?)-1,1,numbers3a051}~			
51	!	{LET rel051,@INFO("release")}~{IF @LEFT(rel051,1)<>"@"} {GOTO}{hereabs051}~{LET counterb051,counterb051+1}~ {IF counterb051<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs051}~{BRANCH cont1a051}			

As we mentioned, the [numbera051] routine is built the same way as the [label051]. The code of this routine is identical to the code of the [label051] routine. Some of the counter names are different and the routine which is activated by `{FOR counter2a051,0,@COLS(Where to ?)-1,1,numbers3a051}~` is [numbers3a051] routine which has the same function as [labels1051]. The next routine is [number051]:

	A	B	C	D	E
34	number051	{GOTO}Which range ?~{LET counterb051,0}~			
35	cont1051	{LET counter2051,0}{LET hereabs051,@CELLPOINTER ("address")}~			
36	!	{FOR counter2051,0,@COLS(Which range ?)-1,1,numbers3051}~			
37	!	{LET rel051,@INFO("release")}~{IF @LEFT(rel051,1)<>"@"} {GOTO}{hereabs051}~{LET counterb051,counterb051+1}~ {IF counterb051<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs051}~{BRANCH cont1051}			
38	!	{GOTO}Which range ?~/RNDWhich range ?~/RNDWhere to ?~			

which returns the [Which range ?] range back to normal. The code of this routine is identical to the code of the [numbera051] routine. Some of the range names are different and the routine which is activated by `{FOR counter2051,0,@COLS(Which range ?)-1,1 ,numbers3051}~` is [numbers3051], which has the same function as the [numbers3a051] routine but it processes the [Which range ?] origin range instead.

	A	B	C	D	E
42	numbers3051	{RIGHT}{LET here2051,@CELLPOINTER("address")}~{LEFT} {FOR counter3051,0,@ROWS(Which range ?)-1,1,numbers1051}~ {GOTO}{here2051}~{LET counter3051,0}~			

The `{FOR counter3051,0,@ROWS(Which range ?)-1,1,numbers1051}~` code activates the [numbers1051] routine as many times as the number of rows in the [Which range ?] origin range.

	A	B	C	D	E
46	numbers1051	{EDIT}{HOME}{DEL}{DOWN}			

This routine is the same as the [numbers1a051] routine. When the macro finishes processing the [Which range ?] range, it returns to the [number051] routine.

	A	B	C	D	E
34	number051	{GOTO}Which range ?~{LET counterb051,0}~			
35	cont1051	{LET counter2051,0}{LET hereabs051,@CELLPOINTER ("address")}~			
36	!	{FOR counter2051,0,@COLS(Which range ?)-1,1,numbers3051}~			
37	!	{LET rel051,@INFO("release")}~{IF @LEFT(rel051,1)<>"@"} {GOTO}{hereabs051}~{LET counterb051,counterb051+1}~ {IF counterb051<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs051}~{BRANCH cont1051}			
38	!	{GOTO}Which range ?~/RNDWhich range ?~/RNDWhere to ?~			

When all the sheets of both ranges are processed, the macro issues `{GOTO}Which range ?` which moves the cell pointer to the upper left cell of the origin range and then issues `/RNDWhich range ?~/RNDWhere to ?~` to delete the [Which range ?] and the [Where to ?] temporary range names to leave a clean worksheet.

[9] Transpose a 2-D Range with Formulas Correctly

```
     A                B                C                D                E
 1 *---A macro to transpose a range in a way that preserves formulas too
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer on the upper leftmost cell of the range
 5     to be transposed, make sure there is at least one empty row
 6     above the range
 7 *---Hold the [MACRO] key and press [Z] to activate the macro
 8 *---The transposed range will begin at the same cell
 9 !
10 \Z               {BREAKON}
11 TRANSPO2         {WINDOWSOFF}{PANELOFF}/WGRM/RNCWhich range ?~~
                    /RNDWhich range ?~/RNC{WINDOWSON}{PANELON}
                    Which range ?~{BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}
12 !                {LET cols510,@COLS(Which range ?)}~
13 !                {LET rows510,@ROWS(Which range ?)}~
14 !                {GOTO}Which range ?~
15 !                {FOR counter1510,1,@MIN(cols510,rows510),1,rout510}
16 !                {GOTO}Which range ?~/MWhich range ?~{DOWN}~{DOWN}
                    /RNDWhich range ?~
17 !
18 rout510          {FOR counter2510,1,rows510,1,rout1510}
19 !                {RECALC firstcol2510}{RECALC firstrow2510}
                    {RECALC secondcol2510}{RECALC cont3510}
20 cont3510         {GOTO}B2~
21 !                {FOR counter3510,1,cols510-1,1,rout2510}
22 !                {RECALC firstcol3510}{RECALC firstrow3510}
                    {RECALC secondcol3510}{RECALC cont4510}
23 cont4510         {GOTO}B3~
24 !                {LET cols510,cols510-1}
25 !                {LET rows510,rows510-1}
26 !
27 rout1510         {RECALC firstcol510}{RECALC firstrow510}
                    {RECALC secondcol510}{RECALC cont1510}
28 cont1510         /M~A1~
29 !                {DOWN}
30 !
31 rout2510         {RECALC firstcol1510}{RECALC firstrow1510}
                    {RECALC secondcol1510}{RECALC cont2510}
32 cont2510         /M~A3~
33 !                {RIGHT}
34 !
35 counter1510                  3
36 !
37 counter3510                  9
38 cols510                      4
39 !
40 counter2510                  2
41 rows510                      3
42 !
43 firstcol510
44 !
45 secondcol510     B
46 !
47 firstrow510      -1
48 !
49 firstcol1510
50 !
51 secondcol1510    8
52 !
53 firstrow1510     9
54 !
55 firstcol2510
56 !
57 firstrow2510     B
58 !
59 secondcol2510    1
```

```
60 !
61 firstcol3510
62 !
63 firstrow3510    :
64 !
65 secondcol3510   2
```

As odd as it seems no spreadsheet on the market today offers the option for transposing a range also containing formulas, in such a way, that the formulas will be updated correctly. In the old releases of Lotus 1-2-3, the result of the transposition on a range containing formulas was unpredictable, and in the new releases 2.2 and up, Lotus transforms all the formulas into values. This macro is a breakthrough allowing you to transpose a range without fear of what will happen to the formulas. This macro transforms any range, and fast, and keeps all the updates of the formula references correctly.

The basis for this macro is the / **M**ove option. When a cell with a range name is moved, the name moves with the cell. Therefore, if we can move all the cell in the original range to the target range to the correct position all the formulas will be correctly updated. This macro works fast because it calculates the target address to move to and then uses the / **M**ove.. commands. Therefore the macro is extensively based on dynamic string formulas to calculate and produce the correct code.

Notice that the code in the B20, B23, B28, B32, B43, B45, B47, B49, B51, B53, B55, B57, B59, B61, B63 and the B65 cells of the macro is the result of the following dynamic string formulas. If you intend to key in the code, you must use the following formulas, NOT the code as it appears in the main listing.

```
20 cont3510       +"{GOTO}"&$B$55&$B$57&$B$59&"~"
23 cont4510       +"{GOTO}"&$B$61&$B$63&$B$65&"~"
28 cont1510       +"/M~"&B43&$B$45&B47&"~"
32 cont2510       +"/M~"&B49&$B$51&B53&"~"
43 firstcol510    @IF(@CHAR(65+(@CELLPOINTER("col")-2+B40-@MOD
                  (@CELLPOINTER("col")-2+B40,26))/26-1)="@","",@CHAR
                  (65+(@CELLPOINTER("col")-2+B40-@MOD(@CELLPOINTER("col")
                  -2+B40,26))/26-1))
45 secondcol510   @CHAR(65+@MOD(@CELLPOINTER("col")-2+B40,26))
47 firstrow510    @STRING(@CELLPOINTER("row")-B40,0)
49 firstcol1510   @IF(@CHAR(65+(@CELLPOINTER("col")-B37-1-@MOD
                  (@CELLPOINTER("col")-B37-1,26))/26-1)="@","",@CHAR(65+
                  (@CELLPOINTER("col")-B37-1-@MOD(@CELLPOINTER("col")
                  -B37-1,26))/26-1))
51 secondcol1510 @CHAR(65+@MOD(@CELLPOINTER("col")-B37-1,26))
53 firstrow1510   @STRING(@CELLPOINTER("row")+B37-1,0)
55 firstcol2510   @IF(@CHAR(65+(@CELLPOINTER("col")-@MOD(@CELLPOINTER
                  ("col"),26))/26-1)="@","",@CHAR(65+(@CELLPOINTER("col")
                  -@MOD(@CELLPOINTER("col"),26))/26-1))
57 firstrow2510   @CHAR(65+@MOD(@CELLPOINTER("col"),26))
59 secondcol2510 @STRING(@CELLPOINTER("row")-B41,0)
61 firstcol3510   @IF(@CHAR(65+(@CELLPOINTER("col")-B38-@MOD(@CELLPOINTER
                  ("col")-B38,26))/26-1)="@","",@CHAR(65+(@CELLPOINTER
                  ("col")-B38-@MOD(@CELLPOINTER("col")-B38,26))/26-1))
63 firstrow3510   @CHAR(65+@MOD(@CELLPOINTER("col")-B38,26))
65 secondcol3510 @STRING(@CELLPOINTER("row")+1,0)
```

	A	B	C	D	E
11	TRANSPO2	{WINDOWSOFF}{PANELOFF}/WGRM/RNCWhich range ?~~ /RNDWhich range ?~/RNC{WINDOWSON}{PANELON} Which range ?~{BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}			

The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint

the range to transpose, and simultaneously assigns the [Which range ?] name to the same range, which acts as a prompt.

	A	B	C	D	E
12	!	{LET cols510,@COLS(Which range ?)}~			
13	!	{LET rows510,@ROWS(Which range ?)}~			
14	!	{GOTO}Which range ?~			
15	!	{FOR counter1510,1,@MIN(cols510,rows510),1,rout510}			
16	!	{GOTO}Which range ?~/MWhich range ?~{DOWN}~{DOWN}			
		/RNDWhich range ?~			

Next, the macro issues `{LET cols510,@COLS(Which range ?)}~`, which store the number of columns in the [Which range ?] range into the B38 cell named [cols510] and then issues `{LET rows510,@ROWS(Which range ?)}~`, which store the number of rows in the [Which range ?] range into cell [rows510]. The macro continues with `{GOTO}Which range ?~`, which moves the cell pointer to the upper left cell of [Which range ?]. Now, the macro issues `{FOR counter1510,1,@MIN(cols510,rows510),1,rout510}` which activates the [rout510] routine as many times as the minimum value of the two numbers: the number of columns in cell [cols510] or the number of rows in cell [rows510]. If the number of columns is greater than the number or rows, the macro transposes the range from row form to column form. If the number of columns is less than the number rows, the macro transposes the range from column form to row form.

Before we continue with the macro code, an example will help you understand how it works. The principle behind this macro is the / **M**ove command instead of the / **C**opy command. When a cell is moved, the range name associated with it also moves. The technique is to move the first column and the first row, then the rest of the second column and the rest of the second row, then the rest of the third column and the rest of the third row, the same with the fourth, the fifth and all the columns and rows. Here is a step-by-step example of the A2..D4 range which is a 3X4 range with three rows and four columns. This is the minimum range which can demonstrate the process. Here is the original range:

	A	B	C	D	E	F	G	H
1								
2	0	3	6	9				
3	1	4	7	10				
4	2	5	8	11				

It has more columns than rows, therefore the macro will transform it into three column by four rows range. The macro uses the row above the range, therefore this row should be empty. The macro first moves the A2 cell into the A1 cell, then the A3 cell into the B1 cell and the A4 cell into the C1 cell. The result is the following range:

	A	B	C	D	E	F	G	H
1	0	1	2					
2		3	6	9				
3		4	7	10				
4		5	8	11				

We can see that the A2..A4 range is empty now (three empty cells) and can accommodate the rest of the first row, the B2..D2 range. Therefore the macro moves the B2 cell into the A2 cell, the C2 cell into the A3 cell and the D2 cell into the A4 cell. The result is:

	A	B	C	D	E	F	G	H
1	0	1	2					
2	3							

3	6	4	7	10
4	9	5	8	11

The first column became the first row and the first row became the first column. The B2..C2 range is empty, therefore it can accommodate what is left from the second column which is the B3..B4 range. Therefore the macro moves the B3 cell into the B2 cell and moves the B4 cell into the C2 cell. The result is the following range:

	A	B	C	D	E	F	G	H
1	0	1	2					
2	3	4	5					
3	6		7	10				
4	9		8	11				

Now the macro needs to move what is left from the second row (the C3..D3 range). The B3..B4 range is now empty, therefore the macro moves the C3 cell into the B3 cell and moves the D3 cell into the B4 cell. The result is the following range:

	A	B	C	D	E	F	G	H
1	0	1	2					
2	3	4	5					
3	6	7						
4	9	10	8	11				

What is left from the third column is the C4 cell, therefore the macro moves it to the empty cell C3, and then moves the last cell of the third row, the D4 cell into the C4 cell. The final result is the transposed range:

	A	B	C	D	E	F	G	H
1	0	1	2					
2	3	4	5					
3	6	7	8					
4	9	10	11					

To return the upper left cell of the range to its origin cell, the macro moves the whole A1..C4 range (named [Which range ?]) to the A2 cell.

	A	B	C	D	E	F	G	H
1								
2	0	1	2					
3	3	4	5					
4	6	7	8					
5	9	10	11					

To complete the picture, let us see step by step how the macro transposes the A2..C5 range back to the previous range. The range now has four rows and three columns, therefore the macro transforms it to a four columns by three rows range. The first step is to change the first column to become the first row, therefore the macro moves the A2 cell into the A1 cell, then moves the A3 cell into the B1 cell. Next the macro moves the A4 cell into the C1 cell and last, the macro moves the A5 cell into the D1 cell. The result is the following range:

	A	B	C	D	E	F	G	H
1	0	3	6	9				
2		1	2					
3		4	5					
4		7	8					
5		10	11					

The second step is to move the rest of the first row to the first column. Therefore the macro moves the B2 cell into the A2 cell and moves the C2 cell into the A3 cell. The result is:

	A	B	C	D	E	F	G	H
1	0	3	6	9				
2	1							
3	2	4	5					
4		7	8					
5		10	11					

Now the macro needs to move the rest of the second column (the B3..B5 range) into the second row (the B2..D2 range), therefore the macro moves the B3 cell into the B2 cell and then moves the B4 cell into the C2 cell and last moves the B5 cell into the D2 cell. The result is the following range:

	A	B	C	D	E	F	G	H
1	0	3	6	9				
2	1	4	7	10				
3	2		5					
4			8					
5			11					

Now the macro needs to move what is left from the third column to the third row. Therefore the macro moves the C3 cell into the B3 cell and then moves the C4 cell into the C3 cell and last moves the C5 cell into the D3 cell. The final result is the following range:

	A	B	C	D	E	F	G	H
1	0	3	6	9				
2	1	4	7	10				
3	2	5	8	11				

To return the upper left cell of the range to its origin cell, the macro moves the whole A1..D3 range (named [Which range ?]) to the A2 cell.

	A	B	C	D	E	F	G	H
1								
2	0	3	6	9				
3	1	4	7	10				
4	2	5	8	11				

Now let's look at the [rout510] routine:

	A	B	C	D	E
18	rout510	{FOR counter2510,1,rows510,1,rout1510}			
19	!	{RECALC firstcol2510}{RECALC firstrow2510}			
		{RECALC secondcol2510}{RECALC cont3510}			
20	cont3510	{GOTO}B2~			
21	!	{FOR counter3510,1,cols510-1,1,rout2510}			
22	!	{RECALC firstcol3510}{RECALC firstrow3510}			
		{RECALC secondcol3510}{RECALC cont4510}			
23	cont4510	{GOTO}B3~			
24	!	{LET cols510,cols510-1}			
25	!	{LET rows510,rows510-1}			

The [rout510] routine issues `{FOR counter2510,1,rows510,1,rout1510}` which executes the [rout1510] routine as many times as the number of rows in [Which range ?].

	A	B	C	D	E
27	rout1510	{RECALC firstcol510}{RECALC firstrow510}			

```
                     {RECALC secondcol510}{RECALC cont1510}
28 cont1510          /M~A1~
29 !                 {DOWN}
```

The [rout1510] routine issues `{RECALC firstcol510}{RECALC firstrow510}{RECALC secondcol510}{RECALC cont1510}` to update the <u>dynamic formulas</u> in the cells named [firstcol510], [firstrow510], [secondcol510], [cont1510] respectively. The formulas are:

```
43 firstcol510    @IF(@CHAR(65+(@CELLPOINTER("col")-2+B40-@MOD
                  (@CELLPOINTER("col")-2+B40,26))/26-1)="@","",@CHAR
                  (65+(@CELLPOINTER("col")-2+B40-@MOD(@CELLPOINTER("col")
                  -2+B40,26))/26-1))
47 firstrow510    @STRING(@CELLPOINTER("row")-B40,0)
45 secondcol510   @CHAR(65+@MOD(@CELLPOINTER("col")-2+B40,26))
28 cont1510       +"/M~"&B43&$B$45&B47&"~"
```

The formula in cell [firstcol510] calculates the first column's letter of the target cell, if the target cell's address has two characters. For example, if the current cell is the BR4 cell and it is the third cell in the first column of the range to be transposed, this formula returns the "D" character. For simplicity let us assume that the range to be transposed is the A2..D4 range. Therefore this formula returns the null string "". You are encouraged to further investigate the formulas when the range to be transposed is something like AB3..AF7 where this formula is relevant.

If the cell pointer is on the first cell of the first column, the row number is "`2`" (the cell is A2). Therefore the counter in cell [counter2510], which counts the cell number in the current column, returns the number "`1`", and the result of the formula is the string "`1`" (the result of 2-1=1). The formula in [secondcol510] returns the second column's letter which is the "A" letter for the A2 cell. The result of the formula in cell [cont1510] is the code: `/M~A1~` which moves the A2 cell into the A1 cell, exactly as we have previously seen in the example. If the current cell is the A4 cell, the counter in cell [counter2510] returns the number "`3`". Therefore the formula in cell [secondcol510] returns `@CHAR(65+1-2+3)=@CHAR(67)`, which is the "`C`" character and the formula in cell [cont1510] returns the code: `/M~C1~` which moves the A4 cell into the C1 cell.

```
   A                 B              C              D              E
18 rout510           {FOR counter2510,1,rows510,1,rout1510}
19 !                 {RECALC firstcol2510}{RECALC firstrow2510}
                     {RECALC secondcol2510}{RECALC cont3510}
20 cont3510          {GOTO}B2~
21 !                 {FOR counter3510,1,cols510-1,1,rout2510}
22 !                 {RECALC firstcol3510}{RECALC firstrow3510}
                     {RECALC secondcol3510}{RECALC cont4510}
23 cont4510          {GOTO}B3~
24 !                 {LET cols510,cols510-1}
25 !                 {LET rows510,rows510-1}
```

When the macro is finished with the first column, the `{FOR}` loop in the [rout510] routine is finished and the macro returns to the next four commands `{RECALC firstcol2510}{RECALC firstrow2510}{RECALC secondcol2510}{RECALC cont3510}` to update the dynamic formulas in cells [firstcol2510], [firstrow2510], [secondcol2510] and [cont3510] respectively.

```
55 firstcol2510   @IF(@CHAR(65+(@CELLPOINTER("col")-@MOD(@CELLPOINTER
                  ("col"),26))/26-1)="@","",@CHAR(65+(@CELLPOINTER("col")
                  -@MOD(@CELLPOINTER("col"),26))/26-1))
57 firstrow2510   @CHAR(65+@MOD(@CELLPOINTER("col"),26))
```

```
59 secondcol2510 @STRING(@CELLPOINTER("row")-B41,0)
20 cont3510      +"{GOTO}"&$B$55&$B$57&$B$59&"~"
```

The formula in B55, [firstcol2510], returns the first column's letter of the first cell of what is left from the first row in the origin range, if the cell's address has two characters. For simplicity let's assume that the range to be transposed is the A2..D4 range. This formula returns the null string "" because the "A" column is a one letter column. You are encouraged to further investigate the formula when the range is something like AB3..AF7 where this formula is relevant.

When the macro is finished with the first column, the cell pointer is located in the A5 cell. The formula in [firstrow2510] returns the second column's letter of the first cell of what is left from the first row in the origin range, or returns the first column's letter if the column is a one letter column such as the "A" column. The formula in cell [firstrow2510] returns `@CHAR(65+1)` which is the "`B`" character. The formula in [secondcol2510] returns the row number of the first cell of what is left from the first row in the origin range. Therefore the formula in [secondcol2510] returns the result of the `@STRING(5-3,0)` formula which is the "`2`" string, (the cell named [rows510] contains the number 3). The result of the formula in cell [cont3510] is the `{GOTO}B2~` code, which moves the cell pointer to the first cell of what is left from the first row.

Now the macro issues `{FOR counter3510,1,cols510-1,1,rout2510}` which activates the [rout2510] routine as many times as the number of columns in the origin range less one (because what is left from the row is the number of columns less one).

	A	B	C	D	E
31	rout2510	{RECALC firstcol1510}{RECALC firstrow1510} {RECALC secondcol1510}{RECALC cont2510}			
32	cont2510	/M~A3~			
33	!	{RIGHT}			

The [rout2510] routine issues `{RECALC firstcol1510}{RECALC firstrow1510}{RECALC secondcol1510}{RECALC cont2510}` updating the formulas in [firstcol1510], [firstrow1510], [secondcol1510] and [cont2510].

```
49 firstcol1510  @IF(@CHAR(65+(@CELLPOINTER("col")-B37-1-@MOD
                 (@CELLPOINTER("col")-B37-1,26))/26-1)="@","",@CHAR
                 (65+(@CELLPOINTER("col")-B37-1-@MOD(@CELLPOINTER
                 ("col")-B37-1,26))/26-1))
51 secondcol1510 @CHAR(65+@MOD(@CELLPOINTER("col")-B37-1,26))
53 firstrow1510  @STRING(@CELLPOINTER("row")+B37-1,0)
32 cont2510      +"/M~"&B49&$B$51&B53&"~"
```

Again the formula in cell [firstcol1510] is not relevant for our specific example because a cell address in the "A" column has only one letter in the address (The "A" letter). The counter in cell [counter3510] holds the position of the cell pointer inside the what is left from the first row after the macro has moved the first column. For example, if the cell pointer is now on the C2 cell (the second cell in the row that left from the first row) the cell [counter3510] holds the number 2. The formula in cell [firstrow1510] returns the `@STRING(2+2-1)` formula which returns the "`3`" string. The formula in [secondcol1510] returns `@CHAR(65+@MOD(3-2-1,26)` which is the `@CHAR(65)` formula which returns the "`A`" character. Therefore, the result of the formula in [cont2510] is the: `/M~A3~` code (as we have seen in the example).

	A	B	C	D	E
18	rout510	{FOR counter2510,1,rows510,1,rout1510}			
19	!	{RECALC firstcol2510}{RECALC firstrow2510}			
		{RECALC secondcol2510}{RECALC cont3510}			
20	cont3510	{GOTO}B2~			
21	!	{FOR counter3510,1,cols510-1,1,rout2510}			
22	!	{RECALC firstcol3510}{RECALC firstrow3510}			
		{RECALC secondcol3510}{RECALC cont4510}			
23	cont4510	{GOTO}B3~			
24	!	{LET cols510,cols510-1}			
25	!	{LET rows510,rows510-1}			

When the macro is finished with the first row, the second {FOR} loop in the [rout510] routine is finished and the macro issues {RECALC firstcol3510}{RECALC firstrow3510} {RECALC secondcol3510}{RECALC cont4510} updating the dynamic formulas in cells [firstcol3510], [firstrow3510], [secondcol3510] and [cont4510] respectively.

61	firstcol3510	@IF(@CHAR(65+(@CELLPOINTER("col")-B38-@MOD(@CELLPOINTER ("col")-B38,26))/26-1)="@","",@CHAR(65+(@CELLPOINTER ("col")-B38-@MOD(@CELLPOINTER("col")-B38,26))/26-1))
63	firstrow3510	@CHAR(65+@MOD(@CELLPOINTER("col")-B38,26))
65	secondcol3510	@STRING(@CELLPOINTER("row")+1,0)
23	cont4510	+"{GOTO}"&$B$61&$B$63&$B$65&"~"

Again the formula in cell [firstcol3510] is irrelevant for our specific example because a cell address in the A2..D4 range has only one letter in the address (A, B, C or D). The cell [cols510] holds the number of the columns in the original range (4 in our example). Because the macro finished processing the first row, the cell pointer is now on the E2 cell. To continue the process the cell pointer must move to the B3 cell. Therefore, the formula in cell [cont4510] should produce the {GOTO}B2~ macro command. Let us see how it is done.

The formula in [firstrow3510] is @CHAR(65+@MOD(5-4,26)), which is the @CHAR(66) formula which gives the "B" character. The formula in [secondcol3510] gives the @STRING (2+1,0) which gives the "3" string. Therefore, the formula in [cont4510] gives the {GOTO}B3~ code as expected. Now that the cell pointer moved to B3, the upper left cell of the B3..D4 range, this new range has one less column and one less row; therefore the macro issues {LET cols510,cols510-1} and {LET rows510,rows510-1} which increase the values in [cols510] and [rows510] by one.

Earlier we explained that the macro processes column and row and then again column and row until it finishes the whole range. Now we can look at it at this way: the macro processes the first column and the first row of the original range, and then the first column and the first row of the remainder range again and again. Every time the remainder range get smaller and smaller, until it disappears. Therefore the [rout510] routine can be used again and again, every time on a smaller range. This is exactly what {FOR counter1510,1,@MIN(cols510 ,rows510),1,rout510} in the main routine does. Notice that the value in [cols510] and the value in [rows510] are smaller by one with every loop of this command.

When the macro finishes processing the original range and the main {FOR} loop is finished, it issues {GOTO}Which range ?~, which moves the cell pointer to the upper left cell of the [Which range ?] range (the A1 cell in our example), and then issues /MWhich range ? ~{DOWN}~, which moves the whole [Which range ?] range (the A1..C4 range in our example) one cell down to the A2..C5 range. Then the macro issues {DOWN} to place the cell pointer on the upper left cell of the [Which range ?] range (the A2 cell in our example), and last the

macro issues `/RNDWhich range ?~` to delete the [Which range ?] range name and leave a clean worksheet.

Note: This is not an easy macro to program and understand. It is a very fine example of what one can achieve using the Lotus macro language. No spreadsheet on the market offers an option to correctly transpose a range with formulas. As you can see, the time to transpose a range is only the function of how many cells the macro needs to move and not how far they are. This is only because the macro dynamically calculates the address to move to and uses the `/M~[target address]~` command. Instead, if we used `/M~{UP [number of rows]}{RIGHT [number of columns]}~` type of code to move the cells, the time will be distance dependent. This is why this macro is fast.

[9] Transpose a 3-D Range with Formulas Correctly

```
     A              B                C                D              E
 1 *---A macro to transpose a 2-D and 3-D range in a way that preserves
 2     formulas too, make sure there is at least one empty row above the
 3     range to be transposed
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 5     the range names in this column (starts with the \Z macro name)
 6 *---Place the cell pointer on the upper leftmost cell
 7 *---Hold the [MACRO] key and press [Z] to activate the macro
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                   IT WILL WORK IN RELEASE 2.0 AND UP
10 \Z              {BREAKON}
11 TRANSPO3        {LET rel700,@INFO("release")}~{IF @LEFT(rel700,1)<>
                   "@"}/WWS/WWP
12 !               {WINDOWSOFF}{PANELOFF}/WGRM/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~~{WINDOWSOFF}{PANELOFF}
13 !               {LET counterb700,0}~
14 !               {LET colsa700,@COLS(Which range ?)}~
15 !               {LET rowsa700,@ROWS(Which range ?)}~
16 !               {LET sheets700,@SHEETS(Which range ?)}~
17 !               {GOTO}Which range ?~
18 cont6700        {LET cols700,colsa700}~
19 !               {LET rows700,rowsa700}~
20 !               {LET hereabs700,@CELLPOINTER("address")}~
21 !               {FOR counter1700,1,@MIN(cols700,rows700),1,rout700}
22 !               {LET rel700,@INFO("release")}~{IF @LEFT(rel700,1)<>
                   "@"}{GOTO}{hereabs700}~{LET counterb700,counterb700+
                   1}~{IF counterb700<sheets700}{NS}{GOTO}{hereabs700}~
                   {BRANCH cont6700}
23 !               {GOTO}Which range ?~/MWhich range ?~{DOWN}~{DOWN}
                   /RNDWhich range ?~
24 !
25 !
26 rout700         {FOR counter2700,1,rows700,1,rout1700}
27 !               {RECALC firstcol2700}{RECALC firstrow2700}
                   {RECALC secondcol2700}{RECALC cont3700}
28 cont3700        {GOTO}C75~~
29 !               {FOR counter3700,1,cols700-1,1,rout2700}
30 !               {RECALC firstcol3700}{RECALC firstrow3700}
                   {RECALC secondcol3700}{RECALC cont4700}
31 cont4700        {GOTO}=76~~
32 !               {LET cols700,cols700-1}
33 !               {LET rows700,rows700-1}
34 !
35 rout1700        {RECALC firstcol700}{RECALC firstrow700}
                   {RECALC secondcol700}{RECALC cont1700}
36 cont1700        /M~B-1~~
37 !               {DOWN}
38 !
39 rout2700        {RECALC firstcol1700}{RECALC firstrow1700}
                   {RECALC secondcol1700}{RECALC cont2700}
40 cont2700        /M~:7~
41 !               {RIGHT}
42 !
43 counter1700              3
44 !
45 counter3700              7
46 cols700                  6
47 !
48 counter2700              2
49 rows700                  0
50 !
51 firstcol700
52 !
53 secondcol700    B
54 !
55 firstrow700     -1
```

```
56 !
57 firstcol1700
58 !
59 secondcol1700   :
60 !
61 firstrow1700    7
62 !
63 firstcol2700
64 !
65 firstrow2700    B
66 !
67 secondcol2700   1
68 !
69 firstcol3700
70 !
71 firstrow3700    <
72 !
73 secondcol3700   2
74 !
75 counterb700                3
76 hereabs700      $B$3
77 !
78 rel700          1.00.00
79 !
80 counterc700                0
81 !
82 colsa700                   8
83 !
84 rowsa700                   2
85 !
86 sheetsa700                 3
87 !
88 sheets700                  3
```

Notice that the code in the B28, B31, B36, B40, B51, B53, B55, B57, B59, B61, B63, B65, B67, 69, 71 and the B73 cells of the macro is the result of the following dynamic string formulas. If you intend to key in the code, you must use the following formulas, NOT the code as it appears in the main listing.

```
28 cont3700        +"{GOTO}"&$B$63&$B$65&$B$67&"~~"
31 cont4700        +"{GOTO}"&$B$69&$B$71&$B$73&"~~"
36 cont1700        +"/M~"&B51&$B$53&B55&"~~"
40 cont2700        +"/M~"&B57&$B$59&B61&"~"
51 firstcol700     @IF(@CHAR(65+(@CELLPOINTER("col")-2+B48-@MOD
                   (@CELLPOINTER("col")-2+B48,26))/26-1)="@","",@CHAR
                   (65+(@CELLPOINTER("col")-2+B48-@MOD(@CELLPOINTER
                   ("col")-2+B48,26))/26-1))
53 secondcol700    @CHAR(65+@MOD(@CELLPOINTER("col")-2+B48,26))
55 firstrow700     @STRING(@CELLPOINTER("row")-B48,0)
57 firstcol1700    @IF(@CHAR(65+(@CELLPOINTER("col")-B45-1-@MOD
                   (@CELLPOINTER("col")-B45-1,26))/26-1)="@","",@CHAR
                   (65+(@CELLPOINTER("col")-B45-1-@MOD(@CELLPOINTER
                   ("col")-B45-1,26))/26-1))
59 secondcol1700   @CHAR(65+@MOD(@CELLPOINTER("col")-B45-1,26))
61 firstrow1700    @STRING(@CELLPOINTER("row")+B45-1,0)
63 firstcol2700    @IF(@CHAR(65+(@CELLPOINTER("col")-@MOD(@CELLPOINTER
                   ("col"),26))/26-1)="@","",@CHAR(65+(@CELLPOINTER
                   ("col")-@MOD(@CELLPOINTER("col"),26))/26-1))
65 firstrow2700    @CHAR(65+@MOD(@CELLPOINTER("col"),26))
67 secondcol2700   @STRING(@CELLPOINTER("row")-B49,0)
69 firstcol3700    @IF(@CHAR(65+(@CELLPOINTER("col")-B46-@MOD
                   (@CELLPOINTER("col")-B46,26))/26-1)="@","",@CHAR(65+
                   (@CELLPOINTER("col")-B46-@MOD(@CELLPOINTER("col")-
                   B46,26))/26-1))
71 firstrow3700    @CHAR(65+@MOD(@CELLPOINTER("col")-B46,26))
73 secondcol3700   @STRING(@CELLPOINTER("row")+1,0)
```

This macro is an improved version of the TRANSPO2.WK1 macro that works in 3-D and 2-D

releases of 2.0 and up. While the TRANSPO2.WK1 macro works in the 2-D releases of 2.0 and up, and works in the 3-D releases as long as the range to be transposed is on the same sheet as the macro. Therefore, it is unnecessary to explain the code of this macro which is identical to the code of the TRANSPO2.WK1 macro. The only difference is the treatment of the 3-D sheets, therefore we are going to explore these additions.

	A	B	C	D	E
11	TRANSPO3	{LET rel700,@INFO("release")}~{IF @LEFT(rel700,1)<> "@"}/WWS/WWP			
12	!	{WINDOWSOFF}{PANELOFF}/WGRM/RNCWhich range ?~~/RND Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~ {BS}{BS}{?}~~{WINDOWSOFF}{PANELOFF}			

The first addition is the `{LET rel700,@INFO("release")}~` macro command which stores the result of the `@INFO("release")` 3-D function in cell [rel700]. Later, the macro uses this result to determine if you are using a 3-D or a 2-D Lotus release. The next `{IF @LEFT(rel700,1)<>"@"}` command immediately checks if the first character of the content of [rel700] is equal to the "`@`" character. If so, you are using a 3-D Lotus release, therefore the macro issues `/WWS` which synchronize scrolling between the sheets and then issues `/WWP` to enter to the windows perspective mode, where you can see three sheets at a time. In the next line, the macro issues `/WGRM` to enter manual recalculation mode to gain speed, crucial for 3-D releases which are slower than 2-D releases.

The next addition to the code is `{LET counterb700,0}~` in the B13 cell, which set the cell [counterb700] as a counter for the number of sheets to process and sets the value in this counter to zero. Next the macro issues `{LET sheets700,@SHEETS(Which range ?)}~` (see the B16 cell), which store the number of sheets of the range named [Which range ?] in cell [sheetsa700]. Next the macro issues `{LET hereabs700,@CELLPOINTER ("address")}~` which store the current cell pointer address in cell [hereabs700] (the address of the upper left cell of the first sheet of the [Which range ?] range). The macro uses this address to move in straight line in the "Z" direction from sheet to sheet.

	A	B	C	D	E
21	!	{FOR counter1700,1,@MIN(cols700,rows700),1,rout700}			
22	!	{LET rel700,@INFO("release")}~{IF @LEFT(rel700,1)<> "@"}{GOTO}{hereabs700}~{LET counterb700,counterb700+ 1}~{IF counterb700<sheets700}{NS}{GOTO}{hereabs700}~ {BRANCH cont6700}			

When the `{FOR counter1700,1,@MIN(cols700,rows700),1,rout700}` loop command is finished with the first sheet, the macro issues `{IF @LEFT(rel700,1)<> "@"}`. If this is true, you are using a 3-D release, therefore the [Which range ?] range may contain more than one sheet. Next the macro issues the `{GOTO}{hereabs700}~` indirect macro command which moves the cell pointer to the origin address of the macro.

The macro continues with `{LET counterb700,counterb700+1}~` which increase the value in cell [counterb700] by one. Now the macro issues `{IF counterb700<@SHEETS (Which range ?)}` to check if the value in [counterb700] is less than the number of sheets in [Which range ?] . If so, there are more sheets to process, therefore the macro issues `{NS}` to move the cell pointer to the next sheet. Next the macro issues the indirect `{GOTO} {hereabs700}~` command, which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of the [Which range ?] range, but in the new sheet. Last, the macro issues `{BRANCH cont6700}` to loop back and process the next sheet in [Which range ?] again and

again, until the macro finishes processing all the sheets in [Which range ?.

[9] Erase the Worksheet Except for a Specified Range

```
     A                B                C                D                E
 1 *---A macro to ERASE all the worksheet EXCEPT a specified range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7             In Release 3 use it for single sheet at a time
 8 !
 9 !
10 \Z            {BREAKON}
11 RANGKEEP      {WINDOWSOFF}{PANELOFF}/RNCKeep range ?~~/RNDKeep range ?~
                 /RNC{WINDOWSON}{PANELON}Keep range ?~{BS}{BS}{?}~
                 {WINDOWSOFF}{GOTO}Keep range ?~
12 !             {LET hereabs529,@CELLPOINTER("address")}~
13 !             /RNCKeep range ?~.{LET botleft1529,@CELLPOINTER
                 ("address")}.{LET upleft1529,@CELLPOINTER("address")}.
                 {LET upright1529,@CELLPOINTER("address")}.
                 {LET botright1529,@CELLPOINTER("address")}~
14 !             {IF upleft1529="$A$1"}{edge1529}{eraseall1529}
                 {BRANCH cont529}
15 !             {IF upright1529="$IV$1"}{edge2529}{eraseall5529}
                 {BRANCH cont529}
16 !             {IF botleft1529="$A$8192"}{edge3529}{eraseall6529}
                 {BRANCH cont529}
17 !             {IF botright1529="$IV$8192"}{edge4529}{eraseall7529}
                 {BRANCH cont529}
18 !             {IF @LEFT(upleft1529,@FIND("$",upleft1529,1))="$A"}
                 {side1529}
                 {eraseall2529}{BRANCH cont529}
19 !             {IF @LEFT(upright1529,@FIND("$",upright1529,1))="$IV"}
                 {side2529}{eraseall3529}{BRANCH cont529}
20 !             {IF @RIGHT(upright1529,@LENGTH(upright1529)-@FIND("$",
                 upright1529,1))="$1"}{side3529}{eraseall4529}
                 {BRANCH cont529}
21 !             {IF @RIGHT(botleft1529,@LENGTH(botleft1529)@FIND("$"
                 ,botleft1529,1))="$8192"}{side4529}{eraseall9529}
                 {BRANCH cont529}
22 !             /RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER
                 ("address")}{U}{R}.{L}{U}{LET upleft529,@CELLPOINTER
                 ("address")}{R}{D}.{R}{U}{LET upright529,@CELLPOINTER
                 ("address")}{L}{D}.{R}{D}{LET botright529,@CELLPOINTER
                 ("address")}{L}{U}~
23 !             {recalc529}{eraseall529}
24 cont529
25 !
26 hereabs529    $A$3
27 upleft529     $B$8179
28 upright529    $F$8179
29 botleft529    $B$8192
30 botright529   $F$8192
31 upleft1529    $A$3
32 upright1529   $B$3
33 botleft1529   $A$10
34 botright1529  $B$10
35 !
36 eraseall529   /RE$B$8179..A1~/RE$B$8179..A8192~/RE$B$8179..IV1~
                 /RE$F$8192..A8192~/RE$F$8192..IV1~/RE$F$8192..IV8192~
37 !
38 eraseall2529  /RE$B$8179..IV1~/RE$B$8192..IV8192~/RE$F$8192..IV8192~
                 /RE$F$8192..IV1~
39 !
40 eraseall3529  /RE$F$8179..A1~/RE$F$8192..A8192~/RE$B$8179..A1~
                 /RE$B$8179..A8192~
41 !
42 eraseall4529  /RE$B$8179..A8192~/RE$B$8192..IV8192~/RE$F$8192..IV1~
43 !
```

```
44 eraseall9529 /RE$B$8192..A1~/RE$B$8179..IV1~/RE$F$8179..IV8192~
45 !
46 eraseall1529 /RE$B$8192..IV8192~/RE$F$8179..IV8192~
47 !
48 eraseall5529 /RE$F$8192..A8192~/RE$B$8179..A8192~
49 !
50 eraseall6529 /RE$B$8179..IV1~/RE$F$8192..IV1~
51 !
52 eraseall7529 /RE$F$8179..A1~/RE$B$8192..A1~
53 !
54 edge1529     /RNCKeep range ?~.{D}{LET botleft529,@CELLPOINTER
               ("address")}{U}.{LET upleft529,@CELLPOINTER("address")}
               .{R}{LET upright529,@CELLPOINTER("address")}{L}.{R}{D}
               {LET botright529,@CELLPOINTER("address")}{L}{U}~
               {recalc529}
55 !
56 edge2529     /RNCKeep range ?~.{L}{D}{LET  botleft529,@CELLPOINTER
               ("address")}{U}{R}.{L}{LET upleft529,@CELLPOINTER
               ("address")}{R}.{LET upright529,@CELLPOINTER("address")}
               .{D}{LET botright529,@CELLPOINTER("address")}{U}~
               {recalc529}
57 !
58 edge3529     /RNCKeep range ?~.{LET botleft529,@CELLPOINTER("address")
               }.{U}{LET upleft529,@CELLPOINTER("address")}{D}.{R}{U}
               {LET upright529,@CELLPOINTER("address")}{L}{D}.{R}
               {LET botright529,@CELLPOINTER("address")}{L}~{recalc529}
59 !
60 edge4529     /RNCKeep range ?~.{L}{LET botleft529,@CELLPOINTER
               ("address")}{R}.{L}{U}{LET upleft529,@CELLPOINTER
               ("address")}{R}{D}.{U}{LET upright529,@CELLPOINTER
               ("address")}{D}.{LET botright529,@CELLPOINTER("address")}
               ~{recalc529}
61 !
62 recalc529    {RECALC eraseall529}{RECALC eraseall1529}
               {RECALC eraseall2529}{RECALC eraseall3529}
               {RECALC eraseall4529}{RECALC eraseall5529}
               {RECALC eraseall6529}{RECALC eraseall7529}
               {RECALC eraseall9529}
63 !
64 side1529     /RNCKeep range ?~.{D}{LET botleft529,@CELLPOINTER
               ("address")}{U}.{U}{LET upleft529,@CELLPOINTER("address")
               }{D}.{R}{U}{LET upright529,@CELLPOINTER("address")}{L}{D}
               .{R}{D}{LET botright529,@CELLPOINTER("address")}{L}{U}~
               {recalc529}
65 !
66 side2529     /RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER
               ("address")}{U}{R}.{L}{U}{LET upleft529,@CELLPOINTER
               ("address")}{R}{D}.{U}{LET upright529,@CELLPOINTER
               ("address")}{D}.{D}{LET botright529,@CELLPOINTER
               ("address")}{U}~{recalc529}
67 !
68 side3529     /RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER
               ("address")}{R}{U}.{L}{LET upleft529,@CELLPOINTER
               ("address")}{R}.{R}{LET upright529,@CELLPOINTER
               ("address")}{L}.{R}{D}{LET botright529,@CELLPOINTER
               ("address")}{L}{U}~{recalc529}
69 !
70 side4529     /RNCKeep range ?~.{L}{LET botleft529,@CELLPOINTER
               ("address")}{R}.{L}{U}{LET upleft529,@CELLPOINTER
               ("address")}{R}{D}.{U}{R}{LET upright529,@CELLPOINTER
               ("address")}{D}{L}.{R}{LET botright529,@CELLPOINTER
               ("address")}{L}~{recalc529}
```

When you build an application in Lotus 1-2-3, the worksheet eventually contains a lot of data which you no longer need. To erase the unnecessary data, you need to manually move all over the worksheet, the more scattered the data, the more time it takes. There is no spreadsheet that I know of that can erase all the worksheet except a range, which you choose to keep, in one command. This is exactly what this macro can do. This macro prompts you to highlight the

range to keep, then the macro erases all the cells surrounding the painted range in one operation. Notice, that the code in the B36, B38, B40, B42, B44, B46, B48, B50 and B52 cells of the macro is the result of the following dynamic string formulas. If you intend to key in the code, you should use the formulas, NOT the code as it appears in the main listing.

```
36 eraseall529  +"/RE"&B27&"..A1~/RE"&B27&"..A8192~/RE"&B27&"..IV1~
                /RE"&B30&"..A8192~/RE"&B30&"..IV1~/RE"&B30&"..IV8192~"
38 eraseall2529 +"/RE"&B27&"..IV1~/RE"&B29&"..IV8192~/RE"&B30&"..IV8192~
                /RE"&B30&"..IV1~"
40 eraseall3529 +"/RE"&B28&"..A1~/RE"&B30&"..A8192~/RE"&B27&"..A1~
                /RE"&B27&"..A8192~"
42 eraseall4529 +"/RE"&B27&"..A8192~/RE"&B29&"..IV8192~/RE"&B30&"..IV1~"
44 eraseall9529 +"/RE"&B29&"..A1~/RE"&B27&"..IV1~/RE"&B28&"..IV8192~"
46 eraseall1529 +"/RE"&B29&"..IV8192~/RE"&B28&"..IV8192~"
48 eraseall5529 +"/RE"&B30&"..A8192~/RE"&B27&"..A8192~"
50 eraseall6529 +"/RE"&B27&"..IV1~/RE"&B30&"..IV1~"
52 eraseall7529 +"/RE"&B28&"..A1~/RE"&B29&"..A1~"
```

	A	B	C	D	E
11	RANGKEEP	{WINDOWSOFF}{PANELOFF}/RNCKeep range ?~~/RNDKeep range ?~ /RNC{WINDOWSON}{PANELON}Keep range ?~{BS}{BS}{?}~ {WINDOWSOFF}{GOTO}Keep range ?~			
12	!	{LET hereabs529,@CELLPOINTER("address")}~			
13	!	/RNCKeep range ?~.{LET botleft1529,@CELLPOINTER ("address")}.{LET upleft1529,@CELLPOINTER("address")}. {LET upright1529,@CELLPOINTER("address")}. {LET botright1529,@CELLPOINTER("address")}~			

The macro issues the {WINDOWSOFF}{PANELOFF} macro commands to freeze the screen and panel display activity. Then the macro uses the "safe technique" to prompt you to paint the range to delete and simultaneously assigns the [Keep range ?] name to the same range which it uses as a prompt. Next, the macro issues {LET hereabs529,@CELLPOINTER ("address")}~ to store the origin location before the macro starts. The macro uses this data later to return to this cell. Now the macro uses the fact that you painted the necessary range to find the addresses of its corners. The macro starts the / **R**ange **N**ame **C**reate process. While in this process, the macro uses the four {LET ...} commands to store the addresses of the range's corners.

The macro starts with /RNCKeep range ?~ which paints the [Keep range ?] range and then issues the period [.] key which moves the cursor to the lower left corner of the painted range, then the macro issues {LET botleft1529,@CELLPOINTER("address")} which stores the address of the lower left corner of the painted range in cell [botleft1529]. Next the macro issues the period [.] key which moves the cursor to the upper left corner of the painted range, and then issues {LET upleft1529,@CELLPOINTER("address")}, which stores the address of the upper left corner of the painted range in cell [upleft1529].

Again the macro issues the period [.] key which moves the cursor to the upper right corner of the painted range, and then issues {LET upright1529,@CELLPOINTER("address")} which stores the address of the upper right corner of the painted range in cell [upright1529]. Last, the macro again issues the period [.] key, which moves the cursor to the lower right corner of the painted range, and then issues {LET botright1529,@CELLPOINTER ("address")} which stores the address of the lower right corner of the painted range in cell [botright1529].

Now, the macro starts a series of {IF} commands to cover the different cases of the range to keep.

```
     A                B                C                D                E
14 !            {IF upleft1529="$A$1"}{edge1529}{eraseall1529}
                {BRANCH cont529}
15 !            {IF upright1529="$IV$1"}{edge2529}{eraseall5529}
                {BRANCH cont529}
16 !            {IF botleft1529="$A$8192"}{edge3529}{eraseall6529}
                {BRANCH cont529}
17 !            {IF botright1529="$IV$8192"}{edge4529}{eraseall7529}
                {BRANCH cont529}
18 !            {IF @LEFT(upleft1529,@FIND("$",upleft1529,1))="$A"}
                {side1529}
                {eraseall2529}{BRANCH cont529}
19 !            {IF @LEFT(upright1529,@FIND("$",upright1529,1))="$IV"}
                {side2529}{eraseall3529}{BRANCH cont529}
20 !            {IF @RIGHT(upright1529,@LENGTH(upright1529)-@FIND("$",
                upright1529,1))="$1"}{side3529}{eraseall4529}
                {BRANCH cont529}
21 !            {IF @RIGHT(botleft1529,@LENGTH(botleft1529)@FIND("$"
                ,botleft1529,1))="$8192"}{side4529}{eraseall9529}
                {BRANCH cont529}
```

The first case is when the range starts at the A1 cell.

```
     A                B                C                D                E
14 !            {IF upleft1529="$A$1"}{edge1529}{eraseall1529}
                {BRANCH cont529}
```

The macro issues `{IF upleft1529="$A$1"}` to check if the A1 cell is the upper left corner of the [Keep range ?] range. If so, then there is no need to erase data left and up to the range. Therefore the macro issues the `{edge1529}` routine command which activates the [edge1529] routine.

```
     A                B                C                D                E
54 edge1529     /RNCKeep range ?~.{D}{LET botleft529,@CELLPOINTER
                ("address")}{U}.{LET upleft529,@CELLPOINTER("address")}
                .{R}{LET upright529,@CELLPOINTER("address")}{L}.{R}{D}
                {LET botright529,@CELLPOINTER("address")}{L}{U}~
                {recalc529}
```

Now, the macro uses the same technique as before to record the cell addresses outside the [Keep range ?] range, and issues the `{recalc529}` routine command.

```
     A                B                C                D                E
62 recalc529    {RECALC eraseall529}{RECALC eraseall1529}
                {RECALC eraseall2529}{RECALC eraseall3529}
                {RECALC eraseall4529}{RECALC eraseall5529}
                {RECALC eraseall6529}{RECALC eraseall7529}
                {RECALC eraseall9529}
```

The [recalc529] routine uses `{RECALC ...}` nine times to update the dynamic formulas in the [eraseallxxxx] cells. The relevant formula for this case is the formula in the [eraseall1529] cell; therefore the macro returns to the main code

```
     A                B                C                D                E
14 !            {IF upleft1529="$A$1"}{edge1529}{eraseall1529}
                {BRANCH cont529}
```

and issues `{eraseall1529}`. The code in B46, [eraseall1529], is the result of the

```
+"/RE"&B29&"..IV8192~/RE"&B28&"..IV8192~"
```

formula. For example, if the range to keep is the A1..F10 range, then this formula returns the following code:

```
/RE$A$11..IV8192~/RE$G$1..IV8192~
```

This erases all of the worksheet except the A1..F10 range. Next, the macro issues `{BRANCH cont529}`, which routes the macro execution to an empty cell and quits. The second case is when the range contains the IV1 cell.

	A	B	C	D	E
15	!	{IF upright1529="$IV$1"}{edge2529}{eraseall5529} {BRANCH cont529}			

The macro issues `{IF upright1529="$IV$1"}` to check if the IV1 cell is the upper right corner of the [Keep range ?] range. If so, then there is no need to erase data right and up to the range. Therefore the macro issues `{edge2529}` which activates the [edge2529] routine

	A	B	C	D	E
56	edge2529	/RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER ("address")}{U}{R}.{L}{LET upleft529,@CELLPOINTER ("address")}{R}.{LET upright529,@CELLPOINTER("address")} .{D}{LET botright529,@CELLPOINTER("address")}{U}~ {recalc529}			

Now, the macro uses the same technique as before to record the cell addresses outside the [Keep range ?] range, and issues `{recalc529}` which we have seen earlier. The relevant formula for this case is the formula in cell [eraseall5529], therefore the macro returns to the main code

	A	B	C	D	E
15	!	{IF upright1529="$IV$1"}{edge2529}{eraseall5529} {BRANCH cont529}			

and issues `{eraseall5529}`. The code in B48, [eraseall5529], is the result of the

```
+"/RE"&B30&"..A8192~/RE"&B27&"..A8192~"
```

formula. For example, if the range to keep is the IR1..IV10 range, then this formula returns the following code:

```
/RE$IV$11..A8192~/RE$IQ$1..A8192~
```

This erases all the worksheet except the IR1..IV10 range. Next, the macro issues `{BRANCH cont529}`, which routes the macro execution to an empty cell and quits. The third case is when the range contains the A8192 cell.

	A	B	C	D	E
16	!	{IF botleft1529="$A$8192"}{edge3529}{eraseall6529} {BRANCH cont529}			

The macro issues `{IF botleft1529="$A$8192"}` to check if the A8192 cell is the lower left corner of the [Keep range ?] range. If so, then there is no need to erase data to the left and down to the range. Therefore the macro issues `{edge3529}` which activates the [edge3529] routine

	A	B	C	D	E
58	edge3529	/RNCKeep range ?~.{LET botleft529,@CELLPOINTER("address") }.{U}{LET upleft529,@CELLPOINTER("address")}{D}.{R}{U} {LET upright529,@CELLPOINTER("address")}{L}{D}.{R} {LET botright529,@CELLPOINTER("address")}{L}~{recalc529}			

Now, the macro uses the same technique as before to record the cell addresses outside the [Keep range ?] range, and issues `{recalc529}` which we have seen earlier. The relevant formula for this case is the formula in cell [eraseall6529], therefore the macro returns to the main code

	A	B	C	D	E
16	!	{IF botleft1529="$A$8192"}{edge3529}{eraseall6529} {BRANCH cont529}			

and issues `{eraseall6529}`. The code in B50, [eraseall6529], is the result of the

```
+"/RE"&B27&"..IV1~/RE"&B30&"..IV1~"
```

formula. For example, if the range to keep is the A8183..F8192 range, then this formula returns the following code:

```
/RE$A$8182..IV1~/RE$G$8192..IV1~
```

This erases all the worksheet except the A8183..F8192 range. Next, the macro issues `{BRANCH cont529}`, which routes the macro execution to an empty cell and quits. The fourth case is when the range contains the IV8192 cell.

	A	B	C	D	E
17	!	{IF botright1529="$IV$8192"}{edge4529}{eraseall7529} {BRANCH cont529}			

The macro issues `{IF botright1529="$IV$8192"}` to check if the IV8192 cell is the lower right corner of the [Keep range ?] range. If so, then there is no need to erase data to the right and down to the range. Therefore the macro issues `{edge4529}` which activates the [edge4529] routine

	A	B	C	D	E
60	edge4529	/RNCKeep range ?~.{L}{LET botleft529,@CELLPOINTER ("address")}{R}.{L}{U}{LET upleft529,@CELLPOINTER ("address")}{R}{D}.{U}{LET upright529,@CELLPOINTER ("address")}{D}.{LET botright529,@CELLPOINTER("address")} ~{recalc529}			

Now, the macro uses the same technique as before to record the cell addresses outside the [Keep range ?] range, and issues `{recalc529}` which we have seen earlier. The relevant formula for this case is the formula in cell [eraseall7529], therefore the macro returns to the main code

	A	B	C	D	E
17	!	{IF botright1529="$IV$8192"}{edge4529}{eraseall7529} {BRANCH cont529}			

and issues `{eraseall7529}`. The code in B52, [eraseall7529], is the result of the

```
+"/RE"&B28&"..A1~/RE"&B29&"..A1~"
```

formula. For example, if the range to keep is the IR8183..IV8192 range, then this formula returns the following code:

```
/RE$IV$8182..A1~/RE$IQ$8192..A1~
```

This erases all the worksheet except the IR8183..IV8192 range. Next, the macro issues `{BRANCH cont529}`, which routes the macro execution to an empty cell and quits. The fifth case is when the left side of the range is on the A column.

	A	B	C	D	E
18	!	{IF @LEFT(upleft1529,@FIND("$",upleft1529,1))="$A"} {side1529}{eraseall2529}{BRANCH cont529}			

The macro issues `{IF @LEFT(upleft1529,@FIND("$",upleft1529,1))="$A"}` check if the left side of the [Keep range ?] range is on the A column of the worksheet. If so, then there is no need to erase data to the left of the range. Therefore the macro issues `{side1529}` which activates the [side1529] routine

	A	B	C	D	E
64	side1529	/RNCKeep range ?~.{D}{LET botleft529,@CELLPOINTER ("address")}{U}.{U}{LET upleft529,@CELLPOINTER("address") }{D}.{R}{U}{LET upright529,@CELLPOINTER("address")}{L}{D} .{R}{D}{LET botright529,@CELLPOINTER("address")}{L}{U}~ {recalc529}			

Now, the macro uses the same technique as before to record the cell addresses outside the [Keep range ?] range, and issues `{recalc529}` which we have seen earlier. The relevant formula for this case is the formula in cell [eraseall2529], therefore the macro returns to the main code

	A	B	C	D	E
18	!	{IF @LEFT(upleft1529,@FIND("$",upleft1529,1))="$A"} {side1529}{eraseall2529}{BRANCH cont529}			

and issues `{eraseall2529}`. The code in B38, [eraseall2529], is the result of the

```
+"/RE"&B27&"..IV1~/RE"&B29&"..IV8192~/RE"&B30&"..IV8192~/RE"
   &B30&"..IV1~"
```

formula. For example, if the range to keep is the A10..F20 range, then this formula returns the following code:

```
/RE$A$9..IV1~/RE$A$21..IV8192~/RE$G$21..IV8192~/RE$G$21..IV1~
```

This erases all the worksheet except the A10..F20 range. Next, the macro issues `{BRANCH cont529}`, which routes the macro execution to an empty cell and quits. The sixth case is when the right side of the range is on the IV column.

	A	B	C	D	E
19	!	{IF @LEFT(upright1529,@FIND("$",upright1529,1))="$IV"} {side2529}{eraseall3529}{BRANCH cont529}			

The macro issues `{IF @LEFT(upright1529,@FIND("$",upright1529,1))="$IV"}` to check if the right side of the [Keep range ?] range is on the IV column of the worksheet. If so,

then there is no need to erase data to the right of the range. Therefore the macro issues {side2529} which activates the [side2529] routine

	A	B	C	D	E
66	side2529	/RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER ("address")}{U}{R}.{L}{U}{LET upleft529,@CELLPOINTER ("address")}{R}{D}.{U}{LET upright529,@CELLPOINTER ("address")}{D}.{D}{LET botright529,@CELLPOINTER ("address")}{U}~{recalc529}			

Now, the macro uses the same technique as before to record the cell addresses outside the [Keep range ?] range, and issues {recalc529} which we have seen earlier. The relevant formula for this case is the formula in cell [eraseall3529], therefore the macro returns to the main code

	A	B	C	D	E
19	!	{IF @LEFT(upright1529,@FIND("$",upright1529,1))="$IV"} {side2529}{eraseall3529}{BRANCH cont529}			

and issues {eraseall3529}. The code in B40, [eraseall3529], is the result of the

```
+"/RE"&B28&"..A1~/RE"&B30&"..A8192~/RE"&B27&"..A1~/RE"&B27
   &"..A8192~"
```

formula. For example, if the range to keep is the IR10..IV20 range, then this formula returns the following code:

```
/RE$IV$9..A1~/RE$IV$21..A8192~/RE$IQ$9..A1~/RE$IQ$9..A8192~
```

This erases all the worksheet except the IR10..IV20 range. Next, the macro issues {BRANCH cont529}, which routes the macro execution to an empty cell and quits. The seventh case is when the upper side of the range is on the first row of the worksheet.

	A	B	C	D	E
20	!	{IF @RIGHT(upright1529,@LENGTH(upright1529)-@FIND("$", upright1529,1))="$1"}{side3529}{eraseall4529} {BRANCH cont529}			

The macro issues the

```
{IF @RIGHT(upright1529,@LENGTH(upright1529)-@FIND("$",
    upright1529,1))="$1"}
```

macro command to check if the upper side of the [Keep range ?] range is on the first row of the worksheet. If so, then there is no need to erase data above the range. Therefore the macro issues {side3529} which activates the [side3529] routine

	A	B	C	D	E
68	side3529	/RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER ("address")}{R}{U}.{L}{LET upleft529,@CELLPOINTER ("address")}{R}.{R}{LET upright529,@CELLPOINTER ("address")}{L}.{R}{D}{LET botright529,@CELLPOINTER ("address")}{L}{U}~{recalc529}			

Now the macro uses the same technique as before to record the cell addresses outside the [Keep range ?] range, and issues {recalc529} which we have seen earlier. The relevant formula for this case is the formula in the [eraseall4529] cell, therefore the macro returns to

the main code

```
     A                B                C                D                E
20 !                {IF @RIGHT(upright1529,@LENGTH(upright1529)-@FIND("$",
                    upright1529,1))="$1"}{side3529}{eraseall4529}
                    {BRANCH cont529}
```

and issues {eraseall4529}. The code in B42, [eraseall4529], is the result of the

```
+"/RE"&B27&"..A8192~/RE"&B29&"..IV8192~/RE"&B30&"..IV1~"
```

formula. For example, if the range to keep is the F1..K10 range, then this formula returns the following code:

```
/RE$E$1..A8192~/RE$E$11..IV8192~/RE$L$11..IV1~
```

This erases all the worksheet except the F1..K10 range. The eighth case is when the lower side of the range is on the last row of the worksheet.

```
     A                B                C                D                E
21 !                {IF @RIGHT(botleft1529,@LENGTH(botleft1529)@FIND("$"
                    ,botleft1529,1))="$8192"}{side4529}{eraseall9529}
                    {BRANCH cont529}
```

The macro issues the

```
{IF @RIGHT(botleft1529,@LENGTH(botleft1529)@FIND("$",
    botleft1529,1))="$8192"}
```

macro command to check if the lower side of the [Keep range ?] range is on the last row of the worksheet. If this is true, then there is no need to erase data below the range. Therefore the macro issues {side4529} which activates the [side4529] routine

```
     A                B                C                D                E
70 side4529         /RNCKeep range ?~.{L}{LET botleft529,@CELLPOINTER
                    ("address")}{R}.{L}{U}{LET upleft529,@CELLPOINTER
                    ("address")}{R}{D}.{U}{R}{LET upright529,@CELLPOINTER
                    ("address")}{D}{L}.{R}{LET botright529,@CELLPOINTER
                    ("address")}{L}~{recalc529}
```

Now, the macro uses the same technique as before to record the cell addresses outside the [Keep range ?] range, and issues {recalc529} which we have seen earlier. The relevant formula for this case is the formula in cell [eraseall4529], therefore the macro returns to the main code.

```
     A                B                C                D                E
20 !                {IF @RIGHT(upright1529,@LENGTH(upright1529)-@FIND("$",
                    upright1529,1))="$1"}{side3529}{eraseall4529}
                    {BRANCH cont529}
```

and issues {eraseall9529}. The code in B44, [eraseall9529], is the result of the

```
+"/RE"&B29&"..A1~/RE"&B27&"..IV1~/RE"&B28&"..IV8192~"
```

formula. For example, if the range to keep is the F8192..K8183 range, then this formula returns the following code:

```
/RE$E$8192..A1~/RE$E$8182..IV1~/RE$L$8182..IV8192~
```

This erases all the worksheet except the F8192..K8183 range. Next, the macro issues {BRANCH cont529} which routes the macro execution to an empty cell and quits. The last case is when the range to keep is inside the worksheet and not touching the corners or the sides of the worksheet

	A	B	C	D	E
22	!	/RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER			
		("address")}{U}{R}.{L}{U}{LET upleft529,@CELLPOINTER			
		("address")}{R}{D}.{R}{U}{LET upright529,@CELLPOINTER			
		("address")}{L}{D}.{R}{D}{LET botright529,@CELLPOINTER			
		("address")}{L}{U}~			
23	!	{recalc529}{eraseall529}			

Now, the macro uses the same technique as before to record the cell addresses outside the [Keep range ?] range, and issues {recalc529} which we have seen earlier. Then the macro issues {eraseall529} which starts the [eraseall529] routine; however the code in this routine is the result of the following formula:

```
+"/RE"&B27&"..A1~/RE"&B27&"..A8192~/RE"&B27&"..IV1~/RE"&B30&
    "..A8192~/RE"&B30&"..IV1~/RE"&B30&"..IV8192~"
```

For example, if the range to keep is the F10..K20 range, then this formula returns the following code:

```
/RE$E$9..A1~/RE$E$9..A8192~/RE$E$9..IV1~/RE$L$21..A8192~
    /RE$L$21..IV1~/RE$L$21..IV8192~
```

This erases all the worksheet except the F10..K20 range. Next the macro reaches an empty cell and quits. This is one of the most complicated macros in ***Super Power***. It demanded a lot of programming and intensive use of string functions to dynamically adapt the code to the various cases available. This macro fills one of the missing functions of the standard spreadsheet. A possible addition to this macro is the option to delete all the range names outside the range to keep.

Because this macro is quite complicated, here is the list of all the cell contents to help you key this macro into 1-2-3.

```
A1: U [W14] '*---A macro to ERASE all the worksheet EXCEPT a specified
          range
A2: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
         define the
A3: [W14] '    range names in this column (starts with the \Z macro name)
A4: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W14] '!
A6: [W14] '!
A7: U [W14] '            In Release 3 use it for single sheet at a time
A8: [W14] '!
A9: [W14] '!
A10: U [W14] '\Z
B10: '{BREAKON}
A11: U [W14] 'RANGKEEP
B11: '{WINDOWSOFF}{PANELOFF}/RNCKeep range ?~~/RNDKeep range ?~/RNC
     {WINDOWSON}{PANELON}Keep range ?~{BS}{BS}{?}~{WINDOWSOFF}
     {GOTO}Keep range ?~
A12: [W14] '!
B12: '{LET hereabs529,@CELLPOINTER("address")}~
A13: [W14] '!
B13: '/RNCKeep range ?~.{LET botleft1529,@CELLPOINTER("address")}.
```

```
     {LET upleft1529,@CELLPOINTER("address")}.{LET upright1529,@CELLPOINTER
     ("address")}.{LET botright1529,@CELLPOINTER("address")}~
A14: [W14] '!
B14: '{IF upleft1529="$A$1"}{edge1529}{eraseall1529}{BRANCH cont529}
A15: [W14] '!
B15: '{IF upright1529="$IV$1"}{edge2529}{eraseall5529}{BRANCH cont529}
A16: [W14] '!
B16: '{IF botleft1529="$A$8192"}{edge3529}{eraseall6529}
     {BRANCH cont529}
A17: [W14] '!
B17: '{IF botright1529="$IV$8192"}{edge4529}{eraseall7529}
     {BRANCH cont529}
A18: [W14] '!
B18: '{IF @LEFT(upleft1529,@FIND("$",upleft1529,1))="$A"}{side1529}
     {eraseall2529}{BRANCH cont529}
A19: [W14] '!
B19: '{IF @LEFT(upright1529,@FIND("$",upright1529,1))="$IV"}{side2529}
     {eraseall3529}{BRANCH cont529}
A20: [W14] '!
B20: '{IF @RIGHT(upright1529,@LENGTH(upright1529)-@FIND("$",upright1529
     ,1))="$1"}{side3529}{eraseall4529}{BRANCH cont529}
A21: [W14] '!
B21: '{IF @RIGHT(botleft1529,@LENGTH(botleft1529)-@FIND("$",botleft1529,1))
     ="$8192"}{side4529}{eraseall9529}{BRANCH cont529}
A22: [W14] '!
B22: '/RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER("address")}
     {U}{R}.{L}{U}{LET upleft529,@CELLPOINTER("address")}{R}{D}.{R}{U}
     {LET upright529,@CELLPOINTER("address")}{L}{D}.{R}{D}
     {LET botright529,@CELLPOINTER("address")}{L}{U}~
A23: [W14] '!
B23: '{recalc529}{eraseall529}
A24: [W14] 'cont529
A25: [W14] '!
A26: [W14] 'hereabs529
B26: '$A$3
A27: [W14] 'upleft529
B27: '$B$8179
A28: [W14] 'upright529
B28: '$F$8179
A29: [W14] 'botleft529
B29: '$B$8192
A30: [W14] 'botright529
B30: '$F$8192
A31: [W14] 'upleft1529
B31: '$A$3
A32: [W14] 'upright1529
B32: '$B$3
A33: [W14] 'botleft1529
B33: '$A$10
A34: [W14] 'botright1529
B34: '$B$10
A35: [W14] '!
A36: [W14] 'eraseall529
B36: U +"/RE"&B27&"..A1~/RE"&B27&"..A8192~/RE"&B27&"..IV1~/RE"&
       B30&"..A8192~/RE"&B30&"..IV1~/RE"&B30&"..IV8192~"
A37: [W14] '!
A38: [W14] 'eraseall2529
B38: U +"/RE"&B27&"..IV1~/RE"&B29&"..IV8192~/RE"&B30&"..IV8192~/RE"&
       B30&"..IV1~"
A39: [W14] '!
A40: [W14] 'eraseall3529
B40: U +"/RE"&B28&"..A1~/RE"&B30&"..A8192~/RE"&B27&"..A1~/RE"&
       B27&"..A8192~"
A41: [W14] '!
A42: [W14] 'eraseall4529
B42: U +"/RE"&B27&"..A8192~/RE"&B29&"..IV8192~/RE"&B30&"..IV1~"
A43: [W14] '!
A44: [W14] 'eraseall9529
B44: U +"/RE"&B29&"..A1~/RE"&B27&"..IV1~/RE"&B28&"..IV8192~"
A45: [W14] '!
A46: [W14] 'eraseall1529
```

```
B46: U +"/RE"&B29&"..IV8192~/RE"&B28&"..IV8192~"
A47: [W14] '!
A48: [W14] 'eraseall5529
B48: U +"/RE"&B30&"..A8192~/RE"&B27&"..A8192~"
A49: [W14] '!
A50: [W14] 'eraseall6529
B50: U +"/RE"&B27&"..IV1~/RE"&B30&"..IV1~"
A51: [W14] '!
A52: [W14] 'eraseall7529
B52: U +"/RE"&B28&"..A1~/RE"&B29&"..A1~"
A53: [W14] '!
A54: [W14] 'edge1529
B54: '/RNCKeep range ?~.{D}{LET botleft529,@CELLPOINTER("address")}{U}.
     {LET upleft529,@CELLPOINTER("address")}.{R}{LET upright529,
     @CELLPOINTER("address")}{L}.{R}{D}{LET botright529,@CELLPOINTER
     ("address")}{L}{U}~{recalc529}
A55: [W14] '!
A56: [W14] 'edge2529
B56: '/RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER("address")}{U}
     {R}.{L}{LET upleft529,@CELLPOINTER("address")}{R}.{LET upright529,
     @CELLPOINTER("address")}.{D}{LET botright529,@CELLPOINTER("address")}
     {U}~{recalc529}
A57: [W14] '!
A58: [W14] 'edge3529
B58: '/RNCKeep range ?~.{LET botleft529,@CELLPOINTER("address")}.{U}
     {LET upleft529,@CELLPOINTER("address")}{D}.{R}{U}
     {LET upright529,@CELLPOINTER("address")}{L}{D}.{R}
     {LET botright529,@CELLPOINTER("address")}{L}~{recalc529}
A59: [W14] '!
A60: [W14] 'edge4529
B60: '/RNCKeep range ?~.{L}{LET botleft529,@CELLPOINTER("address")}
     {R}.{L}{U}{LET upleft529,@CELLPOINTER("address")}{R}{D}.{U}
     {LET upright529,@CELLPOINTER("address")}{D}.
     {LET botright529,@CELLPOINTER("address")}~{recalc529}
A61: [W14] '!
A62: [W14] 'recalc529
B62: '{RECALC eraseall529}{RECALC eraseall1529}{RECALC eraseall2529}
     {RECALC eraseall3529}{RECALC eraseall4529}{RECALC eraseall5529}
     {RECALC eraseall6529}{RECALC eraseall7529}{RECALC eraseall9529}
A63: [W14] '!
A64: [W14] 'side1529
B64: '/RNCKeep range ?~.{D}{LET botleft529,@CELLPOINTER("address")}
     {U}.{U}{LET upleft529,@CELLPOINTER("address")}{D}.{R}{U}
     {LET upright529,@CELLPOINTER("address")}{L}{D}.{R}{D}
     {LET botright529,@CELLPOINTER("address")}{L}{U}~{recalc529}
A65: [W14] '!
A66: [W14] 'side2529
B66: '/RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER("address")}
     {U}{R}.{L}{U}{LET upleft529,@CELLPOINTER("address")}{R}{D}.{U}
     {LET upright529,@CELLPOINTER("address")}{D}.{D}
     {LET botright529,@CELLPOINTER("address")}{U}~{recalc529}
A67: [W14] '!
A68: [W14] 'side3529
B68: '/RNCKeep range ?~.{L}{D}{LET botleft529,@CELLPOINTER("address")}
     {R}{U}.{L}{LET upleft529,@CELLPOINTER("address")}{R}.{R}
     {LET upright529,@CELLPOINTER("address")}{L}.{R}{D}
     {LET botright529,@CELLPOINTER("address")}{L}{U}~{recalc529}
A69: [W14] '!
A70: [W14] 'side4529
B70: '/RNCKeep range ?~.{L}{LET botleft529,@CELLPOINTER("address")}
     {R}.{L}{U}{LET upleft529,@CELLPOINTER("address")}{R}{D}.{U}{R}
     {LET upright529,@CELLPOINTER("address")}{D}{L}.{R}
     {LET botright529,@CELLPOINTER("address")}{L}~{recalc529}
```

[6] Eliminate Completely Empty Rows

```
     A                B                C                D                E
 1 *---A macro to ELIMINATE completely empty rows in a range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---In release 3 and up make sure that the GROUP option is DISABLED
 6 *---In release 2/2.01/2.2/2.3 the macro must be located ABOVE the
       range
 7 !
 8          THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                    IT WILL WORK IN LOTUS 2.0 AND UP
10 \Z              {BREAKON}
11 ROWSELIM        {LET rel157,@INFO("release")}~
12 !               {WINDOWSOFF}{PANELOFF}/RNCwhich range ?~~/RND
                   which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{GOTO}Which range ?~{LET rowsaa157,@ROWS
                   (Which range ?)}~{WINDOWSON}{LET counterb157,0}~
                   {LET colss157,@COLS(Which range ?)}~
13 !               {IF @LEFT(rel157,1)<>"@"}{LET sheets157,@SHEETS
                   (Which range ?)}~
14 cont157         {LET hereabs157,@CELLPOINTER("address")}~
                   {LET counter1157,0}{LET rowsa157,rowsaa157}~
15 !               {labels1157}
16 !               {LET rel157,@INFO("release")}~{IF @LEFT(rel157,1)<>
                   "@"}{GOTO}{hereabs157}~{LET counterb157,counterb157+1}
                   ~{IF counterb157<sheets157}{NS}{GOTO}{hereabs157}~
                   {BRANCH cont157}
17 !               /RNDWhich range ?~{WINDOWSON}
18 !
19 counter1157           0
20 counter1a157          0
21 labels1157      {FOR counter1a157,0,rowsa157-1,1,labels1a157}~
                   {LET counter1a157,0}~
                   {RETURN}
22 !
23 labels1a157     {IF @CELLPOINTER("col")=1}{END}{RIGHT}{IF @CELLPOINTER
                   ("col")=256}{END}{LEFT}{IF @CELLPOINTER("type")="b"}
                   /WDR~{LET rowsa157,rowsa157-1}~{RETURN}
24 !               {IF @CELLPOINTER("type")<>"b"}{DOWN}{RETURN}
25 !               {END}{LEFT}{LET loc1157,@CELLPOINTER("col")}~{END}
                   {RIGHT}{LET loc2157,@CELLPOINTER("col")}~{END}{LEFT}
26 !               {IF loc1157=1#AND#loc2157=256#AND#@CELLPOINTER("type")
                   ="b"}/WDR~{LET rowsa157,rowsa157-1}~{RETURN}
27 !               {DOWN}
28 !
29 ret157
30 !
31 rowsa157             7
32 rowsaa157           10
33 sheets157            7
34 colss157             5
35 !
36 loc1157              1
37 !
38 loc2157            256
39 !
40 counterb157          7
41 hereabs157      $A$1
42 !
43 rel157
```

This macro will delete any row in a range that is completely empty, but the macro first checks that it is completely empty before deleting it. This macro is safer than the ROWDELET.WK1 macro.

	A	B	C	D	E
11	ROWSELIM	{LET rel157,@INFO("release")}~			

The macro issues the {LET rel157,@INFO("release")}~ macro command that stores the result of the @INFO("release") function in cell [rel157] for later use. The macro uses the result to check if you are using a 3-D or a 2-D Lotus release.

	A	B	C	D	E
12	!	{WINDOWSOFF}{PANELOFF}/RNCwhich range ?~~/RND which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~ {BS}{BS}{?}~{GOTO}Which range ?~{LET rowsaa157,@ROWS (Which range ?)}~{WINDOWSON}{LET counterb157,0}~ {LET colss157,@COLS(Which range ?)}~			

Here the macro issues {WINDOWSOFF}{PANELOFF} to freeze the screen and panel activities and uses the "safe technique" to start a / **R**ange **N**ame **C**reate process. Simultaneously it provides a meaningful range name like the [Which range ?] that appears as a guiding prompt. When the {?} is issued, the macro pauses and waits until you paint the range to process and press ENTER. When you press ENTER, the macro issues {LET rowsaa157,@ROWS(Which range ?)}~ to store the number of rows of the range named [Which range ?] in cell [rowsaa157]. Next the macro issues {WINDOWSOFF} to freeze the screen activity, and issues {LET counterb157,0}~ to set the value in cell [counterb157] to zero, which serves as a counter. Next it issues {LET colss157,@COLS(Which range ?)}~ to store the number of columns of [Which range ?] in [colss157].

	A	B	C	D	E
13	!	{IF @LEFT(rel157,1)<>"@"}{LET sheets157,@SHEETS (Which range ?)}~			

The macro uses the stored content of cell [rel157] to check if you are using a 3-D Lotus release and uses the @LEFT(rel157,1) function to check if the first character of the cell content is equal or unequal to the "@" character. If it is equal to the "@" character, you are using a 2-D release, but if it is not equal to the "@" character, you are using a 3-D Lotus release; therefore the macro has to know how many sheets [Which range ?] contains. The macro issues {LET sheets157,@SHEETS(Which range ?)}~ to store the number of sheets of [Which range ?] range in cell [sheets157].

	A	B	C	D	E
14	cont157	{LET hereabs157,@CELLPOINTER("address")}~ {LET counter1157,0}{LET rowsa157,rowsaa157}~			

It is a good practice to record the cell pointer position before moving it to another cell, especially if we need to come back to the same cell later on. Therefore the macro uses {LET hereabs157,@CELLPOINTER("address")}~ to store the current cell pointer address in cell [hereabs157]. Next the macro issues {LET counter1157,0} to set the value in cell [counter1157] to zero and {LET rowsa157,rowsaa157}~ to copy the content of cell [rowsaa157] to cell [rowsa157].

	A	B	C	D	E
15	!	{labels1157}			

Now the macro issues {labels1157} that activates the [labels1157] routine.

	A	B	C	D	E

```
21 labels1157      {FOR counter1a157,0,rowsa157-1,1,labels1a157}~
                   {LET counter1a157,0}~
                   {RETURN}
```

The [labels1157] routine uses this {FOR} command to execute the [labels1a157] routine as many times as the number of rows in [Which range ?]. The number of rows is stored in cell [rowsa157].

```
          A              B                C              D              E
23 labels1a157     {IF @CELLPOINTER("col")=1}{END}{RIGHT}{IF @CELLPOINTER
                   ("col")=256}{END}{LEFT}{IF @CELLPOINTER("type")="b"}
                   /WDR~{LET rowsa157,rowsa157-1}~{RETURN}
24 !               {IF @CELLPOINTER("type")<>"b"}{DOWN}{RETURN}
25 !               {END}{LEFT}{LET loc1157,@CELLPOINTER("col")}~{END}
                   {RIGHT}{LET loc2157,@CELLPOINTER("col")}~{END}{LEFT}
26 !               {IF loc1157=1#AND#loc2157=256#AND#@CELLPOINTER("type")
                   ="b"}/WDR~{LET rowsa157,rowsa157-1}~{RETURN}
27 !               {DOWN}
```

The [labelsa1157] routine issues {IF @CELLPOINTER("col")=1} to check if the cell pointer is on the "A" column. If so, the macro issues {END}{RIGHT} to send the cell pointer to the last unoccupied cell in the row, hopefully the "IV" column. Then it issues {IF @CELLPOINTER("col")=256} to check if the column is truly the "IV" column. If it is, then the macro issues {END}{LEFT} to send the cell pointer to the "A" column. Now it also issues {IF @CELLPOINTER("type")="b"} to check that the cell is blank. If it is, then the macro issues /WDR~ to erase the row. Next the macro uses {LET rowsa157,rowsa157 -1}~ to decrease by one the number of rows stored in cell [rowsa157].

```
          A              B                C              D              E
24 !               {IF @CELLPOINTER("type")<>"b"}{DOWN}{RETURN}
```

If the current column is not the "A" column the macro uses {IF @CELLPOINTER("type") <>"b"} to check if the cell is blank. If not, then it uses {DOWN} to move the cell pointer to the next row and issues {RETURN} to return to the {FOR} loop.

```
          A              B                C              D              E
25 !               {END}{LEFT}{LET loc1157,@CELLPOINTER("col")}~{END}
                   {RIGHT}{LET loc2157,@CELLPOINTER("col")}~{END}{LEFT}
```

If the current cell is blank, the macro issues {END}{LEFT} to send the cell pointer to the first occupied cell or the "A" column. Now it issues {LET loc1157,@CELLPOINTER("col")}~ to store the column number attribute in cell [loc1157] and {END}{RIGHT} to move the cell pointer to the first occupied cell to the right in the IV column. Then issues {LET loc2157 ,@CELLPOINTER("col")}~ to store the column number attribute in cell [loc2157] and {END}{LEFT} to move the cell pointer back to the extreme left cell.

The macro knows the column number of the extreme left occupied cell in the row (stored in cell [loc1157]) and the extreme right occupied cell in the row (stored in cell [loc2157]).

```
          A              B                C              D              E
26 !               {IF loc1157=1#AND#loc2157=256#AND#@CELLPOINTER("type")
                   ="b"}/WDR~{LET rowsa157,rowsa157-1}~{RETURN}
```

The macro checks that the extreme left column number is "1" and the extreme right column number is "256" and the current cell is blank using the

```
{IF loc1157=1#AND#loc2157=256#AND#@CELLPOINTER("type")="b"}
```

{IF} macro command. If this is true, the macro issues /WDR~ to delete the row and issues {LET rowsa157,rowsa157-1}~ to decrease by one the number of rows stored in cell [rowsa157], and then issues {RETURN} to return to the {FOR} loop.

	A	B	C	D	E
16	!	{LET rel157,@INFO("release")}~{IF @LEFT(rel157,1)<> "@"}{GOTO}{hereabs157}~{LET counterb157,counterb157+1} ~{IF counterb157<sheets157}{NS}{GOTO}{hereabs157}~ {BRANCH cont157}			

Till now, the macro processed only one sheet, but what happens if you are using a 3-D Lotus release and the [Which range ?] range has more than one sheet. The code in the B16 cell takes care of this. The macro first issues {LET rel157,@INFO("release")}~ to store the result of the @INFO("release") function in cell [rel157], and then {IF @LEFT(rel157 ,1)<>"@"} to check the first character of the content of cell [rel157]. If the character is not the "@" character then you are using a 3-D Lotus release. To move back to the upper left cell of the [Which range ?] range, the macro issues the {GOTO}{hereabs157}~ indirect macro command. (See the code in the B14 cell where the cell pointer address was stored to the cell named [hereabs157]).

The macro uses {LET counterb157,counterb157+1}~ to increase the counter in cell [counterb157] by one. Before the macro moves to the next sheet, it uses {IF counterb157<sheets157} to check if the counter in cell [counterb157] is less than the number of sheets in the [Which range ?] range. If the counter is still less than the number of sheets in [Which range ?], the macro issues {NS} to move to the next sheet. Here is the trick! The macro uses the {GOTO}{hereabs157}~ indirect command to move the cell pointer to the upper left cell of the range to process in the new sheet because it is the same address as the address as in the previous sheet. The {NS} command combined with the {GOTO} {hereabs157}~ command is the preferred way to move the cell pointer in a straight line in the "Z" direction when you are using a 3-D Lotus release.

	A	B	C	D	E
17	!	/RNCWhich range ?~~/RNDWhich range ?~{WINDOWSON}			

To end this session, the macro deletes the [Which range ?] range name, but there is always the chance that the range name does not exist any more, because if the first row of the range had a zero or a blank cell, the macro deleted the row and the range name with it. To overcome this possibility, the macro issues /RNCWhich range ?~~ to recreate the [Which range ?] range name. If the range name still exists, it stays the same; but if the range name does not exist, the macro creates a new one at the cell pointer location. Now the macro can safely use /RNDWhich range ?~ to delete the range name without the chance of error and leave a clean worksheet.

[6] Eliminate Completely Empty Columns

```
     A                B                C                D                E
 1 *---A macro to ELIMINATE completely empty columns in a range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---In release 3 make sure that the GROUP option is DISABLED
 6 *---In release 2/2.01/2.2 the macro must be located ABOVE the range
 7 !
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                   IT WILL WORK IN RELEASE 2.0 AND UP
10 \Z              {BREAKON}
11 COLSELIM        {LET rel532,@INFO("release")}~
12 !               {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{GOTO}Which range ?~{LET rowsaa532,@ROWS
                   (Which range ?)}~{WINDOWSON}{LET counterb532,0}~
                   {LET kolss532,@COLS(Which range ?)}~
13 !               {IF @LEFT(rel532,1)<>"@" }{LET sheets532,@SHEETS
                   (Which range ?)}~
14 cont532         {LET hereabs532,@CELLPOINTER("address")}~
                   {LET counter1532,0}{LET rowsa532,rowsaa532}~
15 !               {WINDOWSOFF}{labels1532}{WINDOWSON}
16 !               {LET rel532,@INFO("release")}~{IF @LEFT(rel532,1)<>
                   "@"}{GOTO}{hereabs532}~{LET counterb532,counterb532+
                   1}~{IF counterb532<sheets532}{NS}{GOTO}{hereabs532}~
                   {BRANCH cont532}
17 !               /RNDWhich range ?~{WINDOWSON}
18 !
19 counter1532                0
20 counter1a532               0
21 labels1532      {LET colss532,kolss532}~{FOR counter1a532,0,colss532
                   -1,1,labels1a532}~{LET counter1a532,0}~{RETURN}
22 !
23 !
24 labels1a532     {IF @CELLPOINTER("row")=1}{END}{DOWN}{IF @CELLPOINTER
                   ("row")=8192}{END}{UP}{IF @CELLPOINTER("type")="b"}
                   /WDC~{LET colss532,colss532-1}~{RETURN}
25 !               {IF @CELLPOINTER("type")<>"b"}{RIGHT}{RETURN}
26 !               {END}{UP}{LET loc1532,@CELLPOINTER("row")}~{END}
                   {DOWN}{LET loc2532,@CELLPOINTER("row")}~{END}{UP}
27 !               {IF loc1532=1#AND#loc2532=8192#AND#@CELLPOINTER
                   ("type")="b"}/WDC~{LET colss532,colss532-1}~{RETURN}
28 !               {RIGHT}
29 !
30 ret532
31 !
32 rowsa532                    1
33 rowsaa532                   1
34 sheets532                   5
35 colss532                    0
36 kolss532
37 loc1532                     1
38 !
39 loc2532                  8192
40 !
41 counterb532                 5
42 hereabs532      $A$1
43 !
44 rel532          3.10.00
```

This macro deletes any column in a range that is completely empty; it checks that the column is completely empty before deleting it. This macro is safer than the ROWDELET.WK1 macro.

```
     A                B                C                D                E
11 COLSELIM        {LET rel532,@INFO("release")}~
```

The macro issues the {LET rel532,@INFO("release")}~ macro commands which store the result of the @INFO("release") function in cell [rel532] for later use to check if you are using a 3-D or a 2-D Lotus release.

	A	B	C	D	E
12	!	{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~ {BS}{BS}{?}~{GOTO}Which range ?~{LET rowsaa532,@ROWS (Which range ?)}~{WINDOWSON}{LET counterb532,0}~ {LET kolss532,@COLS(Which range ?)}~			

Here the macro uses {WINDOWSOFF}{PANELOFF} to freeze the screen and panel activities while it uses the "safe technique" to assign the [Which range ?] name to the same range, which acts as a guiding prompt. Next the macro issues {?} that pauses and waits until you paint the range and press ENTER. Then the macro issues {LET rowsaa532,@ROWS (Which range ?)}~, which store the number of rows of the range named [Which range ?] in cell [rowsaa532]. Next the macro uses {LET counterb532,0}~ to set the value in cell [counterb532] to zero, which serves as a counter. The macro then issues {LET colss532,@COLS(Which range ?)}~ to store the number of columns of the [Which range ?] range in cell [colss532].

	A	B	C	D	E
13	!	{IF @LEFT(rel532,1)<>"@" }{LET sheets532,@SHEETS (Which range ?)}~			

Now the macro issues {IF @LEFT(rel532,1)<>"@"} to check if you are using a 3-D Lotus release. The macro uses the @LEFT(rel532,1) Lotus string function to check if the first character of the cell content of [rel532] is equal to the "@" character. If it is, you are using a 2-D release, but if it is not equal, you are using a 3-D Lotus release. If you are using a 3-D release, the macro has to know how many sheets the [Which range ?] range has. The macro uses {LET sheets532,@SHEETS(Which range ?)}~ to store the number of sheets in the [Which range ?] range in cell [sheets532].

	A	B	C	D	E
14	cont532	{LET hereabs532,@CELLPOINTER("address")}~ {LET counter1532,0}{LET rowsa532,rowsaa532}~			

It is a good practice to record the cell pointer position before moving it to another cell, especially if we need to come back to the same cell later on. Therefore the macro uses {LET hereabs532,@CELLPOINTER("address")}~ to store the current cell pointer address in cell [hereabs532]. Next the macro issues {LET counter1157,0} to reset the value in [counter1532] to zero and then issues {LET rowsa532,rowsaa532}~ to copy the content of [rowsaa532] to [rowsa532].

	A	B	C	D	E
15	!	{WINDOWSOFF}{labels1532}{WINDOWSON}			

Now the macro issues {WINDOWSOFF} to freeze the screen activity while the macro issues {labels1532} that activates the [labels1532] routine.

	A	B	C	D	E
21	labels1532	{LET colss532,kolss532}~{FOR counter1a532,0,colss532 -1,1,labels1a532}~{LET counter1a532,0}~{RETURN}			

The [labels1532] routine issues {LET colss532,kolss532}~ to copy the content of cell [colss532] to cell [kolss532], and then issues {FOR counter1a532,0,colss532-1,1, labels1a532}~ to execute the [labels1a532] routine as many times as the number of rows in the [Which range ?] range. The number of rows is stored in cell [colss532].

	A	B	C	D	E
24	labels1a532	{IF @CELLPOINTER("row")=1}{END}{DOWN}{IF @CELLPOINTER ("row")=8192}{END}{UP}{IF @CELLPOINTER("type")="b"} /WDC~{LET colss532,colss532-1}~{RETURN}			
25	!	{IF @CELLPOINTER("type")<>"b"}{RIGHT}{RETURN}			
26	!	{END}{UP}{LET loc1532,@CELLPOINTER("row")}~{END} {DOWN}{LET loc2532,@CELLPOINTER("row")}~{END}{UP}			
27	!	{IF loc1532=1#AND#loc2532=8192#AND#@CELLPOINTER ("type")="b"}/WDC~{LET colss532,colss532-1}~{RETURN}			
28	!	{RIGHT}			

The [labels1a532] routine issues {IF @CELLPOINTER("row")=1} to check if the cell pointer is on the first row of the spreadsheet, if so, the macro uses {END}{DOWN} to send the cell pointer to the last unoccupied cell in the row, hopefully the "8192" row. Then the macro issues {IF @CELLPOINTER("row")=8192} to check if the current row is the "8192" row. If so, the macro moves the cell pointer back to the first row using {END}{UP}. Now the macro issues {IF @CELLPOINTER("type")="b"} to check that the cell in the first row is blank. If this is also true, then the macro issues /WDC~ to delete the column. Next the macro issues {LET colss532,colss532-1}~ to decrease the number of columns stored in cell [colss532] by one.

	A	B	C	D	E
25	!	{IF @CELLPOINTER("type")<>"b"}{RIGHT}{RETURN}			

If the current row is not the first row, the macro issues {IF @CELLPOINTER("type") <>"b"} to check if the cell is blank. If not, then the macro uses {RIGHT} to move the cell pointer to the next column and {RETURN} to return to the {FOR} loop.

	A	B	C	D	E
26	!	{END}{UP}{LET loc1532,@CELLPOINTER("row")}~{END} {DOWN}{LET loc2532,@CELLPOINTER("row")}~{END}{UP}			

If the current cell is blank, the macro issues {END}{UP} that send the cell pointer to the first occupied cell or the first row. Now it issues {LET loc1532,@CELLPOINTER("row")}~ to store the row's number attribute in cell [loc1532], and then issues {END}{DOWN} to move the cell pointer to the first occupied cell below or the "8192" row. Now the macro issues {LET loc2532,@CELLPOINTER("row")}~ to store the cell's row attribute in cell [loc2532] and then {END}{UP} to move the cell pointer back to the very top cell in the current column. Now the macro "knows" the row number of the lowest occupied cell in the current column (stored in cell [loc1532]) and the row number of the highest occupied cell in the column (stored in cell [loc2532]).

	A	B	C	D	E
27	!	{IF loc1532=1#AND#loc2532=8192#AND#@CELLPOINTER ("type")="b"}/WDC~{LET colss532,colss532-1}~{RETURN}			

The macro issues the {IF loc1532=1#AND#loc2532=8192#AND#@CELLPOINTER ("type")="b"} command to check that the highest row number is "1" and that the lowest

row number is "8192" and the current cell is blank. If so, the macro issues `/WDC~` to delete the column and `{LET colss532,colss532-1}~` to decrease the number of columns stored in cell [colss532] by one. Then issues `{RETURN}` to return control to the `{FOR}` loop.

	A	B	C	D	E
16	!	{LET rel532,@INFO("release")}~{IF @LEFT(rel532,1)<> "@"}{GOTO}{hereabs532}~{LET counterb532,counterb532+ 1}~{IF counterb532<sheets532}{NS}{GOTO}{hereabs532}~ {BRANCH cont532}			

Till now the macro processed only one sheet, but what happens if you are using a 3-D Lotus release and the [Which range ?] range has more than one sheet. The code in the B16 cell takes care of this. The macro first issues `{LET rel532,@INFO("release")}~` to store the result of the `@INFO("release")` function in cell [rel532], and then issues `{IF @LEFT (rel532,1)<>"@"}` to check the first character of the content of cell [rel532]. If the character is not the "@" character then you are using a 3-D Lotus release, otherwise you are using a 2-D Lotus release. To move back to left top cell of [Which range ?], the macro issues the `{GOTO} {hereabs532}~` indirect command. (See the code in the B14 cell where the cell pointer address was stored in cell [hereabs532]).

Next the macro issues `{LET counterb532,counterb532+1}~` to increase the counter in cell [counterb532] by one. Before the macro moves to the next sheet, it issues `{IF counterb532<sheets532}` to check if the counter in cell [counterb532] is less than the number of sheets in the [Which range ?] range. If the counter is still less than the number of sheets in [Which range ?], the macro issues `{NS}` to move to the next sheet. Here is the trick! The macro uses the `{GOTO}{hereabs532}~` indirect macro command to move to the upper left cell of the range to process in the new sheet because it is the same address as the address in the previous sheet. The `{NS}` command combined with the `{GOTO} {hereabs532}~` command is the preferred way to move in a straight line in the "Z" direction when using a 3-D worksheet.

	A	B	C	D	E
17	!	/RNCWhich range ?~~/RNDWhich range ?~{WINDOWSON}			

To end this session, the macro deletes the [Which range ?] range name, but there is always the chance that the range name does not exist any more, because if the first column of the range had a zero or a blank cell, the macro deleted the column and the range name with it. To overcome this possibility, the macro issues `/RNCWhich range ?~~` to recreate the [Which range ?] range name. If the range name still exists, it stays the same, but if the range name does not exist, the macro creates a new one at the cell pointer location. Now the macro can safely use `/RNDWhich range ?~` to delete the range name without the chance of error and leave a clean worksheet.

[7] Erase to the End of Worksheet

```
     A            B          C       D       E       F       G       H       I
 1 *---A macro to ERASE from the cell next to the cell pointer to ends of
 2     row, columns, sheets
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                   IT WILL WORK IN RELEASE 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 ERASENDS        {LET rel504,@INFO("release")}~{MENUBRANCH menu1504}
12 !
13 menu1504        Right   Left   Up   Down    In    Out    Move  Quit
14 !               Erase alErase aEraseErase alErase Erase aMove tQuit
15 !               {rightaa{leftaa{topa{bottoma{IF @L{IF @LE{MOVE5{BRA
16 !               {MENUBRA{MENUBR{MENU{MENUBRA{MENUB{MENUBR{MENUBR
17 !
18 ret504
19 !
20 rel504          @INFO("release")
21 !
22 top504          $A$1
23 !
24 bottom504       $A$8192
25 !
26 left504         A1
27 !
28 right504        IV35
29 !
30 out504          A:$A$1
31 !
32 ina504          $U:$H$38
33 !
34 dum504          $A:$D$10
35 !
36 eratop504       $G$35..IV35
37 !
38 dumc504         $J:$D$10
39 !
40 topaa504        {WINDOWSOFF}{RECALC top504}{UP}
                   {LET eratop504,@CELLPOINTER("address")&".."&top504}~
                   {DOWN}/RE{eratop504}~~
41 !               {WINDOWSON}
42 !
43 bottomaa504     {WINDOWSOFF}{RECALC bottom504}{DOWN}
                   {LET eratop504,@CELLPOINTER("address")&".."&bottom504}
                   ~{UP}/RE{eratop504}~~
44 !               {WINDOWSON}
45 !
46 leftaa504       {WINDOWSOFF}{RECALC left504}{LEFT}
                   {LET eratop504,@CELLPOINTER("address")&".."&left504}~
                   {RIGHT}/RE{eratop504}~~
47 !               {WINDOWSON}
48 !
49 rightaa504      {WINDOWSOFF}{RECALC right504}{RIGHT}
                   {LET eratop504,@CELLPOINTER("address")&".."&right504}~
                   {LEFT}/RE{eratop504}~~
50 !               {WINDOWSON}
51 !
52 inaa504         {WINDOWSOFF}{IF @LEFT(rel504,1)<>"@"}
                   {LET dum504,@CELLPOINTER("coord")}~{NS}
                   {LET dumc504,@CELLPOINTER("coord")}~{LC}{END}{NS}
                   {LET ina504,@CELLPOINTER("coord")}~
53 !               {IF @LEFT(rel504,1)<>"@" }{LET eratop504,@LEFT(dumc504
                   ,@FIND(":",dumc504,0))&@RIGHT(dum504,@LENGTH(dum504)-
                   @FIND(":",dum504,0))&".."&@LEFT(ina504,@FIND(":",
```

```
                  ina504,0))&@RIGHT(dum504,@LENGTH(dum504)-@FIND(":",
                  dum504,0))}~/RE{eratop504}~~
54 !              {IF @LEFT(rel504,1)<>"@" }{GOTO}{dum504}~
55 !              {WINDOWSON}
56 !
57 outaa504        {WINDOWSOFF}{IF @LEFT(rel504,1)<>"@" }{RECALC out504}
                  {PS}{LET dumc504,@CELLPOINTER("coord")}~{LET eratop504
                  ,out504&".."&@LEFT(dumc504,@FIND(":",dumc504,0))&
                  @RIGHT(out504,@LENGTH(out504)-@FIND(":",out504,0))}~
                  /RE{eratop504}~~{GOTO}{dumc504}~
58 !              {NS}{WINDOWSON}
59 !
60 move504        Use the arrow keys to move the cell pointer and press
                  RETURN: {GET key504}{ESC}
61 !              {IF key504="~"}{RETURN}
62 !              {IF key504="{PGDN}"#OR#key504="{PGUP}"#OR#key504="
                  {DOWN}"#OR#key504="{UP}"#OR#key504="{LEFT}"#OR#key504=
                  "{RIGHT}"}{key504}
63 !              {IF @LEFT(rel504,1)<>"@"#AND#(key504="{NS}"#OR#
                  key504="{PS}")}{key504}
64 !              {BRANCH move504}
65 !
66 key504         ~
```

This macro allows you to erase a column, row or in a Z direction starting from the current cell till the end of the worksheet. For example: let us assume that the cell pointer is on the F35 cell. The macro can erase the G35..IV35, E35..A35, F1..F34, F56..F8192 and all the F35 cell in the next sheets or the previous sheets in a 3-D Lotus release. This is a very useful macro for worksheet cleaning purposes. This macro uses a custom menu with eight menu options, therefore there is no way they can fit on one page without overlapping. Here is the code of every menu option separately. If you intend to key this macro into Lotus 1-2-3, you should use the code as it appears here, NOT the code as it appears in the main listing.

```
Right
Erase all the cells right to the current cell
{rightaa504}
{MENUBRANCH menu1504}

Left
Erase all the cells left to the current cell
{leftaa504}
{MENUBRANCH menu1504}

Up
Erase all the cells up to the current cell
{topaa504}
{MENUBRANCH menu1504}

Down
Erase all the cells down to the current cell
{bottomaa504}
{MENUBRANCH menu1504}

In
Erase all the cells in the next sheets
{IF @LEFT(rel504,1)<>"@" }{inaa504}
{MENUBRANCH menu1504}

Out
Erase all the cells in the previous sheets
{IF @LEFT(rel504,1)<>"@" }{outaa504}
{MENUBRANCH menu1504}

Move
Move the cell pointer to other location
{move504}
{MENUBRANCH menu1504}
```

```
Quit
Quit the macro
{BRANCH ret504}
```

Notice that the code in the B22, B24, B26, B28 and the B30 cells of the macro is the result of dynamic string formulas; therefore if you intend to key this macro into Lotus 1-2-3, you should key the following formulas and not the code as it appears in the main listing.

```
22 top504          @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER
                   ("address"),1)+1)&"1"
24 bottom504       @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER
                   ("address"),1)+1)&"8192"
26 left504         +"A"&@RIGHT(@CELLPOINTER("address"),@LENGTH
                   (@CELLPOINTER("address"))-@FIND("$",@CELLPOINTER
                   ("address"),1)-1)
28 right504        +"IV"&@RIGHT(@CELLPOINTER("address"),@LENGTH
                   (@CELLPOINTER("address"))-@FIND("$",@CELLPOINTER
                   ("address"),1)-1)
30 out504          +"A:"&@CELLPOINTER("address")
```

```
     A          B          C          D          E          F          G          H          I
11 ERASENDS     {LET rel504,@INFO("release")}~{MENUBRANCH menu1504}
```

The macro starts with the `{LET rel504,@INFO("release")}~` macro commands which store the result of the `@INFO("release")` function in cell [rel504], later the macro uses the stored result to check if you are using a 3-D or a 2-D release. Next the macro issues `{MENUBRANCH menu1504}` which starts the [menu1504] custom menu.

The first menu option is [**R**ight]:

```
Right
Erase all the cells right to the current cell
{rightaa504}
{MENUBRANCH menu1504}
```

When you choose this menu option, the macro issues the `{rightaa504}` routine command, which erases all the cells right of the current cell. When the routine is finished, the macro control returns to the `{MENUBRANCH menu1504}` command which re-starts the custom menu.

```
     A          B          C          D          E          F          G          H          I
49 rightaa504   {WINDOWSOFF}{RECALC right504}{RIGHT}
                {LET eratop504,@CELLPOINTER("address")&".."&right504}~
                {LEFT}/RE{eratop504}~~
50 !            {WINDOWSON}
```

The [rightaa504] routine starts with `{WINDOWSOFF}` which freezes the screen activity. Then the macro issues `{RECALC right504}` which updates the dynamic string formula in cell [right504] and issues `{RIGHT}` to move the cell pointer to the adjacent cell.

```
28 right504        +"IV"&@RIGHT(@CELLPOINTER("address"),@LENGTH
                   (@CELLPOINTER("address"))-@FIND("$",@CELLPOINTER
                   ("address"),1)-1)
```

The result of the formula in cell [right504] is the last cell in this row, which is the IV35 cell, if the current cell is F35. Next the macro issues `{LET eratop504,@CELLPOINTER ("address")&".."&right504}~` which store the address of the range to erase in cell [eratop504]. In our example, the range address is the `$G$35..IV35` address. Now the macro

issues {LEFT} to move the cell pointer back to its place of origin and issues /RE {eratop504}~~ to erase the $G$35..IV35 range. Notice the {eratop504} routine command inside the /RE process. The {eratop504} command injects the content of the [eratop504] routine into the panel because it contains only the address of the range to erase.

The second menu option is [**L**eft]:

```
'Left
'Erase all the cells left to the current cell
{leftaa504}
{MENUBRANCH menu1504}
```

When you choose this menu option, the macro issues {leftaa504} which erases all the cells left to the current cell. When the routine is finished, the macro control returns to the {MENUBRANCH menu1504} command which re-starts the custom menu.

```
     A          B         C        D        E        F        G        H        I
46 leftaa504    {WINDOWSOFF}{RECALC left504}{LEFT}
                {LET eratop504,@CELLPOINTER("address")&".."&left504}~
                {RIGHT}/RE{eratop504}~~
47 !            {WINDOWSON}
```

The [leftaa504] routine starts with {WINDOWSOFF} which freezes the screen activity. Then the macro issues {RECALC left504} which updates the dynamic string formula in cell [left504] and then {LEFT}.

```
26 left504      +"A"&@RIGHT(@CELLPOINTER("address"),@LENGTH
                (@CELLPOINTER("address"))-@FIND("$",@CELLPOINTER
                ("address"),1)-1)
```

The result of the formula in cell [left504] is the last cell in this row, is the A35 cell, if the current cell is F35. Next the macro issues {LET eratop504,@CELLPOINTER ("address")&".."&left504}~ which stores the address of the range to erase in cell [eratop504]. In our example, the range address is the $E$35..A35 address. Now the macro issues {RIGHT} to move the cell pointer back to its place of origin and issues /RE {eratop504}~~ to erase the $E$35..A35 range. Notice the {eratop504} routine command inside the /RE process. The {eratop504} command injects the content of the [eratop504] routine into the panel because it contains only the address of the range to erase.

The third menu option is [**U**p]:

```
'Up
'Erase all the cells up to the current cell
{topaa504}
{MENUBRANCH menu1504}
```

When you choose this menu option, the macro issues the {topaa504} routine command which erases all the cells above the current cell. When the routine is finished, the macro control returns to the {MENUBRANCH menu1504} macro command which re-starts the custom menu.

```
     A          B         C        D        E        F        G        H        I
40 topaa504     {WINDOWSOFF}{RECALC top504}{UP}
                {LET eratop504,@CELLPOINTER("address")&".."&TOP504}~
                {DOWN}/RE{eratop504}~~
41 !            {WINDOWSON}
```

The [topaa504] routine starts with the {WINDOWSOFF} macro command which freezes the screen activity. Then the macro issues the {RECALC top504} macro command which updates the dynamic string formula in cell [top504] and issues the {UP} macro command.

```
22 top504          @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER
                   ("address"),1)+1)&"1"
```

The result of the formula in cell [top504] is the upper cell in this column, is the F1 cell, if the current cell is F35. Next the macro issues {LET eratop504,@CELLPOINTER ("address")&".."&top504}~ which store the address of the range to erase in cell [eratop504]. In our example, the range address is $F$34..F1. Now the macro issues {DOWN} to move the cell pointer back to its origin place and issues /RE{eratop504}~~ to erase the $F$34..F1 range. Notice the {eratop504} routine command inside the /RE process. The {eratop504} routine command injects the content of the [eratop504] routine into the panel because it contains only the address of the range to erase.

The fourth menu option is [**D**own]:

```
Down
Erase all the cells down to the current cell
{bottomaa504}
{MENUBRANCH menu1504}
```

When you choose this menu option, the macro issues the {bottomaa504} routine command which erases all the cells below the current cell. When the routine is finished, the macro control returns to the {MENUBRANCH menu1504} command which re-starts the custom menu.

```
          A          B          C          D          E          F          G          H          I
43 bottomaa504     {WINDOWSOFF}{RECALC bottom504}{DOWN}
                   {LET eratop504,@CELLPOINTER("address")&".."&bottom504}
                   ~{UP}/RE{eratop504}~~
44 !               {WINDOWSON}
```

The [bottomaa504] routine starts with {WINDOWSOFF} which freezes the screen activity, then the macro issues {RECALC bottom504} to update the dynamic string formula in cell [bottom504] and then issues {DOWN}.

```
24 bottom504       @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER
                   ("address"),1)+1)&"8192"
```

The result of the formula in cell [bottom504] is the lower cell in this column, is the F8192 cell, if the current cell is F35. Next the macro issues the {LET eratop504,@CELLPOINTER ("address")&".."&bottom504}~ macro command which stores the address of the range to erase in cell [eratop504]. In our example, the range address is the $F$36..F8192 address. Now the macro issues the {UP} macro command to move the cell pointer back to its place of origin and issues the /RE{eratop504}~~ command to erase the $F$36..F8192 range. Notice the {eratop504} routine command inside the /RE process. The {eratop504} routine command injects the content of the [eratop504] routine into the panel because it contains only the address of the range to erase.

The fifth menu option is [**I**n]:

```
In
Erase all the cells in the next sheets
```

```
{IF @LEFT(rel504,1)<>"@" }{inaa504}
{MENUBRANCH menu1504}
```

When you choose this menu option, the macro issues `{IF @LEFT(rel504,1)<>"@"}` which checks if the first character of the content of cell [rel504] is the "@" character. If so, you are using a 3-D release and the macro issues the `{inaa504}` routine command. When the routine is finished, the macro control returns to the `{MENUBRANCH menu1504}` command which re-starts the custom menu. If this is not true, you are using a 2-D release and the macro issues `{MENUBRANCH menu1504}` which re-starts the custom menu.

	A	B	C	D	E	F	G	H	I
52	inaa504	{WINDOWSOFF}{IF @LEFT(rel504,1)<>"@"}							
		{LET dum504,@CELLPOINTER("coord")}~{NS}							
		{LET dumc504,@CELLPOINTER("coord")}~{LC}{END}{NS}							
		{LET ina504,@CELLPOINTER("coord")}~							
53	!	{IF @LEFT(rel504,1)<>"@" }{LET eratop504,@LEFT(dumc504							
		,@FIND(":",dumc504,0))&@RIGHT(dum504,@LENGTH(dum504)-							
		@FIND(":",dum504,0))&".."&@LEFT(ina504,@FIND(":",							
		ina504,0))&@RIGHT(dum504,@LENGTH(dum504)-@FIND(":",							
		dum504,0))}~/RE{eratop504}~~							
54	!	{IF @LEFT(rel504,1)<>"@" }{GOTO}{dum504}~							
55	!	{WINDOWSON}							

The [inaa504] routine is more complicated then the previous four routines because the number of sheets in the current worksheet is not a fixed number as the number of columns or the number of rows. Therefore the macro has to find a way to erase all the cells in the Z direction no matter how many sheets are in the worksheet. The routine starts with `{WINDOWSOFF}` which freezes the screen activity, then the macro issues `{IF @LEFT (rel504,1)<>"@"}` which checks if the first character of the content of cell [rel504] is the "@" character. If so, you are using a 3-D Lotus release, therefore the macro issues `{LET dum504,@CELLPOINTER("coord")}~` which stores the current cell's coordinates (coord) in cell [dum504].

Next the routine issues `{NS}` which moves the cell pointer to the next sheet. Then the macro issues `{LET dumc504,@CELLPOINTER("coord")}~` which store the current cell's coord in cell [dumc504]. The macro will extract the sheet letter from the coord stored in cell [dumc504]. Next the routine issues `{LC}` which moves the cell pointer to the last cell in the last occupied sheet, and `{END}{NS}` to move the cell pointer to the last sheet in the worksheet. Now the macro issues `{LET ina504,@CELLPOINTER("coord")}~` which store the current cell's coord in cell [ina504].

We now have three coordinates. The first is the coordinate of the origin cell which is stored in cell [dum504]. From this coordinate, we can extract the row and the column of the cell. The second is the coordinate of a cell in the next sheet which is stored in cell [dumc504]. We can extract the sheet letter for the first cell in the range to erase. The third coordinate of a cell in the last sheet of the worksheet which is stored in cell [ina504]. From this coordinate, we can extract the last sheet letter. For example let us assume that the origin cell is the F35 cell in the "C" sheet and that the last sheet is the "X" sheet. Therefore the first coordinate is the `C:F35`, the second can be `D:A1` and the last can be `X:B2`. Therefore the range that the macro will erase is the `D:F35..X:F35`. The `{LET}` command in the B53 cell stores the result of the formula

```
@LEFT(dumc504,@FIND(":",dumc504,0))&@RIGHT(dum504,@LENGTH(dum504)
    @FIND(":",dum504,0))&".."&@LEFT(ina504,@FIND(":",ina504,0))
    &@RIGHT(dum504,@LENGTH(dum504)-@FIND(":",dum504,0))
```

in cell [eratop504]. The result in our example is the D:F35..X:F35 range. The macro continues as in the last four menu options and issues /RE{eratop504}~~ to erase the D:F35..X:F35 range. Notice the {eratop504} command inside the /RE process. The {eratop504} command injects the content of the [eratop504] routine into the panel because it contains only the address of the range to erase. Last, the macro issues the indirect {GOTO}{dum504}~ macro command to send the cell pointer back its point of origin.

The sixth menu option is [**O**ut]:

```
Out
Erase all the cells in the previous sheets
{IF @LEFT(rel504,1)<>"@" }{outaa504}
{MENUBRANCH menu1504}
```

When you choose this menu option, the macro issues {IF @LEFT(rel504,1)<>"@"} to check if the first character of the content of cell [rel504] is the "@" character. If so, you are using a 3-D release and the macro issues the {outaa504} routine command. When the routine is finished, macro control returns to the {MENUBRANCH menu1504} command which re-starts the custom menu. If the condition is not true, you are using a 2-D release and the macro issues {MENUBRANCH menu1504} which re-starts the custom menu.

```
     A          B        C       D       E       F       G       H       I
57 outaa504     {WINDOWSOFF}{IF @LEFT(rel504,1)<>"@" }{RECALC out504}
                {PS}{LET dumc504,@CELLPOINTER("coord")}~{LET eratop504
                ,out504&".."&@LEFT(dumc504,@FIND(":",dumc504,0))&
                @RIGHT(out504,@LENGTH(out504)-@FIND(":",out504,0))}~
                /RE{eratop504}~~{GOTO}{dumc504}~
58 !            {NS}{WINDOWSON}
```

The [outaa504] routine starts with {WINDOWSOFF} which freezes the screen activity, then the macro issues {IF @LEFT(rel504,1)<>"@"} to make sure that you are using a 3-D release and {RECALC out504} to update the dynamic string formula in cell [out504] and issues {PS} to move the cell pointer to the previous sheet..

```
30 out504       +"A:"&@CELLPOINTER("address")
```

The result of the formula in cell [out504] is the cell coordinate in the first sheet (the "A" sheet), which is the A:F35 cell if the current cell is the F35 cell. Next, the macro issues {LET dumc504,@CELLPOINTER("coord")}~ to store the current cell coordinate in cell [dumc504]. The routine uses this data to extract the sheet letter of the current sheet which is the "B" sheet (because the origin sheet is the "C" sheet). The next {LET} command stores the result of the

```
out504&".."&@LEFT(dumc504,@FIND(":",dumc504,0))&@RIGHT(out504
   ,@LENGTH(out504)-@FIND(":",out504,0))
```

formula which is the address of the range to be erased in cell [eratop504]. In our example, the range address is A:F35..B:$F$35. Now the macro issues /RE{eratop504}~~ to erase the A:F35..B:$F$35 range, and issues {GOTO}{dumc504}~{NS} to move the cell pointer back to its point of origin. Notice the {eratop504} command inside the /RE process. The {eratop504} command injects the content of the [eratop504] routine into the panel because it contains only the address of the range to erase.

The seventh menu option is [**M**ove]:

```
Move
Move the cell pointer to other location
{move504}
{MENUBRANCH menu1504}
```

When you choose this menu option, the macro issues the {move504} routine command which starts the [move504] routine.

	A	B	C	D	E	F	G	H	I
60	move504	Use the arrow keys to move the cell pointer and press RETURN: {GET key504}{ESC}							
61	!	{IF key504="~"}{RETURN}							
62	!	{IF key504="{PGDN}"#OR#key504="{PGUP}"#OR#key504="{DOWN}"#OR#key504="{UP}"#OR#key504="{LEFT}"#OR#key504="{RIGHT}"}{key504}							
63	!	{IF @LEFT(rel504,1)<>"@"#AND#(key504="{NS}"#OR#key504="{PS}")}{key504}							
64	!	{BRANCH move504}							

The [move504] routine allows you to move the cell pointer to any cell in the worksheet using the direction keys. When the cell pointer is in place, you can use the rest of the menu options in the custom menu. The routine starts with the "Use the arrow keys to move the cell pointer and press RETURN:" message prompt which is displayed in the panel. Lotus writes the message into the panel because it does not contain any Lotus valid command. Then the routine continues with {GET key504} which stores the key that you press in cell [key504], then stops the macro execution and waits for you to read the message and use the direction keys and/or the ENTER key. When you press ENTER, the macro resumes control and immediately issues {ESC} to clear the message from the panel before Lotus writes the message to the current cell.

The routine continues with a series of {IF} commands to check the key that you press. The first is {IF key504="~"} which checks if you pressed the ENTER key. If so, the macro issues {RETURN} to return to the custom menu. The second is the

```
{IF key504="{PGDN}"#OR#key504="{PGUP}"#OR#key504="{DOWN}"#OR#
    key504="{UP}"#OR#key504="{LEFT}"#OR#key504="{RIGHT}"}
```

command which checks if you pressed one of the direction keys in the numeric key pad. If so, the macro issues the {key504} routine command which starts the [key504] routine. Recall that the cell named [key504] contains the key that you pressed, therefore when the macro issues the {key504} command, the key is executed as though pressed from the keyboard. The third is the

```
{IF @LEFT(rel504,1)<>"@"#AND#(key504="{NS}"#OR#key504="{PS}")}
```

macro command which checks if you are using a 3-D Lotus release (and if simultaneously you pressed the NEXT SHEET or the PREVIOUS SHEET commands. If so, the macro issues the {key504} command which starts the [key504] routine. Recall that the cell named [key504] contains the key that you pressed; therefore when the macro issues {key504} the key is executed as though pressed from the keyboard. If none of the {IF} commands is true, the macro issues {BRANCH move504} which loops back to the beginning of the [move504] routine and displays the message prompt. This routine is a fine example of how we can monitor and control the keys you are allowed to use when the macro works.

The eighth menu option is [**Q**uit]:

```
Quit
Quit the macro
{BRANCH ret504}
```

When you choose this menu option, the macro issues `{BRANCH ret504}` which routes macro control to the empty [ret504] routine and quits.

Because this macro is quite complicated, here is a full list of all the cell contents and formulas:

```
A1: U '*---A macro to ERASE from the cell next to the cell pointer to
A2: U '    ends of row, columns, sheets
A3: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define
A4: '    the range names in this column (starts with the \Z macro name)
A5: '*---Hold the [ALT] key and press [Z] to activate the macro
A6: '!
A7: U '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
A8: U '                  IT WILL WORK IN RELEASE 2.0 AND UP
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'ERASENDS
B11: '{LET rel504,@INFO("release")}~{MENUBRANCH menu1504}
A12: '!
A13: 'menu1504
B13: 'Right
C13: 'Left
D13: 'Up
E13: 'Down
F13: 'In
G13: 'Out
H13: 'Move
I13: 'Quit
A14: '!
B14: 'Erase all the cells right to the current cell
C14: 'Erase all the cells left to the current cell
D14: 'Erase all the cells up to the current cell
E14: 'Erase all the cells down to the current cell
F14: 'Erase all the cells in the next sheets
G14: 'Erase all the cells in the previous sheets
H14: 'Move the cell pointer to other location
I14: 'Quit the macro
A15: '!
B15: '{rightaa504}
C15: '{leftaa504}
D15: '{topaa504}
E15: '{bottomaa504}
F15: '{IF @LEFT(rel504,1)<>"@"}{inaa504}
G15: '{IF @LEFT(rel504,1)<>"@"}{outaa504}
H15: '{move504}
I15: '{BRANCH ret504}
A16: '!
B16: '{MENUBRANCH menu1504}
C16: '{MENUBRANCH menu1504}
D16: '{MENUBRANCH menu1504}
E16: '{MENUBRANCH menu1504}
F16: '{MENUBRANCH menu1504}
G16: '{MENUBRANCH menu1504}
H16: '{MENUBRANCH menu1504}
A17: '!
A18: 'ret504
A19: '!
A20: 'rel504
B20: '@INFO("release")
A21: '!
```

```
A22: 'top504
B22: U @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER("address")
       ,1)+1)&"1"
A23: '!
A24: 'bottom504
B24: U @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER("address"),1)
       +1)&"8192"
A25: '!
A26: 'left504
B26: U +"A"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER
       ("address"))-@FIND("$",@CELLPOINTER("address"),1)-1)
A27: '!
A28: 'right504
B28: U +"IV"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER
       ("address"))-@FIND("$",@CELLPOINTER("address"),1)-1)
A29: '!
A30: 'out504
B30: U +"A:"&@CELLPOINTER("address")
A31: '!
A32: 'ina504
B32: '$U:$H$38
A33: '!
A34: 'dum504
B34: '$A:$D$10
A35: '!
A36: 'eratop504
B36: 'A:$D$10..$J:$D$10
A37: '!
A38: 'dumc504
B38: '$J:$D$10
A39: '!
A40: 'topaa504
B40: '{WINDOWSOFF}{RECALC top504}{UP}{LET eratop504,@CELLPOINTER
     ("address")&".."&TOP504}~{DOWN}/RE{eratop504}~~
A41: '!
B41: '{WINDOWSON}
A42: '!
A43: 'bottomaa504
B43: '{WINDOWSOFF}{RECALC bottom504}{DOWN}{LET eratop504,@CELLPOINTER
     ("address")&".."&bottom504}~{UP}/RE{eratop504}~~
A44: '!
B44: '{WINDOWSON}
A45: '!
A46: 'leftaa504
B46: '{WINDOWSOFF}{RECALC left504}{LEFT}{LET eratop504,@CELLPOINTER
     ("address")&".."&left504}~{RIGHT}/RE{eratop504}~~
A47: '!
B47: '{WINDOWSON}
A48: '!
A49: 'rightaa504
B49: '{WINDOWSOFF}{RECALC right504}{RIGHT}{LET eratop504,@CELLPOINTER
     ("address")&".."&right504}~{LEFT}/RE{eratop504}~~
A50: '!
B50: '{WINDOWSON}
A51: '!
A52: 'inaa504
B52: '{WINDOWSOFF}{IF @LEFT(rel504,1)<>"@"}{LET dum504,@CELLPOINTER
     ("coord")}~{NS}{LET dumc504,@CELLPOINTER("coord")}~{LC}{END}{NS}
     {LET ina504,@CELLPOINTER("coord")}~
A53: '!
B53: '{IF @LEFT(rel504,1)<>"@"}{LET eratop504,@LEFT(dumc504,@FIND(":",
     dumc504,0))&@RIGHT(dum504,@LENGTH(dum504)-@FIND(":",dum504,0))&".."&
     @LEFT(ina504,@FIND(":",ina504,0))&@RIGHT(dum504,@LENGTH(dum504)-
     @FIND(":",dum504,0))}~/re{eratop504}~~
A54: '!
B54: '{IF @LEFT(rel504,1)<>"@"}{GOTO}{dum504}~
A55: '!
B55: '{WINDOWSON}
A56: '!
A57: 'outaa504
B57: '{WINDOWSOFF}{IF @LEFT(rel504,1)<>"@"}{RECALC out504}{PS}
```

```
     {LET dumc504,@CELLPOINTER("coord")}~{LET eratop504,out504&".."
     &@LEFT(dumc504,@FIND(":",dumc504,0))&@RIGHT(out504,@LENGTH(out504)-
     @FIND(":",out504,0))}~/RE{eratop504}~~{GOTO}{dumc504}~
A58: '!
B58: '{NS}{WINDOWSON}
A59: '!
A60: 'move504
B60: 'Use the arrow keys to move the cell pointer and press RETURN:
     {GET key504}{ESC}
A61: '!
B61: '{IF key504="~"}{RETURN}
A62: '!
B62: '{IF key504="{PGDN}"#OR#key504="{PGUP}"#OR#key504="{DOWN}"#OR#
     key504="{UP}"#OR#key504="{LEFT}"#OR#key504="{RIGHT}"}{key504}
A63: '!
B63: '{IF @LEFT(rel504,1)<>"@"#AND#(key504="{NS}"#OR#key504="{PS}")}
     {key504}
A64: '!
B64: '{BRANCH move504}
A65: '!
A66: 'key504
B66: '~
```

[9] Delete Partial Columns and Rows (Ranges)

```
     A                B                C                D                E
 1 *---A macro to DELETE a range, like column/row delete but limited
 2     to a specified row/column length
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                 IT WILL WORK IN RELEASE 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 RANGDEL1        {WINDOWSOFF}{PANELOFF}/RNCDelete range ?~~/RND
                   Delete range ?~/RNC{WINDOWSON}{PANELON}Delete range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Delete range ?~
12 !               {LET hereabs527,@CELLPOINTER("address")}~
13 cont527         {LET col1527,@CELLPOINTER("col")}~
14 !               {LET col2527,@CELLPOINTER("col")+@COLS(Delete range ?)
                   -1}~{LET col4527,@COLS(Delete range ?)}~
15 !               {LET row1527,@CELLPOINTER("row")}~
16 !               {LET row2527,@CELLPOINTER("row")+@ROWS(Delete range ?)
                   -1}~{LET row4527,@ROWS(Delete range ?)}~
17 !               {LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<>
                   "@"}{LET sheets527,@SHEETS(Delete range ?)}~
18 !               {END}{HOME}~{LET row3527,@CELLPOINTER("row")}~
                   {LET col3527,@CELLPOINTER("col")}~{GOTO}Delete range ?~
19 !               {MENUCALL menu1527}
20 !               {GOTO}{hereabs527}~/RNDDelete range ?~
21 !
22 !
23 hereabs527       $B$13
24 !
25 counterb527                5
26 !
27 menu1527        Columnwise    Rowwise          Quit
28 !               Delete the ranDelete the range Quit the macro
29 !               {LET counterb5{LET counterb527,{BRANCH ret527}
30 !
31 ret527
32 rel527          3.10.00
33 col1527                    3
34 col2527                    6
35 col3527                  137
36 col4527                    4
37 row1527                    2
38 row2527                  100
39 row3527                    2
40 row4527                    9
41 row5527                   11
42 sheets527                  1
43 chrz1527        G
44 chrz2527        F
45 chrz3527        G
46 chrz4527        C
47 rightaa527      {WINDOWSON}{PANELON}{RECALC chrz1527}{RECALC chrz2527}
                   {RECALC chrz3527}{RECALC chrz4527}{RECALC row5527}
                   {RECALC rng1527}/M
48 rng1527         G2..EG10
49 !               ~
50 rng1a527        ~
51 !               {LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<>"@"}
                   {GOTO}{hereabs527}~{LET counterb527,counterb527+1}~
                   {IF counterb527<sheets527}{NS}{GOTO}{hereabs527}~
                   {BRANCH rightaa527}
52 !               {WINDOWSON}
53 !
54 downaa527       {WINDOWSON}{PANELON}{RECALC chrz1527}{RECALC chrz2527}
                   {RECALC chrz3527}{RECALC chrz4527}{RECALC row5527}
```

```
                  {RECALC rng2527}/M
55 rng2527        C11..F100
56 !              ~
57 rng2a527       ~
58 !              {LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<>"@"}
                  {GOTO}{hereabs527}~{LET counterb527,counterb527+1}~
                  {IF counterb527<sheets527}{NS}{GOTO}{hereabs527}~
                  {BRANCH downaa527}
59 !              {WINDOWSON}
60 !
61 outaa527       {}
```

This macro fills a gap in Lotus 1-2-3. In Lotus you can delete only a complete column or row but cannot a partial row or column or a range. For example, if you use a complicated worksheet with tables positioned one over the other, and you want to delete a column in a table you have a problem because you may also delete a column in other table. This macro allows you to delete partial rows and columns. To better understand this macro let's look at the following table:

	A	B	C	D	E	F
1	21	41	61	81	101	121
2	22	42	62	82	102	122
3	23	43	63	83	103	123
4	24	44	64	84	104	124
6	26	46	65	85	105	125
7	27	47	67	87	107	127
8	28	48	68	88	108	128
9	29	49	69	89	109	129
10	20	40	60	80	100	120

Let's assume that you started the macro and highlighted the B4..E7 range (three rows and four columns) to delete; you can delete in the rowwise direction. In that case, the result will be

	A	B	C	D	E	F
1	21	41	61	81	101	121
2	22	42	62	82	102	122
3	23	43	63	83	103	123
4	24	48	68	88	108	124
6	26	49	69	89	109	125
7	27	40	60	80	100	127
8	28					128
9	29					129
10	20					120

The macro erased the B4..E7 range and pushed the rest of the four columns up to fill the deleted range. The macro moved up all the data beneath the deleted range. If you choose to delete in the columnwise direction, the result is:

	A	B	C	D	E	F
1	21	41	61	81	101	121
2	22	42	62	82	102	122
3	23	43	63	83	103	123
4	24	124				
6	26	126				
7	27	127				
8	28	48	68	88	108	128
9	29	49	69	89	109	129
10	20	40	60	80	100	120

We can see that the macro erased the B4..E7 range and pushed all the data of the right to the left to the B4..E7 range. See the RANGEINS.WK1 macro which also does the opposite, it allows you to insert partial rows and columns. The macro uses a custom menu with three menu options, which cannot fit clearly on one page without overlapping. To help you key this macro into Lotus

1-2-3, here is the code of every menu option separately. See the full list of all cell contents at the end of this section. If you intend to key in this macro, you should use the code as it appears here, NOT the code as it appears in the main listing.

```
Columnwise
Delete the range and move left all the cells right to the range
{LET counterb527,0}~/REDelete range ?~{rightaa527}

Rowwise
Delete the range and move upward all the cells down to the range
{LET counterb527,0}~/REDelete range ?~{downaa527}

Quit
Quit the macro
{BRANCH ret527}
```

Notice too, that the code in the B41, B43, B44, B45, B46, B48 and the B55 cells of the macro is the result of the following dynamic string formulas. If you intend to key in the code, you must use the following formulas, NOT the code as it appears in the main listing.

```
41 row5527          +B40+B37
43 chrz1527         @IF(@MOD(B33+B36,26)=0,"Z",@CHAR(@MOD(B33+B36,26)+64))
44 chrz2527         @IF(@MOD(B34,26)=0,"Z",@CHAR(@MOD(B34,26)+64))
45 chrz3527         @IF(@MOD(B35,26)=0,"Z",@CHAR(@MOD(B35,26)+64))
46 chrz4527         @IF(@MOD(B33,26)=0,"Z",@CHAR(@MOD(B33,26)+64))
48 rng1527          @IF(B33+B36<27,@CHAR(B33+B36+64),@CHAR(@INT((B33+B36)
                    /26)+64)&B43)&@STRING(B37,0)&".."&@IF(B35<27,@CHAR(B35
                    +64),@CHAR(@INT(B35/26)+64)&B45)&@STRING(B38,0)
55 rng2527          @IF(B33<27,@CHAR(B33+64),@CHAR(@INT(B33/26)+64)&B46)
                    &@STRING(B37+B40,0)&".."&@IF(B34<27,@CHAR(B34+64),
                    @CHAR(@INT(B34/26)+64)&B44)&@STRING(B39,0)
```

	A	B	C	D	E
11	RANGDEL1	{WINDOWSOFF}{PANELOFF}/RNCDelete range ?~~/RND Delete range ?~/RNC{WINDOWSON}{PANELON}Delete range ?~ {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Delete range ?~			

The macro issues the {WINDOWSOFF}{PANELOFF} macro commands to freeze the screen and panel display activity. Then the macro uses the "safe technique" to prompt you to paint the range to delete and simultaneously assigns the [Delete range ?] name to the same range, which acts as a prompt.

	A	B	C	D	E
12	!	{LET hereabs527,@CELLPOINTER("address")}~			
13	cont527	{LET col1527,@CELLPOINTER("col")}~			
14	!	{LET col2527,@CELLPOINTER("col")+@COLS(Delete range ?) -1}~{LET col4527,@COLS(Delete range ?)}~			
15	!	{LET row1527,@CELLPOINTER("row")}~			
16	!	{LET row2527,@CELLPOINTER("row")+@ROWS(Delete range ?) -1}~{LET row4527,@ROWS(Delete range ?)}~			
17	!	{LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<> "@"}{LET sheets527,@SHEETS(Delete range ?)}~			
18	!	{END}{HOME}~{LET row3527,@CELLPOINTER("row")}~ {LET col3527,@CELLPOINTER("col")}~{GOTO}Delete range ?~			
19	!	{MENUCALL menu1527}			

The macro stores the current address in cell [hereabs527]. Later the macro will use it to return to this address. The macro continues with a series of {LET} commands which store information that the macro will use later. The {LET col1527,@CELLPOINTER("col")}~ commands store the current column's number in cell [col1527] which is the first column of the [Delete range ?] range. The {LET col2527,@CELLPOINTER("col")+ @COLS(Delete range ?)-

1}~ commands store the column number of the last column of the [Delete range ?] range in cell [col2527]. The {LET col4527,@COLS(Delete range ?)}~ commands store the number of columns of the [Delete range ?] range in cell [col4527]. The {LET row1527,@CELLPOINTER("row")}~ store the row number of the current row (the first row of the [Delete range ?] range) in cell [row1527].

The {LET row2527,@CELLPOINTER("row")+@ROWS(Delete range ?)-1}~ command stores the number of the last row of the [Delete range ?] range in the cell named [row2527], and the {LET row4527,@ROWS(Delete range ?)}~ macro command stores the number of rows of the [Delete range ?] range in cell [row4527]. The {LET sheets527,@SHEETS (Delete range ?)}~ macro command stores the number of sheets of the [Delete range ?] range in cell [sheets527]. This command will not create an error in a 2-D release even though the @SHEETS function is valid only in at 3-D Lotus release. When Lotus encounters a function which it cannot recognize in a {LET} command it sores it as text. For example, {LET A1,@INFO("release")}~ insert the Lotus version number in A1 if you are using a 3-D release, but inserts the "@INFO("release")" string in A1 if you are using a 2-D release.

Next the macro issues {GOTO}{END}{HOME}~, which move the cell pointer to the last occupied cell in the worksheet, and issues {LET row3527,@CELLPOINTER("row")}~, which store the row number of the last cell in cell [row3527], and issues {LET col3527, @CELLPOINTER("col")}~, which store the column number of the last cell in cell [col3527]. Now that all the properties of the range to delete and the size of the worksheet are known, the macro issues {GOTO}Delete range ?~, which moves the cell pointer back to the upper left cell of [Delete range ?], and issues {MENUCALL menu1527}, which activates the [menu1527] custom menu. The first menu option is [**C**olumnwise]:

```
Columnwise
Delete the range and move left all the cells right to the range
{LET counterb527,0}~/REDelete range ?~{rightaa527}
```

The routine starts with {LET counterb527,0} which sets the value in cell [counterb527] to zero. Next the macro issues /REDelete range ?~ to erase the content of the [Delete range ?] range and issues the {rightaa527} routine command to start the [rightaa527] routine.

	A	B	C	D	E
47	rightaa527	{WINDOWSON}{PANELON}{RECALC chrz1527}{RECALC chrz2527} {RECALC chrz3527}{RECALC chrz4527}{RECALC row5527} {RECALC rng1527}/M			
48	rng1527	G2..EG10			
49	!	~			
50	rng1a527	~			
51	!	{LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<>"@"} {GOTO}{hereabs527}~{LET counterb527,counterb527+1}~ {IF counterb527<sheets527}{NS}{GOTO}{hereabs527}~ {BRANCH rightaa527}			
52	!	{WINDOWSON}			

The routine starts with {WINDOWSON}{PANELON} which free the screen and panel activities and then issues the six {RECALC} macro commands to update the formulas in the [chrz1527], [chrz2527], [chrz3527], [chrz4527], [row5527], [rng1527] cells. Let us look at the formulas:

The first is the formula in cell [chrz1527]

43	chrz1527	@IF(@MOD(B33+B36,26)=0,"Z",@CHAR(@MOD(B33+B36,26)+64))

This formula calculates the letter of the column right to the last column of [Delete range ?]. If the column is designated by two letters, this formula calculates only the second letter.

The second is the formula in cell [chrz2527]

```
44 chrz2527        @IF(@MOD(B34,26)=0,"Z",@CHAR(@MOD(B34,26)+64))
```

This formula calculates the last column's letter of [Delete range ?]. If the column is designated by two letters, this formula calculates only the second letter.

The third is the formula in cell [chrz3527]

```
45 chrz3527        @IF(@MOD(B35,26)=0,"Z",@CHAR(@MOD(B35,26)+64))
```

This formula calculates the last occupied column's letter of the worksheet. If the column is designated by two letters, this formula calculates only the second letter.

The fourth is the formula in cell [chrz4527]

```
46 chrz4527        @IF(@MOD(B33,26)=0,"Z",@CHAR(@MOD(B33,26)+64))
```

This formula calculates the first column's letter of [Delete range ?]. If the column is designated by two letters, this formula calculates only the second letter.

The fifth is the formula in cell [row5527]

```
41 row5527         +B40+B37
```

This formula calculates the row number of the row beneath the last row of the [Delete range ?] range. If the column is designated by two letters, this formula calculates only the second letter.

The sixth is the formula in cell [rng1527]

```
48 rng1527         @IF(B33+B36<27,@CHAR(B33+B36+64),@CHAR(@INT((B33+B36)
                   /26)+64)&B43)&@STRING(B37,0)&".."&@IF(B35<27,@CHAR(B35
                   +64),@CHAR(@INT(B35/26)+64)&B45)&@STRING(B38,0)
```

This formula calculates the range to move to the left to fill the gap after the macro erases the [Delete range ?] range, if you choose the [Columnwise] menu option. For example, if the range named [Delete range ?] is the C2..F10 range and the last occupied column of the worksheet is the EG column, the result will be the G2..EG10 range, which contains all the data in the worksheet right to the [Delete range ?] range.

Next the macro issues `/MG2..EG10~~` to move the G2..EG10 range to the current cell position and completes the process.

```
            A                    B                  C                  D            E
51 !                {LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<>"@"}
                    {GOTO}{hereabs527}~{LET counterb527,counterb527+1}~
                    {IF counterb527<sheets527}{NS}{GOTO}{hereabs527}~
                    {BRANCH rightaa527}
52 !                {WINDOWSON}
```

Because a 3-D Lotus worksheet may have more than one sheet, the macro issues `{LET rel527,@INFO("release")}~` which store the result of the `@INFO("release")` function in [rel527]. Then it issues `{IF @LEFT(rel527,1)<>"@"}` to check if you are using a 2-D or a 3-D Lotus release. If "@" is the first character of the content of cell [rel527] then you are using a 2-D Lotus release, otherwise you are using a Lotus 3-D release. If you are using a 3-D release, the macro issues the `{GOTO}{hereabs527}~` indirect command to move the cell pointer to the upper left cell in the current sheet of the [Delete range ?] range, and then issues `{LET counterb527,counterb527+1}~` to increase the counter in cell [counterb527] by one.

Next, the macro issues `{IF counterb527<sheets527}` to compare the counter value in cell [counterb527] to the number of sheets in the [Delete range ?] range. If the counter value is still less than the number of sheets in [Delete range ?], it means that the macro has to process more sheets before it can quit. Therefore the macro issues `{NS}{GOTO} {hereabs527}~` to move the cell pointer to the upper left cell of the [Delete range ?] range in the new sheet and issues `{BRANCH rightaa527}` to loop back to the beginning of the [rightaa527] routine. The routine is finished only when the value in [counterb527] is greater than the number of sheets in the [Delete range ?] range. The second menu option is
[**R**owwise]:

```
Rowwise
Delete the range and move upward all the cells down to the range
{LET counterb527,0}~/REDelete range ?~{downaa527}
```

The routine starts with `{LET counterb527,0}` which sets the value in cell [counterb527] to zero. Next the macro issues `/REDelete range ?~` to erase the content of the [Delete range ?] range and issues the `{downaa527}` routine command to start the [downaa527] routine.

	A	B	C	D	E
54	downaa527	{WINDOWSON}{PANELON}{RECALC chrz1527}{RECALC chrz2527} {RECALC chrz3527}{RECALC chrz4527}{RECALC row5527} {RECALC rng2527}/M			
55	rng2527	C11..F100			
56	!	~			
57	rng2a527	~			
58	!	{LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<>"@"} {GOTO}{hereabs527}~{LET counterb527,counterb527+1}~ {IF counterb527<sheets527}{NS}{GOTO}{hereabs527}~ {BRANCH downaa527}			
59	!	{WINDOWSON}			

The routine starts with `{WINDOWSON}{PANELON}` which free the screen and panel activities and then issues the seven `{RECALC}` commands to update the formulas in the [chrz1527], [chrz2527], [chrz3527], [chrz4527], [row5527], [rng2527], [rng2a527] cells. Let's look at the formulas:

The first five formulas are the same as we have seen in the previous menu option, therefore let's look at the sixth formula which is the formula in cell [rng2527]

```
55 rng2527          @IF(B33<27,@CHAR(B33+64),@CHAR(@INT(B33/26)+64)&B46)
                    &@STRING(B37+B40,0)&".."&@IF(B34<27,@CHAR(B34+64),
                    @CHAR(@INT(B34/26)+64)&B44)&@STRING(B39,0)
```

This formula calculates the range to move up to fill the gap after the macro erases the [Delete range ?] range when you choose the [Rowwise] menu option. For example, if the range named

[Delete range ?] is the C2..F10 range and the last occupied row of the worksheet is the 100 row, the result will be the C11..F100 range which contains all the data in the worksheet below the [Delete range ?] range.

Next, the macro issues `/MC11..F100~~` to move the C11..F100 range to the current cell position and completes the process. The rest of the code is identical to the code of the [rightaa527] except that the macro instead loops back to the [downaa527] routine. The third menu option is [**Q**uit]:

```
Quit
Quit the macro
{BRANCH ret527}
```

When you choose this option, the macro issues `{BRANCH ret527}` which routes macro control to the empty routine [ret527] and quits. Now the macro returns control to the command after the `{MENUCALL menu1527}` command in the main macro which issues `{GOTO} {hereabs527}~/RNDDelete range ?~` to move the cell pointer back to its point of origin and then delete the temporary [Delete range ?] range name to leave a clean worksheet.

Here is the list of all the cell contents to help you to key this macro into 1-2-3.

```
A1: U [W14] '*---A macro to DELETE a range, like column/row delete but
          limited
A2: U [W14] '    to a specified row/column length
A3: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A4: [W14] '    range names in this column (starts with the \Z macro name)
A5: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A6: [W14] '!
A7: U [W14] '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3
          RELEASE
A8: U [W14] '                  IT WILL WORK IN RELEASE 2.0 AND UP
A9: [W14] '!
A10: U [W14] '\Z
B10: '{BREAKON}
A11: U [W14] 'RANGDEL1
B11: '{WINDOWSOFF}{PANELOFF}/RNCDelete range ?~~/RNDDelete range ?~
     /RNC{WINDOWSON}{PANELON}Delete range ?~{BS}{BS}{?}~{WINDOWSOFF}
     {GOTO}Delete range ?~
A12: [W14] '!
B12: '{LET hereabs527,@CELLPOINTER("address")}~
A13: [W14] 'cont527
B13: '{LET col1527,@CELLPOINTER("col")}~
A14: [W14] '!
B14: '{LET col2527,@CELLPOINTER("col")+@COLS(Delete range ?)-1}~
     {LET col4527,@COLS(Delete range ?)}~
A15: [W14] '!
B15: '{LET row1527,@CELLPOINTER("row")}~
A16: [W14] '!
B16: '{LET row2527,@CELLPOINTER("row")+@ROWS(Delete range ?)-1}~
     {LET row4527,@ROWS(Delete range ?)}~
A17: [W14] '!
B17: '{LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<>"@"}
     {LET sheets527,@SHEETS(Delete range ?)}~
A18: [W14] '!
B18: '{END}{HOME}~{LET row3527,@CELLPOINTER("row")}~{LET col3527,
     @CELLPOINTER("col")}~{GOTO}Delete range ?~
A19: [W14] '!
B19: '{menucall menu1527}
A20: [W14] '!
B20: '{GOTO}{hereabs527}~/RNDDelete range ?~
A21: [W14] '!
A22: [W14] '!
A23: [W14] 'hereabs527
```

```
B23: '$B$13
A24: [W14] '!
A25: [W14] 'counterb527
B25: 5
A26: [W14] '!
A27: [W14] 'menu1527
B27: 'Columnwise
C27: 'Rowwise
D27: 'Quit
A28: [W14] '!
B28: 'Delete the range and move left all the cells right to the range
C28: 'Delete the range and move upward all the cells down to the range
D28: 'Quit the macro
A29: [W14] '!
B29: '{LET counterb527,0}~/redelete range ?~{rightaa527}
C29: '{LET counterb527,0}~/redelete range ?~{downaa527}
D29: '{BRANCH ret527}
A30: [W14] '!
A31: [W14] 'ret527
A32: [W14] 'rel527
B32: '3.10.00
A33: [W14] 'col1527
B33: 2
A34: [W14] 'col2527
B34: 5
A35: [W14] 'col3527
B35: 6
A36: [W14] 'col4527
B36: 4
A37: [W14] 'row1527
B37: 13
A38: [W14] 'row2527
B38: 16
A39: [W14] 'row3527
B39: 17
A40: [W14] 'row4527
B40: 4
A41: [W14] 'row5527
B41: U +B40+B37
A42: [W14] 'sheets527
B42: 5
A43: [W14] 'chrz1527
B43: U @IF(@MOD(B33+B36,26)=0,"Z",@CHAR(@MOD(B33+B36,26)+64))
A44: [W14] 'chrz2527
B44: U @IF(@MOD(B34,26)=0,"Z",@CHAR(@MOD(B34,26)+64))
A45: [W14] 'chrz3527
B45: U @IF(@MOD(B35,26)=0,"Z",@CHAR(@MOD(B35,26)+64))
A46: [W14] 'chrz4527
B46: U @IF(@MOD(B33,26)=0,"Z",@CHAR(@MOD(B33,26)+64))
A47: [W14] 'rightaa527
B47: '{WINDOWSON}{PANELON}{RECALC CHRZ1527}{RECALC CHRZ2527}
     {RECALC CHRZ3527}{RECALC CHRZ4527}{RECALC ROW5527}{RECALC rng1527}
     {RECALC rng1a527}/M
A48: [W14] 'rng1527
B48: U @IF(B33+B36<27,@CHAR(B33+B36+64),@CHAR(@INT((B33+B36)/26)+64)&B43)
       &@STRING(B37,0)&".."&@IF(B35<27,@CHAR(B35+64),@CHAR(@INT(B35/26)+
       64)&B45)&@STRING(B38,0)
A49: [W14] '!
B49: '~
A50: [W14] 'rng1a527
B50: '~
A51: [W14] '!
B51: '{LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<>"@"}{GOTO}
     {hereabs527}~{LET counterb527,counterb527+1}~{IF counterb527<
     sheets527}{NS}{GOTO}{hereabs527}~{BRANCH rightaa527}
A52: [W14] '!
B52: '{WINDOWSON}
A53: [W14] '!
A54: [W14] 'downaa527
B54: '{WINDOWSON}{PANELON}{RECALC CHRZ1527}{RECALC CHRZ2527}
     {RECALC CHRZ3527}{RECALC CHRZ4527}{RECALC ROW5527}{RECALC rng2527}
```

```
     {RECALC rng2a527}/M
A55: [W14] 'rng2527
B55: U @IF(B33<27,@CHAR(B33+64),@CHAR(@INT(B33/26)+64)&B46)&@STRING(B37+
       B40,0)&".."&@IF(B34<27,@CHAR(B34+64),@CHAR(@INT(B34/26)+64)&B44)
       &@STRING(B39,0)
A56: [W14] '!
B56: '~
A57: [W14] 'rng2a527
B57: '~
A58: [W14] '!
B58: '{LET rel527,@INFO("release")}~{IF @LEFT(rel527,1)<>"@"}{GOTO}
     {hereabs527}~{LET counterb527,counterb527+1}~{IF counterb527<
     sheets527}{NS}{GOTO}{hereabs527}~{BRANCH downaa527}
A59: [W14] '!
B59: '{WINDOWSON}
A60: [W14] '!
A61: [W14] 'outaa527
B61: '{}
```

[6] Delete Rows with Zeros or Blank Cells in a Range

```
     A                B                C                D                E
 1 *---A macro to DELETE rows containing 0 or empty cell in a range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---In release 3 make sure that the GROUP option is DISABLED
 6 *---In release 2/2.01/2.2 the macro must be located ABOVE the range
 7 !
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                    IT WILL WORK IN LOTUS 2.0 AND UP
10 \Z              {BREAKON}
11 ROWDELET        {LET rel156,@INFO("release")}~
12 !               {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{GOTO}Which range ?~{LET rowsaa156,@ROWS
                   (Which range ?)}~{WINDOWSOFF}{LET counterb156,0}~
                   {LET colss156,@COLS(Which range ?)}~
13 !               {IF @LEFT(rel156,1)<>"@"}{LET sheets156,@SHEETS
                   (Which range ?)}~
14 cont156         {LET hereabs156,@CELLPOINTER("address")}~
                   {LET counter1156,0}{LET rowsa156,rowsaa156}~
11 !               {FOR counter1156,0,colsS156-1,1,labels1156}
16 !               {LET rel156,@INFO("release")}~{IF @LEFT(rel156,1)<>"@"}
                   {GOTO}{hereabs156}~{LET counterb156,counterb156+1}~
                   {IF counterb156<sheets156}{NS}{GOTO}{hereabs156}~
                   {BRANCH cont156}
17 !               /RNCWhich range ?~~/RNDWhich range ?~{WINDOWSON}
18 !
19 counter1156           4
20 counter1a156          0
21 labels1156      {FOR counter1a156,0,rowsa156-1,1,labels1a156}~{RIGHT}
                   {UP rowsa156}{LET counter1a156,0}~
22 !
23 labels1a156     {IF (@CELLPOINTER("contents")=0#AND#@CELLPOINTER
                   ("type")="v")#OR#@CELLPOINTER("type")="b"}/WDR~
                   {LET rowsa156,rowsa156-1}~{RETURN}
24 !               {DOWN}
25 !
26 !
27 rowsa156              4
28 rowsaa156             8
29 sheets156             4
30 COLSS156              4
31 !
32 counterb156           4
33 hereabs156      $A$1
34 !
35 rel156          3.00.00
36 !
37 hereiam156
```

This macro will delete any row in a range that has a blank cell or a cell with a zero. Till now I could not figure out why such a macro is needed, but from consulting with other users it seems that such a macro has some use. The macro starts with the `{LET rel156,@INFO("release")}~` macro commands which store the result of the `@INFO("release")` function in cell [rel156] for later use, to check if you are using a 3-D or a 2-D Lotus release.

```
     A                B                C                D                E
12 !               {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{GOTO}Which range ?~{LET rowsaa156,@ROWS
                   (Which range ?)}~{WINDOWSOFF}{LET counterb156,0}~
                   {LET colss156,@COLS(Which range ?)}~
```

The macro issues {WINDOWSOFF}{PANELOFF} to freeze the screen and panel activities. Then the macro uses the "safe technique" to prompt you to paint the range to process, and assigns the [Which range ?] name to the same range, which acts as a prompt. The {?} pauses the macro and wait until you paint the range and press ENTER. Then the macro issues {GOTO}Which range ?~ to move the cell pointer to the upper left cell of the [Which range ?] range and then stores the number of rows of the [Which range ?] range in cell [rowsaa156] using {LET rowsaa156,@ROWS(Which range ?)}~. The macro, again, freezes screen activity using {WINDOWSOFF} and uses {LET counterb156,0}~ to set the value in cell [counterb156], which serves as a counter, to zero. Next it uses {LET colss156,@COLS (Which range ?)}~ to store the number of columns of the [Which range ?] range in cell [colss156].

	A	B	C	D	E
13	!	{IF @LEFT(rel156,1)<>"@"}{LET sheets156,@SHEETS (Which range ?)}~			

Now the macro uses the stored content of [rel156] to check if you are using a 3-D Lotus release. The macro uses the @LEFT(rel156,1) Lotus string function to check if the first character of the cell content is equal or not to the "@" character. If it is equal you are using a 2-D release, but if it is unequal you are using a 3-D release; therefore the macro has to know how many sheets the [Which range ?] range has. The macro uses {LET sheets156 ,@SHEETS(Which range?)}~ to store the number of sheets in the [Which range ?] range in cell [sheets156].

	A	B	C	D	E
14	cont156	{LET hereabs156,@CELLPOINTER("address")}~ {LET counter1156,0}{LET rowsa156,rowsaa156}~			

It is a good practice to record the cell pointer position before moving it to another cell especially if we need to come back to the same cell later on. Therefore the macro uses {LET hereabs156,@CELLPOINTER("address")}~ to store the current cell pointer address in cell [hereabs156]. Next, the macro issues {LET counter1156,0} to reset the value in cell [counter1156] to zero and {LET rowsa156,rowsaa156}~ to copy the content of cell [rowsaa156] to cell [rowsa156].

	A	B	C	D	E
15	!	{FOR counter1156,0,colsS156-1,1,labels1156}			

Now the macro uses this {FOR} command to execute the [labels1156] routine as many times as the number of columns in [Which range ?]; the number of columns is stored in cell [colss156].

	A	B	C	D	E
21	labels1156	{FOR counter1a156,0,rowsa156-1,1,labels1a156}~{RIGHT} {UP rowsa156}{LET counter1a156,0}~			

The [labels1156] routine includes a second {FOR} command that executes the [labels1a156] routine as many times as the number of rows in the [Which range ?] range, the number of rows is stored in cell [rowsa156]. When the macro finishes processing the row, it moves the cell pointer to the next column using the {RIGHT} macro command and then uses the {UP rowsa156} to move the cell pointer up to the first cell of the new column. The disadvantage of this code is {UP rowsa156}, which may take a very long time if the range has many rows. A better approach is to record the first cell of the next column before processing the current column; therefore the code in the B21 should be:

	A	B	C	D	E
21	labels1156	{RIGHT}{LET hereiam156,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a156,0,rowsa156-1,1,labels1a156}~{GOTO}			
		{hereiam156}~{LET counter1a156,0}			

The macro first issues {RIGHT} which moves the cell pointer to the first cell of the next column, then the macro records the address of the cell into cell [hereiam156] using {LET hereiam156,@CELLPOINTER("address")}~ and uses {LEFT} to move the cell pointer back to its place, and continues with the {FOR} loop macro command. When the {FOR} loop is finis hed, the macro issues the indirect {GOTO}{hereiam156}~ macro command to immediately jump to the first cell of the next column which is much faster than the previous code.

	A	B	C	D	E
23	labels1a156	{IF (@CELLPOINTER("contents")=0#AND#@CELLPOINTER			
		("type")="v")#OR#@CELLPOINTER("type")="b"}/WDR~			
		{LET rowsa156,rowsa156-1}~{RETURN}			
24	!	{DOWN}			

The [labels1a156] routine makes sure that the current cell contains a zero "0" value or is blank. If the condition is met, the macro deletes the row using /WDR~. Because the [Which range ?] range now has fewer rows, the macro decreases the value stored in the cell [rowsa156] using {LET rowsa156,rowsa156-1}~. If the condition is not met, the macro moves the cell pointer to the next cell using {DOWN}.

	A	B	C	D	E
16	!	{LET rel156,@INFO("release")}~{IF @LEFT(rel156,1)<>"@"}			
		{GOTO}{hereabs156}~{LET counterb156,counterb156+1}~			
		{IF counterb156<sheets156}{NS}{GOTO}{hereabs156}~			
		{BRANCH cont156}			

Till now the macro treated only one sheet, but what happen if you are using a 3-D Lotus release and the [Which range ?] range has more than one sheet. The code in the B16 takes care of such a case. The macro first stores the result of the @INFO("release") function in cell [rel156] using {LET rel156,@INFO("release")}~, and then uses {IF @LEFT (rel156,1)<>"@"} to check the first character of the content of the [rel156]. If the character is not the "@" character then you are using a 3-D Lotus release. You may ask why not just type the @INFO("release") function in the cell [rel156] instead of using the {LET rel156,@INFO("release")}~ macro command? A good question, but this method has a disadvantage, this function is allowed only in a 3-D release, therefore a file with a 3-D release live function cannot be saved as a *.WK1 file and cannot be used in a 2-D Lotus release. ***Super Power***'s goal is to design macros that work transparently across all the Lotus releases.

	A	B	C	D	E
17	!	/RNCWhich range ?~~/RNDWhich range ?~{WINDOWSON}			

To end this session the macro deletes the [Which range ?] range name, but there is always the chance that the range name does not exist any more, because, if the first row of the range had a zero or a blank cell, the macro deleted the row and the range name with it. To overcome this possibility, the macro recreates the [Which range ?] range name using /RNCWhich range ?~~. If the range name still exists, nothing is changed but if the range name does not exist, the macro creates a new one at the current cell pointer location. Now the macro can safely use /RNDWhich range ?~ to delete the range name without the chance of error.

This macro is quite dangerous since it will deletes rows, even if they are not completely empty, as long as the row in the painted range contains a zero or a blank cell. For less dangerous macros see the ROWSELIM.WK1 and COLSELIM.WK1 which eliminate only completely empty rows or columns in a range. These two macros check the row or column to make sure that they are completely empty before deleting them.

[3] Delete Even Protected Rows

```
          A                    B                    C                    D
 1 *---A macro to DELETE rows even if the spreadsheet is protected
 2     the global protection is disabled to allow the MACRO MANAGER
 3     to work properly.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 5     the range names in this column (starts with the \Z macro name)
 6 *---Place the cell pointer at the row to be deleted
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8 !
 9 !
10 \Z             {BREAKON}
11 ROWDELPR       {MENUBRANCH menu035}
12 !
13 MENU035        Cancel           Protected       Standard
14 !              Quit without deleUse this option Use this option when
15 !                               {prot035}        /WDR{?}~
16 !
17 prot035        {IF @CELLPOINTER("protect")=1}/WGPD/WDR{?}~/WGPE
18 !              {IF @CELLPOINTER("protect")=0}/WDR{?}~
```

This macro allows you to delete rows even if the worksheet is protected. The macro features a custom menu of three menu options. Because the custom menu cannot be displayed on the screen or on paper without overlapping, we show every menu option separately. If you intend to key this macro into Lotus 1-2-3, you have to key the code as it appears here, NOT the code as it appears in the main listing.

The first menu option is [**C**ancel]:

```
Cancel
Quit without deleting the row
```

It allows you to quit the macro without deleting rows. When you choose this menu option the macro reaches an empty cell and quits.

The second menu option is [**P**rotected]:

```
Protected
Use this option when the spreadsheet is protected
{prot035}
```

It allows you to delete rows even if the worksheet is protected. The main routine of this menu item is the `{prot035}` routine

```
          A                    B                    C                    D
17 prot035        {IF @CELLPOINTER("protect")=1}/WGPD/WDR{?}~/WGPE
18 !              {IF @CELLPOINTER("protect")=0}/WDR{?}~
```

The macro issues the `{IF @CELLPOINTER("protect")=1}` macro command to check if the current cell is protected. If so, the macro issues `/WGPD` to disable the protection, and then issues `/WDR` to start the row delete process. Then the macro issues `{?}` to pause while you paint the rows and press ENTER. When you press ENTER, the macro issues the tilde "`~`" macro command which is the same as the ENTER key and deletes the rows. Then the macro issues `/WGPE` which enable the protection. If the cell is not protected, the macro issues `/WDR` to start the row delete process. Then the macro issues `{?}` to pause while you paint the rows and press ENTER. When you press ENTER, the macro issues the tilde "`~`", and deletes the rows.

The third menu option is [**S**tandard]:

```
Standard
Use this option when the spreadsheet is not protected
/WDR{?}~
```

When you choose this menu option, the macro issues `/WDR` to start the row delete process. Then the macro issues `{?}` to pause while you paint the rows and press ENTER. Then the macro issues the tilde "`~`" and deletes the rows.

[3] Delete a Selected Number of Columns

```
     A                B                C                D                E
 1 *---A macro to DELETE a specified number of COLUMNS
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 COLDELET       {PANELOFF}{WINDOWSOFF}{GETLABEL "Number of columns to
                  delete including the current column ? ",columns003}
                  {RECALC move1003}
12 !              /WDC
13 !              {BS}.
14 move1003       {RIGHT 100}
15 !              {LEFT}~{WINDOWSON}{PANELON}
16 !
17 columns003     100
```

This macro allows you to delete a selected number of columns including the current column without the need to paint the columns to delete. For example, you can delete 100 columns just by typing the 100 number and pressing the ENTER key. Notice that the code in the B14 cell named [move1003] is the result of the following dynamic string formula:

```
114 move1003      +"{RIGHT "&B17&"}"
```

To key this macro into Lotus 1-2-3, you need to key this formula into the B14 cell.

The macro starts with the `{PANELOFF}{WINDOWSOFF}` macro commands which freeze the screen and panel display activities. Then the macro issues `{GETLABEL "Number of columns to delete including the current column ? ",columns003}` which displays:

```
"Number of columns to delete including the current column ? "
```

When you type the number and press ENTER, Lotus stores the number in cell [columns003] as the "100" label. Next, the macro issues `{RECALC move1003}` to update the dynamic string formula in cell [move1003]. For example, if you type 100, the formula in cell [move1003] returns `{RIGHT 100}`.

Lotus recognizes the result of the formula as a command as long as the syntax of the result is legal; therefore the macro will execute:

```
/WDC{BS}.{RIGHT 100}~
```

The macro issues `{BS}` to free the cell pointer in case the spreadsheet remembers a previous column deletion, then the macro issues [`.`] to anchor the cell pointer in the desired location and moves the cell pointer 100 columns to paint the range of columns to delete. The tilde "`~`" command is the same as the ENTER key pressed from the keyboard.

[3] Delete a Selected Number of Rows

```
     A                B                C                D                E
 1 *---A macro to DELETE a specified number of ROWS
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Hold the [MACRO] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 ROWSDLET        {PANELOFF}{WINDOWSOFF}{GETLABEL "Number of rows to
                   delete (including the current row) ? ",rows040}
12 !               {RECALC move040}/WDR
13 !               {BS}.
14 move040         {DOWN 1}
15 !               {UP}~{WINDOWSON}{PANELON}
16 !
17 rows040         1
```

You can delete a selected number of rows including the current row without the need to paint the rows to delete. For example, you can delete 100 rows just by typing the 100 number and pressing the ENTER key. Notice that the code in cell [move040] is the result of the following dynamic string formula:

```
114 move040        +"{DOWN "&B17&"}"
```

If you intend to key this macro into Lotus 1-2-3, you need to key this formula into the B14 cell, NOT the code as it appears in the main listing of the macro.

The macro starts with the `{PANELOFF}{WINDOWSOFF}` macro commands which freeze the screen and panel display activities. Then the macro issues `{GETLABEL "Number of rows to delete including the current row ? ",rows040}` which displays:

```
"Number of rows to delete including the current row ? "
```

When you type the number and press ENTER, Lotus stores the number in the B14 cell named [rows040] as the "100" label. Next, the macro issues `{RECALC move040}` which updates the dynamic string formula in cell [move040]. For example, if you type 100, the formula in cell [move040] returns `{DOWN 100}`.

Lotus recognizes the result of the formula as a legal Lotus command as long as the syntax of the result is a legal command statement, therefore the macro will execute:

```
/WDR{BS}.{DOWN 100}{UP}~
```

The macro issues `{BS}` to free the cell pointer in case the spreadsheet remembers a previous row deletion, then the macro issues [`.`] to anchor the cell pointer in the desired location and then moves it 100 rows to paint the range of rows to delete. The tilde "`~`" is the same as the ENTER key pressed from the keyboard.

[9] Insert Partial Columns and Rows (Ranges)

	A	B	C	D	E
1	*---A macro to INSERT a range, it will push aside all the cells and				
2	will make place in the size of the chosen range				
3	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
4	range names in this column (starts with the \Z macro name)				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE				
8	IT WILL WORK IN RELEASE 2.0 AND UP				
9	!				
10	\Z	{BREAKON}			
11	RANGEINS	{WINDOWSOFF}{PANELOFF}/RNCInsert range ?~~/RND			
		Insert range ?~/RNC{WINDOWSON}{PANELON}Insert range ?~			
		{BS}{BS}{?}~{WINDOWSOFF}{GOTO}Insert range ?~			
12	!	{LET hereabs528,@CELLPOINTER("address")}~			
13	cont528	{LET col1528,@CELLPOINTER("col")}~			
14	!	{LET col2528,@CELLPOINTER("col")+@COLS(Insert range ?)			
		-1}~{LET col4528,@COLS(Insert range ?)}~			
15	!	{LET row1528,@CELLPOINTER("row")}~			
16	!	{LET row2528,@CELLPOINTER("row")+@ROWS(Insert range ?)			
		-1}~{LET row4528,@ROWS(Insert range ?)}~			
17	!	{END}{HOME}~{LET row3528,@CELLPOINTER("row")}~			
		{LET col3528,@CELLPOINTER("col")}~{GOTO}Insert range ?~			
18	!	{MENUCALL menu1528}			
19	!	{GOTO}{hereabs528}~/RNDInsert range ?~			
20	!				
21	!				
22	hereabs528	$H$63			
23	!				
24	counterb528	11			
25	!				
26	menu1528	**C**olumnwise **R**owwise **Q**uit			
27	!	Push all the cPush all the cellQuit the macro			
28	!	{LET counterb5{LET counterb528,{BRANCH ret528}			
29	!				
30	!				
31	ret528				
32	!				
33	rel528	3.00.00			
34	col1528	3			
35	col2528	6			
36	col3528	137			
37	col4528	4			
38	row1528	2			
39	row2528	10			
40	row3528	100			
41	row4528	9			
42	chrz1528	C			
43	chrz2528	F			
44	chrz3528	G			
45	rightaa528	{RECALC chrz1528}{RECALC chrz2528}{RECALC chrz3528}			
		{RECALC rng1528}{RECALC rng1a528}/M			
46	rng1528	C2..EG10			
47	!	~			
48	rng1a528	{RIGHT 4}~			
49	!	{LET rel528,@INFO("release")}~{IF @LEFT(rel528,1)<>"@"}			
		{GOTO}{hereabs528}~{LET counterb528,counterb528+1}~			
		{IF counterb528<@SHEETS(Insert range ?)}{NS}{GOTO}			
		{hereabs528}~{BRANCH rightaa528}			
50	!	{WINDOWSON}			
51	!				
52	downaa528	{RECALC chrz1528}{RECALC chrz2528}{RECALC chrz3528}			
		{RECALC rng2528}{RECALC rng2a528}/M			
53	rng2528	C2..F100			
54	!	~			
55	rng2a528	{DOWN 9}~			
56	!	{LET rel528,@INFO("release")}~{IF @LEFT(rel528,1)<>"@"}			

```
                    {GOTO}{hereabs528}~{LET counterb528,counterb528+1}~
                    {IF counterb528<@SHEETS(Insert range ?)}{NS}{GOTO}
                    {hereabs528}~{BRANCH downaa528}
57 !                {WINDOWSON}
58 !
59 outaa528         {}
```

This macro fills a gap. In Lotus you can insert only a complete column, row or range, but you cannot insert a partial row, column or range. For example, if you are using a complicated worksheet with tables positioned one over the other, and you want to insert a column in a table, you have a problem because you may also insert a column in other tables. This macro allows you to insert partial rows and columns. To better understand this macro, let's look at the following table:

	A	B	C	D	E	F	G
1	21	41	61	81			
2	22	42	62	82			
3	23	43	63	83			
4	24	44	64	84			
6	26	46	65	85			
7	27	47	67	87			
8	28	48	68	88			
9	29	49	69	89			
10	20	40	60	80			

Let's assume that you started the macro and highlighted the B4..C7 range (three rows and two columns) to insert. You can insert in the rowwise direction and the result will be:

	A	B	C	D	E	F	G
1	21	41	61	81			
2	22	42	62	82			
3	23	43	63	83			
4	24			84			
6	26			85			
7	27			87			
8	28	44	64	88			
9	29	46	65	89			
10	30	47	67	80			
11		48	68				
12		49	69				
13		40	60				

The macro pushed the rest of the two columns beneath the B4..C7 range down to insert an empty range with the size of the B4..C7 range. If you choose to insert in the columnwise direction, the result is:

	A	B	C	D	E	F	G
1	21	41	61	81			
2	22	42	62	82			
3	23	43	63	83			
4	24			44	64	84	
6	26			46	65	85	
7	27			47	67	87	
8	28	48	68	88			
9	29	49	69	89			
10	20	40	60	80			

We can see that the macro pushed all the rows right of the B4..C7 range to the right and inserted an empty range with the size of the B4..C7 range. See the RANGDEL1.WK1 macro, which does the opposite. It allows you to delete partial rows and columns. The macro uses a custom menu with three menu options which cannot fit clearly on one page without overlapping. Here is

the code of every menu option separately. If you intend to key this macro into Lotus, you should use the code as it appears here, NOT the code as it appears in the main listing.

```
Columnwise
Push all the cells to the right to make space for the new range
{LET counterb528,0}~{rightaa528}

Rowwise
Push all the cells downward to make space for the new range
{LET counterb528,0}~{downaa528}

Quit
Quit the macro
{BRANCH ret528}
```

Notice, that the code in the B42, B43, B44, B46, B48, B53 and B55 cells of the macro is the result of the following dynamic string formulas. If you intend to key in the code, you should use the formulas, NOT the code as it appears in the main listing.

```
42 chrz1528        @IF(@MOD(B34,26)=0,"Z",@CHAR(@MOD(B34,26)+64))
43 chrz2528        @IF(@MOD(B35,26)=0,"Z",@CHAR(@MOD(B35,26)+64))
44 chrz3528        @IF(@MOD(B36,26)=0,"Z",@CHAR(@MOD(B36,26)+64))
46 rng1528         @IF(B34<27,@CHAR(B34+64),@CHAR(@INT(B34/26)+64)&B42)&
                   @STRING(B38,0)&".."&@IF(B36<27,@CHAR(B36+64),@CHAR
                   (@INT(B36/26)+64)&B44)&@STRING(B39,0)
48 rng1a528        +"{RIGHT "&@STRING(B37,0)&"}~"
53 rng2528         @IF(B34<27,@CHAR(B34+64),@CHAR(@INT(B34/26)+64)&B42)&
                   @STRING(B38,0)&".."&@IF(B35<27,@CHAR(B35+64),@CHAR
                   (@INT(B35/26)+64)&B43)&@STRING(B40,0)
55 rng2a528        +"{DOWN "&@STRING(B41,0)&"}~"
```

	A	B	C	D	E
11	RANGEINS	{WINDOWSOFF}{PANELOFF}/RNCInsert range ?~~/RND Insert range ?~/RNC{WINDOWSON}{PANELON}Insert range ?~ {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Insert range ?~			

The macro issues the {WINDOWSOFF}{PANELOFF} commands to freeze the screen and panel display activity. Then the macro uses the "safe technique" to prompt you to paint the range to delete and simultaneously assigns the [Insert range ?] range name to the same range and uses the range name as a prompt.

	A	B	C	D	E
12	!	{LET hereabs528,@CELLPOINTER("address")}~			
13	cont528	{LET col1528,@CELLPOINTER("col")}~			
14	!	{LET col2528,@CELLPOINTER("col")+@COLS(Insert range ?) -1}~{LET col4528,@COLS(Insert range ?)}~			
15	!	{LET row1528,@CELLPOINTER("row")}~			
16	!	{LET row2528,@CELLPOINTER("row")+@ROWS(Insert range ?) -1}~{LET row4528,@ROWS(Insert range ?)}~			
17	!	{END}{HOME}~{LET row3528,@CELLPOINTER("row")}~ {LET col3528,@CELLPOINTER("col")}~{GOTO}Insert range ?~			
18	!	{MENUCALL menu1528}			
19	!	{GOTO}{hereabs528}~/RNDInsert range ?~			

Now the macro stores the current address in cell [hereabs528]. Later, the macro will use it to return to this address. The macro continues with a series of {LET} macro commands which store information the macro will use later. The {LET col1528,@CELLPOINTER("col")}~ macro command stores the column number of the current column in cell [col1528] which is the first column of the [Insert range ?] range.

The {LET col2528,@CELLPOINTER("col")+@COLS(Insert range ?)-1}~ commands

store the column number of the last column of the [Insert range ?] range in cell [col2528]. The {LET col4528,@COLS(Insert range ?)}~ store the number of columns of the [Insert range ?] range in cell [col4528]. The {LET row1528,@CELLPOINTER ("row")}~ store the row number of the current row (the first row of the [Insert range ?] range) in cell [row1528]. The {LET row2528,@CELLPOINTER("row")+@ROWS(Insert range ?)-1}~ commands store the number of the last row of the [Insert range ?] range in [row2528], and {LET row4528,@ROWS(Insert range ?)}~ store the number of rows of the [Insert range ?] range in cell [row4528].

Next, the macro issues the {END}{HOME}~ commands, which move the cell pointer to the last occupied cell in the worksheet, and issues {LET row3528,@CELLPOINTER("row")}~, which store the row number of the last cell of the worksheet in the cell [row3528] and continues with {LET col3528,@CELLPOINTER("col")}~, which store the column number of the last cell of the worksheet in cell [col3528]. Now that all the properties of the range to delete and the size of the worksheet are known, the macro issues {GOTO}Insert range ?~, which moves the cell pointer back to the upper left cell of the [Insert range ?] range, and issues {MENUCALL menu1528} activating the [menu1528] custom menu.

The first menu option is [**R**owwise] :

```
Rowwise
Push all the cells downward to make space for the new range
{LET counterb528,0}~{downaa528}
```

The routine starts with {LET counterb528,0}, which sets the value in the cell [counterb528] to zero. Next, the macro issues the {downaa528} routine command which starts the [downaa528] routine.

	A	B	C	D	E
45	rightaa528	{RECALC chrz1528}{RECALC chrz2528}{RECALC chrz3528}			
		{RECALC rng1528}{RECALC rng1a528}/M			
46	rng1528	C2..EG10			
47	!	~			
48	rng1a528	{RIGHT 4}~			
49	!	{LET rel528,@INFO("release")}~{IF @LEFT(rel528,1)<>"@"}			
		{GOTO}{hereabs528}~{LET counterb528,counterb528+1}~			
		{IF counterb528<@SHEETS(Insert range ?)}{NS}{GOTO}			
		{hereabs528}~{BRANCH rightaa528}			
50	!	{WINDOWSON}			

The routine issues the five {RECALL} macro commands to update the formulas in the [chrz1528], [chrz2528], [chrz3528], [rng1528], [rng1a528] cells. Let's look at the formulas:

The first formula is in cell [chrz1528]:

42	chrz1528	@IF(@MOD(B34,26)=0,"Z",@CHAR(@MOD(B34,26)+64))

This formula calculates the letter of the first column of the [Insert range ?] range. If the column is designated by two letters, this formula calculates only the second letter.

The second formula is in cell [chrz2528]:

43	chrz2528	@IF(@MOD(B35,26)=0,"Z",@CHAR(@MOD(B35,26)+64))

This formula calculates the letter of the last column of the [Insert range ?] range. If the column is designated by two letters, this formula calculates only the second letter.

The third formula is in cell [chrz3528]:

```
44 chrz3528         @IF(@MOD(B36,26)=0,"Z",@CHAR(@MOD(B36,26)+64))
```

This formula calculates the letter of the last occupied column of the worksheet. If the column is designated by two letters, this formula calculates only the second letter.

The fourth formula is in cell [rng1528]:

```
46 rng1528          @IF(B34<27,@CHAR(B34+64),@CHAR(@INT(B34/26)+64)&B42)&
                    @STRING(B38,0)&".."&@IF(B36<27,@CHAR(B36+64),@CHAR
                    (@INT(B36/26)+64)&B44)&@STRING(B39,0)
```

This formula calculates the range to move to the right to insert an empty range with the same size as that of the [Insert range ?] range. For example, if the range named [Insert range ?] is the C2..F10 range and the last occupied column of the worksheet is the EG column, the result will be the C2..EG10 range, which contains all the data in the worksheet to the right including the [Insert range ?] range.

The fifth formula is in cell [rng1a528]:

```
48 rng1a528         +"{RIGHT "&@STRING(B37,0)&"}~"
```

This formula calculates how many columns to move the C2..EG10 range to the right.

Next, the macro issues `/MC2..EG10~{RIGHT 4}~` to move the C2..EG10 range four columns to the right.

```
     A                B                C                D                E
49 !                 {LET rel528,@INFO("release")}~{IF @LEFT(rel528,1)<>"@"}
                     {GOTO}{hereabs528}~{LET counterb528,counterb528+1}~
                     {IF counterb528<@SHEETS(Insert range ?)}{NS}{GOTO}
                     {hereabs528}~{BRANCH rightaa528}
50 !                 {WINDOWSON}
```

Because a 3-D Lotus worksheet may have more than one sheet, the macro issues the `{LET rel528,@INFO("release")}~` code, which stores the result of the `@INFO("release")` function in the cell [rel528]. Then the macro issues `{IF @LEFT(rel528,1)<>"@"}` to check if you are using a 2-D or a 3-D Lotus release. If the first character of the content of the cell [rel527] is the "`@`" character, then you are using a 2-D Lotus release, otherwise you are using a 3-D release. If you are using a 3-D release, the macro issues `{GOTO}{hereabs528}~` indirect macro command to move to the first cell in the current sheet range, and then issues `{LET counterb528,counterb528+1}~` to increase the counter in cell [counterb528] by one.

Next, the macro issues `{IF counterb528<@SHEETS(Insert range ?)}` to compare the counter value in cell [counterb528] to the number of sheets in the [Insert range ?] range. If the counter value is less than the number of sheets in [Insert range ?], the macro has to process more sheets before it can quit. Therefore the macro issues `{NS}{GOTO}{hereabs528}~` to move the cell pointer to the upper left cell of the [Insert range ?] range in the new sheet, and issues `{BRANCH rightaa528}` to loop back to the beginning of the [rightaa528] routine. The routine

is finished only when the value in [counterb528] is greater than the number of sheets in the [Insert range ?] range.

The second menu option is [**C**olumnwise]:

```
Columnwise
Push all the cells to the right to make space for the new range
{LET counterb528,0}~{rightaa528}
```

The routine starts with `{LET counterb528,0}`, which sets the value in cell [counterb528] to zero. Next the macro issues `{rightaa528}` which starts the [rightaa528] routine.

```
     A              B                C              D              E
52 downaa528      {RECALC chrz1528}{RECALC chrz2528}{RECALC chrz3528}
                  {RECALC rng2528}{RECALC rng2a528}/M
53 rng2528        C2..F100
54 !              ~
55 rng2a528       {DOWN 9}~
56 !              {LET rel528,@INFO("release")}~{IF @LEFT(rel528,1)<>"@"}
                  {GOTO}{hereabs528}~{LET counterb528,counterb528+1}~
                  {IF counterb528<@SHEETS(Insert range ?)}{NS}{GOTO}
                  {hereabs528}~{BRANCH downaa528}
```

The routine issues the five `{RECALC}` macro commands to update the formulas in the [chrz1528], [chrz2528], [chrz3528], [rng2528], [rng2a528] cells. The first three are the same formulas as we have seen in the previous menu option. Therefore, let us look at the last two formulas:

The first formula is in cell [rng2528]:

```
53 rng2528        @IF(B34<27,@CHAR(B34+64),@CHAR(@INT(B34/26)+64)&B42)&
                  @STRING(B38,0)&".."&@IF(B35<27,@CHAR(B35+64),@CHAR
                  (@INT(B35/26)+64)&B43)&@STRING(B40,0)
```

This formula calculates the range to move down to insert an empty range with the same size as that of the [Insert range ?] range. For example, if the range named [Insert range ?] is the C2..F10 range and the last occupied row of the worksheet is the 100 row, the result of this formula will be the C2..EG100 which contains all the data in the worksheet below including the [Insert range ?] range.

The fifth formula is in cell [rng2a528]:

```
55 rng2a528       +"{DOWN "&@STRING(B41,0)&"}~"
```

This formula calculates how many rows down to move the C2..F100 range. Next the macro issues `/MC2..F100~{DOWN 9}~` to move the C2..EG10 range four columns to the right.

```
     A              B                C              D              E
56 !              {LET rel528,@INFO("release")}~{IF @LEFT(rel528,1)<>"@"}
                  {GOTO}{hereabs528}~{LET counterb528,counterb528+1}~
                  {IF counterb528<@SHEETS(Insert range ?)}{NS}{GOTO}
                  {hereabs528}~{BRANCH downaa528}
57 !              {WINDOWSON}
```

The rest of the code is identical to the code of the [rightaa528] except that the macro loops back to the [downaa528] routine instead. The third menu option is [**Q**uit]:

```
Quit
Quit the macro
{BRANCH ret528}
```

When you choose this option, the macro issues `{BRANCH ret527}`, which routes macro control to the empty routine [ret527] and quits. Now the macro returns control to the line after the `{MENUCALL menu1528}` in the main code, which issues `{GOTO}{hereabs528}~/RND Insert range ?~`, to move the cell pointer back to the origin cell and then deletes the temporary [Insert range ?] range name to leave a clean worksheet.

Here is the list of all the cell contents to help you to key this macro into 1-2-3.

```
A1: U [W14] '*---A macro to INSERT a range, it will push aside all the
           cells and
A2: U [W14] '    will make place in the size of the chosen range
A3: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A4: [W14] '    range names in this column (starts with the \Z macro name)
A5: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A6: [W14] '!
A7: U [W14] '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3
           RELEASE
A8: U [W14] '                  IT WILL WORK IN RELEASE 2.0 AND UP
A9: [W14] '!
A10: U [W14] '\Z
B10: '{BREAKON}
A11: U [W14] 'RANGEINS
B11: '{WINDOWSOFF}{PANELOFF}/RNCInsert range ?~~/RNDInsert range ?~
     /RNC{WINDOWSON}{PANELON}Insert range ?~{BS}{BS}{?}~{WINDOWSOFF}
     {GOTO}Insert range ?~
A12: [W14] '!
B12: '{LET hereabs528,@CELLPOINTER("address")}~
A13: [W14] 'cont528
B13: '{LET col1528,@CELLPOINTER("col")}~
A14: [W14] '!
B14: '{LET col2528,@CELLPOINTER("col")+@COLS(Insert range ?)-1}~
     {LET col4528,@COLS(Insert range ?)}~
A15: [W14] '!
B15: '{LET row1528,@CELLPOINTER("row")}~
A16: [W14] '!
B16: '{LET row2528,@CELLPOINTER("row")+@ROWS(Insert range ?)-1}~
     {LET row4528,@ROWS(Insert range ?)}~
A17: [W14] '!
B17: '{END}{HOME}~{LET row3528,@CELLPOINTER("row")}~{LET col3528,@CELLPOINTER("col")}~
     {GOTO}Insert range ?~
A18: [W14] '!
B18: '{menucall menu1528}
A19: [W14] '!
B19: '{GOTO}{hereabs528}~/RNDInsert range ?~
A20: [W14] '!
A21: [W14] '!
A22: [W14] 'hereabs528
B22: '$H$63
A23: [W14] '!
A24: [W14] 'counterb528
B24: 11
A25: [W14] '!
A26: [W14] 'menu1528
B26: 'Columnwise
C26: 'Rowwise
D26: 'Quit
A27: [W14] '!
B27: 'Push all the cells to the right to make space for the new range
C27: 'Push all the cells downward to make space for the new range
D27: 'Quit the macro
A28: [W14] '!
B28: '{LET counterb528,0}~{rightaa528}
```

```
C28: '{LET counterb528,0}~{downaa528}
D28: '{BRANCH ret528}
A29: [W14] '!
A30: [W14] '!
A31: [W14] 'ret528
A32: [W14] '!
A33: [W14] 'rel528
B33: '3.00.00
A34: [W14] 'col1528
B34: 8
A35: [W14] 'col2528
B35: 12
A36: [W14] 'col3528
B36: 13
A37: [W14] 'col4528
B37: 5
A38: [W14] 'row1528
B38: 63
A39: [W14] 'row2528
B39: 70
A40: [W14] 'row3528
B40: 80
A41: [W14] 'row4528
B41: 8
A42: [W14] 'chrz1528
B42: U @IF(@MOD(B34,26)=0,"Z",@CHAR(@MOD(B34,26)+64))
A43: [W14] 'chrz2528
B43: U @IF(@MOD(B35,26)=0,"Z",@CHAR(@MOD(B35,26)+64))
A44: [W14] 'chrz3528
B44: U @IF(@MOD(B36,26)=0,"Z",@CHAR(@MOD(B36,26)+64))
A45: [W14] 'rightaa528
B45: '{RECALC chrz1528}{RECALC chrz2528}{RECALC chrz3528}{RECALC rng1528}
     {RECALC rng1a528}/M
A46: [W14] 'rng1528
B46: U @IF(B34<27,@CHAR(B34+64),@CHAR(@INT(B34/26)+64)&B42)&@STRING(B38,0)
       &".."&@IF(B36<27,@CHAR(B36+64),@CHAR(@INT(B36/26)+64)&B44)&@STRING
       (B39,0)
A47: [W14] '!
B47: '~
A48: [W14] 'rng1a528
B48: U +"{RIGHT "&@STRING(B37,0)&"}~"
A49: [W14] '!
B49: '{LET rel528,@INFO("release")}~{IF @LEFT(rel528,1)<>"@"}{GOTO}
     {hereabs528}~{LET counterb528,counterb528+1}~{IF counterb528<@SHEETS
     (Insert range ?)}{NS}{GOTO}{hereabs528}~{BRANCH rightaa528}
A50: [W14] '!
B50: '{WINDOWSON}
A51: [W14] '!
A52: [W14] 'downaa528
B52: '{RECALC chrz1528}{RECALC chrz2528}{RECALC chrz3528}{RECALC rng2528}
     {RECALC rng2a528}/M
A53: [W14] 'rng2528
B53: U @IF(B34<27,@CHAR(B34+64),@CHAR(@INT(B34/26)+64)&B42)&@STRING(B38,0)
       &".."&@IF(B35<27,@CHAR(B35+64),@CHAR(@INT(B35/26)+64)&B43)&@STRING
       (B40,0)
A54: [W14] '!
B54: '~
A55: [W14] 'rng2a528
B55: U +"{DOWN "&@STRING(B41,0)&"}~"
A56: [W14] '!
B56: '{LET rel528,@INFO("release")}~{IF @LEFT(rel528,1)<>"@"}{GOTO}
     {hereabs528}~{LET counterb528,counterb528+1}~{IF counterb528<@SHEETS
     (Insert range ?)}{NS}{GOTO}{hereabs528}~{BRANCH downaa528}
A57: [W14] '!
B57: '{WINDOWSON}
A58: [W14] '!
A59: [W14] 'outaa528
B59: '{}
```

[3] Insert Selected Number of Rows

```
     A              B                C              D              E
 1 *---A macro to insert a specified number of rows
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 ROWINSRT       {PANELOFF}{WINDOWSOFF}{GETLABEL "Number of rows to
                  insert ? ",rows036}
12 !              {RECALC move036}/WIR
13 !              {BS}.
14 move036        {DOWN 5}
15 !              {UP}~{WINDOWSON}{PANELON}
16 !
17 rows036        5
```

This macro allows you to insert a selected number of rows without having to paint the rows to insert. For example, you can insert 100 rows just by typing the 100 number and pressing the ENTER key. Notice that the code in the B14 cell named [move036] is the result of the following dynamic string formula:

```
14 move040       +"{DOWN "&B17&"}"
```

If you intend to key this macro into Lotus 1-2-3, you need to key this formula into the B14 cell, NOT the code as it appears in the main listing of the macro.

The macro starts with `{PANELOFF}{WINDOWSOFF}` which freeze the screen and panel display activities. Then it issues `{GETLABEL "Number of rows to insert ? ",rows036}` which displays:

```
"Number of rows to insert ? "
```

When you type the number and press ENTER Lotus stores the number in cell [rows036] as the "100" label. Next the macro issues `{RECALC move036}` which updates the dynamic string formula in cell [move036]. For example if you type the 100 number, the formula in cell [move036] returns `{DOWN 100}`.

Lotus recognizes the result of the formula as a legal Lotus command as long as the syntax of the result is a legal command statement, therefore the macro will execute:

```
/WIR{BS}.{DOWN 100}{UP}~
```

The macro issues `{BS}` to free the cell pointer in case the spreadsheet remembers a previous row insertion, then the macro issues [`.`] to anchor the cell pointer in the desired location and moves the cell pointer 100 rows to paint the range of rows to insert. The tilde "`~`" is the same as the ENTER key pressed from the keyboard.

[3] Insert Selected Number of Columns

	A	B	C	D	E
1	*---A macro to INSERT a specified number of COLUMNS				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define				
3	therange names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	COLINSRT	{PANELOFF}{WINDOWSOFF}{GETLABEL "Number of columns to insert ?" ,columns003}{RECALC move1003a}			
12	!	/WIC			
13	!	{BS}.			
14	move1003a	{RIGHT 100}			
15	!	{LEFT}~{WINDOWSON}{PANELON}			
16	!				
17	columns003a	100			

This macro is almost the same as the COLDELET.WK1 macro except that it uses the /WIC macro code instead of the /WDC macro code.

[3] Insert Row's Duplicate

```
     A                B                C                D                E
 1 *---A macro to INSERT a duplicate of the row above
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer under the row to be duplicate
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 ROWPUSH         {WINDOWSOFF}{PANELOFF}
12 !               /WIR~{END}{LEFT}
13 !               {UP}/C.{DOWN}{END}{RIGHT}{UP}~{DOWN}~
```

This macro inserts a duplicate of the row above. Using `{WINDOWSOFF}{PANELOFF}` it freezes the screen and panel activities. Next, the macro issues `/WIR~` to inserts a row, and `{END}{LEFT}` to move the cell pointer to the first cell in the row. Now the macro issues `{UP}` to move the cell pointer to the row above. Because there is no way to know which cells are occupied, the macro uses the adjacent empty row's length to paint the whole row to be copied. The macro uses the `/C.{DOWN}{END}{RIGHT}{UP}~{DOWN}~` macro commands to copy the whole row to the new empty row.

[3] Insert a Row with Above Row's Format

```
     A              B                C                D              E
 1 *---A macro to INSERT a row with the above row's format
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer under the row to be copied (format only)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z               {BREAKON}
11 ROWINSFT         {WINDOWSOFF}{PANELOFF}
12 !                /WIR~{END}{LEFT}/RNCtemp497~~/RNDtemp497~/RNCtemp497~
                    {END}{RIGHT}~
13 !                {UP}/C.{DOWN}{END}{RIGHT}{UP}~{DOWN}~/REtemp497~
                    /RNDtemp497~{DOWN}~
```

This macro inserts a row with the same format as the row above. This macro is based on the idea that when we copy a cell to another cell the contents and the format are copied, therefore this macro inserts an empty row, copies the row above to the empty row and erases the data in the new row. What's left is the format. The macro issues `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel activities. Next the macro issues `/WIR~` to inserts the row, and issues `{END}{LEFT}` to move the cell pointer to the first cell in the row. Now the macro assigns the [temp497] range name to the whole row using the "safe technique" to create a range name from within the macro.

The macro issues `/RNCtemp497~~/RNDtemp497~/RNCtemp497~{END}{RIGHT}~` to assign the [temp497] range name to the whole row. Now the macro issues `{UP}` to move the cell pointer to the row above. Because there is no way to know which cells are occupied, the macro uses the adjacent empty row's length to paint the whole row to be copied. The macro uses the `/C.{DOWN}{END}{RIGHT}{UP}~{DOWN}~` macro commands to copy the whole row to the new empty row. Next the macro issues `/REtemp497~` to erase the new row's content and then `/RNDtemp497~` to delete the [temp497] range name. It was important to assign the range name and to copy the row while the new row was still empty, because the macro could use its full length and the `{END}{RIGHT}` commands to paint the whole row. When the row is not completely empty, `{END}{RIGHT}` expand the highlight only to the first blank cell. Secondly, assigning a range name made it easy to erase the row after the copy operation, leaving only the copied format.

[3] Insert a Column with Same Format

	A	B	C	D	E
1	*---A macro to INSERT a column to the left with the current column				
2	format				
3	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define				
4	the range names in this column (starts with the \Z macro name)				
5	*---Place the cell pointer on the column to be copied (format only)				
6	*---Hold the [ALT] key and press [Z] to activate the macro				
7	!				
8	!				
9	!				
11	\Z	{BREAKON}			
11	COLINSFT	{WINDOWSOFF}{PANELOFF}			
12	!	/WIC~{END}{UP}/RNCtemp250~~/RNDtemp250~/RNCtemp250~			
		{END}{DOWN}~			
13	!	{RIGHT}/C.{LEFT}{END}{DOWN}{RIGHT}~{LEFT}~/REtemp250~			
		/RNDtemp250~			
11		{LEFT}~			

Sometime when users want to insert a column to the left of the current column, they want the cells in the new column to have the same format as the adjacent cells in the current column. This macro creates such a column. The macro starts with {WINDOWSOFF}{PANELOFF} which freeze the screen and panel display activities. Then the macro issues /WIC~ and inserts the new column. Next it issues {END}{UP} which move the cell pointer to the upper cell of the new column. Then the macro uses the "safe technique" to prompt you to paint the new column, and simultaneously assigns the [temp250] range name to the new column.

Now the macro issues {RIGHT} to place the cell pointer on the upper cell of the right column and then issues /C.{LEFT}{END}{DOWN}{RIGHT}~{LEFT}~ which copy the right column to the new inserted column named [temp250]. Because the macro cannot know if the data in the current (origin) column is contiguous, it uses the length of the target column, which is the full length of the new column since this column is empty. Last, the macro issues /RE temp250~, which erases the content of the new column leaving only the format. Then it issues /RNDtemp250~ which deletes the temporary [temp250] range name leaving a clean worksheet.

[7] Invert a Range of Data

```
     A              B                C              D              E
 1 *---A macro to invert a range of numbers or text
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z             {BREAKON}
11 INVRTRNG       {LET rel500,@INFO("release")}~
12 !              {WINDOWSOFF}{PANELOFF}/RNCInvert range ?~~/RND
                  Invert range ?~/RNC{PANELON}Invert range ?~{WINDOWSON}
                  {BS}{?}~{LET coll500,@STRING(@ROWS(Invert range ?)-1
                  ,0)}~{LET colla500,@STRING(@COLS(Invert range ?),0)}~
13 !              {IF @LEFT(rel500,1)<>"@"}{LET sheetsa500,@STRING
                  (@SHEETS(Invert range ?)-1,0)}~
14 !              {RECALC sheets500}{RECALC cont500}{RECALC conta500}
                  {RECALC contb500}{RECALC contc500}
15 contc500       {WINDOWSOFF}{PANELOFF}{GOTO}invert range ?~/WIC{NS 5}~
                  /DF{BS}.
16 cont500        {DOWN 3}{NS 5}~1~1~8192~
17 conta500       /DSRD{BS}.{END}{DOWN}{RIGHT 2}{NS 5}~P~~G
18 contb500       /WDC{NS 5}~/RNDInvert range ?~
19 !
20 coll500        3
21 !
22 colla500       2
23 !
24 rel500         3.00.00
25 !
26 sheets500      {NS 0}
27 !
28 sheetsa500     5
```

This macro allows you to invert a range of data so that the first row of data becomes the last row of data and the second becomes the row before the last row, etc. This macro uses five dynamic string formulas to create the correct code for the macro (the formulas are in the B15, B16, B17, B18 and the B26 cells), therefore if you intend to key this macro into Lotus 1-2-3, you have to key the formulas as they appear here, NOT the code as it appears in the main listing.

```
15 contc500       +"{WINDOWSOFF}{PANELOFF}{GOTO}invert range ?~/WIC"
                  &B26&"~/DF{BS}."
16 cont500        +"{DOWN "&B20&"}"&B26&"~1~1~8192~"
17 conta500       +"/DSRD{BS}.{END}{DOWN}{RIGHT "&B22&"}"&B26&"~P~~G"
18 contb500       +"/WDC"&B26&"~/RNDInvert range ?~"
26 sheets500      @IF(@LEFT(B24,1)<>"@",+"{NS "&B28&"}","{}")
```

The macro starts with the `{LET rel500,@INFO("release")}~` macro commands which store the result of the `@INFO("release")` function in cell [rel500]. Later the macro uses this result to determine if you are using a 2-D or a 3-D Lotus release.

```
     A              B                C              D              E
12 !              {WINDOWSOFF}{PANELOFF}/RNCInvert range ?~~/RND
                  Invert range ?~/RNC{PANELON}Invert range ?~{WINDOWSON}
                  {BS}{?}~{LET coll500,@STRING(@ROWS(Invert range ?)-1
                  ,0)}~{LET colla500,@STRING(@COLS(Invert range ?),0)}~
```

Now the macro issues `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities. Then it uses the "safe technique" to temporarily assign the [Invert range ?] range

name to the range you want to invert and simultaneously uses the range name as a prompt. Next the macro issues `{LET coll500,@STRING(@ROWS(Invert range ?)-1,0)}`, which stores the number of rows in the [Invert range ?] range in the B20 cell named [coll500]. The macro uses the `@ROWS()` Lotus function which returns the number of rows in the range in the parenthesis, and then turns the numeric value into an alphanumeric string using the Lotus `@STRING()` function. Next, the macro issues `{LET colla500 ,@STRING(@COLS(Invert range ?),0)}`, which stores the number of columns of the [Invert range ?] range in B22, [colla500].

Later, the macro uses the string values stored in the B20 and the B22 cells to know how many cells to move down and how many columns to move to the right to process the [Invert range ?] range.

	A	B	C	D	E
13	!	{IF @LEFT(rel500,1)<>"@"}{LET sheetsa500,@STRING (@SHEETS(Invert range ?)-1,0)}~			
14	!	{RECALC sheets500}{RECALC cont500}{RECALC conta500} {RECALC contb500}{RECALC contc500}			
15	contc500	{WINDOWSOFF}{PANELOFF}{GOTO}invert range ?~/WIC{NS 5}~ /DF{BS}.			
16	cont500	{DOWN 3}{NS 5}~1~1~8192~			
17	conta500	/DSRD{BS}.{END}{DOWN}{RIGHT 2}{NS 5}~P~~G			
18	contb500	/WDC{NS 5}~/RNDInvert range ?~			

Now the macro issues `{IF @LEFT(rel500,1)<>"@"}` to check if the first character of the content of cell [rel500] is NOT the "@" character. If so, you are using a 3-D Lotus release, and the range to invert may contain more than one sheet, therefore the macro issues `{LET sheetsa500,@STRING(@SHEETS(Invert range ?)-1,0)}` which stores the number of sheets of [Invert range ?] in cell [sheetsa500] as an alphanumeric string. Now that the macro "knows" the number of rows, the number of columns and the number of sheets, it uses `{RECALC sheets500}{RECALC cont500}{RECALC conta500}{RECALC contb500} {RECALC contc500}` to update the five formulas in the B15, B16, B17, B18 and the B26 cells.

Next the macro issues `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities, and then issues `{GOTO}invert range ?~` which moves the cell pointer to the upper left cell of [Invert range ?]. Now the macro issues `/WIC{NS 5}~` which inserts a column left to the [Invert range ?] range five sheets deep. Then the macro issues `/DF{BS}.{DOWN 3} {NS 5}~ 1~1~8192~` to fill the new column with ascending numbers. Next the macro issues the `/DSRD {BS}.{END}{DOWN}{RIGHT 2}{NS 5}~P~~G` macro code, which sorts the new column and the [Invert range ?] range in descending order together as the data range, and the new column serves as the primary key. Last, the macro issues `/WDC{NS 5}~/RNDInvert range ?~` which delete the inserted column and then delete the temporary [Invert range ?] range name leaving the inverted range and a clean worksheet.

First, we want to remind you that all the code we have just covered is the result of string formulas, therefore let us take a close look at these formulas:

```
15 contc500       +"{WINDOWSOFF}{PANELOFF}{GOTO}Invert range ?~/WIC"
                  &B26&"~/DF{BS}."
16 cont500        +"{DOWN "&B20&"}"&B26&"~1~1~8192~"
17 conta500       +"/DSRD{BS}.{END}{DOWN}{RIGHT "&B22&"}"&B26&"~P~~G"
18 contb500       +"/WDC"&B26&"~/RNDInvert range ?~"
26 sheets500      @IF(@LEFT(B24,1)<>"@",+"{NS "&B28&"}","{}")
```

The formula in the B26 cell named [sheets500] uses the content of B28, [sheetsa500], to create the code for the 3-D worksheet. If you are using a 3-D release and the range has five sheets, the formula returns `{NS 5}`, otherwise the formula returns the do nothing `{}` routine command. The rest of the formulas use the result of the formula in B26, [sheets500], the contents of B20, [coll500], which holds the number of rows of the range minus one, and the content of B22, [colla500], which holds the number of columns of the range to invert. The resulting code of the formulas (where the range has two columns, four rows and six sheets) is:

```
15 contc500        {WINDOWSOFF}{PANELOFF}{GOTO}Invert range ?~/WIC{NS 5}~
                   /DF{BS}.
16 cont500         {DOWN 3}{NS 5}~1~1~8192~
17 conta500        /DSRD{BS}.{END}{DOWN}{RIGHT 2}{NS 5}~P~~G
18 contb500        /WDC{NS 5}~/RNDInvert range ?~
26 sheets500       {NS 5}
```

To clarify the technique let us look at the two columns range A11..B14

	A	B	C	D	E
11	Herbert	Samuel			
12	John	Brioche			
13	Jack	Lemon			
14	Jimmy	Carter			

Let us insert a column to the left of the range and fill the column with numbers in ascending order. The result is:

	A	B	C	D	E
11	1	Herbert	Samuel		
12	2	John	Brice		
13	3	Jack	Lemon		
14	4	Jimmy	Carter		

Now if we will sort the A11..C14 range in descending order where the "A" column is the primary key, the result looks like:

	A	B	C	D	E
11	4	Jimmy	Carter		
12	3	Jack	Lemon		
13	2	John	Brice		
14	1	Herbert	Samuel		

When we delete the "A" column the result is the inverted range:

	A	B	C	D	E
11	Jimmy	Carter			
12	Jack	Lemon			
13	John	Brice			
14	Herbert	Samuel			

[7] Invert a Column of Data

```
     A              B                C              D              E
 1 *---A macro to invert a column of numbers or text
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z             {BREAKON}
11 INVRTCOL       {LET rel020,@INFO("release")}~
12 !              {WINDOWSOFF}{PANELOFF}/RNCInvert column ?~~/RND
                  Invert column ?~/RNC{PANELON}Invert column ?~
                  {WINDOWSON}{BS}{?}~{LET coll020,@STRING(@ROWS
                  (Invert column ?)-1,0)}~
13 !              {IF @LEFT(rel020,1)<>"@"}{LET sheetsa020,@STRING
                  (@SHEETS(Invert column ?)-1,0)}~
14 !              {RECALC sheets020}{RECALC cont020}{RECALC conta020}
                  {RECALC contb020}{RECALC contc020}
15 contc020       {WINDOWSOFF}{PANELOFF}{GOTO}Invert column ?~/WIC{}~/DF
                  {BS}.
16 cont020        {DOWN 410}{}~1~1~8192~
17 conta020       /DSRD{BS}.{END}{DOWN}{RIGHT}{}~P~~G
18 contb020       /WDC{}~/RNDInvert column ?~
19 !
20 coll020        410
21 !
22 rel020         @INFO("release")
23 !
24 sheets020      {}
25 !
26 sheetsa020     3



15 contc020      +"{WINDOWSOFF}{PANELOFF}{GOTO}Invert column ?~/WIC"
                  &C24&"~/DF{BS}."
16 cont020        +"{DOWN "&C20&"}"&C24&"~1~1~8192~"
17 conta020       +"/DSRD{BS}.{END}{DOWN}{RIGHT}"&C24&"~P~~G"
18 contb020       +"/WDC"&C24&"~/RNDInvert column ?~"
24 sheets020      @IF(@LEFT(C22,1)<>"@",+"{NS "&C26&"}","{}")
```

See **Invert a Range of Data** in the previous section.

[6] Squeeze a Column of Data

```
     A              B                C                D              E
 1 *---A macro to SQUEEZE a list of column entries (take out blank cells)
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer at the upper most cell of the list
 5 *---Hold the [MACRO] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 SQZCOLMN       {WINDOWSOFF}{PANELOFF}/RNCPaint column !~~
                  /RNDPaint column !~
12 !              /RNC{PANELON}Paint column !~{WINDOWSON}{?}{LET end042,
                  @CELLPOINTER("address")}~{GOTO}Paint column !~
                  {LET start042,@CELLPOINTER{"address")}~{IF @CELLPOINTER
                  ("type")="b"}{WINDOWSOFF}{END}{DOWN}/C~{start042}~/RE~~
13 loop042        {WINDOWSOFF}{GOTO}{end042}~{IF @CELLPOINTER("type")="b"}
                  {END}{UP}
14 loop2042       {UP}{IF @CELLPOINTER("type")="b"}{DOWN}/M~{END}{UP}
                  {DOWN}~{GOTO}{end042}~{BRANCH loop1042}
15 !              {IF @CELLPOINTER("type")<>"b"}{DOWN}{BRANCH loop3042}
16 !
17 loop1042       {END}{UP}
18 loop3042       {END}{UP}
19 !              {IF @CELLPOINTER("row")>@CELL("row",Paint column !)}
                  /M.{END}{DOWN}~{END}{UP}{DOWN}~{GOTO}{end042}~
                  {BRANCH loop1042}
20 !              /RNDPaint column !~
21 !
22 end042         $U$401
23 !
24 ret042
25 !
26 start042       $U$1
```

This macro squeezes out all the empty cells in a column of data, resulting in a contiguous column of data. The macro begins with `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities. Next the macro uses the "safe technique" to temporarily assign the [Paint column !] range name to the column range and simultaneously uses the range name as a prompt. When the name is defined, the macro issues `{LET end042 ,@CELLPOINTER("address")}~` which record the address of the last cell in the column of data in cell [end042]. Then the macro issues `{GOTO}Paint column!~` which moves the cell pointer to the upper cell of the column named [Paint column !].

Now the macro issues `{LET start042,@CELLPOINTER("address")}~` to record the address of the first cell of the selected column into cell [start042]. The macro now "knows" the start and the end of the column addresses. The macro continues with `{IF @CELLPOINTER("type")="b"}` which checks if the first cell in the selected column is empty. If so, the macro issues `{WINDOWSOFF}` to freeze the screen display activity and then issues `{END}{DOWN}` which move the cell pointer to the first occupied cell in the column. Now the macro issues `/C~` which starts the copy process and then issues the `{start042}` routine command which injects the address in cell [start042] into the panel and then issues the tilde "~" to finish the copy process, which copies the content of the first occupied cell into the first empty cell of the selected column. Next the macro issues the `/RE~~` macro command which erases the current cell.

Note: The macro uses the {LET} macro command to find and record the extreme upper and lower addresses while you use the direction keys to paint the range to be squeezed. This way the macro can record and "know" the cell addresses of the four corners of a range that you painted, and build the full address of the range while the macro runs. The macro uses {?} to pause until you paint the range (column in our example) and simultaneously when the cell pointer is on one of the corners, the {LET} command records the position for later use. Notice that the ENTER key, represented by the tilde "~" macro command, is pressed only after the {LET} command and not after the {?} command.

	A	B	C	D	E
13	loop042	{WINDOWSOFF}{GOTO}{end042}~{IF @CELLPOINTER("type")="b"}{END}{UP}			
14	loop2042	{UP}{IF @CELLPOINTER("type")="b"}{DOWN}/M~{END}{UP}{DOWN}~{GOTO}{end042}~{BRANCH loop1042}			
15	!	{IF @CELLPOINTER("type")<>"b"}{DOWN}{BRANCH loop3042}			

Now that the first cell of the selected column is occupied, the macro issues {WINDOWSOFF} to freeze the screen display activity, and then issues the {GOTO}{end042}~ indirect command which moves the cell pointer to the last cell of the [Paint column !] range. Now that the cell pointer is on the last cell of the column, the macro issues {IF @CELLPOINTER("type")="b"} to check if the cell is empty. If so, the macro issues {END}{UP}{UP}, and again {IF @CELLPOINTER("type")="b"} to check if the cell is empty. If so, the last occupied cell is separated from the rest of the cells, therefore the macro issues {DOWN}/M~{END} {UP}{DOWN}~ which move the content of the current cell to the cell below the above occupied cell, and close the gap between them. Next the macro issues the {GOTO}{end042}~ indirect macro command which moves the cell pointer to cell [end042], the last cell in the selected column, [Paint column !]. Let's see an example:

	A	B	C	D	E
1					
2					
3					
4	200				
5	300				
6	400				
7					
8					
9	500				
10					
11					
12					
13					

Let us assume that you selected to squeeze the A2..A12 range which the macro assigns the [Paint column !] range name. The macro records the A2 address in cell [start042] and then records the A12 address in cell [end042]. Then the macro uses the {GOTO}{start042}~ indirect macro command to move to the A2 cell, and checks if the A2 cell is empty. If it is empty, the macro issues the {END}{DOWN} macro command which move the cell pointer to the A4 cell, and then copies this cell to the A2 cell to establish the A2 cell as the first occupied cell in the range, and then erases the content of the A4 cell. The worksheet now displays:

	A	B	C	D	E
1					
2	200				
3					
4					
5	300				

```
 6          400
 7
 8
 9          500
10
11
```

Next the macro issues `{GOTO}{end042}~` which moves the cell pointer to the A12 cell. The macro checks if the A12 cell is empty. If the cell is empty the macro issues `{END}{UP}` which move the cell pointer to the A9 cell. Next the macro issues `{UP}` to the A8 cell. If the A8 cell is empty the macro understands that the A9 cell is separated from the rest of the cells, therefore the macro issues `{DOWN}` to the A9 cell and then issues `/M~{END}{UP}{DOWN}~` which move the content of the A9 cell to the A7 cell, and the worksheet displays:

```
    A          B          C          D          E
 1
 2          200
 3
 4
 5          300
 6          400
 7          500
 8
 9
10
11
12
13
```

The last cell in A7 is certainly not alone, and there is at least one occupied cell above. The macro issues `{GOTO}{end042}~` which moves the cell pointer to the A12 cell and then issues `{END}{UP}` to reach the last occupied cell (the A7 cell). This is why the macro moves the cell pointer first to the A12 cell, because it can always find the last occupied cell using the `{END}{UP}` macro commands from it's location in the A12 cell. Now the macro issues `{BRANCH loop1042}` which routes the macro control to the [loop1042] routine.

```
    A          B          C          D          E
17 loop1042       {END}{UP}
18 loop3042       {END}{UP}
19 !              {IF @CELLPOINTER("row")>@CELL("row",Paint column !)}
                  /M.{END}{DOWN}~{END}{UP}{DOWN}~{GOTO}{end042}~
                  {BRANCH loop1042}
20 !              /RNDPaint column !~
```

The [loop1042] routine starts with `{END}{UP}` which move the cell pointer to the last occupied cell (A7 in our example). The last occupied cell (A7) is no longer separated, and there is at least one occupied cell just above it (A6). Therefore the macro can issue the `{END}{UP}` macro commands again, which move the cell pointer to the last contiguous cell above A7 (A5 in our example). Now the macro issues `{IF @CELLPOINTER("row") >@CELL("row",Paint column !)}` to check if the current cell pointer position is below A2.

If so, it means that there are more gaps in the column. Therefore the macro issues `/M.{END}{DOWN}~{END}{UP}{DOWN}~` which move the A5..A7 group of cells to the A3 cell. Again the macro issues `{GOTO}{end042}~{BRANCH loop1042}` to close the next gap, if it exists. This process continues until the `{IF @CELLPOINTER("row")>@CELL("row" ,Paint column !)}` macro condition is false, and all the cells are contiguous. Then the macro issues `/RNDPaint column !~` to delete the temporary [Paint column !] range name to leave a clean

worksheet. Now we can go back to the [loop042] routine and see what happens if the last occupied cell (the A9 cell) was not isolated at the beginning.

	A	B	C	D	E
13	loop042	{WINDOWSOFF}{GOTO}{end042}~{IF @CELLPOINTER("type")="b"}			
		{END}{UP}			
14	loop2042	{UP}{IF @CELLPOINTER("type")="b"}{DOWN}/M~{END}{UP}			
		{DOWN}~{GOTO}{end042}~{BRANCH loop1042}			
15	!	{IF @CELLPOINTER("type")<>"b"}{DOWN}{BRANCH loop3042}			

The macro continues with `{IF @CELLPOINTER("type")<>"b"}` which checks if the cell above the last occupied cell is not empty. If so, the macro issues `{DOWN}{BRANCH loop3042}` starting the [loop3042] routine, which is part of the [loop1042] we have already seen.

[6] Squeeze a Row of Data

```
     A                B                C                D                E
 1 *---A macro to SQUEEZE a list of row entries (take out blank cells)
 2 *---Use the /Range Name Label Right [END] [RIGHT] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer at the left most cell of the list
 5 *---Hold the [MACRO] key and press [Z] to activate the macro
 6 *---The most right cell of the highlighted row must not be empty
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 SQZROW          {WINDOWSOFF}{PANELOFF}/RNCPaint row !~~/RNDPaint row !~
12 !               /RNC{PANELON}Paint row !~{WINDOWSON}{?}{LET end043,
                   @CELLPOINTER("address")}~{GOTO}Paint row !~{LET start043
                   ,@CELLPOINTER("address")}~{IF @CELLPOINTER("type")="b"}
                   {WINDOWSOFF}{END}{RIGHT}/C~{start043}~/RE~~
13 loop043         {WINDOWSOFF}{GOTO}{end043}~{IF @CELLPOINTER("type")="b"}
                   {END}{LEFT}
14 loop2043        {LEFT}{IF @CELLPOINTER("type")="b"}{RIGHT}/M~{END}
                   {LEFT}{RIGHT}~{GOTO{end043}~{BRANCH loop1043}
15 !               {IF @CELLPOINTER("type")<>"b"}{RIGHT}{BRANCH loop3043}
17 !
18 loop1043        {END}{LEFT}
19 loop3043        {END}{LEFT}
20 !               {IF @CELLPOINTER("col")>@CELL("col",Paint row !)}/M.
                   {END}{RIGHT}~{END}{LEFT}{RIGHT}~{GOTO{end043}~
                   {BRANCH loop1043}
21 !               /RNDPaint row !~
22 !
23 end043          $BR$1
24 !
25 ret043
26 !
27 start043
```

This macro is the same as the SQZCOLMN.WK1 macro except that the {RIGHT} command replaces the {UP} command and the {LEFT} command replaces the {DOWN} command.

[7] Switch Range Places

	A	B	C	D	E
1	*---A macro to switch 3-D and 2-D ranges places				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE				
7	IT WILL WORK IN LOTUS 2.0 AND UP				
8	!				
9	!				
10	\Z	{BREAKON}			
11	SWITCH	{LET rel209,@INFO("release")}~{RECALC loc209}			
		{RECALC form209}			
12	form209	{LET here1209,@CELLPOINTER("coord")}~			
13	!	{WINDOWSOFF}{PANELOFF}/RNCFirst range ?~~/RND			
		First range ?~/RNC{PANELON}First range ?~{WINDOWSON}			
		{BS}{BS}{?}~			
14	!	{WINDOWSOFF}{PANELOFF}/RNCSecond range ?~~/RND			
		Second range ?~/RNC{PANELON}Second range ?~{WINDOWSON}			
		{BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}			
15	!	{LET colss209,@STRING(@COLS(First range ?)-1,0)}~			
		{LET rowss209,@STRING(@ROWS(First range ?)-1,0)}~			
		{IF @LEFT(rel209,1)<>"@"}{LET sheets209,@STRING(@SHEETS			
		(First range ?)-1,0)}~			
16	!	{IF @COLS(First range ?)<>@COLS(Second range ?)#OR#			
		@ROWS(First range ?)<>@ROWS(Second range ?)}			
		{BRANCH messag209}			
17	!	{IF @LEFT(rel209,1)="@"}{BRANCH place3209}			
18	!	{IF @LEFT(rel209,1)<>"@"#AND#@SHEETS(First range ?)<>			
		@SHEETS(Second range ?)}{BRANCH messag209}			
19	place3209	{END}{HOME}{DOWN}{END}{LEFT}/RNCThird range ?~~/RND			
		Third range ?~/RNCThird range ?~~{RECALC place1209}			
		{{RECALC place2209{RECALC PLACE4209}{RECALC place5209}			
		{PANELON}			
20	place4209	{IF @LEFT(rel209,1)<>"@"}/CFirst range ?~~			
		/CSecond range ?~First range ?~/C.{DOWN 5}{RIGHT 2}			
		{NS 2}~Second range ?~{PANELOFF}{BRANCH place5209}			
21	place1209	/CFirst range ?~~/CSecond range ?~First range ?~/C.			
		{DOWN 5}{RIGHT 2}~Second range ?~{PANELOFF}			
22	place2209	/RE.{DOWN 5}{RIGHT 2}~{GOTO}{here1209}~{BRANCH contt209}			
23	place5209	/RE.{DOWN 5}{RIGHT 2}{NS 5}~{GOTO}{here1209}~			
24	contt209	/RNDFirst range ?~/RNDSecond range ?~/RNDThird range ?~			
25	!				
26	messag209	{BEEP}{PANELON}Ranges are not the same size,			
		overlapping is possible! [C]ont [Q]uit			
27	!	{GET key1209}{ESC}			
28	!	{IF @UPPER(key1209)="Q"}{BRANCH ret209}			
29	!	{BRANCH place3209}			
30	!				
31	here1209	$E:$E$1			
32	!				
33	key1209	q			
34	!				
35	colss209	2			
36	rowss209	5			
37	sheets209	2			
38	ret209				
39	!				
40	rel209				
41	!				
42	loc209	coord			

The code in B12, B20, B21, B22, B23 and the B42 cells is the result of the following formulas. When you type in the code, you should use this code, NOT the code in the main list.

```
12 form209          +"{LET here1209,@CELLPOINTER(""""&B42&""""")}~"
```

```
20 place4209        +"{IF @LEFT(rel209,1)<>""@""}/CFirst range ?~~
                    /CSecond range ?~First range ?~/C.{DOWN "&B36&"}
                    {RIGHT "&B35&"}{NS "&B37&"}~Second range ?~{PANELOFF}
                    {BRANCH place5209}"
21 place1209        +"/CFirst range ?~~/CSecond range ?~First range ?~
                    /C.{DOWN "&B36&"}{RIGHT "&B35&"}~Second range ?~
                    {PANELOFF}"
22 place2209        +"/RE.{DOWN "&B36&"}{RIGHT "&B35&"}~{GOTO}{here1209}~
                    {BRANCH contt209}"
23 place5209        +"/RE.{DOWN "&B36&"}{RIGHT "&B35&"}{NS "&B36&"}~{GOTO}
                    {here1209}~"
42 loc209           @IF(@LEFT(B40,1)<>"@","coord","address")
```

This macro enables you to switch places of same size ranges. The macro uses the / **F**ile **E**xtract commands so the disk should not be write protected. The `{LET rel209,@INFO ("release")}~` commands store the result of the `@INFO("release")` function in cell [rel209]. The macro uses this result to verify if you are using a 2-D or a 3-D Lotus release. Next the macro uses the `{RECALC form209}` command to update the dynamic formula in cell [form209], which is also the next cell of code to process. Before we can continue, let's look at the formula in cell [form209].

```
12 form209          +"{LET here1209,@CELLPOINTER("""&B42&""")}~"
```

We can see that the "dynamic" part of the formula is the content of B42, [loc209]. If we look in cell [loc209], we will find that it also contains the dynamic formula:

```
42 loc209           @IF(@LEFT(B40,1)<>"@","coord","address")
```

The dynamic part is the content of B40, [rel209], that holds the result of the `@INFO ("release")` function. If the first character of the content of [rel209] is the "`@`" character, then you are using a 2-D Lotus release and the result of the formula in cell [loc209] will be the "address" string. If the first character of the content of cell [rel209] is not the "`@`" character, then you are using a 3-D Lotus release and the result of the formula in cell [loc209] will be the "coord" string. Therefore the result of the formula in cell [form209] can accept two forms:

```
12 form209          {LET here1209,@CELLPOINTER("address")}~
```

or

```
12 form209          {LET here1209,@CELLPOINTER("coord")}~
```

This will depend on whether you are using a 2-D or a 3-D Lotus release. In the main listing we see the second form with the "coord" string. The `{LET here1209,@CELLPOINTER ("address")}~` commands store the current cell pointer location in cell [here1209], which the macro will use later to return to this position.

```
     A              B              C              D              E
13 !                {WINDOWSOFF}{PANELOFF}/RNCFirst range ?~~/RND
                    First range ?~/RNC{PANELON}First range ?~{WINDOWSON}
                    {BS}{BS}{?}~
14 !                {WINDOWSOFF}{PANELOFF}/RNCSecond range ?~~/RND
                    Second range ?~/RNC{PANELON}Second range ?~{WINDOWSON}
                    {BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}
```

Before the macro prompts you to paint the two ranges to switch, it issues `{WINDOWSOFF} {PANELOFF}` to freeze the screen and panel displays activity. Then the macro uses the "safe technique" to prompt you to paint the first range and simultaneously assigns the [First range ?]

range name to the first range. Next the macro repeats the same techniques to prompt you to paint the second range and simultaneously assigns [Second range ?] to the second range.

	A	B	C	D	E
15	!	{LET colss209,@STRING(@COLS(First range ?)-1,0)}~			
		{LET rowss209,@STRING(@ROWS(First range ?)-1,0)}~			
		{IF @LEFT(rel209,1)<>"@"}{LET sheets209,@STRING(@SHEETS			
		(First range ?)-1,0)}~			
16	!	{IF @COLS(First range ?)<>@COLS(Second range ?)#OR#			
		@ROWS(First range ?)<>@ROWS(Second range ?)}			
		{BRANCH messag209}			
17	!	{IF @LEFT(rel209,1)="@"}{BRANCH place3209}			
18	!	{IF @LEFT(rel209,1)<>"@"#AND#@SHEETS(First range ?)<>			
		@SHEETS(Second range ?)}{BRANCH messag209}			

The macro issues `{LET colss209,@STRING(@COLS(First range ?),0)}~` to store the number of columns of the first range in cell [colss209], and then uses `{LET rowss209 ,@STRING(@ROWS(First range ?),0)}~` to store number of rows of the first range in cell [rowss209]. The macro then uses `{IF @LEFT(rel209,1)<>"@"}` to make sure that the first character of the content of [rel209] is NOT the "@" character. If so, you are using a 3-D release and the macro issues `{LET sheets209,@STRING(@SHEETS(First range ?),0)}~` to store the number of sheets of the first range in cell [sheets209]. To make sure that both ranges have the same number of columns and the same number of rows the macro issues:

```
{IF @COLS(First range ?)<>@COLS(Second range ?)#OR#
    @ROWS(First range ?)<>@ROWS(Second range ?)}
```

If the two ranges do not have the same size the macro issues `{BRANCH messag209}` to activates the [messag209] routine.

	A	B	C	D	E
26	messag209	{BEEP}{PANELON}Ranges are not the same size,			
		overlapping is possible! [C]ont [Q]uit			
27	!	{GET key1209}{ESC}			
28	!	{IF @UPPER(key1209)="Q"}{BRANCH ret209}			
29	!	{BRANCH place3209}			

Before we continue, let's explore the [messag209] routine. The macro uses `{BEEP}` to worn you that something is wrong, then it uses `{PANELON}` to free the panel display activity, and write the text message:

```
Ranges are not the same size, overlapping is possible! [C]ont [Q]uit
```

directly to the panel. To halt the macro execution, the macro issues `{GET key1209}`. When you make a choice and press a key, the key's macro code is stored in cell [key1209], and the macro immediately issues `{ESC}` to clear the panel from the message before it will be written to the current cell and will change the worksheet. The macro uses `{IF @UPPER(key1209) ="Q"}` to check if you pressed the "Q" or the "q" character. If you pressed one of these characters the macro issues `{BRANCH ret209}` that routes the macro execution to an empty routine [ret209] and quits.

On the other hand, if you pressed any other key, the macro issues `{BRANCH place3209}` that routes the macro execution to the [place3209] routine. If the two ranges have the same number of columns and the same number of rows, the macro again uses

```
{IF @LEFT(rel209,1)<>"@"#AND#@SHEETS(First range ?)<>@SHEETS
    (Second range ?)}
```

to compare the number of sheets in both ranges if you are using a 3-D release. If this is false, the macro activates the [messag209] routine, but if this is true, the macro continues execution with the [place3209] routine.

	A	B	C	D	E
19	place3209	{END}{HOME}{DOWN}{END}{LEFT}/RNCThird range ?~~/RND Third range ?~/RNCThird range ?~~{RECALC place1209} {{RECALC place2209{RECALC PLACE4209}{RECALC place5209} {PANELON}			

To allow the switching, the macro issues `{END}{HOME}{DOWN}{END}{LEFT}` to move the cell pointer below the worksheet area on the "A" column. Then it uses the "safe technique" to assign the [Third range ?] range name to the current cell to create a temporary range outside the worksheet area. To update all the dynamic formulas in the worksheet, the macro issues: `{RECALC place1209}{RECALC place2209}{RECALC PLACE4209}{RECALC place5209}`.

	A	B	C	D	E
20	place4209	{IF @LEFT(rel209,1)<>"@"}/CFirst range ?~~ /CSecond range ?~First range ?~/C.{DOWN 5}{RIGHT 2} {NS 2}~Second range ?~{PANELOFF}{BRANCH place5209}			
21	place1209	/CFirst range ?~~/CSecond range ?~First range ?~/C. {DOWN 5}{RIGHT 2}~Second range ?~{PANELOFF}			
22	place2209	/RE.{DOWN 5}{RIGHT 2}~{GOTO}{here1209}~{BRANCH contt209}			
23	place5209	/RE.{DOWN 5}{RIGHT 2}{NS 5}~{GOTO}{here1209}~			
24	contt209	/RNDFirst range ?~/RNDSecond range ?~/RNDThird range ?~			

The code in cell [place4209] is the result of the following formula:

```
20 place4209        +"{IF @LEFT(rel209,1)<>""@""}/CFirst range ?~~
                    /CSecond range ?~First range ?~/C.{DOWN "&B36&"}
                    {RIGHT "&B35&"}{NS "&B37&"}~Second range ?~{PANELOFF}
                    {BRANCH place5209}"
```

The "dynamic" parts of the formula are the contents of the B36, B35 and B37 cells named [rowss209], [colss209] and [sheets209]. These cells hold the number of rows minus one, the number of column minus one, and the number of sheets minus one in the ranges to be switched respectively. Therefore the `{DOWN 5}`, `{RIGHT 2}` and `{NS 2}` commands are dynamically updated when you define the first and the second ranges.

The [place4209] routine starts with `{IF @LEFT(rel209,1)<>"@"}` to check if you are using a 3-D Lotus release. If this is true, the macro uses `/CFirst range ?~~` to temporarily copy the [First range ?] range to the [Third range ?] range. Then it uses `/CSecond range ?~First range ?~` to copy the [Second range ?] into the [First range ?]. Last the macro uses `/C.{DOWN 5}{RIGHT 2}{NS 2}~Second range ?~` to copy the [First range ?] stored in the [Third range ?] to the [Second range ?]. Next the macro uses `{PANELOFF}` to stop the panel display activity and issues `{BRANCH place5209}`. The [place5209] routine uses `/RE.{DOWN 5}{RIGHT 2}{NS 5}~` to erase the content of the temporary [Third range ?] and the `{GOTO}{here1209}~` indirect command to move the cell pointer its point of origin before the macro was started. To end, the macro issues `/RND First range ?~/RNDSecond range ?~/RNDThird range ?~` to delete the [First range ?], [Second range ?] and [Third range ?] range names and leave the worksheet as

clean as possible.

If you are using a 2-D Lotus release, the macro uses the [place1209] routine which is a simpler 2-D version of the [place4209] routine. Because of the complexity of this macro, here is a full list of all the cell formulas and contents.

```
A1: U [W10] '*---A macro to switch 3-D and 2-D ranges places
A2: [W10] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W10] '    range names in this column (starts with the \Z macro name)
A4: [W10] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W10] '!
A6: U [W10] '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3
            RELEASE
A7: U [W10] '                    IT WILL WORK IN LOTUS 2.0 AND UP
A8: [W10] '!
A9: [W10] '!
A10: U [W10] '\Z
B10: [W13] '{BREAKON}
A11: U [W10] 'SWITCH
B11: [W13] '{LET rel209,@INFO("release")}~{RECALC loc209}{RECALC form209}
A12: [W10] 'form209
B12: U [W13] +"{LET here1209,@CELLPOINTER(""&B42&"")}~"
A13: [W10] '!
B13: [W13] '{WINDOWSOFF}{PANELOFF}/RNCFirst range ?~~/RNDFirst range ?~
            /RNC{PANELON}First range ?~{WINDOWSON}{BS}{BS}{?}~
A14: [W10] '!
B14: [W13] '{WINDOWSOFF}{PANELOFF}/RNCSecond range ?~~/RNDSecond range ?~
            /RNC{PANELON}Second range ?~{WINDOWSON}{BS}{BS}{?}~{WINDOWSOFF}
            {PANELOFF}
A15: [W10] '!
B15: [W13] '{LET colss209,@STRING(@COLS(First range ?)-1,0)}~
            {LET rowss209,@STRING(@ROWS(First range ?)-1,0)}~{IF @LEFT
            (rel209,1)<>"@"}{LET sheets209,@STRING(@SHEETS(First range ?)
            -1,0)}~
A16: [W10] '!
B16: [W13] '{IF @COLS(First range ?)<>@COLS(Second range ?)#or#@ROWS
           (First range ?)<>@ROWS(Second range ?)}{BRANCH messag209}
A17: [W10] '!
B17: [W13] '{IF @LEFT(rel209,1)="@"}{BRANCH place3209}
A18: [W10] '!
B18: [W13] '{IF @LEFT(rel209,1)<>"@"#AND#@SHEETS(First range ?)<>@SHEETS
           (SECOND RANGE ?)}{BRANCH messag209}
A19: [W10] 'place3209
B19: [W13] '{END}{HOME}{DOWN}{END}{LEFT}/RNCThird range ?~~/RND
           Third range ?~/RNCThird range ?~~{RECALC place1209}
           {RECALC place2209}{RECALC PLACE4209}{RECALC place5209}{PANELON}
A20: [W10] 'place4209
B20: U [W13] +"{IF @LEFT(rel209,1)<>""@""}/CFirst range ?~~
             /CSecond range ?~First range ?~/C.{DOWN "&B36&"}{RIGHT "&B35&
             "}{NS "&B37&"}~Second range ?~{PANELOFF}{BRANCH place5209}"
A21: [W10] 'place1209
B21: U [W13] +"/CFirst range ?~~/CSecond range ?~First range ?~/C.{DOWN "
             &B36&"}{RIGHT "&B35&"}~Second range ?~{PANELOFF}"
A22: [W10] 'place2209
B22: U [W13] +"/RE.{DOWN "&B36&"}{RIGHT "&B35&"}~{GOTO}{here1209}~
             {BRANCH contt209}"
A23: [W10] 'place5209
B23: U [W13] +"/RE.{DOWN "&B36&"}{RIGHT "&B35&"}{NS "&B36&"}~{GOTO}
             {here1209}~"
A24: [W10] 'contt209
B24: [W13] '/RNDFirst range ?~/RNDSecond range ?~/RNDThird range ?~
A25: [W10] '!
A26: [W10] 'messag209
B26: [W13] '{BEEP}{PANELON}Ranges are not the same size, overlapping is
           possible!  [C]ont  [Q]uit
A27: [W10] '!
B27: [W13] '{GET key1209}{ESC}
A28: [W10] '!
```

```
B28: [W13] '{IF @UPPER(key1209)="Q"}{BRANCH ret209}
A29: [W10] '!
B29: [W13] '{BRANCH place3209}
A30: [W10] '!
A31: [W10] 'here1209
B31: [W13] '$E:$E$1
A32: [W10] '!
A33: [W10] 'key1209
B33: [W13] 'q
A34: [W10] '!
A35: [W10] 'colss209
B35: [W13] '2
A36: [W10] 'rowss209
B36: [W13] '5
A37: [W10] 'sheets209
B37: [W13] '2
A38: [W10] 'ret209
A39: [W10] '!
A40: [W10] 'rel209
B40: [W13] '
A41: [W10] '!
A42: [W10] 'loc209
B42: U [W13] @IF(@LEFT(B40,1)<>"@","coord","address")
```

[6] Center a Column of Titles Across the Screen

```
     A              B                C                D              E
 1 *---A macro to CENTER a column of titles in middle of screen
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---The labels to be centered must be on the most left column on screen
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                     IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 CENTER2         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb502,0}~
12 cont502         {LET hereabs502,@CELLPOINTER("address")}~
                   {LET counter1502,0}
13 !               {FOR counter1502,0,@COLS(Which range ?)-1,1,labels1502}
14 !               {LET rel502,@INFO("release")}~{IF @LEFT(rel502,1)<>"@"}
                   {GOTO}{hereabs502}~{LET counterb502,counterb502+1}~
                   {IF counterb502<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs502}~{BRANCH cont502}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1502                 1
18 counter1a502                0
19 labels1502      {RIGHT}{LET here502,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a502,0,@ROWS(Which range ?)-1,1,
                   labels1a502}~{IF counter1502<@COLS(Which range ?)-1}
                   {GOTO}{here502}~{LET counter1a502,0}~
20 !
21 here502         $F$1
22 !
23 labels1a502     {IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{DEL}@REPEAT
                   (" ",(72-@LENGTH(@CELLPOINTER("contents")))/2)&"{END}"
                   {CALC}
24 !               {DOWN}
25 !
26 counterb502                 1
27 hereabs502      $A$31
28 !
29 rel502          3.00.00
```

This macro will center all the labels in a range across the screen. It is designed for a Lotus screen 80 characters wide, for other screen configurations, you must change the 72 number inside the formula in the B23 cell. The `{WINDOWSOFF}{PANELOFF}` macro commands freeze the screen and panel display activity. Then the macro uses the "safe technique" to prompt you to paint the range of labels and simultaneously assigns the [Which range ?] name to the same range, which acts as a prompt. Next the macro issues `{LET counterb502,0}~` to set the content of cell [counterb502] to zero, which serves as a counter.

```
     A              B                C                D              E
12 cont502         {LET hereabs502,@CELLPOINTER("address")}~
                   {LET counter1502,0}
13 !               {FOR counter1502,0,@COLS(Which range ?)-1,1,labels1502}
14 !               {LET rel502,@INFO("release")}~{IF @LEFT(rel502,1)<>"@"}
                   {GOTO}{hereabs502}~{LET counterb502,counterb502+1}~
                   {IF counterb502<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs502}~{BRANCH cont502}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro is finished, the macro issues the `{LET`

hereabs502,@CELLPOINTER("address")}~ commands that store the current cell pointer address in cell [hereabs502]. Then it issues {LET counter1502,0} to set the content of cell [counter1502] to zero, which also serves as a counter.

The macro issues the {FOR counter1502,0,@COLS(Which range ?)-1,1, labels1502} loop command that activates the [labels1502] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels1502] routine.

	A	B	C	D	E
19	labels1502	{RIGHT}{LET here502,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a502,0,@ROWS(Which range ?)-1,1,			
		labels1a502}~{IF counter1502<@COLS(Which range ?)-1}			
		{GOTO}{here502}~{LET counter1a502,0}~			

The [labels1502] routine uses the {RIGHT}{LET here502,@CELLPOINTER("address") }~{LEFT} commands to record the address of the first cell of the next column in cell [here502]. This way, when the current column processing is finished, the macro will move the cell pointer directly to the top of the next column using the address stored in [here502]. The {FOR counter1a502,0,@ROWS(Which range ?),1,1,labels1a502}~ commands activate the [labels1a502] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
23	labels1a502	{IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{DEL}@REPEAT			
		(" ",(72-@LENGTH(@CELLPOINTER("contents")))/2)&"{END}"			
		{CALC}			
24	!	{DOWN}			

The [labels1a502] routine issues {IF @CELLPOINTER("type")="l"} to check if the current cell contains a label. If so, the macro creates an enveloping formula around the label in the cell. To make it easier to follow, let's assume the cell contains the "HELLO DOLLY" string. The macro issues {EDIT} to enter the EDIT mode, and then issues {HOME}{DEL} to move the cursor to the beginning of the text in the panel and delete the apostrophe prefix. Now the panel displays:

```
HELLO DOLLY
```

Next the macro types the

```
@REPEAT(" ",(72-@LENGTH(@CELLPOINTER("contents")))/2)&"
```

text before the "HELLO DOLLY" text and the panel displays:

```
@REPEAT(" ",(72-@LENGTH(@CELLPOINTER("contents")))/2)&"HELLO DOLLY
```

To finish the formula, the macro issues the {END} command that moves the cursor to the end of the panel, and then types the closing double quotes ["], which causes the panel to display:

```
@REPEAT(" ",(72-@LENGTH(@CELLPOINTER("contents")))/2)&"HELLO DOLLY"
```

For the final touch, the macro issues {CALC} that turns the formula into a pure label with enough leading spaces such that the "HELLO DOLLY" text appears centered in the row. Last the macro issues {DOWN} that moves the cell pointer to the next cell in the column to be centered. Let us continue with the [labels1502] routine.

	A	B	C	D	E
19	labels1502	{RIGHT}{LET here502,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a502,0,@ROWS(Which range ?)-1,1,			
		labels1a502}~{IF counter1502<@COLS(Which range ?)-1}			
		{GOTO}{here502}~{LET counter1a502,0}~			

When this {FOR} loop is finished with the first column, the macro issues {IF counter1502<@COLS(Which range ?)-1} to check if there are more columns to process. If so, the macro issues the {GOTO}{here502}~ indirect macro command to move to the first cell of the next column ([here502] holds the address of the first cell in the next column). The last macro commands in this routine are {LET counter1a502,0}~ that reset the counter in cell [counter1a502] to zero. Now we can go back to the [cont502] routine.

	A	B	C	D	E
12	cont502	{LET hereabs502,@CELLPOINTER("address")}~			
		{LET counter1502,0}			
13	!	{FOR counter1502,0,@COLS(Which range ?)-1,1,labels1502}			
14	!	{LET rel502,@INFO("release")}~{IF @LEFT(rel502,1)<>"@"}			
		{GOTO}{hereabs502}~{LET counterb502,counterb502+1}~			
		{IF counterb502<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs502}~{BRANCH cont502}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

The macro continues with {LET rel505,@INFO("release")}~ that store the result of the @INFO("release") function in cell [rel502]. Then the macro issues {IF @LEFT (rel5=2,1)<>"@"} to check if you are using a 2-D or a 3-D Lotus release. If the first character of the content of cell [rel502] is the "@" character, then you are using a 2-D Lotus release, otherwise you are using a 3-D release. If it is a 3-D release, the macro issues the {GOTO}{hereabs502}~ indirect macro command to move to the first cell in the current sheet range, and then issues {LET counterb502,counterb502+1}~ to increase the counter in cell [counterb502] by one.

Next the macro uses {IF counterb502<@SHEETS(Which range ?)} to compare the counter value in cell [counterb502] to the number of sheets in the [Which range ?] range. If the counter is still less than the number of sheets in [Which range ?], the macro has to process more sheets before it can quit. Therefore the macro issues {NS}{GOTO} {hereabs502}~ to move the cell pointer to the upper left cell of the range in the new sheet. Next it issues {BRANCH cont502} to loop back to the beginning of the [cont505] routine to process the new sheet.

When the counter in cell [counterb502] is equal to the number of sheets in the [Which range ?] range, the macro issues {GOTO}Which range ?~ to place the cell pointer back on the origin cell just before the macro started. Then it uses the /RNDWhich range ?~ macro command to delete the [Which range ?] range name to leave a clean worksheet.

[3] Center a Title Across a Row

```
     A                B                C                D                E
 1 *---A macro to CENTER a TITLE across the row
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define
 3     the range names in this column (starts with the \Z macro name)
 4 *---Move the cell pointer to the first column
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 CTRTITLE        {GETLABEL "Enter title to center: ",title100}~
                   {RECALC dummy100}~{dummy100}~{BRANCH ret100}
12 !
13 dummy100                                  Hello User
14 !
15 title100        Hello User
16 !
17 ret100
```

Notice that the B12 cell contains the following dynamic string formula:

```
12 !              @REPEAT(" ",@INT((72-@LENGTH(B15))/2))
```

and the B13 cell named [dummy100] contains the following formula:

```
13 dummy100       +B12&B15
```

If you intend to key this macro into Lotus 1-2-3, you have to key these formulas into the B12 and the B13 cells, NOT the code as it appears in the main listing.

This macro adds leading spaces to a string to make the string look centered across the row, assuming that the row is 72 characters wide. The macro starts with the `{GETLABEL "Enter title to center: ",title100}` macro command which displays the "`Enter title to center: `" prompt message in the panel and expects you to insert the title string. When you press the ENTER key, Lotus stores the title in the B15 cell, [title100]. For example let us assume that you typed the "Hello User" string for a title. Then the macro issues `{RECALC dummy100}~` to recalculate the formulas in the B12 cell.

```
@REPEAT(" ",@INT((72-@LENGTH(B15))/2))
```

The formula in the B12 cell uses the `@REPEAT` Lotus function to create the leading spaces to the title. The formula assumes that you are using an 80X25 screen, which in Lotus 1-2-3, leaves a row of 72 characters. If a 132 column screen is used, the 72 in the formula should be changed accordingly.

```
13 dummy100       +B12&B15
```

The formula in the B13 cell sums the spaces in the B12 cell with the title string. Now the macro issues the `{dummy100}` routine command which executes the [dummy100] routine. Because the [dummy100] routine contains the combined string of the leading spaces and the title string, Lotus injects the content of the cell into the panel. Next the macro issues the tilde "`~`" macro command which is the same as the ENTER key and writes the centered title into the current cell. Last the macro issues the `{BRANCH ret100}` macro command which routes the

macro control to the empty routine named [ret100] and quits.

The result in the current cell is the

```
Hello User
```

label which has enough leading spaces so it looks centered across the row.

[3] Continuously Hide Columns

```
     A                B                C                D                E
 1 *---A macro to contiguously HIDE columns
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Place the cell pointer at least one column apart of the columns
 6     you plan to hide. When the macro is activated the cell pointer
 7     position is recorded, hiding the column cause the managers to
 8     loose track and beep few times.
 9 !
10 \Z               {BREAKON}
11 COLSHIDE         {BREAKON}
12 loop096          {ESC}Use the direction keys and press [ENTER] to hide.
                    Press ESC to quit.
13 !                {GET key096}{ESC}{IF key096="{ESC}"}~{ESC 6}
                    {BRANCH ret096}
14 !                {IF key096="~"}/WCH~{BRANCH loop096}
15 !                {key096}{BRANCH loop096}
16 !
17 key096           {ESC}
18 !
19 ret096
```

This macro allows you to continuously hide contiguous on non-contiguous columns easily. The macro starts with the `{ESC}` macro command (its use will be explained later) and then writes:

```
Use the direction keys and press [ENTER] to hide. Press [ESC] to quit.
```

To keep the message in the panel and allow you to read it, the macro issues `{GET key096}` which halts the macro execution. When you press a key, Lotus stores the key's macro code in cell [key096] and then issues `{ESC}` which clears the message from the panel before Lotus writes the message into the current cell. Next the macro issues `{IF key096="{ESC}"}` to check if you pressed the ESC key. If so, the macro issues `{ESC 6}` to return to the READY mode. Next the macro issues `{BRANCH ret096}` which routes the macro control to the empty [ret096] routine and quits.

If the condition was false, the macro issues the second `{IF key096="~"}` condition to check if you pressed the ENTER key. IF so, macro issues `/WCH~` to hide the current column and then issues `{BRANCH loop096}` which routes the macro control back to the beginning to allow you to hide more columns.

If both last were false and you pressed a command key which has an equivalent macro command (such as the `{RIGHT}` macro command which is the same as the RIGHT key pressed from the keyboard), then the `{key096}` routine command executes that key. If you pressed a key like the "A" character, the `{key096}` routine command injects the "`A`" character to the panel and the macro issues `{BRANCH loop096}` which routes the macro control back to the beginning to allow you to hide more columns. The `{ESC}` then clears the "A" character from the panel. This is the reason for the `{ESC}` command before the prompt message. In this macro we have seen that using the `{GET key}` macro command combined with `{IF}` conditions we can monitor and control the user's input.

[3] View Hidden Columns Easily

```
     A                B                C                D                E
 1 *---A macro to VIEW and/or UNHIDE ALL the hidden columns
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 COLSDISP       {MENUBRANCH menu437}
12 !
13 menu437        View_only      Unhide_all     Selective_unhide  Quit
14 !              View hidden colUnhide ALL the View and unhide seQuit th
15 !              /WCD{?}{ESC 5} {WINDOWSOFF}/WC/WCD{?}~
16 !              {MENUBRANCH men{MENUBRANCH men{MENUBRANCH menu437}
```

This macro uses a custom menu with four menu options to make the task of hiding and showing columns easier and more natural. Because the four menu options cannot fit side by side on one screen without overlapping, we show them separately. If you intend to key the macro into 1-2-3, you have to key menu options code as it appears here, NOT as it appears in the main listing. To further help you, we show the full cell contents list at the end of this section.

```
View_only
View hidden columns
/WCD{?}{ESC 5}
{MENUBRANCH menu437}

Unhide_all
Unhide ALL the columns
{WINDOWSOFF}/WCD{HOME}.{END}{HOME}{END}{RIGHT}~
{MENUBRANCH menu437}

Selective_unhide
View and unhide selective column
/WCD{?}~
{MENUBRANCH menu437}

Quit
Quit the macro
```

The macro starts with the `{MENUBRANCH menu437}` macro command which activates the [menu437] custom menu. The first menu option of the custom menu allows you to view hidden columns using fewer key presses:

```
View_only
View hidden columns
/WCD{?}{ESC 5}
{MENUBRANCH menu437}
```

The macro issues `/WCD{?}` to display the hidden columns and allow you to use the arrow keys, to wander around the worksheet and look for the hidden columns. When you press the ENTER key or the ESC key, the macro issues `{ESC}` to return to READY mode. Next the macro issues `{MENUBRANCH menu437}` which re-activates the [menu437] custom menu.

```
Unhide_all
Unhide ALL the columns
{WINDOWSOFF}/WCD{HOME}.{END}{HOME}{END}{RIGHT}~
```

```
{MENUBRANCH menu437}
```

The second menu option displays all the hidden columns in the worksheet in one operation. It starts with `{WINDOWSOFF}` which freezes the screen display activity while the macro continues with the `/WCD{HOME}.{END}{HOME}{END}{RIGHT}` command sequence to re-display all the hidden columns in the working area of the worksheet. Next the macro issues `{MENUBRANCH menu437}` which re-activates the [menu437] custom menu.

```
Selective_unhide
View and unhide selective column
/WCD{?}~
{MENUBRANCH menu437}
```

This menu option is like the Lotus natural / **C**olumn **D**isplay operation, which allows you to paint the desired columns to display. When you are finished, the menu re-appears ready for the next columns to display; thereby saving key presses when selective unhiding is necessary. The macro issues `/WCD{?}` which waits for you to paint the columns to display. When you press the ENTER key, the macro issues the tilde "`~`", which is the same as the ENTER key, and re-displays all the pre-hidden columns. Next the macro issues `{MENUBRANCH menu437}` which re-activates the [menu437] custom menu.

```
Quit
Quit the macro
```

When you choose the [**Q**uit] menu option the macro quits because it reaches an empty cell.

[1] Set the Width of a Group of Columns

```
        A                B                C                D                E
 1 *---A macro to change the width of group of adjacent columns
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8              * * *     FOR LOTUS 2.2 AND UP    * * *
 9 !
10 \Z              {BREAKON}
11 WIDTHSET        /WCCS{?}~{?}~
```

This is a simple macro to change the width of a group of adjacent columns. It saves a few key strokes every time you need to change the column width. The macro starts with the `/WCCS` macro commands and then issues the `{?}` macro command allowing you to paint the range of columns to set. When you press the ENTER key, the macro issues a second `{?}` macro command allowing you to insert the new width of the columns. When you again press the ENTER key, the macro issues the tilde "`~`" macro command, which is the same as pressing the ENTER key from the keyboard, and changes the width of the columns in the range.

[1] Reset the Width of a Group of Columns

```
     A                B                C                D                E
 1 *---A macro to reset the width of group of adjacent columns
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8            * * *     A LOTUS 2.2 AND UP MACRO     * * *
 9 !
10 \Z              {BREAKON}
11 WIDTHRTS        /WCCR{?}~
```

This is a simple macro to reset the width of a group of adjacent columns. It saves a few key strokes every time you need to reset the column width. The macro starts with the `/WCCR` macro commands and then issues the `{?}` macro command allowing you to paint the range of columns to reset. When you press the ENTER key, the macro issues the tilde "`~`" macro command, which is the same as pressing the ENTER key from the keyboard, and resets the range of the columns.

[3] Adjust the Column Width to the Current Label Length

```
     A                B                C                D                E
 1 *---A macro to adjust the column width to the length of the label in
 2     the current cell (1 space longer).
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z               {BREAKON}
11 WIDMATCH         {LET width211,@STRING(@LENGTH(@CELLPOINTER("contents"))
                    +1,0)}~
12 !                /WCS
13 width211         16
14 !                ~
```

The macro adjusts the column width to the length of the label in the current cell plus one. Therefore look for the longest label in the column, place the cell pointer on it and start the macro. The macro issues the

```
{LET width211,@STRING(@LENGTH(@CELLPOINTER("contents"))+1,0)}~
```

commands which store the result of the `@STRING(@LENGTH(@CELLPOINTER ("contents"))+1,0)` formula in the B13 cell named [width211]. For example, if the length of the label in the current cell is 15, this formula calculates the length of the content of the current cell and then changes it into a string format, which becomes the label 16. Then the `{LET}` command stores it in B13, [width211]. Next the macro issues `/WCS16~` to adjust the column width to 16. The result of the `{LET}` command in the B13 cell became a part of the code to change the column width. This is a simple example of using dynamic code in macros.

[5] Adjust the Column Width to the Current Label/Number Length

```
             A                B                C                D                E
 1 *---A macro to SET the column width to one more than the LABEL/NUMBER
 2     length in the current cell.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                     IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 WIDTHRST         {LET rel049,@INFO("release")}~{RECALC loc049}
                    {RECALC form049}
12 form049          {WINDOWSOFF}{PANELOFF}{IF @CELLPOINTER("type")="v"}
                    /RV~width2049~{LET here049,@CELLPOINTER("coord")}~
                    {GOTO}width2049~{EDIT}{HOME}'~{LET width049,@STRING
                    (@LENGTH(width2049)+1,0)}~{WINDOWSON}{PANELON}{GOTO}
                    {here049}~
13 !                {IF @CELLPOINTER("type")="l"}{LET width049,@STRING
                    (@LENGTH(@CELLPOINTER("contents"))+1,0)}~
14 !                /  WCS
15 width049         9
16 !                ~
17 !
18 width2049        12345
19 !
20 here049          $B:$A$1
21 !
22 rel049           3.00.00
23 !
24 loc049           coord
```

This macro changes the column width based on the length of the label or a value in the current cell. For example: if the current cell contains the "BUDGET" string which is six character long, the macro will change the column width to seven. Before we start, notice that the code in the B12 and the B24 cells is the result of the following dynamic string formulas.

```
12 form049          +"{WINDOWSOFF}{PANELOFF}{IF @CELLPOINTER(""type"")=
                    ""v""}/RV~width2049~{LET here049,@CELLPOINTER("""&B24&
                    """)}~{GOTO}width2049~{EDIT}{HOME}'~{LET width049,
                    @STRING(@LENGTH(width2049)+1,0)}~{WINDOWSON}{PANELON}
                    {GOTO}{here049}~"
24 loc049           @IF(@LEFT(B22,1)<>"@","coord","address")
```

Therefore, if you plan to key this macro into Lotus 1-2-3, you have to key these formulas in the B12 and the B24 cells of the macro, NOT the code as it appears in the main listing. The macro issues `{LET rel049,@INFO("release")}~` to store the result of the `@INFO ("release")` function into the B22 cell named [rel049]. Later, the macro uses this result to check if you are using a 3-D or a 2-D Lotus release. Next the macro issues `{RECALC loc049}{RECALC form049}` which update the formulas in the B12 cell [form049] and the B24 cell [loc049] based on the result of the `@INFO("release")` function.

```
             A                B                C                D                E
12 form049          {WINDOWSOFF}{PANELOFF}{IF @CELLPOINTER("type")="v"}
                    /RV~width2049~{LET here049,@CELLPOINTER("coord")}~
                    {GOTO}width2049~{EDIT}{HOME}'~{LET width049,@STRING
                    (@LENGTH(width2049)+1,0)}~{WINDOWSON}{PANELON}{GOTO}
                    {here049}~
```

Next, the macro issues {WINDOWSOFF}{PANELOFF} to freeze the screen and panel display activities, and then issues {IF @CELLPOINTER("type")="v"} to check the type of data in the current cell. If the cell contains a value (number or formula), the macro issues /RV~width2049~ which copies the value in the cell into the B15 cell [width2049] as a number. Because the macro needs to move the cell pointer now, but needs to return, the macro issues {LET here049,@CELLPOINTER("coord")}~ to store the current cell pointer location in cell [here049]. Next the macro issues {GOTO}width2049~ which moves the cell pointer to cell [width2049] and then issues {EDIT}{HOME}'~ to add the apostrophe "'" to the value in the cell and change it into a label.

Next the macro issues {LET width049,@STRING(@LENGTH(width2049)+1,0)}~ to store the result of the @STRING(@LENGTH(width2049)+1,0) formula in cell [width049] as a string. This is a string of a number that is greater by one from the length of the value in cell [width2049] (this cell holds the length of the label/value of the origin cell). To resume the screen and panel display activities, the macro issues {WINDOWSON}{PANELON}, and then the indirect {GOTO}{here049}~ macro command to return the cell pointer to the point of origin. The code in B12 is the result of a string formula which also explain how the "coord" attribute appears here. This attribute is available only in the 3-D releases of Lotus 1-2-3. To understand it, we need to look in the formula in the B24 cell that depends on the content of the B22 cell which contains the result of the @INFO("release") function.

```
24 loc049          @IF(@LEFT(B22,1)<>"@","coord","address")
```

The formula in the B24 cell, [loc049], checks if the first character of the content of cell [rel049] is not the "@" character. If SO, you are using a 3-D Lotus release. Therefore the formula returns the "coord" string. Otherwise you are using a 2-D release and the formula returns the "address" string. This way the macro adapts itself to the type of spreadsheet that you use.

```
12 form049         +"{WINDOWSOFF}{PANELOFF}{IF @CELLPOINTER(""type"")=
                   ""v""}/RV~width2049~{LET here049,@CELLPOINTER(""""&B24&
                   """")}~{GOTO}width2049~{EDIT}{HOME}'~{LET width049,
                   @STRING(@LENGTH(width2049)+1,0)}~{WINDOWSON}{PANELON}
                   {GOTO}{here049}~"
```

The formula in B12, [form049], uses the previous result in B24, [loc049], to create the correct code.

```
     A              B                C              D              E
13 !               {IF @CELLPOINTER("type")="l"}{LET width049,@STRING
                   (@LENGTH(@CELLPOINTER("contents"))+1,0)}~
```

If the current cell does not contain a value the macro issues {IF @CELLPOINTER ("type")="l"} to check if it contains a label. If so, the macro issues {LET width049 ,@STRING(@LENGTH(@CELLPOINTER("contents"))+1,0)}~ to store the length of the label plus one as a string in cell [width049].

```
     A              B                C              D              E
14 !               /  WCS
15 width049        9
16 !               ~
```

If we look at this very simple code, /WCS9~, which changes the column width to nine characters

wide, we should remember that the string "9" is a dynamic result of a quite complicated code. This is an example of a different type of dynamic code which is not directly the result of a formula. To make it easier for you to key this macro into Lotus 1-2-3, below is the full list of cell contents and formulas.

```
A1: U [W9] '*---A macro to SET the column width to one more than the
           LABEL/NUMBER
A2: U [W9] '    length in the current cell.
A3: [W9] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
         define the
A4: [W9] '    range names in this column (starts with the \Z macro name)
A5: [W9] '*---Hold the [ALT] key and press [Z] to activate the macro
A6: U [W9] '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3
           RELEASE
A7: U [W9] '                   IT WILL WORK IN LOTUS 2.0 AND UP
A8: U [W9] 'ATTENTION: In release 3 when used from MACROMG3.WK3 or
           SMALLMG3.WK3 it
A9: U [W9] '           may not work on the manager sheet ( a bug in
           release 3 )
A10: U [W9] '\Z
B10: [W9] '{BREAKON}
A11: U [W9] 'WIDTHRST
B11: [W9] '{LET rel049,@INFO("release")}~{RECALC LOC049}{RECALC form049}
A12: [W9] 'form049
B12: U [W9] +"{WINDOWSOFF}{PANELOFF}{IF @CELLPOINTER(""type"")=""v""}
            /RV~width2049~{LET here049,@CELLPOINTER(""""&B24&""")}~{GOTO}
            width2049~{EDIT}{HOME}'~{LET width049,@STRING(@LENGTH
            (width2049)+1,0)}~{WINDOWSON}{PANELON}{GOTO}{here049}~"
A13: [W9] '!
B13: [W9] '{IF @CELLPOINTER("type")="l"}{LET width049,@STRING(@LENGTH(@CELLPOINTER
           ("contents"))+1,0)}~
A14: [W9] '!
B14: [W9] '/WCS
A15: [W9] 'width049
B15: [W9] '9
A16: [W9] '!
B16: [W9] '~
A17: [W9] '!
A18: [W9] 'width2049
B18: [W9] '12345
A19: [W9] '!
A20: [W9] 'here049
B20: [W9] '$B:$A$1
A21: [W9] '!
A22: [W9] 'rel049
B22: [W9] '3.00.00
A23: [W9] '!
A24: [W9] 'loc049
B24: U [W9] @IF(@LEFT(B22,1)<>"@","coord","address")
```

[5] Change Multiple Column Widths

```
     A                B                C                D                E
 1 *---A macro to change multiple column width
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
11 \Z              {BREAKON}
11 COLWIDTH        {BREAKON}
12 loop004         {WINDOWSON}{PANELON}{MENUBRANCH menu004}
13 !
14 menu004         One_at_a_time   All        Quit
15 !               Set the width foSet the samQuit the macro
16 !               {WINDOWSON}{pane{WINDOWSON}{PANELON}{GOTO}{?}~{LET her
17 !               {GETNUMBER "Numb{GETNUMBER "Number of columns to chang
18 !               {FOR counter004,{GETLABEL "Column width ? ",colwide300
19 !               {GOTO}{here004}~{WINDOWSOFF}{PANELOFF}{FOR counter004,
20 !
21 colwide1004      /WCS{?}~{RIGHT}
22 !
23 colwide2004      /WCS9~{RIGHT}
24 !
25 colwide3004      9
26 counter004                  2
27 columns004                  50
28 here004
```

This macro allows you to simultaneously set the same column width to a specified number of columns, or set the width for a column at a time, each with a different column width. The macro uses a custom menu with three menu options and because one page cannot display all the menu options side by side without overlapping; we show here each menu option separately. If you intend to key this macro into Lotus 1-2-3, you should key the menu options as they appear here, NOT the code as it appears in the main listing.

```
One_at_a_time
Set the width for every column separately
{WINDOWSON}{PANELON}{GOTO}{?}~{LET here004,@CELLPOINTER("address")}~
{GETNUMBER "Number of columns to change? ",columns004}~
{FOR counter004,1,columns004,1,colwide1004}
{GOTO}{here004}~{BRANCH loop004}

All
Set the same width for all columns
{WINDOWSON}{PANELON}{GOTO}{?}~{LET here004,@CELLPOINTER("address")}~
{GETNUMBER "Number of columns to change? ",columns004}~
{GETLABEL "Column width ? ",colwide3004}~{RECALC colwide2004}
{WINDOWSOFF}{PANELOFF}{FOR counter004,1,columns004,1,colwide2004}
    {GOTO}{here004}{BRANCH loop004}

Quit
Quit the macro
```

Notice that the code in the B23 cell named [colwide2004] is the result of the following dynamic string formula; therefore you should key the formula, NOT the code as it appears in the main listing.

```
23 colwide2004      +"/WCS"&B25&"~{RIGHT}"
```

```
     A          B          C          D          E
12 loop004      {WINDOWSON}{PANELON}{MENUBRANCH menu004}
```

The macro starts with {PANELOFF}{WINDOWSOFF} which freeze the screen and panel display activities, and then issues {MENUBRANCH menu004} which starts the [menu004] custom menu. The first option is [**O**ne_at_a_time]:

```
One_at_a_time
Set the width for every column separately
{WINDOWSON}{PANELON}{GOTO}{?}~{LET here004,@CELLPOINTER("address")}~
{GETNUMBER "Number of columns to change? ",columns004}~
{FOR counter004,1,columns004,1,colwide1004}
{GOTO}{here004}~{BRANCH loop004}
```

When you select the [One_at_a_time] menu option, the macro issues {WINDOWSON} {PANELON} which free the screen and panel activities. Then the macro issues {GOTO} and then {?} which forces Lotus to pause and wait for your reply. You can type an address, range name or move to the correct location using the direction keys. When you press the ENTER key, the macro issues the tilde "~" which is the same as the ENTER key pressed from the keyboard. Then the macro issues {LET here004,@CELLPOINTER("address") }~ which records the cell pointer position for later return to the same location. Then the macro issues {GETNUMBER "Number of columns to change?",columns004}~ which displays the "Number of columns to change?" prompt message in the panel, and pauses until you reply and enter the number of columns you want to change. When you type a number and press ENTER, Lotus stores it in cell [columns004].

Now the macro issues {FOR counter004,1,columns004,1,colwide1004} which activates the [colwide1004[routine as many times as the number stored in cell [columns004]. When the loop ends, the macro issues the {GOTO}{here004}~ indirect command which moves the cell pointer back to its point of origin before the macro started, and then issues {BRANCH loop004} to start the macro again. The second menu option is [**A**ll]:

```
All
Set the same width for all columns
{WINDOWSON}{PANELON}{GOTO}{?}~{LET here004,@CELLPOINTER("address")}~
{GETNUMBER "Number of columns to change? ",columns004}~
{GETLABEL "Column width ? ",colwide3004}~{RECALC colwide2004}
{WINDOWSOFF}{PANELOFF}{FOR counter004,1,columns004,1,colwide2004}
    {GOTO}{here004}{BRANCH loop004}
```

This time the macro issues {WINDOWSON}{PANELON} which free the screen and panel activities. Then the macro issues {GOTO} and then {?} to force Lotus to pause and wait for your reply. You can type an address, range name or use the direction keys to move to the correct location. When you press ENTER, the macro issues the tilde "~". Next the macro issues {GETNUMBER "Number of columns to change? ",columns004}~ which displays "Number of columns to change? " on the panel, and pauses until you reply and enter the number of columns to change. When you type a number and press ENTER, Lotus stores the number in cell [columns004].

Next the macro issues {GETLABEL "Column width ? ",colwide3004} which displays "Column width ? " on the panel, and pauses until you enter the width to apply to the group of columns and press ENTER. Then Lotus issues the tilde "~" to store the number as a label in B23, [colwide3004]. Next it issues {RECALC colwide2004} which updates the dynamic

string formula in cell [colwide2004]. For example: if you type the number "9", the formula in B23, [colwide2004], returns the /WCS9~{RIGHT} macro code.

Now the macro issues {WINDOWSOFF}{PANELOFF} which freezes the screen and panel activities while it issues {FOR counter004,1,columns004,1,colwide2004} which changes the next fifty columns if you typed the number "50" in response to the first prompt. When the loop is finished, the macro issues {GOTO}{here004}~{BRANCH loop004} which move the cell pointer back to its point of origin before the macro started and loops back to the beginning of the macro. The third menu option is [**Q**uit]:

```
Quit
Quit the macro
```

When you choose the [**Q**uit] menu option, the macro reaches an empty cell and quits.

Because the code of the menus as it appears in the main listing may confuse you for keying this macro into lotus 1-2-3, we show here the full list of cell formulas and contents.

```
A1: U [W16] '*---A macro to change multiple column width
A2: [W16] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W16] '    range names in this column (starts with the \Z macro name)
A4: [W16] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W16] '!
A6: [W16] '!
A7: [W16] '!
A8: [W16] '!
A9: [W16] '!
A10: U [W16] '\Z
B10: [W16] '{BREAKON}
A11: U [W16] 'COLWIDTH
B11: [W16] '{BREAKON}
A12: [W16] 'loop004
B12: [W16] '{WINDOWSON}{PANELON}{MENUBRANCH menu004}
A13: [W16] '!
A14: [W16] 'menu004
B14: [W16] 'One_at_a_time
C14: [W11] 'All
D14: 'Quit
A15: [W16] '!
B15: [W16] 'Set the width for every column separately
C15: [W11] 'Set the same width for all columns
D15: 'Quit the macro
A16: [W16] '!
B16: [W16] '{WINDOWSON}{PANELON}{GOTO}{?}~{LET here004,@CELLPOINTER
           ("address")}~
C16: [W11] '{WINDOWSON}{PANELON}{GOTO}{?}~{LET here004,@CELLPOINTER
           ("address")}~
A17: [W16] '!
B17: [W16] '{GETNUMBER "Number of columns to change? ",columns004}~
C17: [W11] '{GETNUMBER "Number of columns to change? ",columns004}~
A18: [W16] '!
B18: [W16] '{FOR counter004,1,columns004,1,colwide1004}
C18: [W11] '{GETLABEL "Column width ? ",colwide3004}~{RECALC colwide2004}
A19: [W16] '!
B19: [W16] '{GOTO}{here004}~{BRANCH loop004}
C19: [W11] '{WINDOWSOFF}{PANELOFF}{FOR counter004,1,columns004,1,
            colwide2004}{GOTO}{here004}~{BRANCH loop004}
A20: [W16] '!
A21: [W16] 'colwide1004
B21: [W16] '/WCS{?}~{RIGHT}
A22: [W16] '!
A23: [W16] 'colwide2004
B23: U [W16] +"/WCS"&B25&"~{RIGHT}"
```

```
A24: [W16] '!
A25: [W16] 'colwide3004
B25: [W16] '9
A26: [W16] 'counter004
B26: [W16] 2
A27: [W16] 'columns004
B27: [W16] 1
A28: [W16] 'here004
```

Copy Macros

[3] Copy Adjacent Cell's Format

```
          A           B     C       D        E          F              G          H
 1 *---A macro to copy the adjacent cell's format
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z             {BREAKON}
11 COPYFRMT       {LET rel098,@INFO("release")}~{MENUBRANCH menu1098}
12 !
13 menu1098       Up  Down  Left  Right  Next_sheet Previous_sheet Quit
14 !              CopyCopy tCopy tCopy thCopy the neCopy the previoQuit t
15 !              /C{BS/C{BS}{D/C{BS}{l/C{BS}{RI{IF @LEFT(R{IF @LEFT(rel0
16 !                                  {MENUBRANCH{MENUBRANCH menu1098}
17 !
18 ret098
19 !
20 rel098
```

This macro allows you to copy the adjacent cell's format. You can copy the upper cell's format, the lower cell's format, the right and left cell's format and, in a 3-D release of Lotus 1-2-3, even the next sheet or the previous sheet adjacent cell's format. The macro uses a custom menu with seven menu options which are shown here, in full, because they cannot be displayed clearly on one page without overlapping. If you plan to key this macro into Lotus 1-2-3, you have to key the code in the menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you, you can find a full list of all the cell contents at the end of this section.

```
Up
Copy the up cell's format
/C{BS}{UP}~~/RE~

Down
Copy the down cell's format
/C{BS}{DOWN}~~/RE~

Left
Copy the left cell's format
/C{BS}{LEFT}~~/RE~

Right
Copy the right cell's format
/C{BS}{RIGHT}~~/RE~

Next_sheet
Copy the next sheet cell's format
{IF @LEFT(rel098,1)<>"@"}/C{BS}{NS}~~/RE~{BRANCH ret098}
{MENUBRANCH menu1098}

Previous_sheet
Copy the previous sheet cell's format
{IF @LEFT(rel098,1)<>"@"}/C{BS}{PS}~~/RE~{BRANCH ret098}
{MENUBRANCH menu1098}

Quit
Quit the macro
{BRANCH ret098}
```

The macro starts with the {LET rel098,@INFO("release")} macro command, which stores the result of the @INFO("release") function into the B20 cell named [rel098]. Later the macro uses this result to check if you are using a 3-D or a 2-D Lotus release. The first four menu options are basically the same, they copy the adjacent cell to the current cell position and then erase the current cell contents leaving the cell's format. Let's analyze the [**U**p] menu option:

```
Up
Copy the up cell's format
/C{BS}{UP}~~/RE~
```

When you choose this option, the macro issues /C macro keys which start the copy process, then the macro issues {BS} which frees the cell pointer movement and the macro issues {UP}. Then it issues the tilde "~" command (which is the same as the ENTER key) twice and copies the above cell's content and the format, to the current cell. Last, the macro issues /RE~, which erases the current cell contents and leaves only the format which is the same as the above cell's format.

The 5th and the 6th menu options again have almost the same code. Let's analyze the [**N**ext_sheet] menu option:

```
Next_sheet
Copy the next sheet cell's format
{IF @LEFT(rel098,1)<>"@"}/C{BS}{NS}~~/RE~{BRANCH ret098}
{MENUBRANCH menu1098}
```

When you choose this menu option, the macro issues {IF @LEFT(rel098,1)<>"@"} to check the result of the @INFO("release") function, which is stored in the B20, [rel098]. If this is true, you are using a 3-D Lotus release. Therefore the macro issues /C{BS}{NS} ~~/RE~ and copies the adjacent cell's contents and erases the contents to leave only the format, as we have seen earlier in the [**U**p] menu option. If this is false, then you are using a 2-D release of Lotus 1-2-3 and the macro issues {MENUBRANCH menu1098} to restart the custom menu.

The last menu option is [**Q**uit]:

```
Quit
Quit the macro
{BRANCH ret098}
```

When you choose this menu option, the macro issues {BRANCH ret098} which routes the macro to the empty [ret098] routine and quits.

To help you key this macro into Lotus 1-2-3, here is the full list of cell contents in this macro.

```
A1: U '*---A macro to copy the adjacent cell's format
A2: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define
A3: '    the range names in this column (starts with the \Z macro name)
A4: '*---Hold the [ALT] key and press [Z] to activate the macro
A5: '!
A6: U '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
A7: U '                    IT WILL WORK IN LOTUS 2.0 AND UP
A8: '!
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'COPYFRMT
```

```
B11: '{LET rel098,@INFO("release")}~{MENUBRANCH menu1098}
A12: '!
A13: 'menu1098
B13: 'Up
C13: 'Down
D13: 'Left
E13: 'Right
F13: 'Next_sheet
G13: 'Previous_sheet
H13: 'Quit
A14: '!
B14: 'Copy the up cell's format
C14: 'Copy the down cell's format
D14: 'Copy the left cell's format
E14: 'Copy the right cell's format
F14: 'Copy the next sheet cell's format
G14: 'Copy the previous sheet cell's format
H14: 'Quit the macro
A15: '!
B15: '/C{BS}{UP}~~/RE~
C15: '/C{BS}{DOWN}~~/RE~
D15: '/C{BS}{LEFT}~~/RE~
E15: '/C{BS}{RIGHT}~~/RE~
F15: '{IF @LEFT(rel098,1)<>"@"}/C{BS}{NS}~~/RE~{BRANCH ret098}
G15: '{IF @LEFT(rel098,1)<>"@"}/C{BS}{PS}~~/RE~{BRANCH ret098}
H15: '{BRANCH ret098}
A16: '!
F16: '{MENUBRANCH menu1098}
G16: '{MENUBRANCH menu1098}
A17: '!
A18: 'ret098
A19: '!
A20: 'rel098
```

[7] The Absolute Copy Macro

	A	B	C	D	E
1	*---A macro to copy any 3-D or 2-D range and keep ABSOLUTE references				
2	even though the range includes relative references/addresses.				
3	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
4	range names in this column (starts with the \Z macro name)				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE				
8	IT WILL WORK IN LOTUS 2.0 AND UP				
9	!				
10	\Z	{BREAKON}			
11	ABS-COPY	{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND Which range ?~/RNC{PANELON}Which range ?~{WINDOWSON} {BS}{BS}{?}~{GOTO}Which range ?~			
12	!	{WINDOWSOFF}{PANELOFF}/RNCWhere to ?~~/RNDWhere to ?~ /RNC{PANELON}Where to ?~{BS}{BS}{WINDOWSON}{?}~ {WINDOWSOFF}{label401}			
13	!	/CWhich range ?~Where to ?~			
14	!	{GOTO}Where to ?~/RNCWhere to ?~{BS}{BS}			
15	!	.{DOWN @ROWS(Which range ?)-1}			
16	!	.{RIGHT @COLS(Which range ?)-1}~			
17	!	{numbera401}{number401}			
18	!				
19	label401	{GOTO}Which range ?~{LET counterb401,0}~			
20	cont401	{LET counter1401,0}{LET hereabs401,@CELLPOINTER ("address")}~			
21	!	{FOR counter1401,0,@COLS(Which range ?)-1,1,labels1401}~			
22	!	{LET rel401,@INFO("release")}~{IF @LEFT(rel401,1)<>"@"} {GOTO}{hereabs401}~{LET counterb401,counterb401+1}~ {IF counterb401<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs401}~{BRANCH cont401}			
23	!	{RETURN}			
24	!				
25	counter1401	2			
26	counter1a401	5			
27	labels1401	{RIGHT}{LET here1401,@CELLPOINTER("address")}~{LEFT} {FOR counter1a401,0,@ROWS(Which range ?)-1,1, labels1a401}~{IF counter1401<@COLS(Which range ?)-1} {GOTO}{here1401}~{LET counter1a401,0}~			
28	!				
29	here1401	$C$1			
30	!				
31	labels1a401	{IF @CELLPOINTER("type")<>"b"}{EDIT}{HOME}'{DOWN} {RETURN}			
32	!	{DOWN}			
33	!				
34	number401	{GOTO}Which range ?~{LET counterb401,0}~			
35	cont1401	{LET counter2401,0}{LET hereabs401,@CELLPOINTER ("address")}~			
36	!	{FOR counter2401,0,@COLS(Which range ?)-1,1,numbers3401}~			
37	!	{LET rel401,@INFO("release")}~{IF @LEFT(rel401,1)<>"@"} {GOTO}{hereabs401}~{LET counterb401,counterb401+1}~ {IF counterb401<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs401}~{BRANCH cont1401}			
38	!	{GOTO}Which range ?~/RNDWhich range ?~/RNDWhere to ?~			
39	!				
40	counter2401	2			
41	counter3401	0			
42	numbers3401	{RIGHT}{LET here2401,@CELLPOINTER("address")}~{LEFT} {FOR counter3401,0,@ROWS(Which range ?)-1,1,numbers1401}~ {GOTO}{here2401}~{LET counter3401,0}~			
43	!				
44	here2401	$C$1			
45	!				
46	numbers1401	{IF @CELLPOINTER("type")<>"b"}{EDIT}{HOME}{DEL}{DOWN} {RETURN}			
47	!	{DOWN}			

```
48 !
49 numbera401      {GOTO}Where to ?~{LET counterb401,0}~
50 cont1a401       {LET counter2a401,0}{LET hereabs401,@CELLPOINTER
                   ("address")}~
51 !               {FOR counter2a401,0,@COLS(Where to ?)-1,1,numbers3a401}~
52 !               {LET rel401,@INFO("release")}~{IF @LEFT(rel401,1)<>"@"}
                   {GOTO}{hereabs401}~{LET counterb401,counterb401+1}~
                   {IF counterb401<@SHEET(Which range ?)}{NS}{GOTO}
                   {hereabs401}~{BRANCH cont1a401}
53 !
54 counter3a401          0
55 counter2a401          2
56 !
57 numbers3a401    {RIGHT}{LET here3401,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter3401,0,@ROWS(Where to ?)-1,1,numbers1a401}~
                   {GOTO}{here3401}~{LET counter3a401,0}~
58 !
59 here3401        $F$1
60 !
61 numbers1a401    {IF @CELLPOINTER("type")<>"b"}{EDIT}{HOME}{DEL}{DOWN}
                   {RETURN}
62 !               {DOWN}
63 !
64 counterb401           3
65 hereabs401      $A$1
66 !
67 rel401
```

This macro solves a common issue that most Lotus 1-2-3 users face. When you copy a cell in the worksheet and the cell contains a formula, the new formula is changed depending on the number of rows and columns shifted. This is a beautiful property of the electronic spreadsheet, but what do we do if we do not want the formulas updated? We can use absolute addresses, but what if we want to copy a whole range and have two identical copies of the range, which contain non-absolute formulas, to a data outside the range? The solution is to turn all the formulas to labels, and then copy the range. Turning a 100X100 range to labels is a real punishment. The ABS-COPY.WK1 macro does the job automatically by first turning all the values in the range into labels, then copying the range to the target location and last UN-LABELING the two ranges.

```
     A                B                C                D                E
11 ABS-COPY        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{PANELON}Which range ?~{WINDOWSON}
                   {BS}{BS}{?}~{GOTO}Which range ?~
12 !               {WINDOWSOFF}{PANELOFF}/RNCWhere to ?~~/RNDWhere to ?~
                   /RNC{PANELON}Where to ?~{BS}{BS}{WINDOWSON}{?}~
                   {WINDOWSOFF}{label401}
13 !               /CWhich range ?~Where to ?~
14 !               {GOTO}Where to ?~/RNCWhere to ?~{BS}{BS}
15 !               .{DOWN @ROWS(Which range ?)-1}
16 !               .{RIGHT @COLS(Which range ?)-1}~
17 !               {numbera401}{number401}
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and the panel display activities. Then it uses the "safe technique" to prompt you to paint the origin range to process, and simultaneously assigns the [Which range ?] name to the same range, and uses it as a prompt. Next the macro repeats the same procedure and prompts you to paint the target range to process, and simultaneously assigns the [Where to ?] name to the target range, and uses it as a prompt. Next the macro issues `{label401}` routine command which turns all the values in [Which range ?] into labels.

```
     A                B                C                D                E
19 label401        {GOTO}Which range ?~{LET counterb401,0}~
```

```
20 cont401        {LET counter1401,0}{LET hereabs401,@CELLPOINTER
                  ("address")}~
21 !              {FOR counter1401,0,@COLS(Which range ?)-1,1,labels1401}~
22 !              {LET rel401,@INFO("release")}~{IF @LEFT(rel401,1)<>"@"}
                  {GOTO}{hereabs401}~{LET counterb401,counterb401+1}~
                  {IF counterb401<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs401}~{BRANCH cont401}
23 !              {RETURN}
```

This routine starts with `{GOTO}Which range ?~` which moves the cell pointer to the upper left cell of the origin range, [Which range ?]. Next the macro issues `{LET counterb401 ,0}~` to set the value in cell [counterb401], which serves as a counter, to zero. Next the macro issues `{LET counter1401,0}~` to set the value in cell [counter1401], which also serves as a counter, to zero. Now the macro issues `{LET hereabs401,@CELLPOINTER ("address")}~` to store the address of the upper left cell of the [Which range ?] range in cell [hereabs401] for later use. The macro will use it to process all the sheets of the [Which range ?] range if it contains more than one sheet. To change all the values in the origin range to labels, the macro issues `{FOR counter1401,0,@COLS(Which range?)-1,1, labels1401}~` which activates the [labels1401] routine as many times as the number of columns in the origin range.

```
              A               B              C              D              E
27 labels1401     {RIGHT}{LET here1401,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a401,0,@ROWS(Which range ?)-1,1,
                  labels1a401}~{IF counter1401<@COLS(Which range ?)-1}
                  {GOTO}{here1401}~{LET counter1a401,0}~
```

This routine uses `{RIGHT}{LET here1401,@CELLPOINTER("address")}~{LEFT}` to record the address of the first cell of the next column in cell [here1401]. When the macro finishes processing the current column, it moves the cell pointer directly to the top of the next column using the address stored in cell [here1401]. The `{FOR counter1a401,0,@ROWS (Which range ?),1,1,labels1a401}~` command activates the [labels1a401] routine as many times as the number of rows in the [Which range ?] range.

```
              A               B              C              D              E
31 labels1a401    {IF @CELLPOINTER("type")<>"b"}{EDIT}{HOME}'{DOWN}
                  {RETURN}
32 !              {DOWN}
```

The [labels1a401] routine issues `{IF @CELLPOINTER("type")<>"b"}` to make sure that the cell does not empty. If so, the routine issues `{EDIT}` to enter into the EDIT mode, and then `{HOME}` to move the cursor to the beginning of the text in the panel. The macro precedes the content of the panel with an apostrophe "`'`" which changes it into a label if the cell content is a value, and adds it to an existing label if the content is a label. The `{DOWN}` command moves the cell pointer to the next cell in the current column and simultaneously writes the content of the panel into the current cell.

The `{RETURN}` command returns the control to the `{FOR}` loop in the [labels1401] routine. Let's assume the cell contains the `@ABS(-10)` formula which appears in the cell as the number 10. After the `{EDIT}{HOME}'{DOWN}` sequence, the cell shows the formula as the "`@ABS(-10)`" string of text. If the cell contains a label, the routine adds an extra apostrophe "`'`" which causes the text in the cell to appear with a leading apostrophe "`'`". If the cell is blank, the macro issues `{DOWN}` which moves the cell pointer to the next cell in the current column and returns the control to the `{FOR}` loop in the [labels1401] routine.

	A	B	C	D	E
27	labels1401	{RIGHT}{LET here1401,@CELLPOINTER("address")}~{LEFT} {FOR counter1a401,0,@ROWS(Which range ?)-1,1, labels1a401}~{IF counter1401<@COLS(Which range ?)-1} {GOTO}{here1401}~{LET counter1a401,0}~			

When the value in cell [counter1a401] reaches the number of rows in the [Which range ?] range minus one, the macro issues {IF counter1401<@COLS(Which range ?)-1} to check how many columns were processed. If the value in cell [counter1401] is less than the number of columns in the [Which range ?] range, the macro issues the indirect {GOTO} {here1401}~ macro command which moves the cell pointer to the first cell of the next column. Next it issues {LET counter1a401,0}~ to reset the value in cell [counter1a401] to zero. When the [labels1401] routine is finished, the macro returns control to the [cont401] routine to process the next column.

	A	B	C	D	E
19	label401	{GOTO}Which range ?~{LET counterb401,0}~			
20	cont401	{LET counter1401,0}{LET hereabs401,@CELLPOINTER ("address")}~			
21	!	{FOR counter1401,0,@COLS(Which range ?)-1,1,labels1401}~			
22	!	{LET rel401,@INFO("release")}~{IF @LEFT(rel401,1)<>"@"} {GOTO}{hereabs401}~{LET counterb401,counterb401+1}~ {IF counterb401<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs401}~{BRANCH cont401}			
23	!	{RETURN}			

When the macro finishes processing all the columns in the first sheet of the [Which range ?] range, (in a 2-D release this is the only sheet) it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues {LET rel401,@INFO("release")}~ which stores the result of the @INFO("release") 3-D function in cell [rel401], and then the macro issues {IF @LEFT(rel401,1)<>"@"}, which checks the first character of the content of cell [rel401]. If it is the "@" character, you are using a 2-D release, otherwise you are using a 3-D release which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the {GOTO}{hereabs401}~ indirect command which moves the cell pointer to the address of the upper left cell of the current sheet in the [Which range ?] origin range.

The macro continues with {LET counterb401,counterb401+1}~ which increases the value in cell [counterb401] by one. Now the macro issues {IF counterb401<@SHEETS Which range ?)} to check if the value in cell [counterb401] is less than the number of sheets in the [Which range ?] range. If so, there are more sheets to process. Therefore the macro issues {NS} to move the cell pointer to the next sheet. Next the macro issues the indirect {GOTO} {hereabs401}~ macro command which moves the cell pointer to the same address as the address of the upper left cell of the current sheet of the [Which range ?] range, but in the new sheet. Last, the macro issues {BRANCH cont401} to process the cells of the [Which range ?] range in the new sheet. When all the sheets of the [Which range ?] range are processed, the [cont401] routine returns control to the main macro.

	A	B	C	D	E
11	ABS-COPY	{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND Which range ?~/RNC{PANELON}Which range ?~{WINDOWSON} {BS}{BS}{?}~{GOTO}Which range ?~			
12	!	{WINDOWSOFF}{PANELOFF}/RNCWhere to ?~~/RNDWhere to ?~ /RNC{PANELON}Where to ?~{BS}{BS}{WINDOWSON}{?}~ {WINDOWSOFF}{label401}			
13	!	/CWhich range ?~Where to ?~			

```
14 !              {GOTO}Where to ?~/RNCWhere to ?~{BS}{BS}
15 !              .{DOWN @ROWS(Which range ?)-1}
16 !              .{RIGHT @COLS(Which range ?)-1}~
17 !              {numbera401}{number401}
```

Now that all the cells in the [Which range ?] range are labels or blanks, the macro issues `/CWhich range ?~Where to ?~` which copies the content of the [Which range ?] range into the [Where to ?] range. Next the macro issues `{GOTO}Where to ?~` which moves the cell pointer to the upper left cell of the [Where to ?] range. Now the macro issues:

```
/RNCWhere to ?~{BS}{BS}.{DOWN @ROWS(Which range ?)-1}.
    {RIGHT @COLS(Which range ?)-1}~
```

to re-adjust the size of the [Where to ?] target range to the same size of the first sheet of the [Which range ?] origin range. Notice how the periods [`.`] in this command moves the cursor inside the painted range, and how the macro uses the result of the `@COLS` and the `@ROWS` functions to determine how many rows to expand the target range down and how many columns to expand the target range to the right. Now we have two equal ranges of labels. Therefore the macro issues the `{numbera401}` routine command to un-label the [Where to ?] target range, and `{number401}` to un-label the [Which range ?] origin range .

```
     A              B                C              D              E
49 numbera401     {GOTO}Where to ?~{LET counterb401,0}~
50 cont1a401      {LET counter2a401,0}{LET hereabs401,@CELLPOINTER
                  ("address")}~
51 !              {FOR counter2a401,0,@COLS(Where to ?)-1,1,numbers3a401}~
52 !              {LET rel401,@INFO("release")}~{IF @LEFT(rel401,1)<>"@"}
                  {GOTO}{hereabs401}~{LET counterb401,counterb401+1}~
                  {IF counterb401<@SHEET(Which range ?)}{NS}{GOTO}
                  {hereabs401}~{BRANCH cont1a401}
```

The `{numbera401}` routine is built the same way as the `{label401}`. We probably could use the `{label401}` to do what the `{numbera401}` does using dynamic string formulas to change the code respectively in order to make the macro shorter but more complex. However, we prefer to assign a separate routine for each task. The code of this routine is identical to the code of the [label401] routine. Some of the counter names are different and the routine which is activated by the `{FOR counter2a401,0,@COLS(Where to ?)-1,1,numbers3a401 }~` command is the [numbers3a401] which has the same function as the [labels1401] routine.

```
     A              B                C              D              E
57 numbers3a401   {RIGHT}{LET here3401,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter3401,0,@ROWS(Where to ?)-1,1,numbers1a401}~
                  {GOTO}{here3401}~{LET counter3a401,0}~
```

The `{FOR counter3401,0,@ROWS(Where to ?)-1,1,numbers1a401}~` macro command activates the [numbers1a401] routine exactly as many times as the number of rows in the [Where to ?] target range.

```
     A              B                C              D              E
61 numbers1a401   {IF @CELLPOINTER("type")<>"b"}{EDIT}{HOME}{DEL}{DOWN}
                  {RETURN}
62 !              {DOWN}
```

The routine issues the `{IF @CELLPOINTER("type")<>"b"}` to make sure that the cell is not blank. If this is true, the routine issues the `{EDIT}` to enter the EDIT mode and then issues the `{HOME}` to move the cursor to the beginning of the text in the panel. Then issues the `{DEL}` to

delete the apostrophe "'" to turn the cell content back to value if it was a value first or deletes the extra apostrophe "'" which precedes the previous label. Therefore the [numbers1a401] routine is the opposite of the [labels1a401] routine. Let us return to the [number401] routine.

	A	B	C	D	E
34	number401	{GOTO}Which range ?~{LET counterb401,0}~			
35	cont1401	{LET counter2401,0}{LET hereabs401,@CELLPOINTER ("address")}~			
36	!	{FOR counter2401,0,@COLS(Which range ?)-1,1,numbers3401}~			
37	!	{LET rel401,@INFO("release")}~{IF @LEFT(rel401,1)<>"@"} {GOTO}{hereabs401}~{LET counterb401,counterb401+1}~ {IF counterb401<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs401}~{BRANCH cont1401}			
38	!	{GOTO}Which range ?~/RNDWhich range ?~/RNDWhere to ?~			

The code of this routine is identical to the code of the [numbera401] routine, although some of the range names are different and the routine, which is activated by the `{FOR counter2401,0,@COLS(Where to ?)-1,1,numbers3401}~` command is the [numbers3401], which has the same function as the [numbers3a401] routine, but it processes the [Which range ?] range instead.

	A	B	C	D	E
42	numbers3401	{RIGHT}{LET here2401,@CELLPOINTER("address")}~{LEFT} {FOR counter3401,0,@ROWS(Which range ?)-1,1,numbers1401}~ {GOTO}{here2401}~{LET counter3401,0}~			

Again the macro issues `{RIGHT}{LET here2401,@CELLPOINTER("address")}~ {LEFT}`, which store the address of the upper cell of the next column in cell [here2401], and `{FOR counter3401,0,@ROWS(Which range ?)-1,1,numbers1401}~` activates [numbers1401] as many times as the number of rows in the [Which range ?] origin range.

	A	B	C	D	E
46	numbers1401	{IF @CELLPOINTER("type")<>"b"}{EDIT}{HOME}{DEL}{DOWN} {RETURN}			
47	!	{DOWN}			

This routine is the same as the [numbers1a401] routine. When the macro finishes with the [Which range ?] origin range it returns to the [number401] routine.

	A	B	C	D	E
34	number401	{GOTO}Which range ?~{LET counterb401,0}~			
35	cont1401	{LET counter2401,0}{LET hereabs401,@CELLPOINTER ("address")}~			
36	!	{FOR counter2401,0,@COLS(Which range ?)-1,1,numbers3401}~			
37	!	{LET rel401,@INFO("release")}~{IF @LEFT(rel401,1)<>"@"} {GOTO}{hereabs401}~{LET counterb401,counterb401+1}~ {IF counterb401<@SHEETS(Which range ?)}{NS}{GOTO} {hereabs401}~{BRANCH cont1401}			
38	!	{GOTO}Which range ?~/RNDWhich range ?~/RNDWhere to ?~			

When all the sheets of both ranges are processed, the macro issues `{GOTO}Which range ?` which moves the cell pointer to the upper left cell of the origin range, and then `/RNDWhich range ?~/RNDWhere to ?~` deletes the [Which range ?] and the [Where to ?] temporary range names to leave a clean worksheet.

[6] Copy a Cell Every Other Specified No. of Rows or Columns

```
     A                B                C                D
 1 *---A macro to COPY a cell every other specified rows or columns
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 COPYALT        {MENUBRANCH menu097}
12 !
13 menu097        Down                Right               Quit
14 !              Skip specified rows anSkip specified columnQuit macro
15 !              {GETLABEL "How many ro{GETLABEL "How many columns to sk
16 !              {RECALC don097}{RECALC{RECALC rght097}{RECALC rght1097}
17 !              {MENUBRANCH menu097}  {MENUBRANCH menu097}
18 !
19 don097         {WINDOWSOFF}/C~{DOWN 1}{DOWN}~
20 don1097        {DOWN 1}{DOWN}{WINDOWSON}
21 !              Press [ENTER] to continue, other key to quit to menu
22 !              {GET key097}{ESC}~{IF key097="~"}{don097}
23 !
24 rght097        {WINDOWSOFF}/C~{RIGHT 1}{RIGHT}~
25 rght1097       {RIGHT 1}{RIGHT}{WINDOWSON}
26 !              Press [ENTER] to continue, other key to quit to menu
27 !              {GET key097}{ESC}~{IF key097="~"}{rght097}
28 !
29 !
30 skip097        100
31 !
32 key097         Q
```

This macro allows you to copy a cell to another cell address, a specified number of rows apart or a specified number of columns apart. For example, you can copy the current cell contents into a cell 50 rows down or 101 columns to the right. The macro uses a custom menu with three menu options. Because there is no way to display the custom menu's code on a page without overlapping, we show here every menu option's code separately.

```
Down
Skip specified rows and copy
{GETLABEL "How many rows to skip ? ",skip097}~
{RECALC don097}{RECALC don1097}{don097}
{MENUBRANCH menu097}


Right
Skip specified columns and copy
{GETLABEL "How many columns to skip ? ",skip097}~
{RECALC rght097}{RECALC rght1097}{rght097}
{MENUBRANCH menu097}


Quit
Quit macro
```

Also notice that the code in the B19, B20, B24 and the B25 cells is the result of the following dynamic string formulas.

```
19 don097         +"{WINDOWSOFF}/C~{DOWN "&B30&"}{DOWN}~"
20 don1097        +"{DOWN "&B30&"}{DOWN}{WINDOWSON}"
24 rght097        +"{WINDOWSOFF}/C~{RIGHT "&B30&"}{RIGHT}~"
```

```
25 rght1097       +"{RIGHT "&B30&"}{RIGHT}{WINDOWSON}"
```

If you intend to key the code of this macro into Lotus 1-2-3, you have to key the code of the formulas and the code of the custom menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you, we show the full cell contents list at the end of this section. The macro starts with the {MENUBRANCH menu097} macro command which starts the [menu097] custom menu. The first menu option is [**D**own]:

```
Down
Skip specified rows and copy
{GETLABEL "How many rows to skip ? ",skip097}~
{RECALC don097}{RECALC don1097}{don097}
{MENUBRANCH menu097}
```

When you choose this menu option, the macro issues {GETLABEL "How many rows to skip ? ",skip097} which displays "How many rows to skip ? " in the panel. When you insert a number and press the ENTER key, Lotus stores the number in the B30 cell named [skip097]. Then the macro issues {RECALC don097}{RECALC don1097} which recalculate the dynamic string formulas in the B19 and the B20 cells named [don097] and [don1097] respectively, and then the macro issues {don097} routine command which starts the [don097] routine.

```
     A              B                                                    C                    D
19 don097         {WINDOWSOFF}/C~{DOWN 100}{DOWN}~
20 don1097        {DOWN 100}{DOWN}{WINDOWSON}
21 !              Press [ENTER] to continue, other keys to quit to menu
22 !              {GET key097}{ESC}~{IF key097="~"}{don097}
```

This [don097] routine starts with the {WINDOWSOFF} macro command which freezes the screen display activity and continues with /C~{DOWN 100}{DOWN}~ which copies the current cell contents to a cell 101 rows below, if you inserted 100. Recall that the code in cell [don097] and the cell [don1097] is the result of the following two formulas:

```
19 don097         +"{WINDOWSOFF}/C~{DOWN "&B30&"}{DOWN}~"
20 don1097        +"{DOWN "&B30&"}{DOWN}{WINDOWSON}"
```

Both formulas use the value stored in the B30 cell (which contains the number of rows to skip) to create the correct code of the macro. Next the macro issues {DOWN 100}{DOWN} code which moves the cell pointer to the cell 101 rows below and then issues {WINDOWSON} to resume the screen display activity so you can see the result of the copy process.

Note: when the display activity of the screen is suspended, the cell pointer moves much faster down and speeds the macro process. There is another approach which is much faster, we can calculate the address of the cell 101 rows down and then use the GOTO command. To see how it is done for a more complex macro, see the TRANSPO2.WK1 macro.

To allow you to create more copies the macro writes

```
Press [ENTER] to continue, other key to quit to menu
```

directly to the panel and then issues {GET key097} which halts the macro execution and waits until you respond. When you press the ENTER key, the macro issues {ESC} which clears the message from the panel before Lotus writes the content of the panel into the current cell. Next the macro issues the tilde "~" to finish the previous copy command and then {IF

`key097="~"}` to check if you pressed the ENTER key. If so, the macro issues `{don097}` which starts the [don097] routine again to allow you to make another copy. When this routine is finished, the macro issues `{MENUBRANCH menu097}` which re-activates the [menu097] custom menu. The second menu option is [**R**ight]:

```
Right
Skip specified columns and copy
{GETLABEL "How many columns to skip ? ",skip097}~
{RECALC rght097}{RECALC rght1097}{rght097}
{MENUBRANCH menu097}
```

The code of this menu option is identical to the code of the of the previous menu option. The only difference is that the `{RIGHT}` command replaces the `{DOWN}` command. The last menu option is [**Q**uit]:

```
Quit
Quit macro
```

When you choose this menu option the macro reaches an empty cell and quits.

To help you to key this macro into Lotus 1-2-3 here is the full list of all the cell contents.

```
A1: U [W10] '*---A macro to COPY a cell every other specified rows or columns
A2: [W10] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A3: [W10] '    range names in this column (starts with the \Z macro name)
A4: [W10] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W10] '!
A6: [W10] '!
A7: [W10] '!
A8: [W10] '!
A9: [W10] '!
A10: U [W10] '\Z
B10: [W9] '{BREAKON}
A11: U [W10] 'COPYALT
B11: [W9] '{MENUBRANCH menu097}
A12: [W10] '!
A13: [W10] 'menu097
B13: [W9] 'Down
C13: 'Right
D13: 'Quit
A14: [W10] '!
B14: [W9] 'Skip specified rows and copy
C14: 'Skip specified columns and copy
D14: 'Quit macro
A15: [W10] '!
B15: [W9] '{GETLABEL "How many rows to skip ? ",skip097}~
C15: '{GETLABEL "How many columns to skip ? ",skip097}~
A16: [W10] '!
B16: [W9] '{RECALC don097}{RECALC don1097}{don097}
C16: '{RECALC rght097}{RECALC rght1097}{rght097}
A17: [W10] '!
B17: [W9] '{MENUBRANCH menu097}
C17: '{MENUBRANCH menu097}
A18: [W10] '!
A19: [W10] 'don097
B19: U [W9] +"{WINDOWSOFF}/C~{DOWN "&B30&"}{DOWN}~"
A20: [W10] 'don1097
B20: U [W9] +"{DOWN "&B30&"}{DOWN}{WINDOWSON}"
A21: [W10] '!
B21: [W9] 'Press [ENTER] to continue, other key to quit to menu
A22: [W10] '!
B22: [W9] '{GET key097}{ESC}~{IF key097="~"}{don097}
A23: [W10] '!
A24: [W10] 'rght097
B24: U [W9] +"{WINDOWSOFF}/C~{RIGHT "&B30&"}{RIGHT}~"
```

```
A25: [W10] 'rght1097
B25: U [W9] +"{RIGHT "&B30&"}{RIGHT}{WINDOWSON}"
A26: [W10] '!
B26: [W9] 'Press [ENTER] to continue, other key to quit to menu
A27: [W10] '!
B27: [W9] '{GET key097}{ESC}~{IF key097="~"}{rght097}
A28: [W10] '!
A29: [W10] '!
A30: [W10] 'skip097
B30: [W9] '1
A31: [W10] '!
A32: [W10] 'key097
B32: [W9] 'Q
```

[6] Copy a Range Every Other Specified No. of Rows or Columns

```
     A                    B                    C                    D
 1 *---A macro to create multiple copies of a range with specified
 2     number of rows or columns apart
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 COPYMULT        {MENUBRANCH menu099}
12 !
13 menu099         Down                 Right                Quit
14 !               Skip specified rows Skip specified columQuit macro
15 !               {WINDOWSOFF}{panelof{WINDOWSOFF}{panelof/RNCRange to co
16 !               {GETLABEL "How many {GETLABEL "How many empty columns b
17 !               {GETLABEL "How many {GETLABEL "How many copies ? ",skip
18 !               {LET num1099,@STRING{LET num1099,@STRING((@VALUE(skip09
19 !               {LET num2099,@STRING{LET num2099,@STRING((@VALUE(skip09
20 !               {LET num3099,@STRING{LET num3099,@STRING(@ROWS(Range to
21 !               {RECALC skip099}{REC{RECALC skip099}{RECALC skip1099}{R
22 !               {WINDOWSON}{MENUBRAN{WINDOWSON}{MENUBRANCH menu099}
23 !
24 don099          {WINDOWSOFF}/C.{DOWN 14}{RIGHT 1}~{DOWN 5}~
25 !
26 rght099         {WINDOWSOFF}/C.{RIGHT 14}{DOWN 1}~{RIGHT 5}~
27 !
28 skip099         3
29 !
30 skip1099        3
31 !
32 key099          Q
33 !
34 num1099         14
35 !
36 num2099         5
37 !
38 num3099         1
```

This macro allows you to create multiple copies of a selected range with a specified number of rows/columns apart. For example, you can copy a selected range 23 times 9 rows apart from each other. This macro uses a custom menu which cannot be displayed clearly on one page without overlapping. Therefore, we show here the code of each menu item separately:

```
Down
Skip specified rows and copy
{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~/RNDRange to copy ?~
    /RNC{WINDOWSON}{PANELON}Range to copy ?~{BS}{BS}{?}~
    {WINDOWSOFF}{GOTO}Range to copy ?~
{GETLABEL "How many empty rows between ranges ? ",skip099}~
{GETLABEL "How many copies ? ",skip1099}~
{LET num1099,@STRING((@VALUE(skip099)+@ROWS(Range to copy ?))*
    (@VALUE(skip1099))-1,0)}~
{LET num2099,@STRING((@VALUE(skip099)+@ROWS(Range to copy ?)),0)}~
{LET num3099,@STRING(@COLS(Range to copy ?)-1,0)}~
{RECALC skip099}{RECALC skip1099}{RECALC num1099}{RECALC num2099}
    {RECALC num3099}{RECALC don099}{don099}
{WINDOWSON}{MENUBRANCH menu099}

Right
Skip specified columns and copy
{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~/RNDRange to copy ?~
```

```
    /RNC{WINDOWSON}{PANELON}Range to copy ?~{BS}{BS}{?}~{WINDOWSOFF}
    {GOTO}Range to copy ?~
{GETLABEL "How many empty columns between ranges ? ",skip099}~
{GETLABEL "How many copies ? ",skip1099}~
{LET num1099,@STRING((@VALUE(skip099)+@COLS(Range to copy ?))*
    (@VALUE(skip1099))-1,0)}~
{LET num2099,@STRING((@VALUE(skip099)+@COLS(Range to copy ?)),0)}~
{LET num3099,@STRING(@ROWS(Range to copy ?)-1,0)}~
{RECALC skip099}{RECALC skip1099}{RECALC num1099}{RECALC num2099}
    {RECALC num3099}{RECALC rght099}{rght099}
{WINDOWSON}{MENUBRANCH menu099}

Quit
Quit macro
/RNCRange to copy ?~~/RNDRange to copy ?~
```

The macro also use dynamic string formulas to create the correct code for the macro. The code in the B24 and the B25 cells is the result of the following two formulas:

```
24 don099         +"{WINDOWSOFF}/C.{DOWN "&B34&"}{RIGHT "&B38&"}~
                  {DOWN "&B36&"}~"
26 rght099        +"{WINDOWSOFF}/C.{RIGHT "&B34&"}{DOWN "&B38&"}~
                  {RIGHT "&B36&"}~"
```

If you intend to key this macro into Lotus 1-2-3, you have to key the formulas and the code of menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you to correctly key this macro, we show the full list of all the cell contents at the end of this section. The macro starts with the `{MENUBRANCH menu099}` macro command which starts the [menu099] custom menu. The first menu option is [**D**own]:

```
Down
Skip specified rows and copy
{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~/RNDRange to copy ?~
    /RNC{WINDOWSON}{PANELON}Range to copy ?~{BS}{BS}{?}~
    {WINDOWSOFF}{GOTO}Range to copy ?~
{GETLABEL "How many empty rows between ranges ? ",skip099}~
{GETLABEL "How many copies ? ",skip1099}~
{LET num1099,@STRING((@VALUE(skip099)+@ROWS(Range to copy ?))*
    (@VALUE(skip1099))-1,0)}~
{LET num2099,@STRING((@VALUE(skip099)+@ROWS(Range to copy ?)),0)}~
{LET num3099,@STRING(@COLS(Range to copy ?)-1,0)}~
{RECALC skip099}{RECALC skip1099}{RECALC num1099}{RECALC num2099}
    {RECALC num3099}{RECALC don099}{don099}
{WINDOWSON}{MENUBRANCH menu099}
```

When you choose this menu option, the macro issues `{WINDOWSOFF}{PANELOFF}` which freeze the screen and the panel display activities, and then uses the "safe technique" to assign the [Range to copy ?] name to the selected range, while simultaneously uses the [Range to copy ?] range name as a prompt. Next the macro issues `{GETLABEL "How many empty rows between ranges ? ",skip099}` which displays "`How many empty rows between ranges ? `" in the panel and waits until you respond. When you press the ENTER key, Lotus stores the response in cell [skip099]. Now the macro issues a second `{GETLABEL "How many copies ? ",skip1099}` which displays "`How many copies ? `" in the panel. Again, when you press the ENTER key, Lotus stores the response in cell [skip1099].

Now the macro starts a series of advanced `{LET}` macro commands to calculate the number of rows and columns needed to do the job. The first is

```
{LET num1099,@STRING((@VALUE(skip099)+@ROWS(Range to copy ?))*
    (@VALUE(skip1099))-1,0)}~
```

which stores the result of

```
@STRING((@VALUE(skip099)+@ROWS(Range to copy ?))*(@VALUE
    (skip1099))-1,0)
```

in cell [num1099]. This formula returns the total numbers or rows that occupy the copied ranges and the empty rows between them minus one. The second is

```
{LET num2099,@STRING((@VALUE(skip099)+@ROWS(Range to copy ?)),0)}~
```

which stores the result of

```
@STRING((@VALUE(skip099)+@ROWS(Range to copy ?)),0)
```

in cell [num2099]. This formula returns the sum of the rows to skip plus the number of rows in the range that you want to copy. The third is

```
{LET num3099,@STRING(@COLS(Range to copy ?)-1,0)}~
```

which stores the number of columns in the selected range minus one in [num3099]. Next the macro issues `{RECALC skip099}{RECALC skip1099}{RECALC num1099}{RECALC num2099}{RECALC num3099}{RECALC don099}` to update all the stored values and the formulas in the worksheet and then issues the `{don099}` routine command which starts the [don099] routine:

	A	B	C	D
24	don099	{WINDOWSOFF}/C.{DOWN 14}{RIGHT 1}~{DOWN 5}~		

The [don099] routine starts with `{WINDOWSOFF}` which freezes the screen display activity and then issues `/C.{DOWN 14}{RIGHT 1}~{DOWN 5}~` to create three copies of the range. The code here is the result of the

```
+"{WINDOWSOFF}/C.{DOWN "&B34&"}{RIGHT "&B38&"}~{DOWN "&B36&"}~"
```

string formula. The formula uses the stored values in the B34, B38 and the B36 cells to create the code. The second menu option uses an identical code except that the `{RIGHT}` macro command replaces the `{DOWN}` macro command in the previous code. The last menu option is [**Q**uit]:

```
Quit
Quit macro
```

When you choose this menu option, the macro reaches an empty cell and quits.

To help you key this macro into Lotus 1-2-3, here is the full list of all the cell contents.

```
A1: U [W15] '*---A macro to create multiple copies of a range with
            specified
A2: U [W15] '    number of rows or columns apart
A3: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A4: [W15] '    range names in this column (starts with the \Z macro name)
A5: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A6: [W15] '!
A7: [W15] '!
A8: [W15] '!
A9: [W15] '!
```

```
A10: U [W15] '\Z
B10: [W20] '{BREAKON}
A11: U [W15] 'COPYMULT
B11: [W20] '{MENUBRANCH menu099}
A12: [W15] '!
A13: [W15] 'menu099
B13: [W20] 'Down
C13: [W20] 'Right
D13: 'Quit
A14: [W15] '!
B14: [W20] 'Skip specified rows and copy
C14: [W20] 'Skip specified columns and copy
D14: 'Quit macro
A15: [W15] '!
B15: [W20] '{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~/RNDRange to copy ?~
           /RNC{WINDOWSON}{PANELON}Range to copy ?~{BS}{BS}{?}~{WINDOWSOFF}
           {GOTO}Range to copy ?~
C15: [W20] '{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~/RNDRange to copy ?~
           /RNC{WINDOWSON}{PANELON}Range to copy ?~{BS}{BS}{?}~{WINDOWSOFF}
           {GOTO}Range to copy ?~
D15: '/RNCRange to copy ?~~/RNDRange to copy ?~
A16: [W15] '!
B16: [W20] '{GETLABEL "How many empty rows between ranges ? ",skip099}~
C16: [W20] '{GETLABEL "How many empty columns between ranges ? ",skip099}~
A17: [W15] '!
B17: [W20] '{GETLABEL "How many copies ? ",skip1099}~
C17: [W20] '{GETLABEL "How many copies ? ",skip1099}~
A18: [W15] '!
B18: [W20] '{LET num1099,@STRING((@VALUE(skip099)+@ROWS(Range to copy ?))
           *(@VALUE(skip1099))-1,0)}~
C18: [W20] '{LET num1099,@STRING((@VALUE(skip099)+@COLS(Range to copy ?))
           *(@VALUE(skip1099))-1,0)}~
A19: [W15] '!
B19: [W20] '{LET num2099,@STRING((@VALUE(skip099)+@ROWS(Range to copy ?)),
           0)}~
C19: [W20] '{LET num2099,@STRING((@VALUE(skip099)+@COLS(Range to copy ?)),
           0)}~
A20: [W15] '!
B20: [W20] '{LET num3099,@STRING(@COLS(Range to copy ?)-1,0)}~
C20: [W20] '{LET num3099,@STRING(@ROWS(Range to copy ?)-1,0)}~
A21: [W15] '!
B21: [W20] '{RECALC skip099}{RECALC skip1099}{RECALC num1099}
           {RECALC num2099}{RECALC num3099}{RECALC don099}{don099}
C21: [W20] '{RECALC skip099}{RECALC skip1099}{RECALC num1099}
           {RECALC num2099}{RECALC num3099}{RECALC rght099}{rght099}
A22: [W15] '!
B22: [W20] '{WINDOWSON}{MENUBRANCH menu099}
C22: [W20] '{WINDOWSON}{MENUBRANCH menu099}
A23: [W15] '!
A24: [W15] 'don099
B24: U [W20] +"{WINDOWSOFF}/C.{DOWN "&B34&"}{RIGHT "&B38&"}~{DOWN "&B36&"}~"
A25: [W15] '!
A26: [W15] 'rght099
B26: U [W20] +"{WINDOWSOFF}/C.{RIGHT "&B34&"}{DOWN "&B38&"}~{RIGHT "&B36&"}~"
A27: [W15] '!
A28: [W15] 'skip099
B28: [W20] '3
A29: [W15] '!
A30: [W15] 'skip1099
B30: [W20] '3
A31: [W15] '!
A32: [W15] 'key099
B32: [W20] 'Q
A33: [W15] '!
A34: [W15] 'num1099
B34: [W20] '14
A35: [W15] '!
A36: [W15] 'num2099
B36: [W20] '5
A37: [W15] '!
A38: [W15] 'num3099
```

B38: [W20] '1

[6] Special Copy Options

```
     A                    B                    C          D          E          F
 1 *---A macro with standard and extra SPECIAL COPY options :
 2     1) Standard copy  2) Copy and move to target cell  3) Copy from
 3     there to here  4) Repeat last copy
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 COPYSPCL       {LET flag1102,0}{LET flag2102,0}~{MENUBRANCH menu102}
12 !
13 menu102        Standard_Copy  Copy_&_go  From_copy  Repeat  Quit
14 !              Standard 1-2-3 Copy and moCopy from oRepeat LQuit the m
15 !              {LET flag1102,1{LET flag11Move the ce{LET fla{IF flag11
16 !              /Crange to copy/Crange to {IF key102={GOTO}wh{IF flag21
17 !              {MENUBRANCH men{MENUBRANCH{key102}{?}{menubra{LET flag1
18 !                                  {MENUBRANCH menu102}
19 key102         ~
20 flag1102                  0
21 flag2102                  0
```

This macro adds a few copy option that are not available using the 1-2-3 keyboard. The macro is built around a custom menu with four copy options; however, because the code of this menu cannot be displayed clearly on one page without overlapping, we show here the code of each menu item separately:

```
Standard_Copy
Standard 1-2-3 Copy
{LET flag1102,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~
    /RNDRange to copy ?~/RNC{PANELON}Range to copy ?~{BS}{BS}
    {WINDOWSON}{?}~
/CRange to copy ?~{?}~
{MENUBRANCH menu102}

Copy_&_go
Copy and move the cell pointer to the target cell
{LET flag1102,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~
    /RNDRange to copy ?~/RNC{PANELON}Range to copy ?~{BS}{BS}
    {WINDOWSON}{?}~{GOTO}{?}~~
/CRange to copy ?~~
{MENUBRANCH menu102}

From_copy
Copy from other place to here
Move the cell pointer to the target location and press [ENTER]
    {GET key102}{ESC}~
{IF key102="~"}{LET flag1102,1}{WINDOWSOFF}{PANELOFF}
    /RNCRange to copy ?~~/RNDRange to copy ?~/RNC{PANELON}
    Range to copy ?~{BS}{BS}{WINDOWSON}{?}~/CRange to copy ?~~
    {MENUBRANCH menu102}
{key102}{?}~{LET flag1102,1}{WINDOWSOFF}{PANELOFF}/RNC
    Range to copy ?~~/RNDRange to copy ?~/RNC{PANELON}
    Range to copy ?~{BS}{BS}{WINDOWSON}{?}~/CRange to copy ?~~
{MENUBRANCH menu102}

Repeat
Repeat Last Copy at the Current Cell.
{LET flag1102,1}{LET flag2102,1}{WINDOWSOFF}{PANELOFF}/RNC
    Where to ?~~/RNDWhere to ?~/RNC{PANELON}Where to ?~{BS}{BS}
    {WINDOWSON}{?}~
{GOTO}Where to ?~/CRange to copy ?~~
{MENUBRANCH menu102}
```

```
Quit
Quit the macro
{IF flag1102=1}/RNDRange to copy ?~
{IF flag2102=1}/RNDWHERE to ?~
{LET flag1102,0}{LET flag2102,0}~
```

If you intend to key the code of this macro into Lotus 1-2-3, you have to key the code of the custom menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you, we show complete content list at the end of this section. The macro starts with the `{LET flag1102,0}{LET flag2102,0}~` macro commands which set the values in cell [flag1102] and cell [flag2102] to zero. Then the macro issues `{MENUBRANCH menu102}` which starts the [menu102] custom menu. The first option is [**S**tandard_Copy]:

```
Standard_Copy
Standard 1-2-3 Copy
{LET flag1102,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~
    /RNDRange to copy ?~/RNC{PANELON}Range to copy ?~{BS}{BS}
    {WINDOWSON}{?}~
/CRange to copy ?~{?}~
{MENUBRANCH menu102}
```

This menu option does the same thing as the Lotus standard copy option does, but here the macro adds an extra prompt. When you choose this option, the macro issues `{LET flag1102,1}` which stores the number "`1`" in cell [flag1102]. Then the macro uses the "safe technique" to assign the [Range to copy ?] name to the source range and simultaneously uses the range name as a prompt. When you paint the source range and press the ENTER key, the macro issues `/CRange to copy ?~`, which starts the copy process of the [Range to copy ?] source range, and then issues `{?}`, which halts the macro and waits until you point to the target range and press ENTER.

When you point to the target range and press ENTER, the macro issues the tilde "`~`", which is the macro equivalent of ENTER, and finishes the copy process. Next the macro issues `{MENUBRANCH menu102}` which restarts the custom menu to allow you to continue to copy. The second menu option is [**C**opy_&_go]:

```
Copy_&_go
Copy and move the cell pointer to the target cell
{LET flag1102,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~
    /RNDRange to copy ?~/RNC{PANELON}Range to copy ?~{BS}{BS}
    {WINDOWSON}{?}~{GOTO}{?}~~
/CRange to copy ?~~
{MENUBRANCH menu102}
```

This copy option is different from the previous one in the sense that, when the copy process is finished, the cell pointer is on the target location instead of the origin location. When you choose this option, the macro issues `{LET flag1102,1}` which stores the number "`1`" in cell [flag1102]. Then the macro uses the "safe technique" to assign the [Range to copy ?] name to the source range and simultaneously uses the range name as a prompt. When you paint the source range and press the ENTER key, the macro issues `{GOTO}` which starts the GOTO process and then issues `{?}` which halts the macro and waits until you point to the target location, and press ENTER.

When you press ENTER, the macro issues the tilde "`~`" and moves the cell pointer to the target location. Now the macro continues with `/CRange to copy ?~~` which copies the [Range to

copy ?] source range to the current location. Next the macro issues {MENUBRANCH menu102} which restarts the custom menu to allow you to continue to copy. The third menu option is [**F**rom_copy]:

```
From_copy
Copy from other place to here
Move the cell pointer to the target location and press [ENTER]
    {GET key102}{ESC}~
{IF key102="~"}{LET flag1102,1}{WINDOWSOFF}{PANELOFF}
    /RNCRange to copy ?~~/RNDRange to copy ?~/RNC{PANELON}
    Range to copy ?~{BS}{BS}{WINDOWSON}{?}~/CRange to copy ?~~
    {MENUBRANCH menu102}
{key102}{?}~{LET flag1102,1}{WINDOWSOFF}{PANELOFF}/RNC
    Range to copy ?~~/RNDRange to copy ?~/RNC{PANELON}
    Range to copy ?~{BS}{BS}{WINDOWSON}{?}~/CRange to copy ?~~
{MENUBRANCH menu102}
```

This menu option allows you to copy a range from a distanced location into the current location. When you choose this option, the macro writes

```
Move the cell pointer to the target location and press [ENTER]
```

directly into the panel and then issues {GET key102} which halts the macro execution and waits until you press any key. When you press a key, Lotus stores the key in cell [key102] and then the macro issues {ESC} which immediately clears the panel from the prompt message before Lotus 1-2-3 writes the message into the current cell. In Lotus 1-2-3 2.2 and up the {CE} or the {CLEARENTRY} macro command can also be used, however to make this macro compatible with earlier releases of Lotus 1-2-3, we prefer this technique. Next the macro uses {IF key102="~"} to check if you pressed only the ENTER key. If so, the macro issues {LET flag1102,1} which stores the number "1" in cell [flag1102], and then issues {WINDOWSOFF} {PANELOFF} to freeze the screen and the panel display activities.

Now the macro uses the "safe technique" to assign the [Range to copy ?] range name to the source range and simultaneously uses the range name as a prompt. When you paint the source range and press ENTER, the macro issues /CRange to copy ?~~ which copies the [Range to copy ?] source range to the current location. Next the macro issues {MENUBRANCH menu102} which restarts the custom menu to allow you to continue to copy. If the condition {IF key102="~"} is false, the macro issues the {key102} routine command which executes the macro commands in the [key102] routine. This routine holds only the key that you pressed, therefore {key102} executes this key exactly as it is executed from the keyboard. For example: if you press the DOWN key, Lotus stores the {DOWN} command in cell [key102], and when the macro issues {key102}, Lotus moves the cell pointer one cell down.

Next the macro issues {?} which halts the macro execution and allows you to use any Lotus command from the keyboard. However you are expected only to move the cell pointer to the target location and to press the ENTER key. When you press ENTER, the macro issues {LET flag1102,1} which stores the number "1" in cell [flag1102], and then issues {WINDOWSOFF} {PANELOFF} to freeze the screen and the panel display activities. Now the macro uses the "safe technique" to assign the [Range to copy ?] name to the source range and simultaneously uses the range name as a prompt. When you paint the source range and press ENTER, the macro continues with /CRange to copy ?~~ which copies the [Range to copy ?] source range to the current location. Next the macro issues {MENUBRANCH menu102} which restarts the custom menu to allow you to continue to copy.

Note: In this menu option, we have used limited control of the keys you use, and we have checked if you pressed the ENTER key only. For full control of the keys, you can use in a working macro, see **Monitoring and Controlling User's Input**.

The fourth menu option is [**R**epeat]:

```
Repeat
Repeat Last Copy at the Current Cell.
{LET flag1102,1}{LET flag2102,1}{WINDOWSOFF}{PANELOFF}/RNC
    Where to ?~~/RNDWhere to ?~/RNC{PANELON}Where to ?~{BS}{BS}
    {WINDOWSON}{?}~
{GOTO}Where to ?~/CRange to copy ?~~
{MENUBRANCH menu102}
```

When you choose this option, the macro lets you repeat the copy process. Because the source range is already defined if one of the first two menu options has been used. The macro issues `{LET flag1102,1}{LET flag2102,1}` which set the two flags, ([flag1102] and [flag2102]), this will be explained in the [**Q**uit] menu option. Next the macro uses the "safe technique" as we have seen earlier to assign the [Where to ?] range name to the target range and simultaneously uses the [Where to ?] range name as a prompt in the panel. When you point to the target location and press ENTER, the macro issues `{GOTO}Where to ?~` which moves the cell pointer to the upper left cell of the [Where to ?] range, and than the macro continues with `/CRange to copy ?~~` which copies the [Range to copy ?] source range to the current location. Next the macro issues `{MENUBRANCH menu102}` which restarts the custom menu to allow you to continue to copy. The last menu option is [**Q**uit]:

```
Quit
Quit the macro
{IF flag1102=1}/RNDRange to copy ?~
{IF flag2102=1}/RNDWHERE to ?~
{LET flag1102,0}{LET flag2102,0}~
```

When you choose this option, the macro issues `{IF flag1102=1}` which actually checks if the [Range to copy ?] range name exists. If so, the macro issues `/RNDRange to copy ?~` to delete the temporary range name. Next the macro issues `{IF flag2102=1}` which actually checks if the [Where to ?] range names exists, if so, the macro issues `/RNDWhere to ?~` to delete the temporary range name, leaving a clean worksheet. Next the macro issues `{LET flag1102,0}{LET flag2102,0}~` which stores the number "`0`" in cell [flag1102] and cell [flag2102] and quits because it reaches an empty cell. To help you to key this macro into Lotus 1-2-3, here is the full list of all the cell contents.

```
A1: U [W15] '*---A macro with standard and extra SPECIAL COPY options :
A2: U [W15] '    1) Standard copy  2) Copy and move to target cell
          3) Copy from
A3: U [W15] '    there to here  4) Repeat last copy
A4: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A5: [W15] '    range names in this column (starts with the \Z macro name)
A6: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A7: [W15] '!
A8: [W15] '!
A9: [W15] '!
A10: U [W15] '\Z
B10: [W15] '{BREAKON}
A11: U [W15] 'COPYSPCL
B11: [W15] '{LET flag1102,0}{LET flag2102,0}~{MENUBRANCH menu102}
A12: [W15] '!
```

```
A13: [W15] 'menu102
B13: [W15] 'Standard_Copy
C13: [W11] 'Copy_&_go
D13: [W11] 'From_copy
E13: [W8] 'Repeat
F13: 'Quit
A14: [W15] '!
B14: [W15] 'Standard 1-2-3 Copy
C14: [W11] 'Copy and move the cell pointer to the target cell
D14: [W11] 'Copy from other place to here
E14: [W8] 'Repeat Last Copy at the Current Cell.
F14: 'Quit the macro
A15: [W15] '!
B15: [W15] '{LET flag1102,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~
           /RNDRange to copy ?~/RNC{PANELON}Range to copy ?~{BS}{BS}{WINDOWSON}{?}~
C15: [W11] '{LET flag1102,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to copy ?~~
           /RNDRange to copy ?~/RNC{PANELON}Range to copy ?~{BS}{BS}{WINDOWSON}{?}~
           {GOTO}{?}~~
D15: [W11] 'Move the cell pointer to the target location and press [ENTER]
           {GET key102}
           {ESC}~
E15: [W8] '{LET flag1102,1}{LET flag2102,1}{WINDOWSOFF}{PANELOFF}/RNC
          Where to ?~~/RNDWhere to ?~/RNC{PANELON}Where to ?~{BS}{BS}
          {WINDOWSON}{?}~
F15: '{IF flag1102=1}/RNDRange to copy ?~
A16: [W15] '!
B16: [W15] '/CRange to copy ?~{?}~
C16: [W11] '/CRange to copy ?~~
D16: [W11] '{IF key102="~"}{LET flag1102,1}{WINDOWSOFF}{PANELOFF}/RNC
           Range to copy ?~~/RNDRange to copy ?~/RNC{PANELON}
           Range to copy ?~{BS}{BS}{WINDOWSON}{?}~/CRange to copy ?~~
           {MENUBRANCH menu102}
E16: [W8] '{GOTO}Where to ?~/CRange to copy ?~~
F16: '{IF flag2102=1}/RNDWHERE to ?~
A17: [W15] '!
B17: [W15] '{MENUBRANCH menu102}
C17: [W11] '{MENUBRANCH menu102}
D17: [W11] '{key102}{?}~{LET flag1102,1}{WINDOWSOFF}{PANELOFF}/RNC
           Range to copy ?~~/RNDRange to copy ?~/RNC{PANELON}
           Range to copy ?~{BS}{BS}{WINDOWSON}{?}~/CRange to copy ?~~
E17: [W8] '{MENUBRANCH menu102}
F17: '{LET flag1102,0}{LET flag2102,0}~
A18: [W15] '!
D18: [W11] '{MENUBRANCH menu102}
A19: [W15] 'key102
B19: [W15] '~
A20: [W15] 'flag1102
B20: [W15] 0
A21: [W15] 'flag2102
B21: [W15] 0
```

Database Macros

[7] Sub-Total and Grand-Total Macro

	A	B	C	D	E
1	*---A SUBTOTAL and GRAND TOTAL macro for a database table				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Sort the database with respect to the items column to subtotal				
5	*---There should be at least two empty rows above the database to				
6	accommodate for the criterion area.				
7	*---Place the cell pointer at the upper left cell of the database				
8	*---Hold the [ALT] key and press [Z] to activate the macro				
9	!				
10	\Z	{BREAKON}			
11	SUBTOTAL	{PANELOFF}/DQRQ{ESC}{PANELON}Highlight the table to			
		sub-total (including fields) and press [ENTER]			
		{GET key1164}{ESC 6}{PANELOFF}{WINDOWSOFF}			
12	!	/RNCTable to total?~~/RNDTable to total?~			
		/RNC{PANELON}{WINDOWSON}Table to total?~{key1164}{?}~			
		{WINDOWSOFF}{PANELOFF}/DQITable to total?~Q{PANELON}			
13	!	{WINDOWSON}Point where to put the output table and			
		press [RETURN] {GET key1164}{ESC 6}{key1164}{?}~			
14	!	{WINDOWSOFF}{PANELOFF}/RNCoutputa164~~/RNDoutputa164~			
		/RNCoutputa164~~{GOTO}Table to total?~			
15	!	/C.{END}{RIGHT}~outputa164~{GOTO}outputa164~/C.{END}			
		{RIGHT}~{UP 2}~/DQO.{END}{RIGHT}~Q{UP 2}			
		{BRANCH crit164}			
16	!				
17	crit164	/DQC.{END}{RIGHT}{DOWN}~Q{DOWN}/RNCcriterion164~~/RND			
		criterion164~/RNCcriterion164~~{PANELON}{WINDOWSON}			
		{GOTO}Table to total?~{BRANCH subtot3164}			
18	!				
19	subtot3164	{WINDOWSON}{PANELON}Point to the ITEMS column and			
		press [RETURN] or press [ESC] to quit			
20	!	{GET key1164}{ESC 6}{IF key1164="{ESC}"}{BRANCH end164}			
21	!	{IF key1164<>"~"}{key1164}{BRANCH subtot3164}			
22	!	{WINDOWSOFF}{PANELOFF}{LET offset1164,@CELLPOINTER			
		("col")-@CELL("col",Table to total?)}~			
		{RECALC subtot1164} {BRANCH subtot1164}			
23	!				
24	subtot1164	{LET total164,0}~{GOTO}Table to total?~{RIGHT 0}{DOWN}			
25	again164	{LET here1164,@CELLPOINTER("address")}{WINDOWSOFF}			
		{PANELOFF}{RECALC again1164}			
26	again1164	/C~criterion164~{LET item164,criterion164}~			
		{IF offset1164>0}{GOTO}criterion164~/Ccriterion164~			
		{RIGHT 0}~/REcriterion164~{GOTO}{here1164}~			
27	subtot2164	{WINDOWSON}{PANELON}Point to the amount column and			
		press [RETURN] or press [ESC] to quit			
28	!	{GET key1164}{ESC 6}{IF key1164="{ESC}"}{BRANCH end164}			
29	!	{IF key1164<>"~"}{key1164}{BRANCH subtot2164}			
30	!	{WINDOWSOFF}{PANELOFF}{LET offset2164,@CELLPOINTER			
		("col")-@CELL("col",Table to total?)}~{RECALC queri164}			
		{BRANCH queri164}			
31	!				
32	queri164	/DQF~EQ{GOTO}outputa164~{RIGHT 2}{END}{DOWN}{DOWN 2}			
33	!	@SUM({UP 2}.{END}{UP}{DOWN})~			
34	!	{LET total164,total164+@CELLPOINTER("contents")}~			
		{RECALC movea164}			
35	movea164	{LEFT 2}			
36	!	Subtotal of {item164} =~			
37	!	{BRANCH loop1164}			
38	!				
39	loop1164	{GOTO}{here1164}~			
40	loop2164	{DOWN}{RECALC loopa164}{RECALC loopb164}			
41	!	{IF @CELLPOINTER("contents")<>item164#AND#@CELLPOINTER			
		("type")<>"b"}{BRANCH loop3164}			
42	loopa164	{IF @CELLPOINTER("type")="b"}{GOTO}criterion164~/RE.			
		{UP}{END}{RIGHT}~/RNDcriterion164~{GOTO}outputa164~			
		/RNDoutputa164~{END}{DOWN}{DOWN 4}GRAND TOTAL = ~			
		{RIGHT 2}+TOTAL164~{BRANCH end164}			

```
43 !               {BRANCH loop2164}
44 !
45 loop3164        {LET here1164,@CELLPOINTER("address")}~
46 loopb164        /C~criterion164~{LET item164,criterion164}~
                   {IF offset1164>0}{GOTO}criterion164~/Ccriterion164~
                   {RIGHT 0}~/REcriterion164~
47 !               {GOTO}outputa164~/C.{END}{RIGHT}~{END}{DOWN}{DOWN 4}~
                   {END}{DOWN}{DOWN 4}
48 !               /RNDoutputa164~/RNCoutputa164~~/DQO{BS}.{END}{RIGHT}~Q
                   {BRANCH queri164}
49 !
50 key1164         ~
51 !
52 here1164        $A$12
53 !
54 offset1164             0
55 offset2164             2
56 end164          /RNDTable to total?~
57 !
58 item164
59 !
60 total164
```

The code in B24, B26, B32, B35, B42 and the B46 cells is the result of the following formulas, therefore when you type the code, you should use this code, NOT the code in the main list

```
24 subtot1164      +"{LET total164,0}~{GOTO}Table to total?~{RIGHT "&
                   @STRING(B54,0)&"}{DOWN}"
26 again1164       +"/C~criterion164~{LET item164,criterion164}~
                   {IF offset1164>0}{GOTO}criterion164~/Ccriterion164~
                   {RIGHT "&@STRING(B54,0)&"}~/REcriterion164~{GOTO}
                   {here1164}~"
32 queri164        +"/DQF~EQ{GOTO}outputa164~{RIGHT "&@STRING(B55,0)&"}
                   {END}{DOWN}{DOWN 2}"
35 movea164        +"{LEFT "&@STRING(B55,0)&"}"
42 loopa164        +"{IF @CELLPOINTER("&"""type"""&")="&"""b"""&"}{GOTO}
                   criterion164~/RE.{UP}{END}{RIGHT}~/RNDcriterion164~
                   {GOTO}outputa164~/RNDoutputa164~{END}{DOWN}{DOWN 4}
                   GRAND TOTAL = ~{RIGHT "&@STRING(B55,0)&"}+TOTAL164~
                   {BRANCH end164}"
46 loopb164        +"/C~criterion164~{LET item164,criterion164}~
                   {IF offset1164>0}{GOTO}criterion164~/Ccriterion164~
                   {RIGHT "&@STRING(B54,0)&"}~/REcriterion164~{GOTO}
                   {here1164}~"
```

To understand what this macro does, let's look at the following table:

Code No.	Name	Sales $
001	Camera	10000.00
001	Film	5000.00
001	Lens	6000.00
002	CD Player	20000.00
002	Amplifier	15000.00
002	Speaker	16000.00
003	Laser Paper	4000.00
003	Copy paper	3000.00
003	Toner	2000.00

The macro will extract the sub-totals and the grand-total for every code item (must be sorted). The result will show:

Code No.	Name	Sales $
001	Camera	10000.00
001	Film	5000.00
001	Lens	6000.00
Sub-Total of 001 =		21000.00

```
     Code No.    Name             Sales $
      002       CD Player        20000.00
      002       Amplifier        15000.00
      002       Speaker          16000.00

Sub-Total of 002 =               51000.00

     Code No.    Name             Sales $
      003       Laser Paper       4000.00
      003       Copy paper        3000.00
      003       Toner             2000.00

Sub-Total of 003 =                9000.00

GRAND TOTAL =                    81000.00
```

Let's see how the macro does it.

	A	B	C	D	E
11	SUBTOTAL	{PANELOFF}/DQRQ{ESC}{PANELON}Highlight the table to sub-total (including fields) and press [ENTER] {GET key1164}{ESC 6}{PANELOFF}{WINDOWSOFF}			
12	!	/RNCTable to total?~~/RNDTable to total?~ /RNC{PANELON}{WINDOWSON}Table to total?~{key1164}{?}~ {WINDOWSOFF}{PANELOFF}/DQITable to total?~Q{PANELON}			

The basic idea is to use the **/ D**ata **Q**uery **E**xtract Lotus commands to extract the data from the table. Therefore the macro starts with `{PANELOFF}` that stops the panel display activity, while the macro issues `/DQRQ{ESC}` to reset any previous data query Input and output ranges. Then the macro issues `{PANELON}` to resume the panel activity. Next, because Lotus cannot find any command in the next line of text, the macro types

```
Highlight the table to sub-total (including fields) and press [ENTER]
```

and issues `{GET key1164}` that halts the macro execution and waits until you press any key. When you press a key, Lotus stores it in the B50 cell named [key1164]. To end this session and clear the prompt from the panel, the macro issues `{ESC 6}`. The next step of the macro is to give the table of data a meaningful range name that will also serve as a prompt. To limit the screen and panel activities, the macro issues `{WINDOWSOFF}{PANELOFF}`.

The macro uses the "safe technique" to assign the [Table to total?] range name to the table of data, and then issues the `{key1164}` routine command. The [key1164] routine holds the last key you pressed; therefore when this routine is activated, it executes that key. Normally when Lotus prompts you to Highlight/paint a range, you press the period "**.**" key to anchor the cell.

If you press the period key in response to the prompt to paint the table, the `{key1164}` command executes this key without your knowledge, before the macro continues with the `{?}` command. Using these techniques we have managed to combine a long and informative prompt in the panel with the [Table to total?] range name assignment, which appears to you as a second prompt. All this makes the macro easier to use. When the macro issues `{?}` you paint the table and press the ENTER key.

Before the macro continues, it uses `{PANELOFF}{WINDOWSOFF}` to freeze the screen and panel activities, and then issues `/DQITable to total?~Q{PANELON}{WINDOWSON}` to define the table as the input range for the data query process, and to allow panel and screen activities.

	A	B	C	D	E
13	!	{WINDOWSON}Point where to put the output table and press [RETURN] {GET key1164}{ESC 6}{key1164}{?}~			
14	!	{WINDOWSOFF}{PANELOFF}/RNCoutputa164~~/RNDoutputa164~/RNCoutputa164~~{GOTO}Table to total?~			
15	!	/C.{END}{RIGHT}~outputa164~{GOTO}outputa164~/C.{END}{RIGHT}~{UP 2}~/DQO.{END}{RIGHT}~Q{UP 2}{BRANCH crit164}			

The macro creates the second long informative prompt:

```
Point where to put the output table and press [RETURN]
```

The macro uses the `{GET key1164}{ESC 6}{key1164}{?}`, which is the same as before, except that now the macro does not assign a range name, because there is no need to paint a range.

Now that the output range location is known, the macro issues `{PANELOFF}{WINDOWSOFF}` to freeze the screen and panel activities, and uses the "safe technique" to assign the [outputa164] range name to the upper left cell of the output range. For a **D**ata **Q**uery to work, the output range and the criterion range have to have the same field names as the input range. To copy the field names to the output range, the macro first uses `{GOTO}Table to total?~` to move the cell pointer to the upper left cell of the input table and then uses `/C.{END}{RIGHT}~outputa164~` to copy the fields to the output range. It should be clear from this code why it is convenient to use range names in a macro. All that is left is to create the criterion range.

The easiest and safest ways to create the criterion range (which occupies two rows), is to place it above the input or the output range. Here the macro creates the criterion range two rows above the output range. The macro uses `{GOTO}outputa164~` to move the cell pointer to the output range, and then uses `/C.{END}{RIGHT}~{UP 2}~` to copy the field names to the second row above the output range fields. The output range is not yet defined, therefore the macro issues `/DQO.{END}{RIGHT}~Q` to assign the output range for the **D**ata **Q**uery process, and uses `{UP 2}` to move the cell pointer to the criterion range. Again, the criterion range is not yet defined, therefore the macro issues `{BRANCH crit164}` to route the macro execution to the [crit164] routine.

	A	B	C	D	E
17	crit164	/DQC.{END}{RIGHT}{DOWN}~Q{DOWN}/RNCcriterion164~~/RNDcriterion164~/RNCcriterion164~~{PANELON}{WINDOWSON}{GOTO}Table to total?~{BRANCH subtot3164}			

The [crit164] routine uses `/DQC.{END}{RIGHT}{DOWN}~` to create the criterion range for the **D**ata **Q**uery process, and then assigns the [criterion164] range name to the upper left cell of the criterion range again using the "safe technique" to assign a range name. Now the macro is ready to extract the data to sub-total, therefore it issues `{PANELON}{WINDOWSON}{GOTO} Table to total?` to allow screen and panel activities and to send the cell pointer back to the input table. Next it uses `{BRANCH subtot3164}` to route the macro execution to the [subtot3164] routine.

	A	B	C	D	E
19	subtot3164	{WINDOWSON}{PANELON}Point to the ITEMS column and press [RETURN] or press [ESC] to quit			
20	!	{GET key1164}{ESC 6}{IF key1164="{ESC}"}{BRANCH end164}			
21	!	{IF key1164<>"~"}{key1164}{BRANCH subtot3164}			
22	!	{WINDOWSOFF}{PANELOFF}{LET offset1164,@CELLPOINTER("col")-@CELL("col",Table to total?)}~{RECALC subtot1164} {BRANCH subtot1164}			

The [subtot3164] routine uses {WINDOWSON}{PANELON} to allow screen and panel activities before it writes

```
Point to the ITEMS column and press [RETURN] or press [ESC] to quit
```

into the panel. Use the arrow keys to move the cell pointer to the column of sorted items (in our example, to the codes column) and press the ENTER key. Here, the macro uses the same technique as shown earlier to display the prompt in the panel, but this time the macro checks which key you press after {GET key1164}. The macro uses {IF key1164="{ESC}"} to check if the key is the ESC key. If so, the macro uses {BRANCH end164} to branch to the [end164] routine.

	A	B	C	D	E
56	end164	/RNDTable to total?~			

This uses /RNDTable to total?~ to delete the [Table to total?] range names created by the macro, and then quits the macro because it reaches an empty cell.

The [subtot3164] routine uses {IF key1164<>"~"} to check that you pressed any key except the ENTER key. If so, the macro issues {key1164}{BRANCH subtot3164} to execute the key you pressed, returns to the beginning of the routine, and again displays the prompt in the panel. This is useful when you use the arrow keys to move the cell pointer to the desired column. The routine actually follows the keys that you pressed while the macro is in full control of your input. When you press ENTER, the macro uses {WINDOWSOFF} {PANELOFF} to freeze the screen and panel activies, and uses

```
{LET offset1164,@CELLPOINTER("col")-@CELL("col",Table to total?)}
```

to insert the offset of the current column from the first column of the input table, [Table to total?], into cell [offset1164]. The macro uses the result to calculate how far is the column you point to, from the first column of the input table. Then it issues {RECALC subtot1164} to update the formula in cell [subtot1164] which uses the value stored in cell [offset1164], and then issues {BRANCH subtot1164} to route the macro execution to the [subtot1164] routine.

	A	B	C	D	E
24	subtot1164	{LET total164,0}~{GOTO}Table to total?~{RIGHT 0}{DOWN}			
25	again164	{LET here1164,@CELLPOINTER("address")}{WINDOWSOFF}			
		{PANELOFF}{RECALC again1164}			
26	again1164	/C~criterion164~{LET item164,criterion164}~			
		{IF offset1164>0}{GOTO}criterion164~/Ccriterion164~			
		{RIGHT 0}~/REcriterion164~{GOTO}{here1164}~			

Before we continue with the [subtot1164] routine, we need to look at the dynamic formula that creates the code in the B24 cell

```
24 subtot1164       +"{LET total164,0}~{GOTO}Table to total?~{RIGHT "&
                    @STRING(B54,0)&"}{DOWN}"
```

The formula uses the @STRING(B54,0) function inside the {RIGHT} command to calculate how many cells to move to the right from the first column of the input table. The code in the B26 cell is also a result of the dynamic formula

```
26 again1164        +"/C~criterion164~{LET item164,criterion164}~
                    {IF offset1164>0}{GOTO}criterion164~/Ccriterion164~
```

```
{RIGHT "&@STRING(B54,0)&"}~/REcriterion164~{GOTO}
{here1164}~"
```

This formula also uses the `@STRING(B54,0)` function inside the `{RIGHT}` command to calculate how many cells to move to the right from the first cell of the criterion row.

The [subtot1164] routine starts with `{LET total164,0}~` to set the value in cell [total164] to zero. The [total164] cell holds the grand-total amount. The macro then uses `{GOTO}Table to total?~{RIGHT 0}{DOWN}` to move the cell pointer to the first cell of the column of sorted items that you point to. Now the macro uses `{LET here1164,@CELLPOINTER ("address")}` to store the current cell pointer address in cell [here1164]. The macro will use this address to return to the current location later on. The macro issues `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel activities, and then issues `{RECALC again1164}` to update the string formula in B26, [again1164], before the macro processes it.

The cell pointer is now on the first cell of the column that you point to, therefore it is on the first sorted item. To use this item as a criterion, the macro issues `/C~criterion164~` which copies the first item's code/name to the first cell in the criterion row. The macro now issues `{LET item164,criterion164}~` to store the item's name/code in cell [item164]. The macro will use this cell to make sure not to sub-total the same item twice. The item's name was copied to the first cell in the criterion row, but if you did not point to the first column of the input table the item's name should be copied to the same offset column in the criterion range. Therefore the macro issues the `{IF offset1164>0}...` command.

If the value in cell [offset1164] is greater then zero, the macro issues `{GOTO}criterion164~ /Ccriterion164~{RIGHT 3}~` to copy the item's name to the correct place in the criterion row. Then the macro uses `/REcriterion164~` to erase the first cell in the criterion row. You may ask why not to move the content in the first cell, instead of copy and erase? When a cell or range is moved, the range names associated with this range are also moved; we do not want to move the [criterion164] range name, we only want to move the content. The last macro command in this routine is the `{GOTO}{here1164}~` indirect macro command that uses the content of cell [here1164] to move the cell pointer to the address stored in this cell.

```
     A              B                C              D              E
27 subtot2164     {WINDOWSON}{PANELON}Point to the amount column and
                  press [RETURN] or press [ESC] to quit
28 !              {GET key1164}{ESC 6}{IF key1164="{ESC}"}{BRANCH end164}
29 !              {IF key1164<>"~"}{key1164}{BRANCH subtot2164}
30 !              {WINDOWSOFF}{PANELOFF}{LET offset2164,@CELLPOINTER
                  ("col")-@CELL("col",Table to total?)}~{RECALC queri164}
                  {BRANCH queri164}
```

The [subtot2164] routine uses the same code and techniques to display

```
Point to the amount column and press [RETURN] or press [ESC] to quit
```

The rest of the code is identical to the previous routine, except the last two macro commands: the `{RECALC queri164}` that updates the dynamic formula in cell [queri164], and the `{BRANCH queri164}` that routes the macro execution to the [queri164] routine.

```
     A              B                C              D              E
32 queri164       /DQF~EQ{GOTO}outputa164~{RIGHT 2}{END}{DOWN}{DOWN 2}
33 !              @SUM({UP 2}.{END}{UP}{DOWN})~
34 !              {LET total164,total164+@CELLPOINTER("contents")}~
```

```
                    {RECALC movea164}
35 movea164         {LEFT 2}
36 !                Subtotal of {item164} =~
37 !                {BRANCH loop1164}
```

The code in the B32 cell named [queri164] is, again, a result of a formula.

```
32 queri164         +"/DQF~EQ{GOTO}outputa164~{RIGHT "&@STRING(B55,0)&"}
                    {END}{DOWN}{DOWN 2}"
```

This formula uses the `@STRING(offset2164,0)` formula to calculate how many columns to move to the right, to bring the cell pointer to the amounts column that you pointed to. The code in B35, [movea164], is also a result of the formula

```
35 movea164         +"{LEFT "&@STRING(B55,0)&"}"
```

which uses the `@STRING(offset2164,0)` formula to calculate how many columns to move to the right direction to bring the cell pointer to the amounts column that you pointed to.

The [queri164] routine uses `/DQF~EQ` to extract all the records that match the criterion (the first item's name or code), and issues `{GOTO}outputa164~{RIGHT 2}{END}{DOWN} {DOWN 2}` to move the cell pointer two rows below the amounts column (the "Sales $" column in our example) in the output range. To create the sub-total of the amounts, the macro issues `@SUM({UP 2}.{END}{UP}{DOWN})~`. This set of commands needs explanation. To visualize how the macro works, let's look at the following output range in the A100..C103 range:

	A	B	C	D	E	F
100	Code No.	Name	Sales $			
101	001	Camera	10000.00			
102	001	Film	5000.00			
103	001	Lens	6000.00			
104						
105			|||||||||||||||||||			

The |||||||||||||||||||| represents where the cell pointer is at this moment, which is two rows below the last cell of the sales amounts column. Now the macro issues the `@SUM(` command, but because there is no valid command known to Lotus in this text, it types the text into the panel. The panel shows:

```
          @SUM(
```

The next `{UP 2}` is a valid Lotus command, therefore the cell pointer moves two cells up. The panel will show

```
          @SUM(C103..C103
```

and the output range will look like:

	A	B	C	D	E	F
100	Code No.	Name	Sales $			
101	001	Camera	10000.00			
102	001	Film	5000.00			
103	001	Lens	6000.00			

The period "`.`" command anchors the cell pointer to the C103 cell and `{END}{UP}{DOWN}` expand the highlight to include the C103..C101 range. The panel now shows:

```
@SUM(C103..C101
```

and the output range is:

```
     A          B          C          D          E          F
100  Code No.    Name       Sales $
101   001       Camera     10000.00
102   001       Film        5000.00
103   001       Lens        6000.00
```

The next command is the "`)`" closing parenthesis, which is not a reserved Lotus command; therefore Lotus types it to the panel to finish the formula in the panel which shows:

```
@SUM(C103..C101)
```

The last tilde "`~`" which is the equivalent of the ENTER key, is a valid Lotus command. This command writes the formula in the panel to the C105 cell and the output table shows:

```
     A          B          C          D          E          F
100  Code No.    Name       Sales $
101   001       Camera     10000.00
102   001       Film        5000.00
103   001       Lens        6000.00
104
105                         21000.00
```

To finish the sub-total process for the first item, the macro issues `{LET total164, total164+@CELLPOINTER("contents")}~` to add the sub-total to the grand-total in cell [total164] and then issues `{RECALC movea164}` to update the formula in cell [movea164] before the macro processes the code in this cell.

```
     A                B          C          D          E
35 movea164          {LEFT 2}
```

The code in cell [movea164] is the result of the following formula:

```
35 movea164          +"{LEFT "&@STRING(B55,0)&"}"
```

This formula uses the content of cell [offset2164] to create the `{LEFT 2}` command which moves the cell pointer to the A105 cell in our example. To write the "`Sub-Total of 001 =`" text into the A105 cell, the macro issues the "`Subtotal of {item164} =~`" macro code. Again, the first part of the command is the simple "`Subtotal of `" text which Lotus types into the panel, then Lotus sees the `{item164}` routine command, which is a valid command, because there is a cell named [item164] in the macro. In fact, we have used this cell name to store the item's name/code. When Lotus activates the `{item164}` routine command, the item's name is typed into the panel because it contains only pure text, and the panel shows:

```
Sub-Total of 001
```

Next the macro issues the "` =`" code which is, again, pure text, and the panel displays:

```
Sub-Total of 001 =
```

The tilde "`~`" commands ends the code and types the panel's content into the A105 cell. The first

sub-total table looks like this:

	A	B	C	D	E	F
100	Code No.	Name	Sales $			
101	001	Camera	10000.00			
102	001	Film	5000.00			
103	001	Lens	6000.00			
104						
105	Sub-Total of 001 =		21000.00			

Till now the macro has extracted only the sub-total for the first sorted item in the table, the macro has to do the same for all the other items, including the 002 and 003 items in the input table in our example, and then produce the grand-total. Therefore the macro issues `{BRANCH loop1164}` that routes the macro execution to cell [loop1164].

	A	B	C	D	E
39	loop1164	{GOTO}{here1164}~			
40	loop2164	{DOWN}{RECALC loopa164}{RECALC loopb164}			
41	!	{IF @CELLPOINTER("contents")<>item164#AND#@CELLPOINTER ("type")<>"b"}{BRANCH loop3164}			
42	loopa164	{IF @CELLPOINTER("type")="b"}{GOTO}criterion164~/RE. {UP}{END}{RIGHT}~/RNDcriterion164~{GOTO}outputa164~ /RNDoutputa164~{END}{DOWN}{DOWN 4}GRAND TOTAL = ~ {RIGHT 2}+TOTAL164~{BRANCH end164}			
43	!	{BRANCH loop2164}			

The first macro command in the [loop1164] routine is the `{GOTO}{here1164}~` indirect macro command that sends the cell pointer back to the address which is stored in cell [here1164]. Then the macro uses `{DOWN}` to move the cell pointer one cell down, and uses `{RECALC loopa164}` `{RECALC loopb164}` to update the dynamic formulas in cell [loopa164] and cell [loopb164]. The cell pointer is now on the first item of the column in the input table, and the macro issues

```
{IF @CELLPOINTER("contents")<>item164#AND#@CELLPOINTER("type")<>"b"}
```

to ensure that the item's code in the current cell is different from the previous item's code, which we have used to create the previous sub-total table, and that the cell is not empty. If this is true, it means that we have not yet finished the table process, therefore the macro issues `{BRANCH loop3164}` that activates the [loop3164] routine.

	A	B	C	D	E
45	loop3164	{LET here1164,@CELLPOINTER("address")}~			
46	loopb164	/C~criterion164~{LET item164,criterion164}~ {IF offset1164>0}{GOTO}criterion164~/Ccriterion164~ {RIGHT 0}~/REcriterion164~			
47	!	{GOTO}outputa164~/C.{END}{RIGHT}~{END}{DOWN}{DOWN 4}~ {END}{DOWN}{DOWN 4}			
48	!	/RNDoutputa164~/RNCoutputa164~~/DQO{BS}.{END}{RIGHT}~Q {BRANCH queri164}			

The macro uses `{LET here1164,@CELLPOINTER("address")}~` to store the current address in cell [here1164], and `/C~criterion164~` to copy the current cell's content to the first cell in the criterion row, and then issues `{LET item164,criterion164}~` to copy it to cell [item164]. The macro will use this cell to make sure not to sub-total the same item twice. The item's name was copied to the first cell in the criterion row, but if you did not point to the first column of the input table, the item's name should be copied to the same offset column in the criterion row. Therefore the macro issues the `{IF offset1164>0}...` command. If the value in cell [offset1164] is greater then zero, the macro issues `{GOTO}`

criterion164~/Ccriterion164~{RIGHT 3}~ to copy the item's name to the correct place in the criterion row. Then the macro uses the /RE criterion164~ macro code to erase the first cell in the criterion row. Note that the code in B45, [loopb164], is the result of the dynamic formula:

```
46 loopb164        +"/C~criterion164~{LET item164,criterion164}~
                   {IF offset1164>0}{GOTO}criterion164~/Ccriterion164~
                   {RIGHT "&@STRING(B54,0)&"}~/REcriterion164~{GOTO}
                   {here1164}~"
```

The dynamic part of the formula is the @STRING(offset1164,0) function that uses the value in cell [offset1164] to create the {RIGHT 3} command if the value in cell [offset1164] is "3".

The macro uses {GOTO}outputa164~ to move the cell pointer to the output range and then uses /C.{END}{RIGHT}~{END}{DOWN}{DOWN 4}~ to copy the column field names to the next sub-total table location, which is two rows below the last created sub-total table. Now the macro uses {END}{DOWN}{DOWN 4} to move the cell pointer to the new sub-total table. To re-assign the [outputa164] range name to the new table, the macro uses /RNDoutputa164~ to delete the [outputa164] range name, and then uses /RNCoutputa164~~ to assign it to the new location. To complete the preparations for the **D**ata **Q**uery **E**xtract process, the macro uses /DQO{BS}.{END}{RIGHT}~Q to re-define the output range. Note how {BS} erases the old output range that Lotus displays when the macro issues the /DQO macro keys.

To extract and create the sub-total table, the macro again issues {BRANCH queri164} as we have seen and explained previously. The macro will continue looping and creating sub-total tables as long as the

```
{IF @CELLPOINTER("contents")<>item164#AND#@CELLPOINTER("type")<>"b"}
```

condition is true. To see what happens when this condition is false, let us return to the [loop1164] routine

	A	B	C	D	E
39	loop1164	{GOTO}{here1164}~			
40	loop2164	{DOWN}{RECALC loopa164}{RECALC loopb164}			
41	!	{IF @CELLPOINTER("contents")<>item164#AND#@CELLPOINTER ("type")<>"b"}{BRANCH loop3164}			
42	loopa164	{IF @CELLPOINTER("type")="b"}{GOTO}criterion164~/RE. {UP}{END}{RIGHT}~/RNDcriterion164~{GOTO}outputa164~ /RNDoutputa164~{END}{DOWN}{DOWN 4}GRAND TOTAL = ~ {RIGHT 2}+TOTAL164~{BRANCH end164}			
43	!	{BRANCH loop2164}			

If the previous condition is false, the macro uses {IF @CELLPOINTER("type")="b"} to check if the current cell is blank. If so, the macro uses {GOTO}criterion164~ to move the cell pointer to the first cell in the criterion row and then uses /RE.{UP}{END}{RIGHT}~ to erase the criterion range including the field names and then issues /RNDcriterion164~ to delete the [criterion164] range name. The last thing to do is to write the grand-total amount and title. Therefore the macro moves the cell pointer to the last sub-total table, using {GOTO} outputa164~. Then it uses /RNDoutputa164~ to delete the [outputa164] range name, and issues {END}{DOWN}{DOWN 4} to position the cell pointer two rows below the last sub-total table. Then it types "GRAND TOTAL = " in the panel and issues the tilde "~" to write the content of the panel to the current cell. The macro uses {RIGHT 2} to move the cell pointer two cells to the right, and types the +TOTAL164 formula into the panel.

To write the formula into the cell, the macro issues the tilde "~". The `GRAND TOTAL = ~{RIGHT 2}+TOTAL164~` macro code combines two fine examples of how to manipulate the panel to write text and formulas to the current cell. To quit macro execution, the macro issues the `{BRANCH end164}` command. The next `{BRANCH loop2164}` is used in case the cell is not blank and the item's code is equal to the previous item's code.

Because of the complexity of this macro, here is a full list of all the cell formulas and contents.

```
A1: U [W14] '*---A SUBTOTAL and GRAND TOTAL macro for a database table
             (SEE the
A2: U [W14] '    SAMPLE1.WK1 file) Use the SUBTOTL3.WK3 macro in RELEASE 3
A3: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A4: [W14] '    range names in this column (starts with the \Z macro name)
A5: [W14] '*---Sort the database with respect to the items column to
          subtotal
A6: [W14] '*---There should be at least two empty rows above the database
          to
A7: [W14] '    accommodate for the criterion area.
A8: [W14] '*---Place the cell pointer at the upper left cell of the
          database
A9: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A10: U [W14] '\Z
B10: [W18] '{BREAKON}
A11: U [W14] 'SUBTOTAL
B11: [W18] '{PANELOFF}/DQRQ{ESC}{PANELON}Highlight the table to sub-total
            (including fields) and press [ENTER]{GET key1164}{ESC 6}
            {PANELOFF}{WINDOWSOFF}
A12: [W14] '!
B12: [W18] '{WINDOWSOFF}/RNCTable to total?~~/RNDTable to total?~
           /RNC{PANELON}{WINDOWSON}Table to total?~{key1164}{?}~
           {WINDOWSOFF}{PANELOFF}/DQITable to total?~Q{PANELON}
A13: [W14] '!
B13: [W18] '{WINDOWSON}Point where to put the output table and press
           [RETURN] {GET key1164}{ESC 6}{key1164}{?}~
A14: [W14] '!
B14: [W18] '{WINDOWSOFF}{PANELOFF}/RNCoutputa164~~/RNDoutputa164~/RNC
           outputa164~~{GOTO}Table to total?~
A15: [W14] '!
B15: [W18] '/C.{END}{RIGHT}~outputa164~{GOTO}outputa164~/C.{END}{RIGHT}~
           {UP 2}~/DQO.{END}{RIGHT}~Q{UP 2}{BRANCH crit164}
A16: [W14] '!
A17: [W14] 'crit164
B17: [W18] '/DQC.{END}{RIGHT}{DOWN}~Q{DOWN}/RNCcriterion164~~/RND
           criterion164~/RNCcriterion164~~{PANELON}{WINDOWSON}{GOTO}
           Table to total?~{BRANCH subtot3164}
A18: [W14] '!
A19: [W14] 'subtot3164
B19: [W18] '{WINDOWSON}{PANELON}Point to the ITEMS column and press [RETURN]
           or press[ESC] to quit
A20: [W14] '!
B20: [W18] '{GET key1164}{ESC 6}{IF key1164="{ESC}"}{BRANCH end164}
A21: [W14] '!
B21: [W18] '{IF key1164<>"~"}{key1164}{BRANCH subtot3164}
A22: [W14] '!
B22: [W18] '{WINDOWSOFF}{PANELOFF}{LET offset1164,@CELLPOINTER("col")-@CELL
           ("col",Table to total?)}~{RECALC subtot1164}{BRANCH subtot1164}
A23: [W14] '!
A24: [W14] 'subtot1164
B24: U [W18] +"{LET total164,0}~{GOTO}Table to total?~{RIGHT "&@STRING(B54
             ,0)&"}{DOWN}"
A25: [W14] 'again164
B25: [W18] '{LET here1164,@CELLPOINTER("address")}{WINDOWSOFF}{PANELOFF}
           {RECALC again1164}
A26: [W14] 'again1164
B26: U [W18] +"/C~criterion164~{LET item164,criterion164}~{IF offset1164>0}
```

```
            {GOTO}criterion164~/Ccriterion164~{RIGHT "&@STRING(B54,0)&"}~
            /REcriterion164~{GOTO}{here1164}~"
A27: [W14] 'subtot2164
B27: [W18] '{WINDOWSON}{PANELON}Point to the amount column and press
           [RETURN] or press [ESC] to quit
A28: [W14] '!
B28: [W18] '{GET key1164}{ESC 6}{IF key1164="{ESC}"}{BRANCH end164}
A29: [W14] '!
B29: [W18] '{IF key1164<>"~"}{key1164}{BRANCH subtot2164}
A30: [W14] '!
B30: [W18] '{WINDOWSOFF}{PANELOFF}{LET offset2164,@CELLPOINTER("col")-@CELL
           ("col",Table to total?)}~{RECALC queri164}{BRANCH queri164}
A31: [W14] '!
A32: [W14] 'queri164
B32: U [W18] +"/DQF~EQ{GOTO}outputa164~{RIGHT "&@STRING(B55,0)&"}{END}
            {DOWN}{DOWN 2}"
A33: [W14] '!
B33: [W18] '@SUM({UP 2}.{END}{UP}{DOWN})~
A34: [W14] '!
B34: [W18] '{LET total164,total164+@CELLPOINTER("contents")}~
           {RECALC movea164}
A35: [W14] 'movea164
B35: U [W18] +"{LEFT "&@STRING(B55,0)&"}"
A36: [W14] '!
B36: [W18] 'Subtotal of {item164} =~
A37: [W14] '!
B37: [W18] '{BRANCH loop1164}
A38: [W14] '!
A39: [W14] 'loop1164
B39: [W18] '{GOTO}{here1164}~
A40: [W14] 'loop2164
B40: [W18] '{DOWN}{RECALC loopa164}{RECALC loopb164}
A41: [W14] '!
B41: [W18] '{IF @CELLPOINTER("contents")<>item164#AND#@CELLPOINTER("type")
           <>"b"}{BRANCH loop3164}
A42: [W14] 'loopa164
B42: U [W18] +"{IF @CELLPOINTER("&"""type"""&")="&"""b"""&"}{GOTO}
            criterion164~/RE.{UP}{END}{RIGHT}~/RNDcriterion164~{GOTO}
            outputa164~/RNDoutputa164~{END}{DOWN}{DOWN 4}GRAND TOTAL = ~
            {RIGHT "&@STRING(B55,0)&"}+TOTAL164~{BRANCH end164}"
A43: [W14] '!
B43: [W18] '{BRANCH loop2164}
A44: [W14] '!
A45: [W14] 'loop3164
B45: [W18] '{LET here1164,@CELLPOINTER("address")}~
A46: [W14] 'loopb164
B46: U [W18] +"/C~criterion164~{LET item164,criterion164}~{IF offset1164>0}
            {GOTO}criterion164~/Ccriterion164~{RIGHT "&@STRING(B54,0)&"}~
            /REcriterion164~"
A47: [W14] '!
B47: [W18] '{GOTO}outputa164~/C.{END}{RIGHT}~{END}{DOWN}{DOWN 4}~{END}
           {DOWN}{DOWN 4}
A48: [W14] '!
B48: [W18] '/RNDoutputa164~/RNCoutputa164~~/DQO{BS}.{END}{RIGHT}~Q
           {BRANCH queri164}
A49: [W14] '!
A50: [W14] 'key1164
B50: [W18] '~
A51: [W14] '!
A52: [W14] 'here1164
B52: [W18] '$A$12
A53: [W14] '!
A54: [W14] 'offset1164
B54: [W18] 0
A55: [W14] 'offset2164
B55: [W18] 2
A56: [W14] 'end164
B56: [W18] '/RNDTable to total?~
A57: [W14] '!
A58: [W14] 'item164
A59: [W14] '!
```

```
A60: [W14] 'total164
```

[6] Query and Extract Macro

```
     A              B              C              D              E
 1 *---A QUERY macro with EXTRACT, the macro assumes two empty lines above
 2     the database fields for the criterion range. There is no need
 3     to add "*" etc. just type the searched string. The query is case
 4     insensitive.
 5 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 6     range names in this column (starts with the \Z macro name)
 7 *---Place the highlight on the upper left cell of the database
 8 *---Hold the [ALT] key and press [Z] to activate the macro
 9 *---Leave two empty rows above database and one empty column to the left
10 \Z              {BREAKON}
11 QUERY           Expand the highlight to include the database and press
                   [ENTER]! {GET key142}{ESC 6}{WINDOWSOFF}{PANELOFF}
                   /RNCDbase range ?~~/RNDDbase range ?~/RNC{PANELON}
                   {WINDOWSON}Dbase range ?~{key142}{?}~
12 !               Move the cell pointer to the output range and press
                   [ENTER] ! {GET key142}{ESC 6}{key142}{?}~{WINDOWSOFF}
                   {PANELOFF}/RNCoutput142~~/RNDoutput142~/RNCoutput142~~
13 !               {GOTO}Dbase range ?~/C.{END}{RIGHT}~{UP 2}~{UP 2}{LEFT}
                   criterion142~/RNLR~/DQRC.{END}{RIGHT}{DOWN}~Q{GOTO}
                   Dbase range ?~/DQIDbase range ?~Q
14 !               /C.{END}{RIGHT}~output142~{GOTO}output142~/DQO.{END}
                   {RIGHT}~Q{GOTO}criterion142~{DOWN}/RE{UP}{END}{RIGHT}
                   {DOWN}~
15 !               {BRANCH crit2142}
16 !
17 key142          {NS}
18 !
19 crit2142        {WINDOWSON}{PANELON}{GETLABEL "Insert criterion string
                   or [F] to Find or [S] to Skip: ",crit3142}~{IF @UPPER
                   (crit3142)<>"F"#AND#@UPPER(crit3142)<>"S"}
                   {RECALC critform142}{critform142}{WINDOWSOFF}{PANELOFF}
20 !               {IF @UPPER(crit3142)="F"}/DQF{ESC 4}{QUERY}{?}~
                   {WINDOWSOFF}{PANELOFF}{PGDN 2}{HOME}{RIGHT}{END}{DOWN}
                   {DOWN}{GOTO}output142~{WINDOWSON}{PANELON}
                   {MENUBRANCH extrct142}
21 !               {IF @UPPER(crit3142)="S"}{RIGHT}{BRANCH crit2142}
22 !               {IF @LENGTH(crit3142)=0}{RIGHT}{BRANCH crit2142}
23 !               {RIGHT}{BRANCH crit2142}
24 !
25 crit3142        F
26 !
27 critform142     #NOT#@ISERR(@FIND("ROBERT",@UPPER({DOWN 2}),0))~
28 !
29 extrct142       Extract     Quit
30 !               Extract and Quit the macro
31 !               /DQEQ       {namesdel}
32 !               {namesdel}
33 !
34 namesdel        {GOTO}criterion142~{LEFT}/RE.{END}{RIGHT}{DOWN}~
35 !               /RNDoutput142~/RNDcriterion142~/RNDDbase range ?~
```

Notice that the code in the B27 cell named [critform142] is the result of the following formula:

```
27 critform142     +"#NOT#@ISERR(@FIND(""&@UPPER(B25)&"",@UPPER
                   ({DOWN 2}),0))~"
```

This is the complete code listing of the two menu options in the B29..C32 range:

```
Extract
Extract and view the matching addresses
/DQEQ
{namesdel}

Quit
```

```
Quit the macro
{namesdel}
```

This macro makes the QUERY and EXTRACT processes easier. All the macros in ***Super Power*** and the SUPER MACRO LIBRARY which are related to database operation, assume two empty rows above the database fields for the criterion range. The macros define the criterion range in the two empty rows above the database. In this macro, there is no need to use astrix "*" in the criterion string for a wild card search, just type the searched string. This macro uses a smart criterion formula which can find the search string **any where** in the record, not only at the beginning of the records. The query is case insensitive. Place the cell pointer on the upper left cell of the database (including the fields) and leave two empty rows above the database fields and one empty column to the left of the database.

	A	B	C	D	E
11	QUERY	Expand the highlight to include the database and press [ENTER]! {GET key142}{ESC 6}{WINDOWSOFF}{PANELOFF} /RNCDbase range ?~~/RNDDbase range ?~/RNC{PANELON} {WINDOWSON}Dbase range ?~{key142}{?}~			

The macro starts with a prompt in the panel:

```
Expand the highlight to include the database and press [ENTER]!
```

because there are no macro keys or commands in this line, the macro just types it to the panel which appears as a guiding prompt. The macro immediately follows the prompt with the `{GET key142}` command which halts the macro and waits until you press a key. When press a key, the macro issues `{ESC 6}` to clear the panel and starts a process to assign the [Dbase range ?] name to the database using the "safe technique" to create a range name in a macro and simultaneously uses the [Dbase range ?] range name as a prompt. However there is one improvement here, the macro issues the `{key142}` routine command which carries the key you pressed with a response to the prompt in the panel. For example, if you pressed the DOWN arrow key in the numeric key pad in response to the

```
Expand the highlight to include the database and press [ENTER]!
```

prompt, the `{GET key142}` command stored the `{DOWN}` command in cell [key142]. When the macro issues the `{key142}` routine command, the macro executes the `{DOWN}` command which moves the cell pointer one cell down. There is no delay and you do not need to press the DOWN key again when the macro issues `{?}`. Combining the prompt in the panel technique with the safe range name create technique makes a more powerful and more professional approach to macro programming.

	A	B	C	D	E
12	!	Move the cell pointer to the output range and press [ENTER] ! {GET key142}{ESC 6}{key142}{?}~{WINDOWSOFF} {PANELOFF}/RNCoutput142~~/RNDoutput142~/RNCoutput142~~			

Now the macro displays a second prompt in the panel:

```
Move the cell pointer to the output range and press [ENTER] !
```

to point to the output range for the extract process. The macro uses the same technique as before but, because this time only one cell is enough to define the output range, the macro does not use the range name as a prompt, but only uses the safe technique to assign the [output142] name to

the output range using:

```
/RNCoutput142~~/RNDoutput142~/RNCoutput142~~
```

	A	B	C	D	E
13	!	{GOTO}Dbase range ?~/C.{END}{RIGHT}~{UP 2}~{UP 2}{LEFT} criterion142~/RNLR~/DQRC.{END}{RIGHT}{DOWN}~Q{GOTO} Dbase range ?~/DQIDbase range ?~Q			

Now that the database range and the output range for the extracted data are defined, the macro issues `{GOTO}Dbase range ?~` to move the cell pointer to the database and then `/C.{END}{RIGHT}~{UP 2}~` to copy the field names to the criterion range two rows above the database. Next the macro uses `{UP 2}{LEFT}` to move the cell pointer to the column to the left of the criterion range and then issues the "`criterion142~`" code which types the "`criterion142`" string into the current cell. As we previously explained this code is just a text typed to the panel until Lotus encounter the tilde "`~`" command which is the same as ENTER from the keyboard and writes the "`criterion142`" string into the current cell. Now the macro issues `/RNLR~` to assign the [criterion142] name to the criterion range, and starts the data query process using:

```
/DQRC.{END}{RIGHT}{DOWN}~Q{GOTO}Dbase range ?~/DQIDbase range ?~Q
```

The `/DQRC.{END}{RIGHT}{DOWN}~Q` macro code first resets the data query range and then defines the second row of field names and the row under this as the criterion range. Then the macro uses `{GOTO}Dbase range ?~` to move the cell pointer to the database range and issues `/DQIDbase range ?~Q` to define the database range as the input range. Now the input range and the criterion range are defined. What is left to define is the output range. The place for the output range already exists: the [output142] range . But the output range must also include the same field names as the input range and the criterion range, therefore the macro continues with

	A	B	C	D	E
14	!	/C.{END}{RIGHT}~output142~{GOTO}output142~/DQO.{END} {RIGHT}~Q{GOTO}criterion142~{DOWN}/RE{UP}{END}{RIGHT} {DOWN}~			
15	!	{BRANCH crit2142}			

The macro issues `/C.{END}{RIGHT}~output142~` to copy the row of field names to cell [output142] and then issues `{GOTO}output142~` to move the cell pointer to the same address. The macro uses `/DQO.{END}{RIGHT}~Q` to define the output range and `{GOTO}criterion142~{DOWN}` to return the cell pointer to the criterion range. Now the macro issues `/RE{UP}{END}{RIGHT}{DOWN}~` and erases the content in the criterion range from any previous data. Notice how the macro uses the length of the row of fields to define the output range and to erase the content of the criterion range. Next the macro issues `{BRANCH crit2142}` which routes the macro control to the [crit2142] routine.

	A	B	C	D	E
19	crit2142	{WINDOWSON}{PANELON}{GETLABEL "Insert criterion string or [F] to Find or [S] to Skip: ",crit3142}~{IF @UPPER (crit3142)<>"F"#AND#@UPPER(crit3142)<>"S"} {RECALC critform142}{critform142}{WINDOWSOFF}{PANELOFF}			

The [crit2142] routine asks you to insert the criterion string using

```
{GETLABEL "Insert criterion string or [F] to Find or [S] to
    Skip: ",crit3142}~
```

which displays the following prompt in the panel

```
Insert criterion string or [F] to Find or [S] to Skip:
```

You now have three options: to key in the criterion, to press the "S" key to skip to the next field or to press the "F" key to start the query process. Your response to the prompt is stored in cell [crit3142]. When you press the ENTER key the macro issues

```
{IF @UPPER(crit3142)<>"F"#AND#@UPPER(crit3142)<>"S"}
```

to make sure that you inserted data to the field and did not insert the "F" or the "S" characters. If so, the macro issues `{RECALC critform142}` to update the criterion formula in the B27 cell named [critform142].

```
27 critform142      +"#NOT#@ISERR(@FIND(""""&@UPPER(B25)&""",@UPPER
                    ({DOWN 2}),0))~"
```

The formula in B27, [critform142], is a string formula that produces the correct code for the macro and incorporates the content of the B25 cell, [crit3142]. For example, if B25, [crit3142], contains the "`Robert`" string, the formula in B27 will show:

```
#NOT#@ISERR(@FIND("ROBERT",@UPPER({DOWN 2}),0))~
```

This kind of criterion formula will extract all the records that include the "`Robert`" string in the same field no matter where they are in the record. To insert the criterion formula into the criterion range, the macro issues the `{critform142}` routine command injecting

```
#NOT#@ISERR(@FIND("ROBERT",@UPPER(
```

into the panel because Lotus does not see any macro key or command in the text until it reaches `{DOWN 2}`. Because the text in the panel is actually a formula, `{DOWN 2}` acts exactly as you use the DOWN arrow key from the keyboard, and the cell address of the cell located two rows below the current cell position will be written to the panel which will display:

```
#NOT#@ISERR(@FIND("ROBERT",@UPPER(G20
```

assuming that the current cell pointer position is the G18 cell in the criterion range. The macro uses the pointing and painting technique to point to the cell for the first record two rows below to write a formula into the criterion range. Therefore the G20 cell which belongs to the first record of the database appears in the formula. Next the [critform142] routine completes the formula and writes the right parenthesis using the `),0))~` code to complete the formula:

```
#NOT#@ISERR(@FIND("ROBRET",@UPPER(G20),0))
```

We have seen here a unique example of a formula which creates a dynamic macro code, manipulates the panel, and uses the pointing technique to create a criterion formula to extract data from a database. You might invest some time studying this code. Try to perform the exact commands from the keyboard to see how the criterion formula evolved.

	A	B	C	D	E
20	!	{IF @UPPER(crit3142)="F"}/DQF{ESC 4}{QUERY}{?}~ {WINDOWSOFF}{PANELOFF}{PGDN 2}{HOME}{RIGHT}{END}{DOWN} {DOWN}{GOTO}output142~{WINDOWSON}{PANELON} {MENUBRANCH extrct142}			

The macro issues {IF @UPPER(crit3142)="F"} to check if you pressed the "F" or "f" keys. If so, the macro issues /DQF{ESC 4}{QUERY}{?} and starts the **D**ata **Q**uery **F**ind process. The {ESC 4} returns Lotus to the READY mode. The {QUERY}, which is equivalent to the F7 function key pressed from the keyboard, renews the query session, but this time when you finish and press ENTER, the macro automatically returns to the READY mode without the need to use the {ESC 4} command. The {?} allows you to use the DOWN and the UP arrow keys to scroll through the matched records. When you finish scrolling through the matched records and press ENTER, the macro issues {MENUBRANCH extrct142} and activates the custom menu. Before looking into the custom menu, let us look at the other {IF} conditions:

	A	B	C	D	E
21	!	{IF @UPPER(crit3142)="S"}{RIGHT}{BRANCH crit2142}			
22	!	{IF @LENGTH(crit3142)=0}{RIGHT}{BRANCH crit2142}			
23	!	{RIGHT}{BRANCH crit2142}			

If you press the "S" or the "s" character and ENTER, the macro issues {RIGHT} which moves the cell pointer to the next cell in the criterion range and then issues {BRANCH crit2142} which branches the macro control back to the beginning of the routine. If you just press ENTER, the {IF @LENGTH(crit3142)=0} condition is true, therefore the macro issues {RIGHT} which moves the cell pointer to the next cell in the criterion range, and then issues {BRANCH crit2142} which returns the macro control back to the beginning of the routine. If you press any other key such as the ESC key, the macro issues {RIGHT} which moves the cell pointer to the next cell in the criterion range, and then issues {BRANCH crit2142} which returns the macro control back to the beginning of the routine.

Now we can continue with the custom menu. The first menu item is [**E**xtract]:

```
Extract
Extract and view the matching addresses
/DQEQ
{namesdel}
```

If you select this menu option, the macro issues /DQEQ to extract the data into the output range, and then issues the {namesdel} routine command which activates the [namesdel] routine.

The second menu item is [**Q**uit]:

```
Quit
Quit the macro
{namesdel}
```

If you select this menu option, the macro issues the {namesdel} routine command which activates the [namesdel] routine.

	A	B	C	D	E
34	namesdel	{GOTO}criterion142~{LEFT}/RE.{END}{RIGHT}{DOWN}~			
35	!	/RNDoutput142~/RNDcriterion142~/RNDDbase range ?~			

The [namesdel] routine uses {GOTO}criterion142~{LEFT}/RE.{END}{RIGHT} {DOWN}~ to clear the criterion range from the field strings and the data. Last, issues /RND criterion142~/RNDDbase range ?~/RNDoutput142~ to delete the temporary [output142], [criterion142] and the [Dbase range ?] range names, to leave a clean worksheet as

it was before the macro started.

[6] Add Data to Database

```
     A                    B                    C                    D                    E
 1 *---A macro to add data to a database. If you type a number the macro
 2     changes it to a label.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Place the cell pointer on the upper left cell of your database
 6     field names. The data base should have at least two records.
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8
 9 !
10 \Z              {BREAKON}
11 ADDDATA         {WINDOWSOFF}{PANELOFF}/RNCdatabase~~/RNDdatabase~
                   /RNCdatabase~~{GOTO}IV1~{GOTO}database~{DOWN}/WTH{END}
                   {DOWN}{DOWN}{WINDOWSON}{PANELON}
12 !               {GETLABEL "How many fields in the database ? ",
                   fields092}~{MENUBRANCH menu1092}
13 !
14 menu1092        Add/edit       Quit
15 !               Add a record toQuit the macro
16 !               {LET counter092/WTC/RNDdatabase~{BRANCH ret092}
17 !               {MENUBRANCH menu1092}
18 !
19 move1092        {IF counter092>@VALUE(fields092)}{LET counter092,1}~
                   {WINDOWSOFF}{GOTO}IV1~{GOTO}database~{PGDN}{PGUP}
                   {END}{DOWN}{DOWN}{WINDOWSON}
20 move092         Insert data and press [ENTER] or [ESC] to quit:
                   {GET key1092}{ESC}{IF key1092="{ESC}"}{ESC 6}{RETURN}
21 !               {IF key1092="~"}{RIGHT}{LET counter092,counter092+1}~
                   {BRANCH move1092}
22 !               {IF key1092="{RIGHT}"#OR#key1092="{LEFT}"#OR#key1092=
                   "{UP}"#OR#key1092="{DOWN}"#OR#key1092="{PGDN}"#OR#
                   key1092="{PGUP}"}{key1092}{BRANCH move092}
23 !               {IF @CODE(key1092)>=48#AND#@CODE(key1092)<=57}{EDIT}'
24 !               {key1092}{?}~{RIGHT}{LET counter092,counter092+1}~
                   {BRANCH move1092}
25 !
26 counter092                  1
27 !
28 key1092         {ESC}
29 !
30 fields092       2
31 !
32 ret092
```

The add data macro makes the task of building and filling data in a database easier. As with any Lotus database, you must insert the field names into the top row of the database and then insert the data into the database. This macro assumes that the field names are already in the worksheet and at least two records are included in the database. It also demands the cell pointer be placed on the first field name before the macro starts. The macro is built around a custom menu with two menu options; however, because the code of this menu cannot be displayed clearly on one page without overlapping, we show here the code of each menu item separately:

```
Add/edit
Add a record to the database, press the [ENTER] after typing a data
LET counter092,1}~{move1092}
{MENUBRANCH menu1092}


Quit
Quit the macro
/WTC/RNDdatabase~{BRANCH ret092}
```

If you intend to key the code of this macro into Lotus 1-2-3, you have to key the code of the

custom menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you, we show the full cell contents list at the end of this section.

	A	B	C	D	E
11	ADDDATA	{WINDOWSOFF}{PANELOFF}/RNCdatabase~~/RNDdatabase~ /RNCdatabase~~{GOTO}IV1~{GOTO}database~{DOWN}/WTH{END} {DOWN}{DOWN}{WINDOWSON}{PANELON}			
12	!	{GETLABEL "How many fields in the database ? ", fields092}~{MENUBRANCH menu1092}			

The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands which freeze the screen and the panel display activities and then it uses the "safe technique" to assign the [database] range name to the first field name. To make sure that the first field will be also on the upper left corner of the screen, the macro issues {GOTO}IV1~{GOTO}database~. Next, the macro issues {DOWN} to move the cell pointer to the first record of the database and then issues /WTH to assign the row of fields as a horizontal title. Then the macro issues {END} {DOWN} {DOWN} to place the cell pointer on the first empty record of the first field. Last, it issues {WINDOWSON}{PANELON} to resume the screen and panel display activities to allow you to enter data into the database.

To make it easier to understand the macro code, let's look at the following database sample:

	A	B	C	D	E
1	FNAME	LNAME	COMPANY	ADDRESS	CITY	STATE	ZIP
2	DAVID	SAMUEL	ABC CO.	P.O. BOX 48	NEW YORK	NY	11111
3	JOHN	KENNEDY	DEF CO.	611 NW OAK DR	SALEM	NY	12111

In this example, the macro assigned the [database] range name to the A1 cell. Therefore, when the macro issues {GOTO}IV1~{GOTO}database~ the cell pointer moves to the A1 cell and simultaneously the A1 cell is placed on the upper left corner of the screen. Next the macro moves the cell pointer to the A2 cell and then assigns the row of the fields as a horizontal title. Last the macro issues {END}{DOWN}{DOWN} to place the cell pointer on the A4 cell ready to start. Now the macro issues

```
{GETLABEL "How many fields in the database ? ",fields092}
```

which displays "How many fields in the database ? " in the panel. When you insert the number of fields (seven in our example) and press ENTER, Lotus stores the number of fields in cell [fields092], and then uses {MENUBRANCH menu1092} to start the [menu1092] custom menu.

The first menu option is [**A**dd/edit]:

```
Add/edit
Add a record to the database, press the [ENTER] after typing a data
{LET counter092,1}~{move1092}
{MENUBRANCH menu1092}
```

When you choose this option, the macro issues {LET counter092,1}~ which increases the value in cell [counter092] by one. Then the macro issues the {move1092} routine command which starts the [move1092] routine.

	A	B	C	D	E
19	move1092	{IF counter092>@VALUE(fields092)}{LET counter092,1}~ {WINDOWSOFF}{GOTO}IV1~{GOTO}database~{PGDN}{PGUP}			

```
                {END}{DOWN}{DOWN}{WINDOWSON}
20 move092      Insert data and press [ENTER] or [ESC] to quit:
                {GET key1092}{ESC}{IF key1092="{ESC}"}{ESC 6}{RETURN}
21 !            {IF key1092="~"}{RIGHT}{LET counter092,counter092+1}~
                {BRANCH move1092}
22 !            {IF key1092="{RIGHT}"#OR#key1092="{LEFT}"#OR#key1092=
                "{UP}"#OR#key1092="{DOWN}"#OR#key1092="{PGDN}"#OR#
                key1092="{PGUP}"}{key1092}{BRANCH move092}
23 !            {IF @CODE(key1092)>=48#AND#@CODE(key1092)<=57}{EDIT}'
24 !            {key1092}{?}~{RIGHT}{LET counter092,counter092+1}~
                {BRANCH move1092}
```

The routine starts with `{IF counter092>@VALUE(fields092)}` which checks if the value in cell [counter092], which serves as a counter is greater than the number of fields in the database. If so, it means that you have finished inserting the data into the record. Therefore the routine issues `{LET counter092,1}~`, to reset the counter to "1". Next the routine issues `{WINDOWSOFF}` to freeze the screen display activity, and then issues `{GOTO}IV1~ {GOTO}database~` to place the first field name on the upper left corner of the screen. Because the row of fields is also serve as a horizontal title the screen displays it twice. Therefore the macro uses `{PGDN}{PGUP}` to correct it.

	A	B	C	D	E	F	G
1	FNAME	LNAME	COMPANY	ADDRESS	CITY	STATE	ZIP
2	DAVID	SAMUEL	ABC CO.	P.O. Box 48	NEW YORK	NY	11111
3	JOHN	KENNEDY	DEF CO.	611 NW OAK DR	SALEM	NY	12111
4	MAC	HERBERT	GHI CO.	P.O. Box 100	PORTLAND	OR	97444
5							

To understand this, let's look back at the sample database. Assuming you entered the third record in the A4..G4 range, when the macro issues `{GOTO}IV1~{GOTO}database~` the screen should display:

	A	B	C	D	E	F	G
1	FNAME	LNAME	COMPANY	ADDRESS	CITY	STATE	ZIP
1	FNAME	LNAME	COMPANY	ADDRESS	CITY	STATE	ZIP
2	DAVID	SAMUEL	ABC CO.	P.O. Box 48	NEW YORK	NY	11111
3	JOHN	KENNEDY	DEF CO.	611 NW OAK DR	SALEM	NY	12111
4	MAC	HERBERT	GHI CO.	P.O. Box 100	PORTLAND	OR	97444
5							

After `{PGDN}{PGUP}` the display looks like:

	A	B	C	D	E	F	G
1	FNAME	LNAME	COMPANY	ADDRESS	CITY	STATE	ZIP
2	DAVID	SAMUEL	ABC CO.	P.O. Box 48	NEW YORK	NY	11111
3	JOHN	KENNEDY	DEF CO.	611 NW OAK DR	SALEM	NY	12111
4	MAC	HERBERT	GHI CO.	P.O. Box 100	PORTLAND	OR	97444
5							

The cell pointer is now on A5, ready for you to insert the next record. If the counter value in cell [counter092] is less or equal to the number of fields in the database, the macro directly writes

```
Insert data and press [ENTER] or [ESC] to quit:
```

into the panel and then issues `{GET key1092}` which halts the macro execution so you can read the message and respond. When you press a key, Lotus stores the key in cell [key1092] and the macro immediately issues `{ESC}` to clear the message from the panel before Lotus writes it

into the current cell. Now the macro start a series of {IF} conditions to monitor and control your response. First the macro issues {IF key1092="{ESC}"} to check if you pressed ESC. If so, the macro issues {ESC 6}{RETURN} to return to the custom menu.

Next the macro issues {IF key1092="~"} to check if you pressed ENTER. If so, the macro issues {RIGHT} to move the cell pointer to the next field in the current record, then issues {LET counter092,counter092+1}~ to increase the counter value in cell [counter092] by one. Last, issues {BRANCH move1092} to route the macro control back to the beginning of the [move1092] routine to allow you to insert the data for the next field in the record. Every time you press ENTER, the macro increases the counter by one and moves the cell pointer to the next field in the record.

The next condition is the

```
{IF key1092="{RIGHT}"#OR#key1092="{LEFT}"#OR#key1092="{UP}"#OR#
    key1092="{DOWN}"#OR#key1092="{PGDN}"#OR#key1092="{PGUP}"}
```

Here the macro checks if you pressed one of the direction keys. If so, it issues the {key1092} routine command which executes the [key1092] routine. Recall that [key1092] holds the key that you pressed, if you pressed the DOWN direction key, Lotus stored the {DOWN} command in cell [key1092]. Therefore {key1092} executes the DOWN key exactly as you meant to. Then the macro issues {BRANCH move092} to route macro control to the [move092] routine and displays:

```
Insert data and press [ENTER] or [ESC] to quit:
```

in the panel again. Next the macro issues

```
{IF @CODE(key1092)>=48#AND#@CODE(key1092)<=57}{EDIT}'
```

to check if you pressed a number, for example when you insert a zip code or an address. If so, the macro issues {EDIT} to enter the EDIT mode and then directly writes the apostrophe "'" into the panel before it issues the {key1092} command which injects the number you pressed into the panel following the apostrophe "'". Next the macro immediately follows with {?} to allow you to continue to enter the data which follows the number. When you press ENTER, the macro issues the tilde "~" and writes the content of the panel into the current cell. Finally, it issues {RIGHT}{LET counter092,counter092+1}~{BRANCH move1092} which moves the cell pointer to the next field in the record, increases the counter by one, and routes the macro back to the beginning of the routine to allow you to enter the data to the next field.

To summarize: The macro allows you to use the direction keys to edit previous records or just to move around as long as you do not press the enter key. You should move the cell pointer back to the current field before you press ENTER. Only when you press ENTER, does the macro count the number of fields and move the cell pointer to the next field. In addition, the macro identifies when you start the data with a number and therefore precedes the number with an apostrophe so that the database contains only labels. The macro will turn a number automatically into a label the minute you start to type data beginning with a number or a value. This is especially important when you type an address or a zip code into the cell.

Lotus cannot accept data that starts as a number and also contains labels. For example, trying to type "123 Garfield Street" will result in an error, therefore you usually type the

apostrophe "'" first to signal Lotus that the data is a label. The macro does it automatically. It checks the @CODE number of the character that you pressed. If the code is between 48 and 57, then it is a number and the macro issues {EDIT} and types the apostrophe "'" into the panel. Then the macro issues {key1092} activating the [key1092] routine which injects the number previously stored by {GET key1092} into the panel. When the value in cell [counter092] is greater than the number of fields, the first {IF} condition in the B19 cell applies as we have previously seen.

The second menu option is [**Q**uit]:

```
Quit
Quit the macro
/WTC/RNDdatabase~{BRANCH ret092}
```

When you choose this option, the macro issues /WTC/RNDdatabase~{BRANCH ret092} which clear the titles, delete the temporary [database] range name, and route the macro control to the empty [ret092] routine and quit.

To help you to key this macro into Lotus 1-2-3, here is the full list of all the cell contents.

```
A1: U [W15] '*---A macro to add data to a database. If you type a number
            the macro
A2: U [W15] '    changes it to a label.
A3: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A4: [W15] '    range names in this column (starts with the \Z macro name)
A5: [W15] '*---Place the cell pointer on the upper left cell of your
            database field
A6: [W15] '    names. The data base should have at least one record.
A7: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A8: U [W15] '         * * *  Use the SAMPBASE.WK1 file as an example * * *
A9: [W15] '!
A10: U [W15] '\Z
B10: [W15] '{BREAKON}
A11: U [W15] 'ADDDATA
B11: [W15] '{WINDOWSOFF}{PANELOFF}/RNCdatabase~~/RNDdatabase~/RNCdatabase~~
           {GOTO}IV1~{GOTO}database~{DOWN}/WTH{END}{DOWN}{DOWN}{WINDOWSON}
           {PANELON}
A12: [W15] '!
B12: [W15] '{GETLABEL "How many fields in the database ? ",fields092}~
            {MENUBRANCH menu1092}
A13: [W15] '!
A14: [W15] 'menu1092
B14: [W15] 'Add/edit
C14: [W15] 'Quit
A15: [W15] '!
B15: [W15] 'Add a record to the database, press the [ENTER] after typing a
           data
C15: [W15] 'Quit the macro
A16: [W15] '!
B16: [W15] '{LET counter092,1}~{move1092}
C16: [W15] '/WTC/RNDdatabase~{BRANCH ret092}
A17: [W15] '!
B17: [W15] '{MENUBRANCH menu1092}
A18: [W15] '!
A19: [W15] 'move1092
B19: [W15] '{IF counter092>@VALUE(fields092)}{LET counter092,1}~
           {WINDOWSOFF}{GOTO}IV1~{GOTO}database~{PGDN}{PGUP}{END}{DOWN}
           {DOWN}{WINDOWSON}
A20: [W15] 'move092
B20: [W15] 'Insert data and press [ENTER] or [ESC] to quit: {GET key1092}
           {ESC}{IF key1092="{ESC}"}{ESC 6}{RETURN}
```

```
A21: [W15] '!
B21: [W15] '{IF key1092="~"}{RIGHT}{LET counter092,counter092+1}~
           {BRANCH move1092}
A22: [W15] '!
B22: [W15] '{IF key1092="{RIGHT}"#OR#key1092="{LEFT}"#OR#key1092="{UP}"
           #OR#key1092="{DOWN}"#OR#key1092="{PGDN}"#OR#key1092="{PGUP}"}
           {key1092}{BRANCH move092}
A23: [W15] '!
B23: [W15] '{IF @CODE(key1092)>=48#AND#@CODE(key1092)<=57}{EDIT}'
A24: [W15] '!
B24: [W15] '{key1092}{?}~{RIGHT}{LET counter092,counter092+1}~
           {BRANCH move1092}
A25: [W15] '!
A26: [W15] 'counter092
B26: [W15] 1
A27: [W15] '!
A28: [W15] 'key1092
B28: [W15] '{ESC}
A29: [W15] '!
A30: [W15] 'fields092
B30: [W15] '2
A31: [W15] '!
A32: [W15] 'ret092
```

[3] Sort Descending by One Key

	A	B	C	D	E
1	*---A macro to descending sort by primary-key only				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	SORT1DES	{WINDOWSOFF}{PANELOFF}/DSRQ/RNCWhich range ?~~/RND Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~ {BS}{BS}{?}~{GOTO}Which range ?~			
12	!	{WINDOWSOFF}{PANELOFF}/DSRDWhich range ?~{WINDOWSON} {PANELON}P{?}~D~G/DSRQ/RNDWhich range ?~			

This macro saves you some key strokes and makes the task of sorting a range a bit easier. You concentrate only on the important issues i.e. the primary key and the range to sort, the rest is done by the macro. The macro starts with the `/DSRQ` macro keys to reset any previous sorting information and then issues `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range, and simultaneously assigns the [Which range ?] name to the same range, which acts as a prompt.

Next the macro issues `{GOTO}Which range ?~` which moves the cell pointer to the upper left cell of the range to sort. Again, the macro issues `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities, and issues `/DSRDWhich range ?~` to define the [Which range ?] as the range to sort. Now the macro issues `{WINDOWSON}{PANELON}` to resume the screen and panel activities, allowing you to point to the primary key for the sorting operation. Next the macro issues `P{?}` pausing the macro to wait until you point to the column of the primary key. When you press ENTER, the macro issues the tilde "~" which is the same as the ENTER key and continues with the `D~G` macro keys, which inform Lotus that the sort with reference to the primary key is a descending sort and executes the sort. Next the macro issues `/DSRQ` to reset the data sort information, and continues with `/RNDWhich range ?~` which deletes the [Which range ?] temporary range name to leave a clean worksheet.

[3] Sort Ascending by One Key

	A	B	C	D	E
1	*---A macro to ascending sort by primary-key only				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	SORT1ASC	{WINDOWSOFF}{PANELOFF}/DSRQ/RNCWhich range ?~~/RND Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~ {BS}{BS}{?}~{GOTO}Which range ?~			
12	!	{WINDOWSOFF}{PANELOFF}/DSRDWhich range ?~{WINDOWSON} {PANELON}P{?}~A~G/DSRQ/RNDWhich range ?~			

This macro is almost identical to the previous SORT1DES.WK1 macro except that this macro sorts the data in ascending order.

[3] Sort Descending by Two Keys

```
     A                    B                    C                    D                    E
 1 *---A macro to descending sort by primary-key and secondary key
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 SORT2DES        {WINDOWSOFF}{PANELOFF}/DSRQ/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{GOTO}Which range ?~
12 !               {WINDOWSOFF}{PANELOFF}/DSRDWhich range ?~{WINDOWSON}
                   {PANELON}P{?}~D~S{?}~D~G/DSRQ/RNDWhich range ?~
```

This macro saves you some key strokes and makes the task of sorting a range a bit easier. You concentrate only on the important issues i.e. the primary key and/or the secondary key and the range to sort, the rest is done by the macro.

The macro starts with the `/DSRQ` macro keys to reset any previous sorting information and then issues `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range, and simultaneously assigns the [Which range ?] range name to the same range, which acts as a prompt. Next the macro issues `{GOTO}Which range ?~` to move the cell pointer to the upper left cell of the range to sort. Again, the macro issues `{WINDOWSOFF}{PANELOFF}` to the freeze screen and panel display activities, and `/DSRDWhich range ?~` to define [Which range ?] as the range to sort. Now the macro issues `{WINDOWSON}{PANELON}` to resume the screen and panel activities, allowing you to point to the primary and the secondary keys for the sorting operation.

Next the macro issues `P{?}` to pause the macro wait until you point to the column of the primary key. When you press ENTER, the macro issues the tilde "`~`" which is the same as the ENTER key and continues with `D~S{?}` which instruct Lotus that the sort is in descending order. The macro pauses again and waits until you point to the column of the secondary key. When you press ENTER, the macro issues the tilde "`~`" and continues with `D~G` which instruct Lotus that the sort with reference to the secondary key is also a descending sort and executes the sort. Next the macro issues `/DSRQ` to reset the data sort information, and continues with `/RNDWhich range ?~` which deletes the [Which range ?] temporary range name to leave a clean worksheet.

[3] Sort Ascending by Two Keys

```
     A                    B                    C                    D                    E
 1 *---A macro to ascending sort by primary-key and secondary key
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 SORT2ASC        {WINDOWSOFF}{PANELOFF}/DSRQ/RNCWhich range ?~~/RND
                   Which range ?~Which range ?~{BS}{BS}{?}~{GOTO}
                   Which range ?~
12 !               {WINDOWSOFF}{PANELOFF}/DSRDWhich range ?~{WINDOWSON}
                   {PANELON}P{?}~A~S{?}~A~G/DSRQ/RNDWhich range ?~
```

This macro is almost identical to the previous SORT2DES.WK1 macro except that this macro sorts the data in ascending order.

[6] Sort by Multiple Keys

```
     A                B                C                D                E
 1 *---A macro to SORT by MULTIPLE KEYS
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Highlight the data range PLUS one more empty column to the right,
 6     the macro uses the extra column for the sorting process.
 7 *---Works with labels only. Example: zip codes must be entered as labels
 8                      WILL NOT WORK WITH RELEASE 3
 9 !
10 \Z              {BREAKON}
11 SORTKEYS        {WINDOWSOFF}{PANELOFF}/DSRQ/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}{GOTO}Which range ?~
12 !               {LET here2041,@CELLPOINTER("address")}~{RIGHT @COLS
                   (Which range ?)-1}/RNCdummy041~~{LET dummy041,"+""""""}~
13 cont1041        {GOTO}{here2041}~{PANELON}{WINDOWSON}Move ROWWISE to
                   next sort key and press [ENTER] or [ESC] or [S]ort ...
                   {GET key1041}{ESC 6}
                   {IF key1041="{ESC}"}{BRANCH key1041}
14 !               {IF key1041="~"}{ESC}{WINDOWSOFF}{PANELOFF}{LET here2041
                   ,@CELLPOINTER("address")}~{LET here1041,@CELLPOINTER
                   ("col")}~{temp1041}
15 !               {IF key1041="{RIGHT}"#OR#key1041="{LEFT}"}{ESC}{key1041}
                   {?}~{WINDOWSOFF}{PANELOFF}{LET here2041,@CELLPOINTER
                   ("address")}~{LET here1041,@CELLPOINTER("col")}~
                   {temp1041}
16 !               {IF @UPPER(key1041)="S"}{GOTO}dummy041~{EDIT}{HOME}
                   {DEL}~/C~.{DOWN @ROWS(Which range ?)-1}~{temp2041}
                   {BRANCH ret041}
17 !               {BRANCH cont1041}
18 !
19 here1041             10
20 !
21 key1041         S
22 !
23 here2041        $J$37
24 !
25 ret041
26 !
27 temp1041        {GOTO}dummy041~{EDIT}&{HOME}{DEL}{END}{DOWN}{UP}
                   {LEFT -here1041+@CELL("col",dummy041)}~{DOWN}{UP}{EDIT}
                   {HOME}'~{GOTO}{here2041}~
28 !
29 temp2041        /RV.{END}{DOWN}~~/DSDWhich range ?~P~{WINDOWSON}
                   {PANELON}{?}~G{WINDOWSOFF}{PANELOFF}/RE.{END}{DOWN}~
                   {GOTO}Which range ?~/RNDWhich range ?~/RNDdummy041~
```

One of the limitations imposed by Lotus 1-2-3 is the number of sort keys available. In Lotus 2.0/2.01/2.2/2.3/2.4 you can use only two sort keys, in the 3-D releases more keys can be used. The macro displayed here can sort a database of virtually unlimited sort keys; the only limitation is that Lotus can accept up to 240 characters in a cell, which puts the limit on the number of sort keys. This will be clear shortly. Let's look at the following simple database:

```
     A          B          C            D          E
 1   LNAME      FNAME      PROFESSION   SALARY
 2   Jimmy      Carter     President    1000000
 3   Jack       Lemon      Actor        5000000
 4   John       Brice      Accountant   100000
 5   Herbert    Samuel     Minister     200000
```

The database has four data fields. If we need to sort the database on more than two sort keys, we have a problem. To solve the problem let us combine the data of the four fields into a fifth field

the TEMP_FIELD:

	A	B	C	D	E
1	LNAME	FNAME	PROFESSION	SALARY	TEMP_FIELD
2	Jimmy	Carter	President	1000000	JimmyCarterPresident1000000
3	Jack	Lemon	Actor	5000000	JackLemonActor5000000
4	John	Brice	Accountant	100000	JohnBriceAccountant100000
5	Herbert	Samuel	Minister	200000	HerbertSamuelMinister200000

All we have left to do is to sort the new database with respect to the TEMP_FIELD. The result will be as though we have sorted it by all four fields. Now it should be clear that the 240 characters limitation in a cell dictates the maximum number of sort keys. The macro here works exactly the same way; it creates a temporary field which contains the sum of the fields that you chose and sorts the database with respect to the temporary field. The macro uses string formulas to add the fields. If the database contains values, they must be changed to labels before sorting and must be changed back to values after sorting. To easily transform the database into labels and vice versa, see the DO_LABEL.WK1 and the UN_LABEL.WK1 macros.

	A	B	C	D	E
11	SORTKEYS	{WINDOWSOFF}{PANELOFF}/DSRQ/RNCWhich range ?~~/RND Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~ {BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}{GOTO}Which range ?~			

The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands which freeze the screen and panel display activities. Next the macro issues /DSRQ which reset the sorting area in case a previous sorting process took place. Now that the worksheet is fresh, the macro uses the "safe technique" to temporarily assign the [Which range ?] range name to the database range and simultaneously uses the range name as a prompt. Now that the range name is defined, the macro issues {GOTO}Which range ?~ which moves the cell pointer to the upper left cell of the database named as [Which range ?].

	A	B	C	D	E
12	!	{LET here2041,@CELLPOINTER("address")}~{RIGHT @COLS (Which range ?)-1}/RNCdummy041~~{LET dummy041,"+""""""}~			

The macro continues with {LET here2041,@CELLPOINTER("address")} which stores the current cell pointer address in cell [here2041] for later use, and issues {RIGHT @COLS (Which range ?)-1} to move the cell pointer to the column right to the database range. Notice how we used the {RIGHT} command with the @COLS function to "know" how many columns to move to the right. Next the macro issues /RNCdummy041~~ to assign the [dummy041] range name to the first cell of the new column. To be able to sum or combine the data, the macro creates a formula. The formula has to start with the "+" character, so the macro issues {LET dummy041,"+""""""}, which writes the '+"" string into the current cell named [dummy041].

	A	B	C	D	E
13	cont1041	{GOTO}{here2041}~{PANELON}{WINDOWSON}Move ROWWISE to next sort key and press [ENTER] or [ESC] or [S]ort ... {GET key1041}{ESC 6} {IF key1041="{ESC}"}{BRANCH key1041}			
14	!	{IF key1041="~"}{ESC}{WINDOWSOFF}{PANELOFF}{LET here2041 ,@CELLPOINTER("address")}~{LET here1041,@CELLPOINTER ("col")}~{temp1041}			
15	!	{IF key1041="{RIGHT}"#OR#key1041="{LEFT}"}{ESC}{key1041} {?}~{WINDOWSOFF}{PANELOFF}{LET here2041,@CELLPOINTER ("address")}~{LET here1041,@CELLPOINTER("col")}~ {temp1041}			
16	!	{IF @UPPER(key1041)="S"}{GOTO}dummy041~{EDIT}{HOME}			

```
                {DEL}~/C~.{DOWN @ROWS(Which range ?)-1}~{temp2041}
                {BRANCH ret041}
17 !            {BRANCH cont1041}
```

To allow you to pick the keys to sort, the macro issues the indirect `{GOTO}{here2041}~` macro command which returns the cell pointer to its previous location, and then issues `{PANELON}` `{WINDOWSON}` to resume the screen and panel display activities. Next the macro writes the

```
Move ROWWISE to next sort key and press [ENTER] or [ESC] or [S]ort ...
```

prompt message directly into the panel, and then issues `{GET key1041}` which halts the macro execution and allows you to read the message and press a key. When you press a key, Lotus stores the key in cell [key1041]. The macro continues with `{ESC 6}`, which clears the message from the panel, and returns to the READY mode before Lotus writes the message into the current cell. Now the macro starts a series of `{IF}` conditions to check what key you pressed. The first is `{IF key1041="{ESC}"}` which checks if the key is the ESC key. If so, the macro issues `{BRANCH key1041}` which routes macro control to the [key1041] routine. However, this routine contains only `{ESC}`; therefore the macro executes the `{ESC}` command and reaches an empty cell and quits.

Next is `{IF key1041="~"}` which checks if you pressed the ENTER key. If so, the macro issues `{ESC}` which clears the message from the panel and then issues `{WINDOWSOFF}` `{PANELOFF}` which freeze the screen and panel display activities. The macro continues with `{LET here2041,@CELLPOINTER("address")}` which records the current cell pointer position in cell [here2041], and then issues `{LET here1041,@CELLPOINTER("col")}` which records the current column's number in cell [here1041]. Now the macro issues the `{temp1041}` routine command which starts the [temp1041] routine.

```
     A          B                C          D          E
27 temp1041     {GOTO}dummy041~{EDIT}&{HOME}{DEL}{END}{DOWN}{UP}
                {LEFT -here1041+@CELL("col",dummy041)}~{DOWN}{UP}{EDIT}
                {HOME}'~{GOTO}{here2041}~
```

The [temp1041] routine starts with `{GOTO}dummy041~`, which moves the cell pointer to the cell named [dummy041] containing the `'+""` string and then issues `{EDIT}` to enter the EDIT mode. Now the macro writes the ampersand "`&`" character into the panel and then issues `{HOME}` to move the cursor to the beginning of the panel. The `{DEL}` command deletes the apostrophe "`'`" and turns the text in the panel into a formula. Now the macro issues `{DOWN}` `{UP}` to enter to POINT mode and then moves the cell pointer to the left to point to the column, containing the sorting key.

The macro issues `{LEFT -here1041+@CELL("col",dummy041)}` which uses the `-here1041+@CELL("col",dummy041)` formula to calculate how many columns to move the cell pointer to point to the sorting key. Previously, the macro used `{LET here1041 ,@CELLPOINTER("col")}` to record the number of the column with the sorting key. Next the macro issues `~{DOWN}{UP}` to force Lotus to write the content of the panel into the current cell. Next the macro continues with `{EDIT}{HOME}'~` to write the apostrophe to change the formula back to text, and issues `{GOTO}{here2041}~` to move the cell pointer to the column with the sorting key.

Note: It may seem that the tilde "~" is enough to force Lotus to write the content of the panel

into the current cell, and there is no need for {DOWN}{UP} following the tilde "~"; however from our experience, the tilde "~" is not always enough, therefore we used {DOWN}{UP} which have been proven safer. When the tilde "~" is not working correctly, we add {DOWN} {UP} to enter the data in the panel into the cell.

This code may seem quite confusing, but we will use sample database to see step by step how the macro combines the data to the added column.

	A	B	C	D	E
1	LNAME	FNAME	PROFESSION	SALARY	
2	Jimmy	Carter	President	1000000	+""
3	Jack	Lemon	Actor	5000000	
4	John	Brice	Accountant	100000	
5	Herbert	Samuel	Minister	200000	

Let's assume that you want to sort the A2..D5 range. When the macro prompts you to point to the sorting key, you point to the A2 cell. Previously the macro wrote the '+"" text into the E2 cell [dummy014]. Just before the macro starts the [temp1041] routine, it records the A2 address in cell [here2041] and the column number "1" in cell [here1041]. Next it issues {GOTO}dummy041~ to move the cell pointer to the E2 cell which contains the '+"" text. Therefore when the macro issues {EDIT} and writes the ampersand "&" character to the panel, the panel displays:

```
'+""&_
```

where the underscore represents the cursor position. To add the first key to the text on the panel, the macro first needs to delete the apostrophe to change the text in the panel into an active formula. Therefore the macro issues {HOME} which moves the cursor to the beginning of the text in the panel now displaying:

```
'+""&
^
```

where the "^" under the apostrophe represents the cursor position. Now the macro issues {DEL}, which deletes the apostrophe and changes the panel to display:

```
+""&
^
```

where the "^" under the "+" character represents the cursor position. Next the macro issues {DOWN}{UP} to enter the POINT mode, therefore the panel displays:

```
+""&E2
```

To change the E2 in the panel to the correct cell address the macro issues {LEFT -here1041+@CELL("col",dummy041)} which is equivalent to {LEFT 4} because the -here1041+@CELL("col",dummy041) formula returns the value "4", therefore the panel displays:

```
+""&A2
```

Next the macro issues the tilde "~" and writes the content of the panel into the E2 cell. Next, the macro issues {EDIT} which enters the EDIT mode, displayed as:

```
+""&A2
```

The macro issues {HOME} to move the cursor to the beginning of the text in the panel and writes the apostrophe "'" which changes the formula back to text, therefore the panel displays:

```
'+""&A2
```

To write the content of the panel into the E2 cell, the macro again issues the tilde "~". When the [temp1041] routine is finished, the macro routes control back to the B15 cell in the [cont1041] routine. Let's continue with the code in the [cont1041] routine.

	A	B	C	D	E
13	cont1041	{GOTO}{here2041}~{PANELON}{WINDOWSON}Move ROWWISE to next sort key and press [ENTER] or [ESC] or [S]ort ... {GET key1041}{ESC 6} {IF key1041="{ESC}"}{BRANCH key1041}			
14	!	{IF key1041="~"}{ESC}{WINDOWSOFF}{PANELOFF}{LET here2041 ,@CELLPOINTER("address")}~{LET here1041,@CELLPOINTER ("col")}~{temp1041}			
15	!	{IF key1041="{RIGHT}"#OR#key1041="{LEFT}"}{ESC}{key1041} {?}~{WINDOWSOFF}{PANELOFF}{LET here2041,@CELLPOINTER ("address")}~{LET here1041,@CELLPOINTER("col")}~ {temp1041}			
16	!	{IF @UPPER(key1041)="S"}{GOTO}dummy041~{EDIT}{HOME} {DEL}~/C~.{DOWN @ROWS(Which range ?)-1}~{temp2041} {BRANCH ret041}			
17	!	{BRANCH cont1041}			

The code in B15 checks if you pressed one of the RIGHT or LEFT direction keys. If so, the macro issues {ESC} to clear the

```
Move ROWWISE to next sort key and press [ENTER] or [ESC] or [S]ort ...
```

prompt message from the panel and then issues the {key1041} routine command. However the [key1041] routine holds the key you pressed i.e. the {RIGHT} or the {LEFT}. Therefore when the macro issues {key1041}, the macro either moves the cell pointer to the left or to the right as you meant to. Then the macro follows with {?} which allows you to continue and use the direction keys.

Note: The use of the {?} command to allow you to use the direction keys is not the safest way to control the keys that you use, because you can use any Lotus key that you desire including the slash "/" key. However it seems adequate in this case because there is no reason for you to use other keys when you need to point to the sorting keys.

When you press ENTER, the macro issues {WINDOWSOFF}{PANELOFF} to resume the screen and panel display activities, and {LET here2041,@CELLPOINTER("address")} to record the current cell pointer position in cell [here2041], and then issues {LET here1041 ,@CELLPOINTER("col")} to record the current column's number in cell [here1041]. Then it issues {temp1041}, which starts the [temp1041] routine exactly as we have seen earlier in the {IF key1041="~"} condition.

The next condition is {IF @UPPER(key1041)="S"}, which checks if you pressed either of the "s" or the "S" keys to sort the database. If so, the macro issues {GOTO} dummy041~ which moves the cell pointer to the E2 cell in our sample database, and then issues {EDIT} {HOME}

{DEL}~ sequence to delete the apostrophe and turn the text into a formula. Then the macro follows with /C~.{DOWN @ROWS(Which range ?)-1}~, which copies the formula in the E2 cell to the E3..E5 range. The result is that every cell in the E2..E5 range contains a formula that combines the sort keys into one string. But, before the macro can sort the database with respect to the strings in the "E" column as the primary key, it needs to transform the formulas in the E2..E5 into real text. Therefore the macro issues the {temp2041} routine command which start the [temp2041] routine.

	A	B	C	D	E
29	temp2041	/RV.{END}{DOWN}~~/DSDWhich range ?~P~{WINDOWSON}{PANELON}{?}~G{WINDOWSOFF}{PANELOFF}/RE.{END}{DOWN}~{GOTO}Which range ?~/RNDWhich range ?~/RNDdummy041~			

The {temp2041} routine issues /RV.{END}{DOWN}~~ which turn all the formulas in the E2..E5 range, the last column of the database range,[Which range ?], into real text. The macro continues with /DSDWhich range ?~P~ to start the sort process of the database with the last temporary column as the primary key. Then the macro issues {WINDOWSON}{PANELON} to resume the screen and panel display activities, and then the {?} allows you to choose between the **D**escending or the **A**scending sort options. When you choose the sort method and press ENTER, the macro issues ~G which instruct Lotus to finish the sort.

To clean the worksheet after the sort, the macro issues {WINDOWSOFF}{PANELOFF} which freeze the screen and the panel display activities, while the macro continues with /RE.{END}{DOWN}~ and erases the temporary last column of the database. Then the macro issues {GOTO}Which range ?~ to return the cell pointer to the upper left cell of the database. Last the macro issues /RNDWhich range ?~/RNDdummy041~ to delete the [Which range ?] and the [dummy041] temporary range names, to leave a clean worksheet with a sorted database.

When the [temp2041] routine is finished, the macro routes control back to the [cont1041] routine which issues {BRANCH ret041} and routes macro control to the empty [ret041] routine and quits. If you do not press the "S" or the "s" key, the macro issues {BRANCH cont1041} which routes the macro control back to the beginning of the [cont1041] routine to allow you to choose more sorting keys.

Date Macros

[4] Number of Days Between Two Dates

```
      A                B                C                D                E
 1 *---A macro to calculate the number of days between two dates
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---The macro prompts for the first date and then the second date
 6 *---Make sure that the keyboard is in [INSERT] mode and not [OVR]
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 DAYSDIFF        @DATE(yy,mm,dd){EDIT}{HOME}'{END}{LEFT 9}{?}{END}{EDIT}-
12 !               @DATE(yy,mm,dd){EDIT}{LEFT 9}{?}{HOME}{DEL}{CALC}{HOME}
13 !               There are {END} days difference. Press any key to quit.
11                 {GET key203}{ESC}{CALC}
14 !
15 key203          ~
```

Many times we need to calculate how many days are between two dates. This macro makes date entering easier, and displays a message in the panel with the number of days between the two dates. The macro starts with the technique we call "directly writing to the panel" and types the `@DATE(yy,mm,dd)` function text into the panel, and issues `{EDIT}{HOME}'` to turn it into a label. To save you key presses, the macro issues `{END}{LEFT 9}` placing the cursor on the first "`y`" in the `@DATE(yy,mm,dd)` function text and issues `{?}` to pause for your response. The panel displays:

```
'@DATE(yy,mm,dd)
       ^
```

Where the "^" represents the cursor location. Press the INS key to change into OVR mode and type the year, month and day of the date, and press ENTER. The macro issues `{END}{EDIT}` to move the cursor to the end of the text in the panel and types `-@DATE (yy,mm,dd)`, and again issues `{EDIT}{LEFT 9}{?}` to move the cursor to the first "`y`" in the second date function. The panel displays:

```
'@DATE(yy,mm,dd)-@DATE(yy,mm,dd)
                       ^
```

where the "^" represents the cursor location. You are expected to fill the second date function instead of the "yy,mm,dd" and press ENTER. The macro issues `{HOME}{DEL}` to move the cursor to the beginning of the text in the panel and delete the apostrophe to change the text back into a formula. When the formula in the panel is complete, the macro issues `{CALC}` which is the same as pressing the CALC (F9) key from the keyboard. Pressing the CALC key while in the EDIT mode when a formula is in the panel, transforms the formula in the panel into its result. Therefore `{CALC}` transforms the formula into a number which is exactly the number of days between the two dates. Let's assume that you inserted the following dates:

```
@DATE(92,12,23)-@DATE(92,8,23)
```

Therefore the result after `{CALC}` is:

```
122
```

To display the result, the macro issues `{HOME}` which moves the cursor to the first number in the panel and writes the "`There are `" string into the panel, which now displays:

```
There are 122
```

Next the macro issues `{END}`, which moves the cursor to the end of the text in the panel. Then the macro continues to type the " `days difference. Press any key to quit.`" text into the panel which displays:

```
There are 122 days difference. Press any key to quit.
```

To allow you to read the message prompt in the panel, the macro issues `{GET key203}` which stops the macro execution and forces the macro to wait until you press a key. When you press a key, Lotus stores it in the B15 cell named [key203]. Next the macro issues `{ESC}` to clear the text from the panel before Lotus writes it into the current cell.

This is an excellent example of panel manipulating, once to use the panel to easily enter dates and then to use the result of the date formula as part of an answering message in the panel.

[4] Set the System Time and Date from Lotus (2.0/2.01)

```
      A                B                C                D                E
 1 *---A macro to set the system time and date
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---The macro creates a batch file C.BAT to update the date and time
 6 *---When the DOS prompt appears press [C] and [ENTER] to run C.BAT
 7     the macro will automatically EXIT to 1-2-3
 8 !
 9 !
10 \Z              {BREAKON}
11 SETTIME         {GETLABEL "Type the DOS default DRIVE/DIRECTORY - ",
                   drive037}
12 !               {OPEN drive037&"\C.BAT",W}{QUIT}
13 !               {WRITELN "TIME"}
14 !               {WRITELN "DATE"}
15 !               {WRITELN "EXIT"}{CLOSE}
16 !               /S
17 !               {CALC}
18 !
19 drive037        C:\123
```

This macro allows you to set the time and date from within Lotus 1-2-3, version 2.0/2.01, without leaving Lotus. This macro creates a batch file which contains the "TIME", "DATE" and "EXIT" DOS commands and names it C.BAT. Because this macro is for Lotus 2.0/2.01, it cannot use the `{SYSTEM}` macro command as in Lotus 2.2 and up. The macro uses the `/S` macro keys to shell to DOS, you type "C" and press ENTER to manually run the C.BAT file. When the C.BAT file runs, it executes the TIME and next the DATE DOS commands. When you finish updating the time and the date, the file issues the EXIT command and returns to Lotus.

```
{GETLABEL "Type the DOS default DRIVE/DIRECTORY - ",drive037}
```

displays the "`Type the DOS default DRIVE/DIRECTORY -` " prompt message in the panel. You need to enter the drive and directory where Lotus files are located (the drive/directory where Lotus shells when you issue the /S keys from the keyboard). When you type the drive and the directory and press ENTER, Lotus stores it in cell [drive037]. Next the macro issues the `{OPEN drive037&"\C.BAT",W}` macro command which opens the batch file. For example: if the directory where Lotus is located is the C:\123 directory, the macro issues the `{OPEN C:\123\C.BAT,W}` command which opens the C.BAT file in the C:\123 directory for writing. Notice the `drive037&"\C.BAT"` string formula inside the `{OPEN}` macro command. Also notice the `{QUIT}` macro command, which follows the `{OPEN}` macro command **at the same line**. This command is executed only if `{OPEN}` fails due to an error. This little known property of the `{OPEN}` command is sometimes used to verify if a file exists on the disk (see the FILEXIST.WK1 macro).

Now that the C.BAT file exists in the disk and is opened for writing into it, the macro issues the three `{WRITELN}` macro commands which write the "`TIME`", "`DATE`" and "`EXIT`" strings to the C.BAT file, and then issues the `{CLOSE}` command, which closes the opened file, and issues the `/S` macro keys to shell to DOS. The screen now displays:

```
C:\123>
```

If you issue the `TYPE C.BAT` DOS command and press ENTER, the screen displays:

```
C:\123>TYPE C.BAT
TIME
DATE
EXIT

C:\123>
```

which shows that the C.BAT batch file contains the "TIME", "DATE" and "EXIT" DOS commands. When you type the "C" and press ENTER, DOS executes the TIME and the DATE commands. When you finish updating the date and the time and press ENTER, DOS issues the EXIT command returning control to Lotus and to the same position before the macro started.

[4] Set The System Time and Date from Lotus (2.2 and Up)

```
     A                    B                    C                    D                    E
 1 *---A macro to set the system time and date
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---The macro creates a batch file C.BAT to update the date and time
 6 *---When the DOS prompt appears press [C] and [ENTER] to run C.BAT
 7     the macro will automatically EXIT to 1-2-3
 8                      * * *  LOTUS 2.2 AND UP * * *
 9 !
10 \Z              {BREAKON}
11 SETIME22        {OPEN "\C.BAT",W}{BRANCH ret235}
12 !               {WRITELN "TIME"}
13 !               {WRITELN "DATE"}
14 !               {WRITELN "EXIT"}{CLOSE}
15 !               {SYSTEM C}
16 !
17 !
18 ret235
```

This macro allows you to set the time and date from within Lotus 1-2-3 without leaving Lotus. This macro creates a batch file which contains the "TIME", "DATE" and "EXIT" DOS commands and names it C.BAT. Because this macro is for Lotus 2.2 and up, we can use the `{SYSTEM}` macro command.

The macro starts with `{OPEN "\C.BAT",W}` which opens a file named C.BAT in the root directory of the C: drive. Next the macro issues `{WRITELN}` to write the "TIME", "DATE" and "EXIT" strings to the C.BAT file, and then issues `{CLOSE}` to close the C.BAT file. In this macro, there is no need for you to shell to DOS in order to run the C.BAT batch file (see the SETTIME.WK1 macro), the macro does it automatically using `{SYSTEM C}` which runs the C.BAT file directly from Lotus. When the macro issues `{SYSTEM C}`, Lotus shells to DOS and activates the C.BAT batch file, which issues the TIME DOS command, and the DATE DOS command. When you update the time and the date and press ENTER, DOS issues the EXIT command which automatically returns control to Lotus 1-2-3 without your interference.

[4] Set the Time and Date from Lotus (3.0 and Up)

```
        A                B                C                D                E
 1 *---A macro to set the system time and date
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---The macro creates a batch file TEMP.BAT to update the date and time
 6     and deletes it later
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 SETTIME3       {RECALC form1037}{RECALC form2037}{RECALC form3037}
12 form1037       ERR
13 !              {WRITELN "TIME"}
14 !              {WRITELN "DATE"}
15 !              {WRITELN "EXIT"}
16 !              {CLOSE}
17 form2037       ERR
18 form3037       ERR
19 !
20 !
21 rel037
```

This macro allows you to set the time and the date from within Lotus 1-2-3 without leaving Lotus. This macro creates a batch file which contains the "TIME", "DATE" and "EXIT" DOS commands and names it TEMP.BAT. Because this macro is for the 3-D releases of Lotus 3.0 and up, including Lotus for Windows, we can use the `{SYSTEM}` macro command. You may ask why we need a separate macro for the 3-D releases of Lotus, since they also allow the use of the `{SYSTEM}` macro command as do 2.2/2.3 and 2.4 (see the SETIME22.WK1 macro)? If you will try the SETIME22.WK1 in one of the 3-D releases, you will find that the macro does not work. After extensive testing, we have found that all the 3-D releases of Lotus do not recognize the `{CLOSE}` command which follows `{WRITELN "EXIT"}` if it is in the same cell, therefore the `{CLOSE}` command is in a separate cell, the cell after the `{WRITELN "EXIT"}` command.

The second and main reason is that, with the 3-D releases, Lotus introduced the `@INFO ("directory")` function which allows us to find the default directory. If you will look at the SETTIME.WK1 macro for Lotus 2.0/2.01, you will see that you have to input the drive and directory where Lotus resides specifically, and in the SETIME22.WK1 macro for Lotus 2.2/2.3 and 2.4, the macro assumes that the drive is the C: drive. Here we do not have to assume anything; the macro can use the default directory automatically. To accomplish the task, the macro makes use of dynamic string formulas to create the correct code. Therefore if you intend to key this macro into Lotus 1-2-3, you have to key the following formulas in the B12, B17 and the B18 cells of the macro instead of the ERR strings in the main listing.

```
12 !              +"{OPEN """&@INFO("directory")&"TEMP.BAT"",W}{QUIT}"
17 !              +"{SYSTEM """&@INFO("directory")&"TEMP""}"
18 !              +"{SYSTEM ""DEL "&@INFO("directory")&"TEMP.BAT""}"
```

The macro starts with the `{RECALC form1037}{RECALC form2037}{RECALC form3037}` macro commands which update the dynamic formulas in the B12, B17 and the B18 cells to reflect the current default directory. Next the macro reaches the code in the B12 cell which is the result of the `+"{OPEN """&@INFO("directory")&"TEMP.BAT"",W} {QUIT}"` dynamic formula. For example, if C:\MACROS is the default directory, then the result of this formula is `{OPEN "C:\MACROS\TEMP.BAT",W}{QUIT}` which opens the TEMP.BAT file for writing.

Next the macro issues `{WRITELN}` three times to write the "`TIME`", "`DATE`" and "`EXIT`" string to the TEMP.BAT file and then issues `{CLOSE}` to close the TEMP.BAT file. Now the macro reaches the B17 cell which contains the `+"{SYSTEM """&@INFO("directory")&"TEMP""}"` string formula which results as the `{SYSTEM "C:\MACROS\TEMP"}` command, which runs the TEMP.BAT batch file in the C:\MACROS directory. When the TEMP.BAT batch file is executed, DOS executes the TIME and DATE files. When you finish updating the date and the time and press ENTER, DOS issues the EXIT command which returns control to Lotus.

Last the macro reaches the code in the B18 cell which contains the `+"{SYSTEM ""DEL"&@INFO("directory")&"TEMP.BAT""}"` string formula which results in the `{SYSTEM "DEL C:\MACROS\TEMP"}` macro command, which deletes the TEMP.BAT file from the C:\MACROS directory.

[5] Insert Dates Easily

```
                 A                       B                          C                  D
 1 *---A macro to insert a DATE function (see the MACROHLP macro too)
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---The @DATE(Yr,Mo,Dy prompt will appear. Change the [INS] to [OVR]
 6     and type the year, month and day and press [ENTER]. The macro will
 7     add the closing bracket ")".
 8 !
 9 !
10 \Z              {BREAKON}
11 DATEMAC         @DATE(YY,MM,DD{EDIT}{LEFT 8}{?})~{WINDOWSOFF}{PANELOFF}
12 !               {MENUBRANCH menu103}
13 !
14 menu103         Full date             Standard              Quit
15 !               Full date in words as Lotus standard as a dQuit the mac
16 !               {RECALC date1103}{RECA/RF{PANELON}D{?}~~
17 !               {fulldate103}~
18 !
19 date1103                          0
20 day1103         Saturday
21 day2103                         ERR
22 month1103                       ERR
23 year1103                        ERR
24 fulldate103                     ERR
```

This macro saves you nine key strokes every time you enter a date to the worksheet and uses a custom menu, which cannot be displayed clearly on one page without overlapping. Here is the code of each menu item separately:

```
Full date
Full date in words as label
{RECALC date1103}{RECALC day1103}{RECALC month1103}
    {RECALC year1103}{RECALC day2103}{RECALC fulldate103}
{fulldate103}~

Standard
Lotus standard as a date function
/RF{PANELON}D{?}~~

Quit
Quit the macro
```

The macro also uses dynamic string formulas to create the correct code for the macro, therefore the code in the B19, B20, B21, B22, B23 and the B24 cells is the result of the following six formulas:

```
19 date1103       @CELLPOINTER("contents")
20 day1103        @CHOOSE(@MOD(B19,7),"Saturday","Sunday","Monday",
                  "Tuesday","Wednesday","Thursday","Friday")
21 day2103        @DAY(B19)
22 month1103      @CHOOSE(@MONTH(B19)-1,"January","February","March",
                  "April","May","June","July","August","September",
                  "October","November","December")
23 year1103       @YEAR(B19)+1900
24 fulldate103    +B20&" "&B22&" "&@STRING(B21,0)&", "&@STRING(B23,0)
```

If you intend to key this macro into Lotus 1-2-3, you have to key the formulas and the code of menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you to correctly key this macro, we show the full list of all the cell

contents at the end of this section. The macro directly writes the "@DATE(YY,MM,DD" string of text into the panel and enters the EDIT mode using {EDIT}. It issues {LEFT 8} to move the cursor eight characters to the left where the cursor lands on the "Y" of the "YY" and then issues {?} which halts the macro execution and waits until you insert the date.

You can make the task even easier if you press the INS key to enter the OVR mode, and then type the year, the month and the day. When you press ENTER, the macro adds the closing bracket and issues the tilde "~" which is the same as pressing the ENTER key, and issues {WINDOWSOFF}{PANELOFF}, which freeze the screen and panel activities, and then {MENUBRANCH menu103} branches to the [menu103] custom menu.

The first menu option of the custom menu is [**F**ull date]:

```
Full date
Full date in words as label
{RECALC date1103}{RECALC day1103}{RECALC month1103}
    {RECALC year1103}{RECALC day2103}{RECALC fulldate103}
{fulldate103}~
```

This menu option transforms the date that you entered into a string of the form like "Sunday May 15, 1992". The macro is based on the six formulas in the B19..B24 range, therefore the macro issues

```
{RECALC date1103}{RECALC day1103}{RECALC month1103}
    {RECALC year1103}{RECALC day2103}{RECALC fulldate103}
```

which update the following six formulas based on the inserted date

```
19 date1103       @CELLPOINTER("contents")
20 day1103        @CHOOSE(@MOD(B19,7),"Saturday","Sunday","Monday",
                  "Tuesday","Wednesday","Thursday","Friday")
21 day2103        @DAY(B19)
22 month1103      @CHOOSE(@MONTH(B19)-1,"January","February","March",
                  "April","May","June","July","August","September",
                  "October","November","December")
23 year1103       @YEAR(B19)+1900
24 fulldate103    +B20&" "&B22&" "&@STRING(B21,0)&", "&@STRING(B23,0)
```

The first formula returns the date you entered into the current cell. The second formula uses the @CHOOSE Lotus function to extract the day string. The third formula calculates the day as number using the @DAY function. The fourth formula again uses the @CHOOSE function to extract the month string. The fifth formula calculates the year number using the @YEAR function, and the sixth formula uses the results of the formulas in the B20, B22, B21 and the B23 cell and the @STRING function to create the full date string. Next the macro issues {fulldate103} which starts the [fulldate103] routine, which writes the content of the cell named [fulldate103] into the panel because it contains the date as a text (the result of the formula in the B24 cell). Last the macro issues the tilde "~" macro command which writes the full date string into the current cell. The second menu option is the [**S**tandard] menu option:

```
Standard
Lotus standard as a date function
/RF{PANELON}D{?}~~
```

When you choose this option, the macro issues /RF to start the date format sequence and then issues {PANELON} to resume the panel display activity to allow you to see and use the

Lotus standard date menu. Next the macro issues the "{D}" macro key and then issues {?} which halts the macro execution and waits until you choose the date format from the menu.

The last menu option is [**Q**uit]:

```
Quit
Quit macro
```

When you choose this menu option, the macro reaches an empty cell and quits.

To help you key this macro into Lotus 1-2-3, here is the full list of all the cell contents.

```
A1: U '*---A macro to insert a DATE function (see the MACROHLP macro too)
A2: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A3: '    range names in this column (starts with the \Z macro name)
A4: '*---Hold the [ALT] key and press [Z] to activate the macro
A5: '*---The @DATE(Yr,Mo,Dy prompt will appear. Change the [INS] to [OVR]
A6: '    and type the year, month and day and press [ENTER]. The macro will
A7: '    add the closing bracket ")".
A8: '!
A9: '!
A10: U '\Z
B10: [W10] '{BREAKON}
A11: U 'DATEMAC
B11: (,2) [W10] '@DATE(Yr,Mo,Dy{EDIT}{LEFT 8}{?})~{WINDOWSOFF}{PANELOFF}
A12: '!
B12: [W10] '{MENUBRANCH menu103}
A13: '!
A14: 'menu103
B14: [W10] 'Full date
C14: [W10] 'Standard
D14: [W12] 'Quit
A15: '!
B15: [W10] 'Full date in words as label
C15: [W10] 'Lotus standard as a date function
D15: [W12] 'Quit the macro
A16: '!
B16: [W10] '{RECALC date1103}{RECALC day1103}{RECALC month1103}
           {RECALC year1103}{RECALC day2103}{RECALC fulldate103}
C16: [W10] '/RF{PANELON}d{?}~~
A17: '!
B17: [W10] '{fulldate103}~
A18: '!
A19: 'date1103
B19: U [W10] @CELLPOINTER("contents")
A20: 'day1103
B20: U [W10] @CHOOSE(@MOD(B19,7),"Saturday","Sunday","Monday","Tuesday"
             ,"Wednesday","Thursday","Friday")
A21: 'day2103
B21: U [W10] @DAY(B19)
A22: 'month1103
B22: U [W10] @CHOOSE(@MONTH(B19)-1,"January","February","March","April",
             "May","June","July","August","September","October","November"
             ,"December")
A23: 'year1103
B23: U [W10] @YEAR(B19)+1900
A24: 'fulldate103
B24: U [W10] +B20&" "&B22&" "&@STRING(B21,0)&", "&@STRING(B23,0)
```

[5] Same Day for List of Months

```
         A                    B              C          D          E          F
 1 *---A macro to insert the same day for list of months (column, row or
 2     sheet)
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                      IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 SAMEDAY         {WINDOWSOFF}{PANELOFF}{LET rel159,@INFO("release")}~
                   /RNCwx~~/RNDwx~/RNCwx~~{PANELON}
12 !               {GETNUMBER "How many months ? ",months159}~
13 !               {GETLABEL "Enter the first date (MM/DD/YY) or press
                   [ENTER] for today: ",wx}{PANELOFF}
14 !               {IF wx=""}{LET wx,+@INT(@NOW)}~{BRANCH loop1159}
15 !               {EDIT}{HOME}{DEL}@DATEVALUE("{END}")~
16 loop1159        {MENUBRANCH menu159}
17 !
18 menu159         Columnwise  Rowwise  In       Quit
19 !               Insert the dInsert thInsert thQuit the macro
20 !               {WINDOWSOFF}{WINDOWSO{IF @LEFT/RNDwx~
21 !               /RNDwx~     /RNDwx~  {WINDOWSOFF}{RECALC steps3159}{st
22 !                                    /RNDwx~
23 !
24 steps1159       {DOWN}{formula159}~/C~.{DOWN 18}~{UP}/RFD4.{END}{DOWN}~
                   /RV.{END}{DOWN}~~
25 !
26 steps2159       {RIGHT}{formula159}~/C~.{RIGHT 18}~{LEFT}/RFD4.{END}
                   {RIGHT}~/RV.{END}{RIGHT}~~
27 !
28 steps3159       {NS}{formula159}~/C~.{NS 18}~{PS}/RFD4.{END}{NS}~/RV.
                   {END}{NS}~~
29 !
30 months159             20
31 !
32 formula159      @IF(@DAY(wx)>3,wx+28-@DAY(wx+28),wx+31-@DAY(wx+31))+
                   @IF(@MONTH(wx)=1,@MIN(@IF(@MOD(@YEAR(wx),4)=0,29,28)
                   ,@DAY($wx)),@IF(@MONTH(wx)=3#OR#@MONTH(wx)=5#OR#@MONTH
                   (wx)=8#OR#@MONTH(wx)=10,@MIN(30,@DAY($wx)),@DAY($wx)))
33 !
34 rel159          3.10.00
35 ret159
```

This macro inserts the same date for a list of months. For example you can create this list:

```
01/16/92
02/16/92
03/16/92
04/16/92
05/16/92
06/16/92
07/16/92
08/16/92
09/16/92
10/16/92
11/16/92
12/16/92
```

starting January 16, 1992, or the this list

```
01/31/92
02/29/92
03/31/92
```

```
04/30/92
05/31/92
06/30/92
07/31/92
08/31/92
09/30/92
10/31/92
11/30/92
12/31/92
```

starting January 31, 1992. However notice that in February, the macro inserted the 02/29/92 date because in there were 29 days in February. The same is true for months with less than 31 days. The macro can produce lists in the row direction, the column direction, and the "Z" direction in multi sheet 3-D worksheet. The lists can be any size.

This macro uses a custom menu with four menu items, which cannot be displayed in full on one page, therefore we show them separately. If you intend to key the macro into Lotus 1-2-3, you should use the following code for the menu items.

```
Columnwise
Insert the dates along a column
{WINDOWSOFF}{RECALC steps1159}{steps1159}
/RNDwx~

Rowwise
Insert the dates along a row
{WINDOWSOFF}{RECALC steps2159}{steps2159}
/RNDwx~

In
Insert the dates along the sheets (release 3 only!)
{IF @LEFT(rel159,1)="@"}/RE~{BRANCH ret159}
{WINDOWSOFF}{RECALC steps3159}{steps3159}
/RNDwx~

Quit
Quit the macro
/RNDwx~
```

This macro uses string formulas to create dynamic code. If you intend to key it into Lotus 1-2-3, you should key the following formulas into the B24, B26 and the B28 cells of the macro, instead of the code as it appears in the main listing.

```
24 steps1159       +"{DOWN}{formula159}~/C~.{DOWN "&@STRING(B30-2,0)&"}~
                   {UP}/RFD4.{END}{DOWN}~/RV.{END}{DOWN}~~"
26 steps2159       +"{RIGHT}{formula159}~/C~.{RIGHT "&@STRING(B30-2,0)&"}~
                   {LEFT}/RFD4.{END}{RIGHT}~/RV.{END}{RIGHT}~~"
28 steps3159       +"{NS}{formula159}~/C~.{NS "&@STRING(B30-2,0)&"}~{PS}
                   /RFD4.{END}{NS}~/RV.{END}{NS}~~"
```

We will analyze these formulas when we reach their code.

	A	B	C	D	E	F
11	SAMEDAY	{WINDOWSOFF}{PANELOFF}{LET rel159,@INFO("release")}~ /RNCwx~~/RNDwx~/RNCwx~~{PANELON}				
12	!	{GETNUMBER "How many months ? ",months159}~				
13	!	{GETLABEL "Enter the first date (MM/DD/YY) or press [ENTER] for today: ",wx}{PANELOFF}				
14	!	{IF wx=""}{LET wx,+@INT(@NOW)}~{BRANCH loop1159}				
15	!	{EDIT}{HOME}{DEL}@DATEVALUE("{END}")~				
16	loop1159	{MENUBRANCH menu159}				

To freeze screen and panel activities the macro issues {WINDOWSOFF}{PANELOFF} and uses

{LET rel159,@INFO("release")}~ to store the result of the @INFO("release") function in cell [rel159]. Later on the macro will use it to check if you are using a 2-D or a 3-D Lotus release. To assign the [wx] range name to the current cell, the macro uses the "safe technique": /RNCwx~~/RNDwx~/RNCwx~~. We have chosen a range name that has no resemblance to a column heading. After assigning the range name, the macro issues {PANELON} to allow panel activity and continues with {GETNUMBER "How many months ? ",months159}~ which displays "How many months ? " in the panel. Lotus stores your response in the B30 cell named [months159]. Next the macro issues a second {GETLABEL} command which displays:

```
"Enter the first date (MM/DD/YY) or press [ENTER] for today:
```

in the panel. The macro stores the date you insert in cell [wx], but if you just press the ENTER key the macro stores the null "" string in cell [wx]. Therefore the next {IF wx=""} condition is true and the macro issues the {LET wx,+@INT(@NOW)}~ advanced macro command to enter the current date into cell [wx], and then issues {BRANCH loop1159} to branch to the [loop1159] routine which uses {MENUBRANCH menu159} to activate the custom menu. If you insert the date as the MM/DD/YY string, the macro issues {EDIT}{HOME} {DEL}@DATEVALUE("{END}")~ to manipulate the panel and change it into an active date and uses {MENUBRANCH menu159} to activate the custom menu. The first menu option is [**C**olumnwise]:

```
Columnwise
Insert the dates along a column
{WINDOWSOFF}{RECALC steps1159}{steps1159}
/RNDwx~
```

This routine starts with {WINDOWSOFF} that freezes the screen activities while the macro uses {RECALC steps1159} to update the dynamic formula in cell [steps1159] and then issues the {steps1159} routine command to start the [steps1159] routine. Before continuing, let's look at the formula in cell [steps1159] and then at the code of the [steps1159] routine which is the result of this formula:

```
24 steps1159        +"{DOWN}{formula159}~/C~.{DOWN "&@STRING(B30-2,0)&"}~
                    {UP}/RFD4.{END}{DOWN}~/RV.{END}{DOWN}~~"
```

Before we can go into the code that this formula creates, we can see that the formula uses the @STRING(B30-2,0) function which uses the number of months stored in B30, [months159], to create the code for how many cells to move down. In the macro list, we can see that [months159] contains the number 20. Therefore the @STRING(B30-2,0) function returns 18. The result of this formula is the code:

A	B	C	D	E	F
24 steps1159	{DOWN}{formula159}~/C~.{DOWN 18}~{UP}/RFD4.{END}{DOWN}~ /RV.{END}{DOWN}~~				

The [steps1159] routine uses the {DOWN} command to move the cell pointer one cell down and then issues the {formula159} routine command.

A	B	C	D	E	F
32 formula159	@IF(@DAY(wx)>3,wx+28-@DAY(wx+28),wx+31-@DAY(wx+31))+ @IF(@MONTH(wx)=1,@MIN(@IF(@MOD(@YEAR(wx),4)=0,29,28) ,@DAY($wx)),@IF(@MONTH(wx)=3#OR#@MONTH(wx)=5#OR#@MONTH (wx)=8#OR#@MONTH(wx)=10,@MIN(30,@DAY($wx)),@DAY($wx)))				

The {formula159} routine contains a 240 character formula to calculate the same day date as in the cell named [wx] but one month ahead. Now we can see why we have used a two characters long name. If we used a longer name, we could not fit the formula into one cell. Therefore, when the macro issues {formula159}, Lotus injects the formula is into the panel and the tilde "~" writes it into the current cell. To insert the 18 dates, the macro uses /C~.{DOWN 18}~. The number 18 in the {DOWN 18} is the result of the @STRING(B30-2 ,0) function. Now we have the 20 dates as required, but we need to format them, therefore the macro issues {UP}/RFD4.{END}{DOWN}~ to format the date to type 4 format. To turn all the formulas into date values, the macro issues /RV.{END}{DOWN}~~.

To end the [Columnwise] menu option, the macro issues /RNDwx~ to delete the [wx] range name before the macro end. The second menu option is [**R**owwise]:

```
Rowwise
Insert the dates along a row
{WINDOWSOFF}{RECALC steps2159}{steps2159}
/RNDwx~
```

This routine uses the same code as the previous menu option except that now the formula is in cell [steps2159].

```
26 steps2159        +"{RIGHT}{formula159}~/C~.{RIGHT "&@STRING(B30-2,0)&"}~
                    {LEFT}/RFD4.{END}{RIGHT}~/RV.{END}{RIGHT}~~"
```

This is almost the same formula as the formula in cell [steps1159] but instead of the {DOWN} and {UP}, the macro uses {RIGHT} and {LEFT}. The result of the formula is the code:

A	B	C	D	E	F
26 steps2159	{RIGHT}{formula159}~/C~.{RIGHT 18}~{LEFT}/RFD4.{END}				
	{RIGHT}~/RV.{END}{RIGHT}~~				

Again this code is similar to the previous one. The third menu option is [**I**n]:

```
In
Insert the dates along the sheets (release 3 only!)
{IF @LEFT(rel159,1)="@"}/RE~{BRANCH ret159}
{WINDOWSOFF}{RECALC steps3159}{steps3159}
/RNDwx~
```

This menu option is for a 3-D worksheet when you want to insert dates in the "Z" direction across the sheets. The macro uses the {IF @LEFT(rel159,1)="@"} to find if you are using a 3-D Lotus release. You are referred to the code at the beginning of the macro where it stored the result of the 3-D @INFO("release") function in the cell named [rel159] for later use, which is now the time. If the first character of the result is the "@" character you are using a 2-D release, otherwise you are using a 3-D release. With a 2-D release, the macro issues the /RE~ to erase the date from the current cell and uses {BRANCH ret159} to branch to an empty routine to quit. However, with a 3-D Lotus release, the macro continues pretty much the same as the two previous menu options except that this time the formula is in cell [steps3159].

```
28 steps3159        +"{NS}{formula159}~/C~.{NS "&@STRING(B30-2,0)&"}~{PS}
                    /RFD4.{END}{NS}~/RV.{END}{NS}~~"
```

This is almost the same as the formula in cell [steps1159] but instead of the {DOWN} and {UP}, the macro uses {NS} and {PS}. The result of the formula is the code:

	A	B	C	D	E	F
28	steps3159	{NS}{formula159}~/C~.{NS 18}~{PS}/RFD4.{END}{NS}~/RV.{END}{NS}~~				

Again, this code is similar to the previous two. The last menu option is [**Q**uit]:

```
Quit
Quit the macro
/RNDwx~
```

Before the macro quits it uses `/RNDwx~` to delete the [wx] range name to leave no trace of the macro. Next the macro reaches an empty cell and quits.

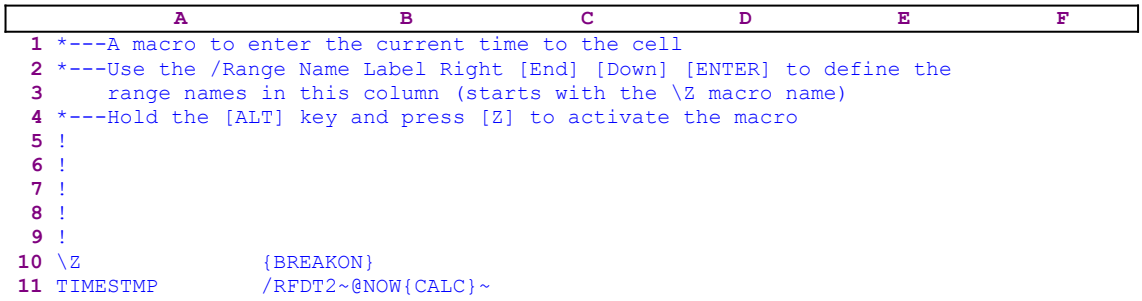
Here is the list of all the cell contents to help you key this macro into Lotus 1-2-3.

```
A1: U '*---A macro to insert the same day for list of months (column, row
      or
A2: U '    sheet)
A3: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A4: '    range names in this column (starts with the \Z macro name)
A5: '*---Hold the [ALT] key and press [Z] to activate the macro
A6: '!
A7: U '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
A8: U '                    IT WILL WORK IN LOTUS 2.0 AND UP
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'SAMEDAY
B11: '{WINDOWSOFF}{PANELOFF}{LET rel159,@INFO("release")}~/RNCwx~~/RNDwx~
     /RNCwx~~{PANELON}
A12: '!
B12: '{GETNUMBER "How many months ? ",months159}~
A13: '!
B13: '{GETLABEL "Enter the first date (MM/DD/YY) or press [ENTER] for
     today: ",wx}{PANELOFF}
A14: '!
B14: '{IF wx=""}{LET wx,+@INT(@NOW)}~{BRANCH loop1159}
A15: '!
B15: '{EDIT}{HOME}{DEL}@DATEVALUE("{END}")~
A16: 'loop1159
B16: '{MENUBRANCH menu159}
A17: '!
A18: 'menu159
B18: 'Columnwise
C18: 'Rowwise
D18: 'In
E18: 'Quit
A19: '!
B19: 'Insert the dates along a column
C19: 'Insert the dates along a row
D19: 'Insert the dates along the sheets (release 3 only!)
E19: 'Quit the macro
A20: '!
B20: '{WINDOWSOFF}{RECALC steps1159}{steps1159}
C20: '{WINDOWSOFF}{RECALC steps2159}{steps2159}
D20: '{IF @LEFT(REL159,1)="@"}/RE~{BRANCH RET159}
E20: '/RNDwx~
A21: '!
B21: '/RNDwx~
C21: '/RNDwx~
D21: '{WINDOWSOFF}{RECALC steps3159}{steps3159}
A22: '!
D22: '/RNDwx~
A23: '!
A24: 'steps1159
B24: U +"{DOWN}{formula159}~/C~.{DOWN "&@STRING(B30-2,0)&"}~{UP}/RFD4.{END}
```

```
       {DOWN}~/RV.{END}{DOWN}~~"
A25: '!
A26: 'steps2159

B26: U +"{RIGHT}{formula159}~/C~.{RIGHT "&@STRING(B30-2,0)&"}~{LEFT}/RFD4.
       {END}{RIGHT}~/RV.{END}{RIGHT}~~"
A27: '!
A28: 'steps3159
B28: U +"{NS}{formula159}~/C~.{NS "&@STRING(B30-2,0)&"}~{PS}/RFD4.{END}
       {NS}~/RV.{END}{NS}~~"
A29: '!
A30: 'months159
B30: 20
A31: '!
A32: 'formula159
B32: '@IF(@DAY(wx)>3,wx+28-@DAY(wx+28),wx+31-@DAY(wx+31))+@IF(@MONTH(wx)
     =1,@MIN(@IF(@MOD(@YEAR(wx),4)=0,29,28),@DAY($wx)),@IF(@month(wx)
     =3#OR#@MONTH(wx)=5#OR#@MONTH(wx)=8#OR#@MONTH(wx)=10,@MIN(30,@DAY($wx))
     ,@DAY($wx)))
A33: '!
A34: 'rel159
B34: '3.10.00
A35: '!
A36: 'ret159
```

[1] Time Stamp Macro



```
     A                    B                C              D              E              F
 1 *---A macro to enter the current time to the cell
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z               {BREAKON}
11 TIMESTMP         /RFDT2~@NOW{CALC}~
```

This is a simple macro that enters the current date as constant into the current cell pointer position. The macro starts with the standard `/RFDT2~` macro keys that format the current cell to the number 2 date format. Then the macro issues the `@NOW` function. Because this is not a valid Lotus command, Lotus types it into the panel. If the macro will now issue the tilde "`~`" command the `@NOW` formula will be written to the current cell. Note that the `@NOW` function returns the current date and time; therefore if you save the worksheet with the `@NOW` function in the cell, the next time you retrieve the worksheet, the `@NOW` function will display the new date. Therefore the macro issues `{CALC}` while the `@NOW` function is still in the panel. When `{CALC}` or F9 is issued while a formula is still in the panel, Lotus turns the formula into a value equal to the result of the formula. Now when the macro issues the tilde "`~`", Lotus writes the value to the current cell instead of the formula.

[6] Turn Date Labels Into Date Values

```
     A                    B              C          D          E          F
 1  *---A macro to turn a 3-D OR 2-D range of DATE LABELS into DATE VALUES.
 2      This macro creates format 4 dates (mm/dd/yy), to change to other
 3      formats change to /RFD[Format Mo.] in the second line of
        labels1a402.
 4  *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5      range names in this column (starts with the \Z macro name)
 6  *---Hold the [ALT] key and press [Z] to activate the macro
 7  *---In release 3 the date label can be any of the 5 dates formats
 8           THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                       IT WILL WORK IN RELEASE 2.0 AND UP
10 \Z             {BREAKON}
11 DATELABL       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~{BS}
                  {BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb402,0}~
12 cont402        {LET hereabs402,@CELLPOINTER("address")}~
                  {LET counter1402,0}
13 !              {FOR counter1402,0,@COLS(Which range ?)-1,1,labels1402}
14 !              {LET rel402,@INFO("release")}~{IF @LEFT(rel402,1)<>"@"}
                  {GOTO}{hereabs402}~{LET counterb402,counterb402+1}~
                  {IF counterb402<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs402}~{BRANCH cont402}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1402           5
18 counter1a402          0
19 labels1402     {RIGHT}{LET here402,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a402,0,@ROWS(Which range ?)-1,1,labels1a402}~
                  {IF counter1402<@COLS(Which range ?)-1}{GOTO}{here402}~
                  {LET counter1a402,0}~
20 !
21 here402        $F$1
22 !
23 labels1a402    {IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{DEL}@DATEVALUE
                  ("{END}")~/RFD4~/RV~~
24 !              {DOWN}
25 !
26 !
27 ret402
28 !
29 counterb402           4
30 hereabs402     $A$1
31 !
32 rel402
```

Many times, especially when you exchange data with other types of computers, dates are accepted as strings. When imported to Lotus 1-2-3, they need to be transformed into date values. This macro turns labels that like date (mm/dd/yy) into real dates that can be manipulated and used for calculations. The macro uses the `@DATEVALUE()` function. This is done in the `{labels1a402}` routine using the enveloping method. Every date label is "enveloped" by `@DATEVALUE()`.

The macro starts with `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the first range, and simultaneously assigns the [Which range ?] name to the same range, which acts as a prompt. Next the macro uses `{LET counterb402,0}~` to set the content of the B29 cell named [counterb402] to zero, which serves as a counter.

```
     A                    B              C          D          E          F
12 cont402        {LET hereabs402,@CELLPOINTER("address")}~
```

```
                {LET counter1402,0}
13 !            {FOR counter1402,0,@COLS(Which range ?)-1,1,labels1402}
14 !            {LET rel402,@INFO("release")}~{IF @LEFT(rel402,1)<>"@"}
                {GOTO}{hereabs402}~{LET counterb402,counterb402+1}~
                {IF counterb402<@SHEETS(Which range ?)}{NS}{GOTO}
                {hereabs402}~{BRANCH cont402}
15 !            {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to its place of origin when the macro is finished, the macro issues `{LET hereabs402,@CELLPOINTER("address")}~`, which store the current cell pointer address in the cell [hereabs402], and then issues `{LET counter1402,0}` to set the content of the cell [counter1402] to zero, which also serves as a counter.

The macro issues the `{FOR counter1402,0,@COLS(Which range ?)-1,1, labels1402}` loop command which activates the [labels1402] routine as many times as the number of columns in the [Which range ?] range. Before continuing, let's look at the [labels1402] routine.

```
     A              B              C         D         E         F
19 labels1402   {RIGHT}{LET here402,@CELLPOINTER("address")}~{LEFT}
                {FOR counter1a402,0,@ROWS(Which range ?)-1,1,labels1a402}~
                {IF counter1402<@COLS(Which range ?)-1}{GOTO}{here402}~
                {LET counter1a402,0}~
```

This routine uses `{RIGHT}{LET here402,@CELLPOINTER("address")}~{LEFT}` to record the address of the first cell of the next column in cell [here402]. When the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here402]. The `{FOR counter1a402,0,@ROWS (Which range ?)-1,1,labels1a402}~` command activates the [labels1a402] routine as many times as the number of rows in the [Which range ?] range.

```
     A              B              C         D         E         F
23 labels1a402  {IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{DEL}@DATEVALUE
                ("{END}")~/RFD4~/RV~~
24 !            {DOWN}
```

The [labels1a402] routine issues `{IF @CELLPOINTER("type")="l"}` to check if the current cell contains a label. If so, the macro issues `{EDIT}` to enter the EDIT mode, then `{HOME}` to move the cursor to the beginning of the panel, and `{DEL}` to delete the apostrophe "`'`" prefix. To better explain the macro, let's assume that the current cell contains the `'12/20/92` label. When the macro issues `{EDIT}` the panel shows:

```
'12/20/92_
```

The underscore represents the position of the cursor. When the macro issues `{HOME}` the panel displays:

```
'12/20/92
^
```

The "^" represents the position of the cursor. When the macro issues `{DEL}` the panel displays:

```
12/20/92
^
```

The "^" represents the position of the cursor. Now the macro issues `@DATEVALUE(` text, but

because there is no valid command in this text, Lotus writes it to the panel which now displays:

```
@DATEVALUE("12/20/92
```

Now the macro issues {END} which moves the cursor to the end of the panel which now displays:

```
@DATEVALUE("12/20/92_
```

Where the underscore represents the position of the cursor. Last the macro writes the ") text to the panel which displays the full formula:

```
@DATEVALUE("12/20/92")
```

This formula turns the date label into a date value. Now the macro issues the tilde "~" command which is the same as pressing the ENTER key, and then issues /RFD4~/RV~~, which formats the date into date format 4, and then the formula into a value. Last the macro issues {DOWN} which moves the cell pointer down to the next cell in the column. When the [labels1a402] routine ends, the macro returns control to the {FOR} loop in the [labels1402] routine to process the date string in the next cell.

	A	B	C	D	E	F
19	labels1402	{RIGHT}{LET here402,@CELLPOINTER("address")}~{LEFT}				
		{FOR counter1a402,0,@ROWS(Which range ?)-1,1,labels1a402}~				
		{IF counter1402<@COLS(Which range ?)-1}{GOTO}{here402}~				
		{LET counter1a402,0}~				

When the value in cell [counter1a402] reaches the number of rows in the [Which range ?] range minus one, the macro issues {IF counter1402<@COLS(Which range ?)-1} to check how many column were processed. If the value in [counter402] is less than the number of columns in [Which range ?], the macro issues the indirect {GOTO}{here402}~ command which moves the cell pointer to the first cell of the next column. Next the macro issues {LET counter1a402,0}~ to reset the value in [counter1a402] to zero. When the [labels1402] routine is finished, the macro returns control to the [cont402] routine.

	A	B	C	D	E	F
12	cont402	{LET hereabs402,@CELLPOINTER("address")}~				
		{LET counter1402,0}				
13	!	{FOR counter1402,0,@COLS(Which range ?)-1,1,labels1402}				
14	!	{LET rel402,@INFO("release")}~{IF @LEFT(rel402,1)<>"@"}				
		{GOTO}{hereabs402}~{LET counterb402,counterb402+1}~				
		{IF counterb402<@SHEETS(Which range ?)}{NS}{GOTO}				
		{hereabs402}~{BRANCH cont402}				
15	!	{GOTO}Which range ?~/RNDWhich range ?~				

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues {LET rel402,@INFO("release")}~ to store the result of the @INFO("release") 3-D function in cell [rel402]. Then it issues {IF @LEFT (rel402,1)<>"@"} to check the first character of the content of [rel402]. If the character is the "@" character, you are using a 2-D release, otherwise you are using a 3-D release which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the indirect {GOTO}{hereabs402}~ command which moves the cell pointer to the origin address of the macro.

The macro continues with `{LET counterb402,counterb402+1}~` to increase the value in cell [counterb402] by one. Now the macro issues `{IF counterb402<@SHEETS(Which range ?)}` to check if the value in [counterb402] is less than the number of sheets in the [Which range ?] range. If so, there are more sheets to process, therefore the macro issues `{NS}` to move the cell pointer to the next sheet. Next the macro issues the indirect `{GOTO}{hereabs402}~` command which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of [Which range ?], but in the new sheet. Last, the macro issues `{BRANCH cont402}` to process the cells in the new sheet. When all the sheets are processed, the macro issues `{GOTO}Which range ?~` which moves the cell pointer to the upper left cell of the first sheet of [Which range ?], and then issues `/RNDWhich range ?~` to delete the temporary [Which range ?] range name to leave a clean worksheet.

[6] Turn Date Values into Date Labels

```
     A                B                C          D          E          F
 1 *---A macro to turn a 3-D or 2-D range of DATE VALUES into DATE LABELS.
 2     This macro accepts and creates format 4 dates (mm/dd/yy).
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                  IT WILL WORK IN RELEASE 2.0 AND UP
 9 !
10 \Z             {BREAKON}
11 DATEVAL        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~{BS}
                  {BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb404,0}~
12 cont404        {LET hereabs404,@CELLPOINTER("address")}~
                  {LET counter404,0}
13 !              {FOR counter404,0,@COLS(Which range ?)-1,1,labels404}
14 !              {LET rel404,@INFO("release")}~{IF @LEFT(rel404,1)<>"@"}
                  {GOTO}{hereabs404}~{LET counterb404,counterb404+1}~
                  {IF counterb404<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs404}~{BRANCH cont404}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter404           1
18 countera404         10
19 labels404      {RIGHT}{LET here404,@CELLPOINTER("address")}~{LEFT}
                  {FOR countera404,0,@ROWS(Which range ?)-1,1,labelsa404}~
                  {IF counter404<@COLS(Which range ?)-1}{GOTO}{here404}~
                  {LET countera404,0}~
20 !
21 here404        $Y$1
22 !
23 labelsa404     {RECALC day404}{RECALC month404}{RECALC year404}
                  {RECALC string404}
24 !              {IF @CELLPOINTER("type")="v"}{string404}{EDIT}{HOME}'
25 !              {DOWN}
26 !
27 day404         ERR
28 month404       ERR
29 year404        ERR
30 string404      ERR
31 !
32 counterb404          0
33 hereabs404     $X$1
34 !
35 rel404         @INFO("release")
```

This macro does the opposite of the DATELABL.WK1 macro. It will turn perfectly good date numbers into date labels with the mm/dd/yy format so that they can be transferred to other computers. Notice that this macro uses dynamic string formulas to create the correct code, therefore if you intend to key this macro into Lotus 1-2-3, you should key the following formulas (in the B27, B28, B29 and the B30 cells) instead of the ERR in the main listing.

```
27 day404         @STRING(@DAY(@CELLPOINTER("contents")),0)
28 month404       @STRING(@MONTH(@CELLPOINTER("contents")),0)
29 year404        @STRING(@YEAR(@CELLPOINTER("contents")),0)
30 string404      +B28&"/"&B27&"/"&B29
```

The macro starts with the {WINDOWSOFF}{PANELOFF} commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the first range, and simultaneously assigns the [Which range ?] name to the same range, which

acts as a prompt. Next the macro issues {LET counterb404,0}~ to set the content of cell [counterb404] to zero, which also serves as a counter.

	A	B	C	D	E	F
12	cont404	{LET hereabs404,@CELLPOINTER("address")}~				
		{LET counter404,0}				
13	!	{FOR counter404,0,@COLS(Which range ?)-1,1,labels404}				
14	!	{LET rel404,@INFO("release")}~{IF @LEFT(rel404,1)<>"@"}				
		{GOTO}{hereabs404}~{LET counterb404,counterb404+1}~				
		{IF counterb404<@SHEETS(Which range ?)}{NS}{GOTO}				
		{hereabs404}~{BRANCH cont404}				
15	!	{GOTO}Which range ?~/RNDWhich range ?~				

To be able to return to its point of origin when the macro is finished, the macro issues {LET hereabs404,@CELLPOINTER("address")}~ to store the current cell pointer address in the cell [hereabs404], and then it issues {LET counter404,0} to set the content of cell [counter404] which also serves as a counter to zero. The macro issues the {FOR counter404,0,@COLS(Which range ?)-1,1,labels404} loop command which activates the [labels404] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels404] routine.

	A	B	C	D	E	F
19	labels404	{RIGHT}{LET here404,@CELLPOINTER("address")}~{LEFT}				
		{FOR countera404,0,@ROWS(Which range ?)-1,1,labelsa404}~				
		{IF counter404<@COLS(Which range ?)-1}{GOTO}{here404}~				
		{LET countera404,0}~				

This routine uses {RIGHT}{LET here404,@CELLPOINTER("address")}~{LEFT} to record the address of the first cell of the next column in cell [here404]. This way, when the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here404]. The {FOR countera404,0,@ROWS (Which range ?)-1,1,labelsa404}~ commands activate the [labelsa404] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E	F
23	labelsa404	{RECALC day404}{RECALC month404}{RECALC year404}				
		{RECALC string404}				
24	!	{IF @CELLPOINTER("type")="v"}{string404}{EDIT}{HOME}'				
25	!	{DOWN}				

This routine issues {RECALC day404}{RECALC month404}{RECALC year404} {RECALC string404} to update the following formulas:

27	day404	@STRING(@DAY(@CELLPOINTER("contents")),0)
28	month404	@STRING(@MONTH(@CELLPOINTER("contents")),0)
29	year404	@STRING(@YEAR(@CELLPOINTER("contents")),0)
30	string404	+B28&"/"&B27&"/"&B29

The formula in the B27 cell named [day404] calculates and extracts the day from the date value in the current cell using the @DAY function. The formula in the B28 cell named [month404] calculates and extracts the month from the date value in the current cell using the @MONTH function. The formula in the B29 cell named [year404] calculates and extracts the year from the date value in the current cell using the @YEAR function. The string formula in the B30 cell named [string404] combines the results of the three previous formulas and creates the date label.

The routine continues with {IF @CELLPOINTER("type")="v"} which checks if the current cell contains a value. If so, the routine issues the {string404} routine command which activates the [string404] routine. However the [string404] routine contains the result of a formula which returns the date in the mm/dd/yy form. Therefore the {string404} routine command injects the content of the cell [string404] to the panel which may display something like

```
12/20/92
```

but Lotus recognize this as a number, therefore the routine issues {HOME}{EDIT} which move the cursor to the first character in the panel and then types the apostrophe "'" prefix to change it to the

```
'12/20/92
```

label. Now the routine issues the {DOWN} command which moves the cell pointer to the next cell and simultaneously writes the '12/20/92 label into the cell. Without the apostrophe "'" prefix the result would be 0.006521739130435 which is the result of 12 divided by 20 divided by 92.

When the [labelsa404] routine ends, the macro returns control to the {FOR} loop in the [labels404] routine to start the process for the next cell in the current column.

	A	B	C	D	E	F
19	labels404	{RIGHT}{LET here404,@CELLPOINTER("address")}~{LEFT}				
		{FOR countera404,0,@ROWS(Which range ?)-1,1,labelsa404}~				
		{IF counter404<@COLS(Which range ?)-1}{GOTO}{here404}~				
		{LET countera404,0}~				

When the value in [countera404] reaches the number of rows in [Which range ?] minus one, the macro issues {IF counter404<@COLS(Which range ?)-1} to check how many columns were processed. If the value in [counter404] is less than the number of columns in [Which range ?], the macro issues the indirect {GOTO}{here404}~ command which moves the cell pointer to the first cell of the next column. Next the macro issues {LET countera404,0}~ to reset the value in [countera404] to zero. When the [labels404] routine is finished, the macro returns control to the [cont404] routine.

	A	B	C	D	E	F
12	cont404	{LET hereabs404,@CELLPOINTER("address")}~				
		{LET counter404,0}				
13	!	{FOR counter404,0,@COLS(Which range ?)-1,1,labels404}				
14	!	{LET rel404,@INFO("release")}~{IF @LEFT(rel404,1)<>"@"}				
		{GOTO}{hereabs404}~{LET counterb404,counterb404+1}~				
		{IF counterb404<@SHEETS(Which range ?)}{NS}{GOTO}				
		{hereabs404}~{BRANCH cont404}				
15	!	{GOTO}Which range ?~/RNDWhich range ?~				

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First, the macro issues {LET rel404,@INFO("release")}~ to store the result of the @INFO("release") 3-D function in cell [rel404], and then it issues {IF @LEFT(rel404,1)<>"@"} which checks the first character of the content of [rel404]. If the character is the "@" character, you are using a 2-D release, otherwise you are using a 3-D release which may have more than one sheet in the [Which range ?] range. Therefore the macro issues

the {GOTO}{hereabs404}~ indirect command which moves the cell pointer to the origin address of the macro.

The macro continues with {LET counterb404,counterb404+1}~ to increase the value in the [counterb404] by one. Now the macro issues {IF counterb404<@SHEETS(Which range ?)} to check if the value in [counterb404] is less than the number of sheets in the [Which range ?] range. If so, there are more sheets to process, therefore the macro issues {NS} to move the cell pointer to the next sheet. Next the macro issues the indirect {GOTO}{hereabs404}~ command which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of [Which range ?], but in the new sheet. Last, the macro issues {BRANCH cont404} to process the cells of [Which range ?] in the new sheet. When all the sheets are processed, the macro issues {GOTO}Which range ?~ to move the cell pointer to the upper left cell of the first sheet of [Which range ?], and then issues /RND Which range ?~ to delete the temporary [Which range ?] range name to leave a clean worksheet.

[5] Create a List of Dates

```
     A          B          C          D          E          F          G
 1 *---A macro to enter a series of DATES
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z           {BREAKON}
11 DATELIST     {GETLABEL "Input first date (MM/DD/YY) or press [ENTER]
                for today: ",date1008}~{IF date1008=""}{LET date2008,
                @STRING(@INT(@NOW),0)}~{RECALC form008}{BRANCH loop008}
12 !            {LET date2008,@STRING(@INT(@DATEVALUE(date1008)),0)}~
                {RECALC form008}{BRANCH loop008}
13 loop008      {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                Which range ?~/RNC{PANELON}Which range ?~{WINDOWSON}{BS}
                {BS}{?}~{WINDOWSOFF}/DFWhich range ?~
14 date2008     33081
15 !            ~
16 !            1
17 !            ~
18 form008      41081
19 !            ~{WINDOWSON}{PANELON}{MENUCALL format008}/RND
                Which range ?~
20 !
21 date1008     07/27/90
22 !
23 format008    1 (DD-MMM 2 (DD-MMM) 3 (MMM-Y 4 (Long in 5 (Short  Quit
24 !            Lotus stanLotus standLotus staAs configurAs config Quit
25 !            /RFD1Which/RFD2Which /RFD3Whic/RFD4Which /RFD5Which range
```

This macro allows you to insert a series of dates up to 8000 cells long. The macro prompts you to insert the first date and how many dates to enter and the macro creates the series of dates. You can type the first date or press ENTER for the current date. This macro uses a custom menu with six menu options which cannot be displayed across a page or the screen without overlapping. Therefore we show each menu option separately. If you intend to key the macro into Lotus 1-2-3, you should key the menus code as it appears here, NOT the code as it appears in the main listing.

```
1 (DD-MMM-YY)
Lotus standard long form
/RFD1Which range ?~~

2 (DD-MMM)
Lotus standard short form
/RFD2Which range ?~~

3 (MMM-YY)
Lotus standard short form
/RFD3Which range ?~~

4 (Long intn'l)
As configured (MM/DD/YY or others)
/RFD4Which range ?~~

5 (Short intn'l)
As configured (MM/DD or others)
/RFD5Which range ?~~

Quit
Quit the macro
```

Notice that the code in the B17 cell named [form008] also contains a dynamic string formula which returns the correct code for the macro. If you intend to key this macro into Lotus 1-2-3, you should key the formula's code as it appears here, NOT the code as it appears in the main listing. The formula is:

```
17 form008      @STRING(@INT(@NOW)+8000,0)
```

```
          A            B          C          D          E          F          G
11 DATELIST     {GETLABEL "Input first date (MM/DD/YY) or press [ENTER]
                for today: ",date1008}~{IF date1008=""}{LET date2008,
                @STRING(@INT(@NOW),0)}~{RECALC form008}{BRANCH loop008}
12 !            {LET date2008,@STRING(@INT(@DATEVALUE(date1008)),0)}~
                {RECALC form008}{BRANCH loop008}
```

The macro starts with

```
{GETLABEL "Input first date (MM/DD/YY) or press [ENTER]
    for today: ",date1008}~
```

which displays

```
Input first date (MM/DD/YY) or press [ENTER] for today:
```

prompt message in the panel and waits until you respond. You can type an `MM/DD/YY` starting date or press ENTER to start the list of date from the current date. When you press the ENTER key, Lotus stores your response in cell [date1008] as a label, because of the `{GETLABEL}` command. Next the macro issues `{IF date1008=""}` to check if you only pressed the ENTER key (which left the cell [date1008] empty). If so, the macro issues `{LET date2008,@STRING(@INT(@NOW),0)}~` to write the current date as a string in cell [date2008]. Now the macro issues `{RECALC form008}` which updates the formula in the cell [form008] and then issues `{BRANCH loop008}` which routes the macro control to the [loop008] routine. The `@STRING(@INT(@NOW),0)` formula uses the `@NOW` function to calculate today's date/time number. Then it uses the `@INT` function to strip the time from the date number and leave only the integer part of the number (only the date). Then it changes it to a string using the `@STRING(@INT(@NOW),0)` formula. For more details on these formulas see the Lotus manual.

If this is not true, it means that you typed a date, therefore the macro issues `{LET date2008,@STRING(@INT(@DATEVALUE(date1008)),0)}~` to extract the integer part of the date (the non integer part is the time) from the content of [date1008] into [date2008]. Now the macro issues `{RECALC form008}` which updates the formula in [form008] and then `{BRANCH loop008}` which routes the macro control to the [loop008] routine. This time, the macro uses the same string formula but on the `@DATEVALUE` function instead of the `@NOW` function.

```
          A            B          C          D          E          F          G
13 loop008      {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                Which range ?~/RNC{PANELON}Which range ?~{WINDOWSON}{BS}
                {BS}{?}~{WINDOWSOFF}/DFWhich range ?~
14 date2008     33081
15 !            ~
16 !            1
17 !            ~
18 form008      41081
```

```
19 !            ~{WINDOWSON}{PANELON}{MENUCALL format008}/RND
                Which range ?~
```

This routine starts with {WINDOWSOFF}{PANELOFF} which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range of the dates, and simultaneously assigns the [Which range ?] name to the same range, which acts as a prompt. When you paint the range and press ENTER, the macro issues /DFWhich range ? ~33081~1~41081~ to fill the assigned range with dates. The starting number of the data is the date string in the B13 cell [date2008] that the macro has recorded earlier. This is another example of using dynamic code, the starting date is entered to the macro code before the macro processes this code, thereby dynamically change the code. Then the step is entered as "1" and the ending date string is greater than the current date by 8000 which insure that you can create a list 8000 dates long.

Next the macro issues {WINDOWSON}{PANELON} to resume the screen and panel activities and then issues {MENUCALL format008} activating the [format008] custom menu to duplicate the standard Lotus menu. The first menu option is [**1** (DD-MMM-YY)]:

```
1 (DD-MMM-YY)
Lotus standard long form
/RFD1Which range ?~~
```

When you choose this menu option, the macro issues /RFD1Which range ?~~ which formats all the cells in the [Which range ?] range as Date format number 1. The next four menu items are the same, the only difference being the date format number. The last menu option is [**Q**uit]:

```
Quit
Quit the macro
```

When you choose this menu option, the macro reaches an empty cell and quits. Because we have used {MENUCALL} to activate the custom menu, the macro returns to the /RNDWhich range ?~ command which deletes the temporary [Which range ?] range name and leaves a clean worksheet. Because this macro occupies more than the A and the B columns, and it may be difficult for you to key this macro into Lotus 1-2-3, we show the full list of all the cell formulas and contents.

```
A1: U [W10] '*---A macro to enter a series of DATES
A2: [W10] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W10] '    range names in this column (starts with the \Z macro name)
A4: [W10] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W10] '!
A6: [W10] '!
A7: [W10] '!
A8: [W10] '!
A9: [W10] '!
A10: U [W10] '\Z
B10: [W15] '{BREAKON}
A11: U [W10] 'DATELIST
B11: [W15] '{GETLABEL "Input first date (MM/DD/YY) or press [ENTER] for
           today: ",date1008}~{IF date1008=""}{LET date2008,@STRING(@INT
           (@NOW),0)}~{RECALC form008}{BRANCH loop008}
A12: [W10] '!
B12: [W15] '{LET date2008,@STRING(@INT(@DATEVALUE(date1008)),0)}~
           {RECALC form008}{BRANCH loop008}
A13: [W10] 'loop008
B13: [W15] '{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RNDWhich range ?~
            /RNC{PANELON}Which range ?~{WINDOWSON}{BS}{BS}{?}~{WINDOWSOFF}
            /DFWhich range ?~
```

```
A14: [W10] 'date2008
B14: [W15] '33081
A15: [W10] '!
B15: [W15] '~
A16: [W10] '!
B16: [W15] '1
A17: [W10] '!
B17: [W15] '~
A18: [W10] 'form008
B18: [W15] @STRING(@INT(@NOW)+8000,0)
A19: [W10] '!
B19: [W15] '~{WINDOWSON}{PANELON}{MENUCALL format008}/RNDWhich range ?~
A20: [W10] '!
A21: [W10] 'date1008
B21: (G) [W15] '07/27/90
A22: [W10] '!
A23: [W10] 'format008
B23: [W15] '1 (DD-MMM-YY)
C23: [W11] '2 (DD-MMM)
D23: [W11] '3 (MMM-YY)
E23: [W16] '4 (Long intn'l)
F23: [W17] '5 (Short intn'l)
G23: 'Quit
A24: [W10] '!
B24: [W15] 'Lotus standard long form
C24: [W11] 'Lotus standard short form
D24: [W11] 'Lotus standard short form
E24: [W16] 'As configured (MM/DD/YY or others)
F24: [W17] 'As configured (MM/DD or others)
G24: 'Quit the macro
A25: [W10] '!
B25: [W15] '/RFD1Which range ?~~
C25: [W11] '/RFD2Which range ?~~
D25: [W11] '/RFD3Which range ?~~
E25: [W16] '/RFD4Which range ?~~
F25: [W17] '/RFD5Which range ?~~
```

[4] Past Date and Future Date

```
     A          B          C          D          E          F          G
 1 *---A macro to insert dates for the future or the past. You can
 2     find the date several days ago (-) or ahead (+).
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---When you are prompted for the date format use the direction keys
 7     and press ENTER. Do not use the first character option.
 8 !
 9 !
10 \Z             {BREAKON}
11 DATEPAST       {WINDOWSOFF}{GETNUMBER "Enter date increment ? "
                  ,increment009}~{PANELOFF}
12 !              @NOW+increment009{CALC}{DOWN}{UP}~
13 !              {WINDOWSON}/RF{PANELON}D{?}~~
14 !
15 increment009
```

This macro helps you calculate the date of a specified number of days ahead or past. For example, we can calculate the date 56 days ahead from now or 789 days ago. The macro inserts the result in the current cell, therefore you should place the cell pointer where you want the result and then activate the macro.

The macro issues `{WINDOWSOFF}` to freeze the screen activity and then issues `{GETNUMBER "Enter date increment ? ",increment009}` to display "`Enter date increment ?`" in the panel. When you insert the number and press ENTER, Lotus stores the number in the B15 cell named [increment009]. Next the macro writes the `@NOW+increment009` formula into the panel and then issues `{CALC}` to turn the formula into its value (the date number). Then it issues `{DOWN}` and `{UP}` to write the content of the panel to the current cell. Last the macro issues `{WINDOWSON}/RF{PANELON}D{?}~~` to resume screen and panel activities and to issue / **R**ange **F**ormat **D**ates menu commands and `{?}` which stops the macro and waits until you choose the date format from the menu. Use only the direction keys and press ENTER to choose the date format.

[4] Date Stamp with Extras

```
         A              B          C          D          E          F      G
 1 *---A macro to INSERT the current TIME or DATE in the current cell
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 DATESTMP       {WINDOWSOFF}@NOW~{MENUBRANCH menu010}
12 !
13 menu010        Standard_Da Permanent_st Date_fu Time_st Fixed_ti  Quit
14 !              Date stamp aDate stamp asFull datTime staTime as vaquit
15 !              /RFD{?}~~   {EDIT}{CALC}~{RECALC /RFDT{?}{EDIT}{CAL/RE~
16 !                                        {fulldate010}~
17 !
18 !
19 date010                     0
20 day1010        Saturday
21 day2010                  ERR
22 month010                 ERR
23 year010                  ERR
24 fulldate010              ERR
```

This macro is like DATE STAMP. You place the cell pointer where you want the date and activate the macro. The macro allows few types of dates:

- Alive date formula that will be updated every time the worksheet is retrieved
- Date stamp as value only
- Full date in words (Tuesday September 10, 1991 for example)
- Time stamp as formula that will be updated every time the worksheet is retrieved
- Time stamp as value only

This macro uses a custom menu with six menu options which cannot be displayed across a page or the screen without overlapping. Therefore we show each menu option separately. If you intend to key the macro into Lotus 1-2-3, you should key the menu code as it appear here, NOT the code as it appears in the main listing.

```
Standard_Date
Date stamp as formula that updates every time the worksheet is
    retrieved
/RFD{?}~~

Permanent_std
Date stamp as value only
{EDIT}{CALC}~/RFD{?}~~


Date_full
Full date in words as label
{RECALC date010}{RECALC day1010}{RECALC month010}{RECALC year010}
    {RECALC day2010}{RECALC fulldate010}
{fulldate010}~

Time_std
Time stamp as formula that updates every time the worksheet is
    retrieved
/RFDT{?}~~
```

```
Fixed_time
Time stamp as value only
{EDIT}{CALC}~/RFDT{?}~~

Quit
Quit the macro
/RE~
```

Notice that the code in the B19..B24 range is the result of dynamic string formulas which return the correct code for the macro. If you intend to key the macro into Lotus 1-2-3, you should key the formulas code as it appears here, NOT the code as it appears in the main listing. The formulas are:

```
19 date010        @CELLPOINTER("contents")
20 day1010        @CHOOSE(@MOD(B19,7),"Saturday","Sunday","Monday","Tuesday"
                  ,"Wednesday","Thursday","Friday")
21 day2010        @DAY(B19)
22 month010       @CHOOSE(@MONTH(B19)-1,"January","February","March",
                  "April","May","June","July","August","September","October"
                  ,"November","December")
23 year010        @YEAR(B19)+1900
24 fulldate010    +B20&" "&B22&" "&@STRING(B21,0)&", "&@STRING(B23,0)
```

The macro starts with the {WINDOWSOFF} command which freezes the screen activity, then writes the @NOW function into the panel and issues the tilde "~" command (which is the same as the ENTER key from the keyboard) to write the current date into the current cell. Then the macro issues {MENUBRANCH menu010} which activates the [menu010] custom menu which includes six menu options. The first menu option is [**S**tandard_Date]:

```
Standard_Date
Date stamp as formula that updates every time the worksheet is
    retrieved
/RFD{?}~~
```

When you choose this menu option, the macro issues /RFD which starts the / **D**ate **F**ormat process and then issues {?} which halts the macro and pauses until you choose one of the date formats from the standard Lotus menu. The second menu option is [**P**ermanent_std]:

```
Permanent_std
Date stamp as value only
{EDIT}{CALC}~/RFD{?}~~
```

When you choose this menu option, the macro issues {EDIT} to enter to EDIT mode and then {CALC} to turn the formula into a value. Then it issues the tilde "~" to write the current date as a fixed number into the current cell. Next the macro issues /RFD which starts the / **D**ate **F**ormat process and then issues {?} which halts the macro and pauses until you choose one of the date formats from the standard Lotus menu. The third menu option is [**D**ate_full]:

```
Date_full
Full date in words as label
{RECALC date010}{RECALC day1010}{RECALC month010}{RECALC year010}
    {RECALC day2010}{RECALC fulldate010}
{fulldate010}~
```

When you choose this menu option, the macro issues {RECALC date010}{RECALC day1010}{RECALC month010}{RECALC year010}{RECALC day2010} to update the respective formulas in the cells [date010], [day1010], [month010], [year010], and [day2010].

```
19 date010        @CELLPOINTER("contents")
```

The formula in the B19 cell named [date010], returns the result of the @NOW function in the current cell.

```
20 day1010        @CHOOSE(@MOD(B19,7),"Saturday","Sunday","Monday","Tuesday"
                  ,"Wednesday","Thursday","Friday")
```

The formula in B20, [day1010], returns the day of the week as one of the words "Saturday", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday" or "Friday". The formula uses the @CHOOSE function which returns one of the names as a function of the result of the @MOD (B19,7) formula which returns a number between 1 to 7. If the number is 1 the result is "Sunday" and respectively. The @MOD function returns the remainder after dividing the date number by 7 which points to the current day in the current week.

```
21 day2010        @DAY(B19)
```

The formula in B21, [day2010], returns the day of the week as a number.

```
22 month010       @CHOOSE(@MONTH(B19)-1,"January","February","March",
                  "April","May","June","July","August","September","October"
                  ,"November","December")
```

The formula in B22, [month010], returns the month in words using the same technique as we have seen for the day in the B20 cell.

```
23 year010        @YEAR(B19)+1900
```

The formula in B23, [year010], returns the year as a numbers.

```
24 fulldate010    +B20&" "&B22&" "&@STRING(B21,0)&", "&@STRING(B23,0)
```

The formula in B24, [fulldate010], combines all the results in the previous formulas into the full date. Now the macro issues the {fulldate010} routine command which injects the content of cell [fulldate010] into the panel and then the macro issues the tilde "~" to write the current date as a full date into the current cell. The last menu option is [**Q**uit]:

```
Quit
Quit the macro
/RE~
```

When you choose this menu option, the macro issues /RE~ to erase the @NOW function from the current cell and then reaches an empty cell and quits. Because this macro occupies more than the A and the B columns and it may be difficult for you to key this macro into Lotus 1-2-3, here is the full list of all the cell formulas and contents.

```
A1: U [W14] '*---A macro to INSERT the current TIME or DATE in the current
          cell
A2: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W14] '    range names in this column (starts with the \Z macro name)
A4: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W14] '!
A6: [W14] '!
A7: [W14] '!
A8: [W14] '!
```

```
A9: [W14] '!
A10: U [W14] '\Z
B10: '{BREAKON}
A11: U [W14] 'DATESTMP
B11: (F2) '{WINDOWSOFF}@NOW~{MENUBRANCH menu010}
A12: [W14] '!
A13: [W14] 'menu010
B13: (D6) 'Standard_Date
C13: 'Permanent_std
D13: 'Date_full
E13: 'Time_std
F13: 'Fixed_time
G13: 'Quit
A14: [W14] '!
B14: (F2) 'Date stamp as formula that updates every time the worksheet is
          retrieved
C14: (F2) 'Date stamp as value only
D14: 'Full date in words as label
E14: (F2) 'Time stamp as formula that updates every time the worksheet is
          retrieved
F14: (F2) 'Time stamp as value only
G14: 'Quit the macro
A15: [W14] '!
B15: (D6) '/RFD{?}~~
C15: (D6) '{EDIT}{CALC}~/RFD{?}~~
D15: '{RECALC date010}{RECALC day1010}{RECALC month010}{RECALC year010}
     {RECALC day2010}{RECALC fulldate010}
E15: '/RFDT{?}~~
F15: '{EDIT}{CALC}~/RFDT{?}~~
G15: '/RE~
A16: [W14] '!
D16: '{fulldate010}~
A17: [W14] '!
A18: [W14] '!
A19: [W14] 'date010
B19: @CELLPOINTER("contents")
A20: [W14] 'day1010
B20: @CHOOSE(@MOD(B19,7),"Saturday","Sunday","Monday","Tuesday","Wednesday"
     ,"Thursday","Friday")
A21: [W14] 'day2010
B21: @DAY(B19)
A22: [W14] 'month010
B22: @CHOOSE(@MONTH(B19)-1,"January","February","March","April","May",
     "June","July","August","September","October","November","December")
A23: [W14] 'year010
B23: @YEAR(B19)+1900
A24: [W14] 'fulldate010
B24: +B20&" "&B22&" "&@STRING(B21,0)&", "&@STRING(B23,0)
```

File Macros

[1] Change the Default Directory

	A	B	C	D	E
1	*---A macro to change the DEFAULT FILE/DIRECTORY				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	FILEDIR	/FD*{ESC}{?}~			

This is a simple macro to save a few key strokes every time you need to change the default file directory. The macro starts with the standard `/FD` (/ **F**ile **D**irectory) macro keys. At this point, Lotus offers the last file directory as the default directory. To clear the suggested directory, we cannot simply use the ESC key because we may need to press the ESC key more than once. To safely delete the proposed directory, the macro types the astrix "`*`" character (can be any character). When the macro types the character, Lotus clears the panel and types the character into the panel instead. Now one `{ESC}` can safely clear the panel. Next the macro issues `{?}` which pauses the macro execution and allows you to type the new file directory. When you press ENTER, Lotus understand that you are finished and issues the tilde "`~`" to finish the process.

[1] Change the Default Drive

	A	B	C	D	E
1	*---A macro to change and update the DEFAULT DRIVE/DIRECTORY				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	DEFLTDRV	/WGDD{ESC}{?}~UQ			

If you change the default directory as much as I do, this macro can save many key strokes. The macro starts with the standard `/WGDD` (**/ W**orksheet **G**lobal **D**efault **D**irectory) key sequence and then issues `{ESC}` to clear the current default directory from the panel. Then it issues `{?}` which halts the macro and waits until you type the new directory and press the ENTER key. When you press ENTER, the macro issues the `UQ` macro keys to update the new configuration and quit to

[1] Save a File for Lotus 2.0 and Up

```
     A                B                C                D                E
 1 *---A macro to SAVE A FILE with the current name or a new name
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---If the file is the current file just press [ENTER] otherwise
 6     print the new name and press [ENTER]
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 SAVEFILE       /FS {BS}{?}~R{ESC}
```

It seems odd to use a macro to save a file, but this simple macro can save you many key strokes and much frustration every time you want to save a file under a new name. When you try to save a file, Lotus offers the retrieved file name as the first suggestion. If you try to use the BACKSPACE key to erase the suggested file name, it will not work. First you have to type a character, and only then Lotus allows you to use the BACKSPACE key. This macro types the " " space character, issues the `{BS}` command to erases it, and then `{?}` which allows you to use the editing keys to change the file name or to type a new file name.

When you press the ENTER key, the macro issues the "`R`" key that stands for the "**R**eplace" menu option, however "**R**eplace" does not always appears, it appears only if the file name already exists. But, if the file name is new, Lotus prints the "`R`" character to the panel, therefore the macro follows

[1] Save a File for Lotus 2.2 and Up

```
        A              B              C              D              E
 1 *---A macro to SAVE A FILE with the current name or a new name
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---After the file is saved press [ENTER] if the [CMD] indicator is
 6     highlighted
 7 !
 8                * * *     FOR LOTUS 2.2 AND UP    * * *
 9 !
10 \Z              {BREAKON}
11 SAVEFIL2        /FS {BS}{?}~{?}~
```

This macro is a little different from the SAVEFILE.WK1 previous macro because Lotus 2.2 and up includes a new menu option, the **B**ackup menu option when you save a file. Therefore we introduced the second `{?}` macro command which allows you to choose one of the three menu options (**C**ancel **R**eplace **B**ackup).

Note: Every time a standard Lotus menu appears in the panel, you must use the direction keys in the numeric key pad to select the menu options, and press the ENTER key. If you try to press the capital letter in the menu option, the macro may not continue to function correctly. Somehow Lotus does not like you to use the one key commands from the keyboard when the Lotus menu is activated from a macro. However, it is fine to use the one key commands in a custom menu as we did in the SAVEFILE.WK1 macro with the "R" key to activate the **R**eplace menu option.

[1] Save File to the A: Drive

```
     A                B                C                D                E
 1 *---A macro to save file with a new name to the A: drive
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 SAVE_A         /FSa{ESC}A:{?}~R{ESC}
```

This macro is an upgrade to the SAVEFILE.WK1 macro which adds the A: to the panel in front of the file name that you want to save. This macro saves you two keys every time you save a file to the A: drive. The macro types "A:" and waits until you type the file name. It is real handy when you do much saving to different drives than the default drive. You can prepare such a macro for every one of your drives and save many key strokes if you do much saving.

[1] Save a File to the A: Drive for Lotus 2.2 and Up

```
     A                B                C                D                E
 1 *---A macro to save file with a new name to the A: drive
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---After the file is saved press [ENTER] if the [CMD] indicator is
 6     highlighted
 7 !
 8           THIS MACRO WORKS IN LOTUS 2.2 AND UP
 9 !
10 \Z              {BREAKON}
11 SAVE_A22        /FSa{ESC}A:{?}~{?}~
```

This macro is an upgrade to the SAVEFIL2.WK1 macro which adds the A: to the panel in front of the file name that you want to save. This macro saves you two keys every time you save a file to the A: drive. The macro types "A:" and waits until you type the file name. You can prepare such a macro for every one of your drives and save many key strokes if you do much saving.

[5] Use DOS Commands and Execute DOS Files from 1-2-3

```
     A                B                C                D
 1 *---A file and application manager macro that allows activating any
 2     executable file without quitting Lotus application. When using
 3     batch files the options are unlimited. To add or edit the command
 4     list add to/or change the list starting at the LIST522 range name.
 5 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 6     range names in this column (starts with the \Z macro name)
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8                  THIS MACRO WORKS IN LOTUS 2.2 AND UP
 9 !
10 \Z               {BREAKON}
11 FILEMNGR         {MENUCALL menu522}
12 !
13 coma522          dir/p
14 !
15 key522           ~
16 !
17 menu522          Command_line       Application_list      Quit
18 !                Use DOS command linExecute any executableQuit the macro
19 !                {GETLABEL "Enter DO{WINDOWSOFF}{PANELOFF}{GOTO}IV1~{GO
20 !                {RECALC sys522 }{sy{move522}{LET coma522,@CELLPOINTER(
21 !                {MENUBRANCH menu522{RECALC sys522}{sys522}
22 !                                   {MENUBRANCH menu522}
23 !
24 sys522           {SYSTEM "dir/p"}~
25 !
26 move522          {GOTO}list522~{END}{DOWN}{DOWN}
27 loop522          Use arrow keys to move and press [ENTER] to execute or
                    [ESC] to quit..{GET key522}{ESC}
28 !                {IF key522="{UP}"#OR#key522="{RIGHT}"}{UP}
                    {BRANCH loop522}
29 !                {IF key522="{DOWN}"#OR#key522="{LEFT}"}{DOWN}
                    {BRANCH loop522}
30 !                {IF key522="{PGUP}"}{PGUP}{BRANCH loop522}
31 !                {IF key522="{PGUP}"#OR#key522="{HOME}"}{PGUP}
                    {BRANCH loop522}
32 !                {IF key522="{PGDN}"#OR#key522="{END}"}{PGDN}
                    {BRANCH loop522}
33 !                {IF key522="{ESC}"}{MENUBRANCH menu522}
34 !                {IF key522="~"}{RETURN}
35 !
36 list522          LIST OF APPLICATIONS
37                  ====================
38                  DIR|MORE
39                  C:\BASIC\GWBASIC
40                  DISKCOPY A: A:
42                  DISKCOPY A: B:
```

This macro allows you to issue DOS commands directly from inside Lotus 1-2-3 without quitting or shelling from Lotus. The macro allows you to use commands as though from a command line or to build a list of predefined DOS commands and applications to choose from and execute. The macro already includes a list of four commands, which you can replace and/or add as many as you like. When you activate the macro, the second menu option allows you to access the list and pick the command to execute. The macro uses a custom menu with three menu options which cannot fit clearly on one page without overlapping. Therefore we show the code of every menu option separately. If you intend to key in this macro into Lotus, you should use the code as it appears here, NOT the code as it appears in the main listing.

```
Command_line
Use DOS command line to activate applications
{GETLABEL "Enter DOS command and press [ENTER] ...",coma522}~
{RECALC sys522}{sys522}
{MENUBRANCH menu522}
```

```
Application_list
Execute any executable file using POINT and SHOOT
{WINDOWSOFF}{PANELOFF}{GOTO}IV1~{GOTO}list522~{WINDOWSON}{PANELON}
{move522}{LET coma522,@CELLPOINTER("contents")}~
{RECALC sys522}{sys522}
{MENUBRANCH menu522}

Quit
Quit the macro
```

Notice that the code in B24 is the result of a dynamic string formula; therefore, if you intend to key in the code you should use the following formula, NOT the code as it appears in the main listing.

```
24 sys522          +"{SYSTEM """&B13&"""}~"
```

The macro starts with `{MENUCALL menu522}` which starts the [menu522] custom menu. The first menu option in the [menu522] custom menu is [**C**ommand_line]:

```
Command_line
Use DOS command line to activate applications
{GETLABEL "Enter DOS command and press [ENTER] ...",coma522}~
{RECALC sys522}{sys522}
{MENUBRANCH menu522}
```

When you choose this menu option, the macro issues `{GETLABEL "Enter DOS command and press [ENTER] ...",coma522}~` to display "`Enter DOS command and press [ENTER] ...`" in the panel. When you insert a DOS command and press ENTER, Lotus stores the DOS command in cell [coma522] and the macro issues `{RECALC sys522}` to update the dynamic string formula in cell [sys522].

```
24 sys522          +"{SYSTEM """&B13&"""}~"
```

The formula in the B24 cell named [sys522] uses the content of [coma522] to create the command to execute the DOS command. For example, if you inserted the "`DIR/P`" command, [coma522] holds the "`DIR/P`" string, therefore the result of the formula is:

```
24 sys522          {SYSTEM "DIR/P"}~
```

Next the macro issues the `{sys522}` routine command which issues `{SYSTEM "DIR/P"}~` which executes the DOS command in the [sys522] routine which is the "`DIR/P`" DOS command. You can run any application such as GWBSIC or any other DOS command. When you are finished with the DOS command, the macro issues `{MENUBRANCH menu522}` to start the menu again.

The second menu option is [**A**pplication_list]:

```
Application_list
Execute any executable file using POINT and SHOOT
{WINDOWSOFF}{PANELOFF}{GOTO}IV1~{GOTO}list522~{WINDOWSON}{PANELON}
{move522}{LET coma522,@CELLPOINTER("contents")}~
{RECALC sys522}{sys522}
{MENUBRANCH menu522}
```

When you choose this menu option, the macro issues `{WINDOWSOFF}{PANELOFF}` to freeze the unwanted screen and panel activities. Then the macro issues `{GOTO}IV1~{GOTO}`

`list522~` to move the cell pointer to cell [list522] and simultaneously puts [list522] on the upper left corner of the screen display. The screen should display now:

```
LIST OF APPLICATIONS
====================
DIR|MORE
C:\BASIC\GWBASIC
DISKCOPY A: A:
DISKCOPY A: B:
```

This is the list that came with the macro as an example. You can add or remove DOS commands, batch files, *.exe files and *.com files to the macro before the macro starts. The macro continues with the `{move522}` routine command which starts the [move522] routine.

	A	B	C	D
26	move522	{GOTO}list522~{END}{DOWN}{DOWN}		
27	loop522	Use arrow keys to move and press [ENTER] to execute or [ESC] to quit..{GET key522}{ESC}		
28	!	{IF key522="{UP}"#OR#key522="{RIGHT}"}{UP} {BRANCH loop522}		
29	!	{IF key522="{DOWN}"#OR#key522="{LEFT}"}{DOWN} {BRANCH loop522}		
30	!	{IF key522="{PGUP}"}{PGUP}{BRANCH loop522}		
31	!	{IF key522="{PGUP}"#OR#key522="{HOME}"}{PGUP} {BRANCH loop522}		
32	!	{IF key522="{PGDN}"#OR#key522="{END}"}{PGDN} {BRANCH loop522}		
33	!	{IF key522="{ESC}"}{MENUBRANCH menu522}		
34	!	{IF key522="~"}{RETURN}		

The routine allows you to use the direction keys move the cell pointer while it maintains tight control on the keys that you are allowed to use. The routine starts with `{GOTO}list522~` which brings the cell pointer to the top of the list of the DOS commands. Then the macro issues `{END}{DOWN}{DOWN}` which move the cell pointer to the cell just below the list. Next the macro types

```
Use arrow keys to move and press [ENTER] to execute or [ESC] to quit..
```

to the panel. Lotus types the message into the panel because it cannot find a valid Lotus command in the message text. To hold the message in the panel and halt the macro execution, the macro issues `{GET key522}` and waits until you press a key. Now you can use the direction keys to point to the DOS command to execute and then press the ENTER key. When you press a key, Lotus stores the key in cell [key522] and the macro immediately issues `{ESC}` to clear the panel before Lotus writes it into the current cell. The macro starts with a series of `{IF}` commands to make sure that you use only the direction key or the ENTER key. From the first five conditions, we can see that you are limited only to move UP, DOWN, PGUP and PGDN. When you use one of these keys the macro issues the same command, and loops back to display the message again. The sixth condition checks if you pressed the ESC key. If so, the macro issues `{MENUBRANCH menu522}` to activate the custom menu.

The last condition checks if you pressed the ENTER key (represented by the tilde "~" in the Lotus macro language). If so, the macro issues `{RETURN}` to route the macro control back to the [**A**pplication_list] menu routine. The [**A**pplication_list] menu routine issues `{LET coma522,@CELLPOINTER("contents")}~` to store the current cell content (the DOS

command) in cell [coma522]. Now the routine issues `{RECALC sys522}` to update the dynamic code formula in the [sys522] cell, and then issues the `{sys522}` routine command which shells to DOS and activates the DOS command in cell [sys522]. When DOS finishes executing the DOS command, Lotus regains control and issues `{MENUBRANCH menu522}` to reactivate the custom menu.

The third menu option is [**Q**uit]:

```
Quit
Quit the macro
```

When you choose this menu option, Lotus reaches an empty cell and quits.

[1] Import Text Files

	A	B	C	D	E
1	*---A macro to IMPORT a file from the default directory, with a full				
2	screen of files list.				
3	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
4	range names in this column (starts with the \Z macro name)				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	*---Highlight the file name and press the [ENTER] key				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	IMPORT	/FI{?}~{NAME}{?}~			

This macro allows you to select a file to import from a full screen list of file names. The macro starts with the `/FI` macro keys (/ **F**ile **I**mport) and then issues the `{?}` macro command which halts the macro and pauses until you choose the type of file to import (**T**ext or **N**umbers) from the standard Lotus menu. When you press the ENTER key to exit the `{?}`, the macro issues the tilde "`~`", which is the same as the ENTER key, and issues `{NAME}`, which displays a full screen list of all the *.PRN files in the default directory. The macro again issues `{?}` which allows you to use the direction keys to highlight the name of the file to import and press ENTER. When you press the ENTER key, the macro issues the tilde "`~`" and imports the file.

[5] Create Files Table

```
     A                B                C                D                E
 1 *---A macro to create file names table
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer where you want the names table, the table
 5     occupies 4 columns and as many rows as necessary
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8                       FOR LOTUS 2.2 AND UP
 9 !
10 \Z              {BREAKON}
11 FILETABL        {LET rel492,@INFO("release")}~{MENUBRANCH menu492}
12 !
13 menu492         Worksheet  Print  Graph   Other  Active   Linked   Quit
14 !               List workshList prList graList alList fileList fileQuit
15 !               /FATW~~    /FATP~~/FATG~~ /FATO~~{IF @LEFT/FATL~~
16 !               {tabla492} {tabla4{tabla49{tabla4/FATA~~  {tabla492}
17 !                                          {tabla492}
18 !
19 tabla492        /WCS13~{RIGHT}/RFD4.{END}{DOWN}~{RIGHT}/RFDT3.{END}
                   {DOWN}~
20 !
21 rel492          @INFO("release")
22 !
23 ret492
```

This is the code of the custom menu called [menu492]

```
Worksheet
List worksheet files (*.WK*)
/FATW~~
{tabla492}

Print
List print files (*.PRN)
/FATP~~
{tabla492}

Graph
List graph files (*.PIC
/FATG~~
{tabla492}

Other
List all files
/FATO~~
{tabla492}

Active
List files in memory
{IF @LEFT(rel492,1)="@"}{BRANCH ret492}
/FATA~~
{tabla492}

Linked
List file linked to the current file
/FATL~~
{tabla492}

Quit
Quit the macro
```

This macro not only saves key strokes, but formats the file table automatically. The macro uses a custom menu that cannot be printed in full on one page, therefore we show every menu item separately. The code as it appears in the main listing is not complete because the menu items

overlap. This custom menu duplicates the standard Lotus menu, so it appears that you are using a Lotus menu, but you are really using a custom menu. The macro starts with `{LET rel492,@INFO("release")}~` to store the result of the `@INFO("release")` function in cell [rel492] for later use. It will use this data to determine if you are using a 2-D Lotus release or a 3-D Lotus release. Next the macro issues `{MENUBRANCH menu492}` to activate the [menu492] custom menu.

The first menu option in the custom menu is [**W**orksheet]:

```
Worksheet
List worksheet files (*.WK*)
/FATW~~
{tabla492}
```

The macro uses the `/FATW~~` macro command to create the table of the *.WK* files in the default directory and then issues the `{tabla492}` routine command which starts the [tabla492] routine.

	A	B	C	D	E
19	tabla492	/WCS13~{RIGHT}/RFD4.{END}{DOWN}~{RIGHT}/RFDT3.{END}{DOWN}~			

The [tabla492] routine first sets the width of the column of file names to 13 characters wide, and then uses `{RIGHT}` to move to the next column which is the date column and issues `/RFD4.{END}{DOWN}~` to format the date column to the type 4 format. Next the macro uses `{RIGHT}` to move to the next column, which is the time column and issues `/RFDT3.{END} {DOWN}~` to format it to type 3 format. Let's look at the following example:

```
COLSELIM.WK1  33289  0                    3963
CONTEST8.WK1  33845  0.908333333333333333 6096
CONTEST8.WK3  33845  0.906597222222222222    5331
FORMLETR.WK1  33835  0.451342592592592593 6501
FRAMERNG.WK1  33289  0                    3154
LECTURE2.WK1  33868  0.128240740740740741   17916
MENURANG.WK1  33842  0.568125             5775
PRINTWKS.WK1  33016  0.0523148148148148148  4576
PRNTHEAD.WK1  33572  0.899583333333333333 3729
QUERY.WK1     33062  0.0286342592592592593   4180
RANGSPRT.WK1  33869  0.956203703703703704 6497
ROWSELIM.WK1  33572  0.901805555555555556 3934
```

This table is an unfomatted table created using the `/FATW~~` macro keys, and here is the formatted table created by the macro:

```
COLSELIM.WK1   02/20/91      00:00:00      3963
CONTEST8.WK1   08/29/92      21:48:00      6096
CONTEST8.WK3   08/29/92      21:45:30      5331
FORMLETR.WK1   08/19/92      10:49:56      6501
FRAMERNG.WK1   02/20/91      00:00:00      3154
LECTURE2.WK1   09/21/92      03:04:40      17916
MENURANG.WK1   08/26/92      13:38:06      5775
PRINTWKS.WK1   05/23/90      01:15:20      4576
PRNTHEAD.WK1   11/30/91      21:35:24      3729
QUERY.WK1      07/08/90      00:41:14      4180
RANGSPRT.WK1   09/22/92      22:56:56      6497
ROWSELIM.WK1   11/30/91      21:38:36      3934
```

The [**P**rint], [**G**raph] and the [**O**ther] menu options use the same line of code, and therefore do not need further explanation. The next menu option is [**A**ctive]:

```
Active
```

```
List files in memory
{IF @LEFT(rel492,1)="@"}{BRANCH ret492}
/FATA~~
{tabla492}
```

First, the macro checks the first character of the label in cell [rel492] which contains the result of the `@INFO("release")` function. If "`@`" is the first character of the label in [rel492], you are using a 2-D Lotus release which does not support multiple files in the memory. Therefore the macro issues `{BRANCH rel492}` which routes macro execution control to the [ret492] routine which is an empty routine and quits. If "`@`" is not the first character of the label in [rel492], you are using a 3-D Lotus release which supports multiple files in the memory. Therefore the macro issues `/FATA~~` to create the active file table and then issues `{tabla492}` to format the table. The [**L**inked] menu option is similar to the first four menu options. The last menu option is [**Q**uit] that quits because the macro reaches an empty cell.

[4] Print a Worksheet as an ASCII File

```
     A              B              C              D              E
 1 *---A macro to save a range as unformatted ASCII file
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer at the leftmost cell of the range to file
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---Paint the range to be printed to file and press [ENTER]
 7 *---The macro saves the file with .PRN extension
 8 !
 9 !
10 \Z             {BREAKON}
11 ASCIIFIL       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~{BS}
                  {BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
12 !              {PANELOFF}/RNCRWhich range ?~~/RNDRWhich range ?~/RNC
                  RWhich range ?~.{DOWN @ROWS(Which range ?)-1}
                  {RIGHT @COLS(Which range ?)-1}~
13 !              /P{PANELON}F{?}~{PANELOFF}RRWhich range ?~OML0~MR240~
                  OUQGQ
14 !              /RNDWhich range ?~/RNDrWhich range ?~
```

Use this macro to save a range as an unformatted ASCII file. When you start the macro, it prompts you to enter the range to save. Paint the range and press the ENTER key. Remember Lotus prints only the visible painted area and not the painted cells. The macro starts with the {WINDOWSOFF}{PANELOFF} commands which freeze the screen and the panel display activity and then uses the "Safe technique" to assign the [Which range ?] name to the range that you paint. At the same time, the macro uses [Which range ?] as a prompt. But here the macro does it twice; once the macro assigns the [Which range ?] name to the range using:

```
     A              B              C              D              E
11 ASCIIFIL       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~{BS}
                  {BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
```

and then the macro issues the [RWhich range ?] range name to the same range (note the "R" before the "Which range ?") using the following code:

```
     A              B              C              D              E
12 !              {PANELOFF}/RNCRWhich range ?~~/RNDRWhich range ?~/RNC
                  RWhich range ?~.{DOWN @ROWS(Which range ?)-1}
                  {RIGHT @COLS(Which range ?)-1}~
```

When you try to print a worksheet into a file or the printer using the keyboard manually, you have to select the [**R**ange] menu option in the Lotus print menu using the "R" key or highlight the [**R**ange] menu option and press the ENTER key. Then you have to assign a file name if you want to print it into a file. However, if the file name already exists, you have to choose the [**R**eplace] menu option; therefore you have to press the "R" character the second time. If the file name is new, the [**R**eplace] menu option does not appear, so the "R" character will be pressed only once. It is not a problem when you use the keyboard, but how can we make the macro know when to issue the "R" command once or twice? To handle both cases, we issue two range names to the same range to print, the first is [Which range ?], and the second is the [RWhich range ?]. When the file is new, the macro thinks that the range name is [RWhich range ?], but when the file name already exists, the macro thinks that the range name is [Which range ?].

The macro uses the fact that the range already has the [Which range ?] name, so it issues {DOWN

`@ROWS(Which range ?)-1}` which uses the `@ROWS(Which range ?)` function to calculate how many rows down to paint in order to define [RWhich range ?]. Next the macro issues `{RIGHT @COLS(Which range ?)-1}` which uses the `@COLS(Which range ?)` function to calculate how many columns to paint to the right.

	A	B	C	D	E
13	!	/P{PANELON}F{?}~{PANELOFF}RRWhich range ?~OML0~MR240~OUQGQ			
14	!	/RNDWhich range ?~/RNDrWhich range ?~			

Last the macro issues `/P{PANELON}F` to resume the panel activity and prompts you to insert the file name. Then the macro issues `{?}` which halts the macro execution and allows you to point to the file name or to type a new name. When you press the ENTER key, the macro issues the tilde "`~`" (ENTER in macro language) and then `{PANELOFF}` to freeze the panel activity while the macro issues `RRWhich range ?~` to assign the [RWhich range ?] or the [Which range ?] as the printed range. This depends on whether the file name is new or old, as we explained earlier. Next the macro issues the `OML0~MR240~` macro keys which set the left margin to zero 0 and the right margin to 240 characters. Last, the macro issues the `OUQGQ` macro keys which choose the **U**nformatted printing option and prints the range.

[5] Combine Files and Ranges

```
     A                B                C                D                E
 1 *---A macro to COMBINE a file from the default directory, with a full
 2     screen list of files.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---Use the arrow keys to point where to combine the file
 7 *---Highlight the file name and press the [ENTER] key
 8 !
 9 !
11 \Z              {BREAKON}
11 COMBINE         {BREAKON}
11 loop005         Use the arrow keys to point where to combine the file
                   and press [ENTER]{GET key005}{ESC}{IF key005="{ESC}"}
                   {ESC 6}{BRANCH ret005}
13 !               {IF key005="{DOWN}"#OR#key005="{UP}"#OR#key005="{LEFT}"
                   #OR#key005="{RIGHT}"#OR#key005="{PGDN}"#OR#key005=
                   "{PGUP}"}{key005}{BRANCH loop005}
14 !               {IF key005="~"}~{MENUBRANCH menu005}
15 !               {BRANCH loop005}
16 !
17 key005          {UP}
18 !
19 menu005         Entire-file     Named/Specified-Range
20 !               Incorporate entiIncorporate a named or coordinate spec
21 !               /FCCE{NAME}{?}~ /FCCN{?}~{NAME}{?}~
22 !               {BRANCH loop005}{BRANCH loop005}
23 !
24 ret005
```

This macro mimics the Lotus **F**ile **C**ombine operation. But in this macro, you see a full screen list of the files in the disk, and not just the five files in the bar. The macro uses a custom menu with two menu options, identical to the Lotus standard menu options. Because one page cannot display the menu options without overlapping, we show each menu option separately. If you intend to key this macro into Lotus 1-2-3, you should key the menu options as they appear here, NOT the code as it appears in the main listing. You will find a full list of all the cell contents and formulas at the end of this section.

```
Entire-file
Incorporate entire file into worksheet
/FCCE{NAME}{?}~
{BRANCH loop005}
Named/Specified-Range
Incorporate a named or coordinate specified range from file
    into worksheet
/FCCN{?}~{NAME}{?}~
{BRANCH loop005}
```

The macro starts with a line of text which Lotus cannot compare to any of its commands. Therefore Lotus writes the text into the panel exactly as when you insert text from the keyboard without using the slash "/" key. The text that the macro writes to the panel is:

```
Use the arrow keys to point where to combine the file and press [ENTER]
```

This is a prompt message to use the direction keys to move the cell pointer to the combine location, if it is different from the current location. To hold the message in the panel so you can read it, the macro follows the message with the `{GET key005}` command which stops the macro execution until you press a key. When you press a key, Lotus stores it in cell [key005], and immediately issues `{ESC}` which clears the message from the panel before Lotus writes it

into the current cell.

Now the macro starts a series of {IF} conditions to check the key that you pressed. The first is {IF key005="{ESC}"} to check if you pressed the ESC key. If so, the macro issues {ESC 6} and then issues {BRANCH ret005} which routes the macro control to the empty routine named [ret005] and quits. The next {IF} condition is more complicated. The macro issues

```
{IF key005="{DOWN}"#OR#key005="{UP}"#OR#key005="{LEFT}"#OR#
    key005="{RIGHT}"#OR#key005="{PGDN}"#OR#key005="{PGUP}"}
```

which checks if you pressed one of the following keys: DOWN, UP, LEFT, RIGHT, PGDN or PGUP. If so, the macro issues the {key005} routine command which executes the command stored in cell [key005]. Recall that this cell holds the key that you pressed, therefore the macro issues the key that you pressed earlier. The macro carries your keys as long as they allowed by the macro. In so doing we can control and monitor the keys that you can use. Next the macro issues {BRANCH loop005} which again displays the message in the panel, and allows you to continue to move the cell pointer. The next {IF} condition is {IF key005="~"} which checks if you pressed the ENTER key. If so, the macro issues the tilde "~" command which is the same as pressing the ENTER key from the keyboard, and issues {MENUBRANCH menu005} which activates the [menu005] custom menu. If you press any other key, the macro issues {BRANCH loop005} which re-displays the prompt message. The first menu option is [**E**ntire-file]:

```
Entire-file
Incorporate entire file into worksheet
/FCCE{NAME}{?}~
{BRANCH loop005}
```

When you choose this menu option, the macro issues /FCCE and then issues {NAME}{?} to display a full screen list of all the files in the default directory, and wait until you point to the file name and press ENTER. Then the macro combines the entire file to the current cell pointer location and issues {BRANCH loop005} which branches back to the beginning, and displays the message again so that you can move to another location to combine another file. The second menu option is [**N**amed/Specified-Range]:

```
Named/Specified-Range
Incorporate a named or coordinate specified range from file
    into worksheet
/FCCN{?}~{NAME}{?}~
{BRANCH loop005}
```

When you choose this menu option, the macro issues /FCCN and then issues {?} which allows you to specify the range name or the range address to combine. When you choose a range and press ENTER, the macro issues {NAME}{?} to display a full screen list of all the files in the default directory, and wait until you point to the file name and press ENTER. Then the macro combines the chosen range from the external file to the current cell pointer location and issues {BRANCH loop005} which branches back to the beginning, and displays the message again so that you can move to another location to combine another file.

We have seen that this macro offers more options to make the file combine process easier and more productive. We have duplicated the Lotus menu because you are familiar with it, but there is another advantage. When a standard Lotus menu is activated from a macro, you must not use

the one key option to choose a menu option. Instead, you must use the direction keys to highlight the menu option and press ENTER, otherwise the macro may not work correctly. However when a custom menu is used, there is no such limitation. Therefore, when you want to have a solid macro, sometimes its better to duplicate the standard Lotus menu structure.

Here is the complete list of all the cell contents in this macro.

```
A1: U '*---A macro to COMBINE a file from the default directory, with a
      full
A2: U '    screen list of files.
A3: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A4: '    range names in this column (starts with the \Z macro name)
A5: '*---Hold the [ALT] key and press [Z] to activate the macro
A6: '*---Use the arrow keys to point where to combine the file
A7: '*---Highlight the file name and press the [ENTER] key
A8: '!
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'COMBINE
B11: '{BREAKON}
A12: 'loop005
B12: '{WINDOWSON}{PANELON}Use the arrow keys to point where to combine the
     file and press [ENTER]{GET key005}{ESC}{IF key005="{ESC}"}{ESC 6}
     {BRANCH ret005}
A13: '!
B13: '{IF key005="{DOWN}"#OR#key005="{UP}"#OR#key005="{LEFT}"#OR#key005="
     {RIGHT}"#OR#key005="{PGDN}"#OR#key005="{PGUP}"}{key005}
     {BRANCH loop005}
A14: '!
B14: '{IF key005="~"}~{MENUBRANCH menu005}
A15: '!
B15: '{BRANCH loop005}
A16: '!
A17: 'key005
B17: '{UP}
A18: '!
A19: 'menu005
B19: 'Entire-file
C19: 'Named/Specified-Range
A20: '!
B20: 'Incorporate entire file into worksheet
C20: 'Incorporate a named or coordinate specified range from file into
     worksheet
A21: '!
B21: '/FCCE{NAME}{?}~
C21: '/FCCN{?}~{NAME}{?}~
A22: '!
B22: '{BRANCH loop005}
C22: '{BRANCH loop005}
A23: '!
A24: 'ret005
```

[3] Retrieve a File from Four Drives (A:, B:, C: or D:)

```
         A                  B                  C                  D                  E
 1 *---A macro to RETRIEVE a file from the A:, B:, C: and D: drives
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 RETRIEV2       {MENUBRANCH menu032}
12 !
13 menu032        A-drive        B-drive       C-drive     D-drive
14 !              Retrieve files Retrieve filesRetrieve filRetrieve files
15 !              /FR{ESC}{ESC}A:/FR{ESC}{ESC}B/FR{ESC}{ESC/FR{ESC}{ESC}D
```

This macro can be a great help if you do much retrieving from different drives, such as when you need to update or change your files. This macro uses a custom menu to handle four drives and is also a good, simple example of how to create a custom menu in Lotus 1-2-3. Because the custom menu cannot be displayed on the screen or the paper without overlapping, we show every menu option separately. If you intend to key this macro into Lotus 1-2-3, you have to key the code as it appears here, NOT the code as it appears in the main listing. You will find a full list of all the cell contents and formulas at the end of this section.

```
A-drive
Retrieve files from drive A:
/FR{ESC}{ESC}A:\~{NAME}{?}~

B-drive
Retrieve files from drive B:
/FR{ESC}{ESC}B:\~{NAME}{?}~

C-drive
Retrieve files from drive C:
/FR{ESC}{ESC}C:\~{NAME}{?}~

D-drive
Retrieve files from drive D:
/FR{ESC}{ESC}D:\~{NAME}{?}~
```

The macro starts with the `{MENUBRANCH menu032}` command which activates the [menu032] custom menu. The first menu item is [**A**-drive]:

```
A-drive
Retrieve files from drive A:
/FR{ESC}{ESC}A:\~{NAME}{?}~
```

When you choose this option, the macro issues the `/FR` (/ **F**ile **R**etrieve) macro keys. At this point Lotus usually displays all the file names in the default directory. This may be the root directory or a second level sub-directory or even a higher level sub-directory. It appears that no matter how long the directory name is, we need to press the ESC key only twice to clear it from the panel. Therefore, the macro issues `{ESC}` twice to clear the panel, then types the "`A:\`" string into the panel and then issues the tilde "`~`", which is the same as the ENTER key. Now Lotus lists the worksheet files in the A: drive, therefore the macro issues `{NAME}`, which changes the bar display to a full screen list display, and issues `{?}` which halts the macro execution and waits until you type the name of the file or highlight a file name and press the

ENTER key. When you press ENTER, Lotus issues the tilde "~" and retrieves the file. The three other menu options are basically the same; therefore we leave them for you.

Note: Using the {?} command saves macro programming, however it is not the safest approach. You are encouraged to further investigate the case where the file to retrieve is in the second or even the third level of a sub-directory. Therefore, if the file that you want to retrieve is not in the root directory, you should type the file name and directory instead of using the BACKSPACE and pointing method.

Here is a list of all the cell contents.

```
A1: U '*---A macro to RETRIEVE a file from the A:, B:, C: and D: drives
A2: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A3: '    range names in this column (starts with the \Z macro name)
A4: '*---Hold the [ALT] key and press [Z] to activate the macro
A5: '!
A6: '!
A7: '!
A8: '!
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'RETRIEV2
B11: '{MENUBRANCH menu032}
A12: '!
A13: 'menu032
B13: 'A-drive
C13: 'B-drive
D13: 'C-drive
E13: 'D-drive
A14: '!
B14: 'Retrieve files from drive A:
C14: 'Retrieve files from drive B:
D14: 'Retrieve files from drive C:
E14: 'Retrieve files from drive D:
A15: '!
B15: '/FR{ESC}{ESC}A:\~{NAME}{?}~
C15: '/FR{ESC}{ESC}B:\~{NAME}{?}~
D15: '/FR{ESC}{ESC}C:\~{NAME}{?}~
E15: '/FR{ESC}{ESC}D:\~{NAME}{?}~
```

[1] Retrieve a File from the Default Directory

```
         A                   B                   C                   D                   E
 1 *---A macro to RETRIEVE a file from the default directory, with a full
 2     screen list
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---Highlight the file name and press the [ENTER] key
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 RETRIEVE        /FR{NAME}{?}~
```

This is a simple macro which allows you to save few key strokes each time you retrieve a file from the default directory. The macro starts with the `/FR` (/ **F**ile **R**etrieve) macro keys, at this point Lotus usually displays all the file names in the default directory. Then it issues the `{NAME}` command which changes the bar display to a full screen list display, and then issues `{?}` which halts the macro execution and waits until you type the name of the file or highlight a name of a file and press the ENTER key. When you press ENTER, Lotus issues the tilde "`~`" (same as ENTER from the keyboard) and retrieves the file.

[3] Extract a File to Four Drives (A:, B:, C:, or D:)

```
         A              B              C              D              E
 1 *---A macro to EXTRACT a file to A:, B:, C: and D: to REDUCE it's size
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 EXTRACT         {MENUBRANCH menu106}
12 !
13 menu106         A-drive     B-drive        C-drive       D-drive
14 !               Extract fileExtract files tExtract files Extract files
15 !               {HOME}/FX{?}{HOME}/FX{?}~{E{HOME}/FX{?}~{ESC/FX{?}~*{E
```

This macro can be a great help if you need to extract the file to more than one disk drive, for example when you need to update or change your files on different drives. This macro uses a custom menu to handle four drives, and it is also a good and simple example of how to create a custom menu in Lotus 1-2-3. Because the custom menu cannot be displayed on the screen or the paper without overlapping, we show every menu option separately. If you intend to key this macro into Lotus 1-2-3, you have to key the code as it appears here, NOT the code as it appears in the main listing. A full list of all the cell contents and formulas is listed at the end of this section.

```
A-drive
Extract files to drive A:
{HOME}/FX{?}~*{ESC}A:\~{NAME}{?}~{?}~R{ESC}
```

When you choose the [A-drive] menu option to extract the worksheet into a file in the A: drive, the macro issues `{HOME}` which places the cell pointer on the A1 cell, then issues `/FX{?}`, which start the extract process and then pauses until you select the correct menu option (**F**ormulas or **V**alues). Next the macro issues `*{ESC}` to clear the panel, and then types `A:\` and issues the tilde "`~`" which is the same as the ENTER key. Next, the macro issues `{NAME}` which displays a full screen list of all the *.WK* files in the A: drive and then issues `{?}` to allow you to highlight a file name or type a file name to extract. When you finish defining the file name and press the ENTER key, the macro issues the tilde "`~`" and a second `{?}` to allow you to highlight the range to extract. When you press ENTER, the macro again issues the tilde "`~`" to extract the file.

The end of the code is interesting; you may ask why press the "`R`" character and then the `{ESC}` command? Here is the trick: because the second `{?}` pauses the macro in the middle of a built-in Lotus menu, the menu can route differently if the extracted file name exists or not. If the file name already exists the "`R`" serves as the answer to the **R**eplace menu option, the macro returns to the READY mode and the `{ESC}` does nothing. However, if the file name is a new name the "`R`" is printed to the panel, therefore the `{ESC}` clears the panel before Lotus writes the "`R`" character to the current cell.

Note: Notice the `*{ESC}` macro commands which we have used to clear the panel before the macro writes the `A:\` as the drive to extract the file to. When we issue the `/FX` keys from the keyboard or a macro, Lotus displays a list of the *.WK* files in the default directory and types something to the panel which may look like `C:\123\*.WK1`. To clear this prompt from the

panel, we usually need to press the ESC key twice. Until we wrote ***Super Power***, this was the solution that we used in this macro. However we found that in the 3-D releases, only one ESC was needed and in some cases it didn't work correctly in the various versions of the 2-D releases. After extensive testing, we found that when the panel displays the `C:\123\*.WK1` (for example) and we type any character, then only one ESC is needed to clear the panel, no matter how long the prompt in the panel is and it works correctly in all the Lotus releases including 2-D and 3-D. This is why the macro writes the astrix "`*`" before the `{ESC}` command.

When the `{?}` command is used in a macro and the macro enters a Lotus menu (not a custom menu), you are expected to use one of the arrow keys and press ENTER. If you press the first capital letter of the menu option to choose it, the macro works differently. Lotus may jump to the previous menu and the macro may lose track. It appears that this bug happens only in the built-in Lotus menus, but a custom menu is free from this bug. To overcome this limitation and write a foolproof macro, we have to duplicate the built-in Lotus menu using a macro, so that the macro will look like the Lotus macro, but it is a custom menu.

Rule: If you use a macro or macro based application, NEVER USE the first capital letter of the menu option to activate a menu option if the menu is or looks like one of the Lotus standard menus. ALWAYS use the arrow keys to highlight the menu option and press the ENTER key.

```
B-drive
Extract files to drive B:
{HOME}/FX{?}~*{ESC}B:\~{NAME}{?}~{?}~R{ESC}

C-drive
Extract files to drive C:
{HOME}/FX{?}~*{ESC}C:\~{NAME}{?}~{?}~R{ESC}

D-drive
Extract files to drive D:
{HOME}/FX{?}~*{ESC}D:\~{NAME}{?}~{?}~R{ESC}
```

The three other menu options are exactly the same except for the drive name. If you intend to key this macro into Lotus 1-2-3, here is the full list of all the cell contents and formulas.

```
A1: U '*---A macro to EXTRACT a file to A:, B:, C: and D: to REDUCE it's
      size
A2: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A3: '    range names in this column (starts with the \Z macro name)
A4: '*---Hold the [ALT] key and press [Z] to activate the macro
A5: '!
A6: '!
A7: '!
A8: '!
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'EXTRACT
B11: '{MENUBRANCH menu106}
A12: '!
A13: 'menu106
B13: 'A-drive
C13: 'B-drive
D13: 'C-drive
E13: 'D-drive
A14: '!
B14: 'Extract files to drive A:
C14: 'Extract files to drive B:
D14: 'Extract files to drive C:
E14: 'Extract files to drive D:
```

```
A15: '!
B15: '{HOME}/FX{?}~{ESC}{ESC}A:\~{NAME}{?}~{?}~R{ESC}
C15: '{HOME}/FX{?}~{ESC}{ESC}B:\~{NAME}{?}~{?}~R{ESC}
D15: '{HOME}/FX{?}~{ESC}{ESC}C:\~{NAME}{?}~{?}~R{ESC}
E15: '{HOME}/FX{?}~{ESC}{ESC}D:\~{NAME}{?}~{?}~R{ESC}
```

[4] Improved File Extract to Four Drives (A:, B:, C:, or D:)

	A	B	C	D	E	F
1	*---A macro to EXTRACT a file to A:, B:, C: and D: to REDUCE it's size					
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the					
3	range names in this column (starts with the \Z macro name)					
4	*---Hold the [ALT] key and press [Z] to activate the macro					
5	!					
6	!					
7	!	THIS MACRO WORKS ONLY IN LOTUS 2.2/2.3/2.4				
8	!					
9	!					
10	\Z	{BREAKON}				
11	EXTRACT2	{MENUBRANCH menua213}				
12	!					
13	menua213	**F**ormulas	**V**alues	**Q**uit		
14	!	Saves data iSaves curre	Quit the macro			
15	!	{LET stat213{LET stat213,"V"}~				
16	!	{MENUBRANCH {MENUBRANCH menu213}				
17	!					
18	!					
19	menu213	**A**-drive	**B**-drive	**C**-drive	**D**-drive	
20	!	Extract a fi	Extract a file to drive		Extract a file to	
21	!	{LET drive213,{LET drive213,"B:"}~		{LET drive213,"C:"		
22	!	{RECALC extrac	{RECALC extract213}{ex	{RECALC extract213}		
23	!					
24	drive213	D:				
25	stat213	F				
26	!					
27	extract213	{HOME}/FXF*{ESC}D:*.WK*~{NAME}{?}~{?}~R{ESC}				

This macro does the same job as the previous EXTRACT.WK1 macro except that it does not suffer from the same limitation and you can use the first capital letter to choose the menu options. This macro saves a few strokes every time you want to extract part of a worksheet into a file on a disk. When you want to change a drive using Lotus, you may need quite a few key strokes to change the drive letter; but using this macro's custom menus, the drive selection is a snap. In this macro, we have created a custom menu which duplicates the Lotus menu. Notice that the code in the B27 cell is the result of a dynamic string formula, therefore if you intend to key this macro into 1-2-3, you should key the formula's code as it appear here, NOT as it appears in the main listing.

```
27 extract213      +"{HOME}/FX"&B25&"*{ESC}"&B24&"\*.WK*~{NAME}{?}~{?}~
                   R{ESC}"
```

The macro uses two nested custom menus which cannot be displayed clearly across a page or across the screen without overlapping, therefore we show every menu option separately. If you intend to key this macro into Lotus 1-2-3, you have to key the menu codes as they appear here, NOT the code as it appears in the main listing. A full list of all the cell contents and formulas is listed at the end of this section.

Here is the code of the first custom menu called [menua213]

```
Formulas
Saves data including formulas
{LET stat213,"F"}~
{MENUBRANCH menu213}

Values
Saves current values and labels
```

```
{LET stat213,"V"}~
{MENUBRANCH menu213}

Quit
Quit the macro
```

This is the code of the second custom menu called [menu213]

```
A-drive
Extract a file to drive A:
{LET drive213,"A:"}~
{RECALC extract213}{extract213}

B-drive
Extract a file to drive B:
{LET drive213,"B:"}~
{RECALC extract213}{extract213}

C-drive
Extract a file to drive C:
{LET drive213,"C:"}~
{RECALC extract213}{extract213}

D-drive
Extract a file to drive D:
{LET drive213,"D:"}~
{RECALC extract213}{extract213}
```

The macro starts with the `{MENUBRANCH menua213}` macro command which activates the [menua213] custom menu. This custom menu duplicates the standard Lotus menu, you think that you are using a Lotus menu but this is actually a custom menu. The first menu option in the first custom menu called [menua213] is [**F**ormulas]:

```
Formulas
Saves data including formulas
{LET stat213,"F"}~
{MENUBRANCH menu213}
```

When you choose the [**F**ormulas] menu option, the macro issues `{LET stat213,"F"}~` to store the "`F`" character in cell [stat213]. Later, the macro uses the data in [stat213] to determine if you chose the [**F**ormulas] or the [**V**alues] menu option. Next, the macro issue `{MENUBRANCH menu213}` which activates the next custom menu. The [**V**alues] menu option uses the same code and the [**Q**uit] menu option quits the macro because it reaches an empty cell. The first menu option in the second custom menu is [**A**-drive]:

```
A-drive
Extract a file to drive A:
{LET drive213,"A:"}~
{RECALC extract213}{extract213}
```

When you choose the [**A**-drive] menu option, the macro issues `{LET drive213,"A:"}~` to store the "`A`:" string in cell [drive213]. Then the macro issues `{RECALC extract213}`, which updates the dynamic code formula in cell [extract213]. This formula creates the accurate code to extract the data into a file on the A: drive. Next the macro issues the `{extract213}` routine command, and activates the [extract213] routine, which is a dynamic code routine resulting from the following formula:

```
27 extract213       +"{HOME}/FX"&B25&"*{ESC}"&B24&"\*.WK*~{NAME}{?}~{?}~
                    R{ESC}"
```

The formula uses the data in B25, [stat213], and the data in B24, [drive213]. Therefore, if you choose to extract the data as formulas, [stat213] contains the "F" label, and if you choose the A: drive, [drive213] contains the "A:" label. The result of the formula is the following code:

	A	B	C	D	E	F
27	extract213	{HOME}/FXF*{ESC}A:*.WK*~{NAME}{?}~{?}~R{ESC}				

The macro starts with {HOME} to move the cell pointer to the A1 cell, then issues /FXF, and *{ESC} to clear the panel from its content (see the previous EXTRACT.WK1 macro). Then the macro types the A:*.WK* text into the panel and issues the tilde "~", which is the same as pressing ENTER. To display a full screen list of all the files in the A: drive, the macro issues {NAME} and {?} which allows you to point to any file name or to type a new name. When you select a file name or type a file name and press the ENTER key, the macro issues a second {?} allowing you to paint (highlight) the range to extract. When you press ENTER, Lotus extracts the range to the chosen file.

When you extract a range into a file and the file name already exists, Lotus displays an extra menu which prompts you to press the **R**eplace menu option. If the file is new, Lotus does not display the menu, therefore to safely extract the range, the macro issues R{ESC}. If the file exists Lotus interprets the "R" as the **R**eplace command. However if the file name is new, Lotus considers the "R" as pure text and types it into the panel, therefore {ESC} clears the "R" from the panel. The rest of the menu options are the same except for the drive letter.

Here is the full list of cell contents in this macro.

```
A1: U '*---A macro to EXTRACT a file to A:, B:, C: and D: to REDUCE it's
      size
A2: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A3: '    range names in this column (starts with the \Z macro name)
A4: '*---Hold the [ALT] key and press [Z] to activate the macro
A5: '!
A6: '!
A7: U '          THIS MACRO WORKS ONLY IN LOTUS 2.2/2.3/2.4
A8: '!
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'EXTRACT2
B11: '{MENUBRANCH MENUA213}
A12: '!
A13: 'MENUA213
B13: 'Formulas
C13: 'Values
D13: 'Quit
A14: '!
B14: 'Saves data including formulas
C14: 'Saves current values and labels
D14: 'Quit the macro
A15: '!
B15: '{LET stat213,"F"}~
C15: '{LET stat213,"V"}~
A16: '!
B16: '{MENUBRANCH menu213}
C16: '{MENUBRANCH menu213}
A17: '!
A18: '!
A19: 'menu213
B19: 'A-drive
C19: 'B-drive
D19: 'C-drive
```

```
E19: 'D-drive
A20: '!
B20: 'Extract a files to drive A:
C20: 'Extract a files to drive B:
D20: 'Extract a files to drive C:
E20: 'Extract a files to drive D:
A21: '!
B21: '{LET drive213,"A:"}~
C21: '{LET drive213,"B:"}~
D21: '{LET drive213,"C:"}~
E21: '{LET drive213,"D:"}~
A22: '!
B22: '{RECALC extract213}{extract213}
C22: '{RECALC extract213}{extract213}
D22: '{RECALC extract213}{extract213}
E22: '{RECALC extract213}{extract213}
A23: '!
A24: 'drive213
B24: 'D:
A25: 'stat213
B25: 'F
A26: '!
A27: 'extract213
B27: U +"{HOME}/FX"&B25&"*{ESC}"&B24&"\*.WK*~{NAME}{?}~{?}~R{ESC}"
```

[4] Check File Existence from Inside a Macro

```
     A            B                C          D          E
 1 *---A macro to check for file existance
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 FILEXIST       {GETLABEL "Input File Name: ",filename017}~
                  {RECALC check017}
12 check017       {OPEN "BALANCE.WK1",R}{LET status017,1}
                  {BRANCH condition017}
13 !              {FILESIZE size017}
14 !              {LET size1017,@STRING(size017,0)}~
15 !              {LET status017,0}~{BRANCH condition017}
16 !
17 filename017    BALANCE.WK1
18 !
19 status017
20 !
21 condition017   {IF status017=0}{BEEP}File Exists! and its size is
                  {size1017} bytes.  Press any key to continue.
                  {GET key017}{ESC}{CLOSE}
22 !              {IF status017=1}{BEEP}File Is Missing! Press any key to
                  continue. {GET key017}{ESC}{CLOSE}
23 !
24 key017         ~
25 !
26 size017            56124
27 !
28 size1017       56124
```

Lotus does not offer a function or a command which allows you to check if a file exists in the disk. This macro allows you to check if a file exists in the disk. Using this kind of macro in an application can overcome an error situation when an issued file name is missing. Notice that the code in the B12 cell named [check017] is the result of the following dynamic string formula:

```
+"{OPEN """&B17&""",R}{LET status017,1}{BRANCH condition017}"
```

If you intend to key this macro into Lotus 1-2-3, you have to key the formula into the B12 cell, NOT the code as it appears in the main listing. When you start the macro, it issues the `{GETLABEL "Input File Name: ",filename017}` command, which displays the "`Input File Name: `" prompt message in the panel. When you enter the file name and press ENTER, Lotus stores the name in the B17 cell named [filename017]. To force Lotus to immediately update the content of [filename017], the macro follows the `{GETLABEL}` command with the tilde "~", which is the same as the ENTER key. Next the macro issues `{RECALC check017}` to update the dynamic string formula in cell [check017]. If the name of the file is BALANCE.WK1 and it is located in the default directory, the formula in B12, [check017], returns the

```
{OPEN "BALANCE.WK1",R}{LET status017,1}{BRANCH condition017}
```

code. Lotus executes the code which follows `{OPEN "BALANCE",R}` only if `{OPEN}` produces an error, meaning that the BALANCE.WK1 file does not exist. If the BALANCE.WK1 file does not exist, the macro issues `{LET status017,1}` which sets the

value in [status017] to "1" and issues {BRANCH condition017} which routes the macro control to the [condition017] routine. Before we continue with the [condition017] routine, let's look at the main code. The macro continues with {FILESIZE size017} which enters the size of the opened file into the cell [size017]. Next the macro issues {LET size1017,@STRING(size017,0)}~ to transform the file size number in cell [size017] and insert it into cell [size1017] as a label.

Now the macro issues {LET status017,0}~ to insert the "0" number into [status017] and issues {BRANCH condition017} which starts the [condition017] routine.

	A	B	C	D	E
21	condition017	{IF status017=0}{BEEP}File Exists! and its size is {size1017} bytes. Press any key to continue. {GET key017}{ESC}{CLOSE}			
22	!	{IF status017=1}{BEEP}File Is Missing! Press any key to continue. {GET key017}{ESC}{CLOSE}			

In the [condition017] routine, the macro first issues {IF status017=0} to check if the file exists, if it does, the macro issues {BEEP} to sound the computer's bell and then writes the "File Exists! and its size is " into the panel. Next the macro issues the {size1017} routine command, which injects the label in cell [size1017] into the panel following the previous text, and then writes "bytes. Press any key to continue." into the panel to form a complete message. To better understand the process, let's use an example. Assume that the size of the BALANCE.WK1 file is 56124 bytes, therefore the cell [size1017] contains the "56124" label.

First the macro writes "File Exists! and its size is " into the panel which displays:

File Exists! and its size is

Next the macro issues the {size1017} routine command which injects the label in [size1017] into the panel following the previous text. Now the panel displays:

File Exists! and its size is 56124

Last the macro writes "bytes. Press any key to continue." into the panel displaying:

File Exists! and its size is 56124 bytes. Press any key to continue.

We can see that the macro manipulates the panel to create a sophisticated message, which not only states that the file exists, but, also, specifies its size. To keep the message in the panel and allow you to read it, the macro issues {GET key017}, which halts the macro execution until you press a key. Then the macro issues {ESC}, which clears the message from the panel before Lotus writes the message into the current cell, and {CLOSE} to close the opened file. If the file does not exist cell [status017] contains the number "1". Therefore the {IF status017=1} condition is true and the file does not exist, the macro issues a {BEEP} to sound the computer's bell and writes "File Is Missing! Press any key to continue." into the panel, and then issues {GET key017}, which halts the macro execution until you press a key. Then the macro issues {ESC}, which clears the message from the panel before Lotus writes it into the current cell. Finally, the macro issues {CLOSE} to close the opened file.

[1] List All the Files in the Default Directory

```
        A                   B                   C                   D                   E
 1 *---A macro to LIST files and view SIZE, creation DATE and TIME
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Press [ENTER] to quit the file list display
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 FILELIST        {WINDOWSOFF}/FLO{NAME}{WINDOWSON}{NAME}{?}~
```

This is quite a simple macro. It lists not just the *.WK* macros, but all kind of files in the default file directory. The macro starts with the `{WINDOWSOFF}` command which freezes the screen display activity when it is not needed, and the `/FLO{NAME}` to display a full screen list of the files in the default file directory. Then the macro issues `{WINDOWSON}` to allow you to view the list of the file names, and `{?}` which pauses the macro execution and allows you to use the arrow keys to browse through the list of files. When you press ENTER, it tells Lotus you are finished; therefore the macro ends `{?}` and issues the tilde "`~`", which is the same as ENTER to return to READY mode.

[3] Use Full Screen Display and Point and Shoot to Delete Files

```
     A                B                C                D                E
 1 *---A macro to DELETE files from the default directory, with a full
 2     screen of files list.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---Highlight the file name and press the [ENTER] key
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 FILEDEL         {BREAKON}
12 loop107         /FE{?}~{NAME}{?}~Y~{ESC 6}
13 !               Delete more files ? (Y/N) Y {GET key107}{ESC}
14 !               {IF @UPPER(key107)<>"N"}{BRANCH loop107}
15 !
16 !
17 key107          n
```

Deleting or erasing a file from the disk using Lotus 1-2-3 is a simple task when you need to erase one or two files from time to time, but if you need to erase many files in one session, this macro will save many key strokes and will make the process almost enjoyable.

The macro starts with the `/FE` (/ **F**ile **E**rase) macro keys and then issues the `{?}` command which pauses the macro and allows you to select the type of file from the Lotus standard menu (see the note at the end of this section). The macro continues with `{NAME}` which displays a full screen list of the files in the default directory and then issues `{?}` to allow you to point to the file name and press ENTER.

When you press ENTER the macro issues `~Y~{ESC 6}` to erase the file and returns to READY mode. Next the macro writes the "`Delete more files ? (Y/N) Y` " prompt message to the panel and issues `{GET key107}` to halt macro execution and allow you to read the message. When you press a key Lotus stores the key in cell [key107] and issues `{ESC}` to clear the message from the panel before Lotus writes the message into the current cell. Next the macro issues `{IF @UPPER(key107)<>"N"}` to check if you pressed any key but the "`N`" or the "`n`" characters. If so, the macro issues `{BRANCH loop107}` which routes macro control back to the beginning, allowing you to delete more files. If this is false, the macro reaches an empty cell and quits.

Note: When the `{?}` macro command is used and the macro enters a Lotus menu (not a custom menu), you are expected to use one of the arrow keys and press ENTER. If you press the first capital letter of the menu option to choose that menu option, the macro works differently. Lotus may jump to the previous menu and the macro may lose track. It appears that this bug happens only in the built-in Lotus menus, but a custom menu is free from it. Therefore to overcome this limitation and write a foolproof macro, we have to duplicate the built-in Lotus menu using a macro, so that the menu will look like a Lotus menu, but it is really a custom menu.

Rule: If you use a macro or macro based application, you should NEVER use the first capital letter of the menu option to activate that menu option if this is one of the Lotus standard menus. ALWAYS use the arrow keys to highlight the menu option and press the ENTER key.

Format Macros

[6] Center Titles in the Middle of the Screen

```
*---A macro to CENTER a column of titles in middle of screen
*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
    range names in this column (starts with the \Z macro name)
*---Hold the [ALT] key and press [Z] to activate the macro
*---The labels to be centered must be on the most left column on screen
!
        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
                    IT WILL WORK IN LOTUS 2.0 AND UP
!
\Z              {BREAKON}
CENTER2         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~{BS}
                {BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                {LET counterb502,0}~
cont502         {LET hereabs502,@CELLPOINTER("address")}~
                {LET counter1502,0}
!               {FOR counter1502,0,@COLS(Which range ?)-1,1,labels1502}
!               {LET rel502,@INFO("release")}~{IF @LEFT(rel502,1)<>"@"}
                {GOTO}{hereabs502}~{LET counterb502,counterb502+1}~
                {IF counterb502<@SHEETS(Which range ?)}{NS}{GOTO}
                {hereabs502}~{BRANCH cont502}
!               {GOTO}Which range ?~/RNDWhich range ?~
!
counter1502             1
counter1a502            0
labels1502      {RIGHT}{LET here502,@CELLPOINTER("address")}~{LEFT}
                {FOR counter1a502,0,@ROWS(Which range ?)-1,1,
                labels1a502}~{IF counter1502<@COLS(Which range ?)-1}
                {GOTO}{here502}~{LET counter1a502,0}~
!
here502         $F$1
!
labels1a502     {IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{DEL}@REPEAT
                (" ",(72-@LENGTH(@CELLPOINTER("contents")))/2)&"{END}"
                {CALC}
!               {DOWN}
!
counterb502             1
hereabs502      $A$31
!
rel502          3.00.00
```

This macro adds enough leading spaces to a label to make it look centered in the row across the screen. For example, if the first column contains the following titles in the "A" column:

```
K.I.T.A.L. Software
612 NW Linden Ave.
Corvallis, OR 97330
```

After applying the macro on the A2..A4 range, the screen will look like this:

```
                        K.I.T.A.L. Software
                         612 NW Linden Ave.
                        Corvallis, OR 97330
```

```
11 CENTER2        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~{BS}
                  {BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb502,0}~
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the unwanted screen and panel display activities. Then the macro uses the "safe technique" to temporarily assign the [Which range ?] name to the range to be processed and simultaneously uses the [Which range ?] name as a prompt to paint the range to center. Then the macro issues `{GOTO}Which range ?~`, which moves the cell pointer to the upper left cell of [Which range ?] and `{LET counterb502,0}~`, which sets the content of cell [counterb502] to zero.

```
     A              B              C              D              E
12 cont502        {LET hereabs502,@CELLPOINTER("address")}~
                  {LET counter1502,0}
13 !              {FOR counter1502,0,@COLS(Which range ?)-1,1,labels1502}
14 !              {LET rel502,@INFO("release")}~{IF @LEFT(rel502,1)<>"@"}
                  {GOTO}{hereabs502}~{LET counterb502,counterb502+1}~
                  {IF counterb502<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs502}~{BRANCH cont502}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to its point of origin when the macro is finished, the macro issues `{LET hereabs502,@CELLPOINTER("address")}~` to store the current cell pointer address in the cell [hereabs502], and then issues `{LET counter1502,0}` to set the content of [counter1502] to zero. The `{FOR counter1502,0,@COLS(Which range ?)-1,1, labels1502}` command activates the [labels1502] routine as many times as the number of columns in [Which range ?]. Before we continue with this code, let's look at the [labels1502] routine.

```
     A              B              C              D              E
19 labels1502     {RIGHT}{LET here502,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a502,0,@ROWS(Which range ?)-1,1,
                  labels1a502}~{IF counter1502<@COLS(Which range ?)-1}
                  {GOTO}{here502}~{LET counter1a502,0}~
```

The routine issues `{RIGHT}{LET here502,@CELLPOINTER("address")}~{LEFT}` to record the address of the first cell of the next column in cell [here502]. This way, when the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here502]. The `{FOR counter1a502,0, @ROWS(Which range ?)-1,1,labels1a502}~` activates the [labels1a502] routine as many times as the number of rows in [Which range ?].

```
     A              B              C              D              E
23 labels1a502    {IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{DEL}@REPEAT
                  (" ",(72-@LENGTH(@CELLPOINTER("contents")))/2)&"{END}"
                  {CALC}
24 !              {DOWN}
```

The [labels1a502] routine issues `{IF @CELLPOINTER("type")="l"}` to check if the current cell content is a label. If so, the macro issues `{EDIT}` to switch to the EDIT mode and then `{HOME}`, which moves the cursor to the beginning of the panel, and finally `{DEL}` to delete the apostrophe "'" prefix. To clarify the macro code, let's use an example. Assume that the worksheet looks like:

```
     A              B              C              D              E
 1
 2 K.I.T.A.L. Software
```

```
3 612 NW Linden Ave.
4 Corvallis, OR 97330
5
```

and the cell pointer is on the A2 cell. Therefore, when the macro issues `{EDIT}` the panel displays:

```
'K.I.T.A.L. Software_
```

The underscore represents the cursor. When the macro issues `{HOME}` the cursor jumps to the beginning of the panel as displayed:

```
'K.I.T.A.L. Software
^
```

The "^" under the apostrophe prefix represents the cursor. Next the macro issues `{DEL}` and the panel displays:

```
K.I.T.A.L. Software
^
```

Now the macro writes `@REPEAT(" ",(72-@LENGTH(@CELLPOINTER("contents")))/2)&"` into the panel which now displays:

```
(" ",(72-@LENGTH(@CELLPOINTER("contents")))/2)&"K.I.T.A.L. Software
```

Next the macro issues the `{END}` command, which moves the cursor to the end of the text in the panel. The macro writes the closing double quotes ["] to finish the string formula which creates an enveloping formula around the "K.I.T.A.L. Software" text. The enveloping formula calculates the length of the text and adds enough leading spaces using the `@REPEAT()` Lotus function. The macro immediately follows with `{CALC}`, which turns the formula in the panel into a label, and then issues the `{DOWN}`, which moves the cell pointer to the next cell in the column and simultaneously writes the result of the formula as a label into the current cell. Therefore the worksheet will look like:

	A	B	C	D	E
1					
2		K.I.T.A.L. Software			
3	612 NW Linden Ave.				
4	Corvallis, OR 97330				
5					

When the [labels1a502] routine ends, the macro returns control to the `{FOR}` loop command in the [labels1502] routine to start the process for the next cell in the current column.

	A	B	C	D	E
19	labels1502	{RIGHT}{LET here502,@CELLPOINTER("address")}~{LEFT} {FOR counter1a502,0,@ROWS(Which range ?)-1,1, labels1a502}~{IF counter1502<@COLS(Which range ?)-1} {GOTO}{here502}~{LET counter1a502,0}~			

When the value in cell [counter1a502] reaches the number of rows in the [Which range ?] range minus one, the macro issues `{IF counter1502<@COLS(Which range ?)-1}` to check how many columns were processed. If the value in [counter1502] is less than the number of columns in [Which range ?], the macro issues the indirect `{GOTO}{here502}~` command which moves the cell pointer to the first cell of the next column. Next the macro issues `{LET`

counter1a502,0}~ to reset the value in [counter1a502] to zero. When the [labels1502] routine is finished, the macro returns control to the [cont502] routine.

	A	B	C	D	E
12	cont502	{LET hereabs502,@CELLPOINTER("address")}~			
		{LET counter1502,0}			
13	!	{FOR counter1502,0,@COLS(Which range ?)-1,1,labels1502}			
14	!	{LET rel502,@INFO("release")}~{IF @LEFT(rel502,1)<>"@"}			
		{GOTO}{hereabs502}~{LET counterb502,counterb502+1}~			
		{IF counterb502<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs502}~{BRANCH cont502}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues {LET rel502,@INFO("release")}~ to store the result of the @INFO("release") 3-D function in [rel502], and then the macro issues {IF @LEFT(rel502,1)<>"@"} which checks the first character of the content of [rel502]. If the character is the "@" character, you are using a 2-D release, otherwise you are using a 3-D release which may have more than one sheet in [Which range ?]. Therefore the macro issues the {GOTO}{hereabs502}~ indirect command which moves the cell pointer to the origin address of the macro.

The macro continues with {LET counterb502,counterb502+1}~ to increase the value in [counterb502] by one. Now the macro issues {IF counterb502<@SHEETS(Which range ?)} to check if the value in [counterb502] is less than the number of sheets in the [Which range ?] range. If so, it means that there are more sheets to process, therefore the macro issues the {NS} macro command to move the cell pointer to the next sheet. Next the macro issues the indirect {GOTO}{hereabs502}~ macro command which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of the [Which range ?] range, but in the new sheet. Last, the macro issues {BRANCH cont502} to process the cells of [Which range ?] in the new sheet.

When all the sheets are processed, the macro issues {GOTO}Which range ?~, which moves the cell pointer to the upper left cell of the first sheet of [Which range ?], and finally issues /RNDWhich range ?~ which deletes the temporary [Which range ?] range name and leaves a clean worksheet.

Split Full Name

	A	B	C	D	E
1	*---A macro to EXTRACT and SPLIT a full name to its components, i.e.				
2	FIRST NAME, M.I., LAST NAME, FIRST+M.I., and ALL THREE TOGETHER				
3	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
4	range names in this column (starts with the \Z macro name)				
5	*---Point to the cell right to the upper cell of the NAMES list				
6	*---Hold the [ALT] key and press [Z] to activate the macro				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	NAMESPLT	{WINDOWSOFF}{PANELOFF}{MENUBRANCH menu230}			
12	!				
13	menu230	First name	Last name	M.I. & First	Break all Quit
14		.	.	.	. .
15		.	.	.	.
16		.	.	.	.
17				.	.
18				.	.
19					.

This macro will split a column of full names into its components: first name, M.I. and last name in the three columns right to the column that holds full names. The macro uses a custom menu of five menu items, each of them in different column, therefore the code of all the items will not fit on the paper. We show only the titles and bring each menu item's code separately. In addition, a full list of all the cell contents is given at the end of this section to help you key this macro into 1-2-3. When you key the code, do not key the apostrophe "'" Lotus adds it automatically if a label is typed into the cell. You are asked to place the cell pointer right at the upper cell of the column of names.

	A	B	C	D	E
11	NAMESPLT	{WINDOWSOFF}{PANELOFF}{MENUBRANCH menu230}			

The macro issues the {WINDOWSOFF}PANELOFF} macro commands which freeze the screen and panel display activities and then issues {MENUBRANCH menu230} to activate the [menu230] custom menu .

The first menu option is the [**F**irst name] which allows you to extract the first name from the full name. Let us look into the macro code of this menu option.

```
First name
Extract the FIRST NAME from the left column of full names
{PANELON}@LEFT({LEFT},@FIND(" ",{LEFT},1))~{PANELOFF}{WINDOWSOFF}
/C~.{LEFT}{END}{DOWN}{RIGHT}~/RV.{END}{DOWN}~~
{WINDOWSON}{PANELON}{MENUBRANCH menu230}
```

The macro issues {PANELON} which frees the panel display activity and then types the @LEFT(text into the panel which displays:

```
@LEFT(
```

To understand the code, let's look at the following example:

	A	B	C	D	E
1	Bernard Davis				

The A1 cell contains "Bernard Davis" and the cell pointer is on the B1 cell. The macro writes the text formula directly to the panel and then issues `{LEFT}` which moves the cell pointer to the A1 cell. This is a well-known technique to write formulas using the pointer when working with Lotus from the keyboard. Here we used it from within the macro. Therefore the panel displays:

```
@LEFT(A1
```

Next the macro continues to write the `,@FIND(" ",` text into the panel which now displays:

```
@LEFT(A1,@FIND(" ",
```

The macro now issues `{LEFT}` which moves the cell pointer to the A1 cell. Therefore the panel displays:

```
@LEFT(A1,@FIND(" ",A1
```

To finish the formula, the macro types the `,1))` text, therefore the panel displays:

```
@LEFT(A1,@FIND(" ",A1,1))
```

Next the macro issues the tilde "~", the macro equivalent of the ENTER key, and writes the formula into the B1 cell. Therefore the sample worksheet will look like:

	A	B	C	D	E
1	Bernard Davis	Bernard			

To fully understand the technique, you are advised to issue the code manually from the keyboard and see the way the formula is built.

Now that we have demonstrated the basic technique, we can continue. The macro again issues `{PANELOFF}{WINDOWSOFF}` to avoid screen and panel activities during execution. Then the macro issues `/C~.{LEFT}{END}{DOWN}{RIGHT}~` which copy the new formula all the way down based on the adjacent left column length. Next the macro issues `/RV.{END}{DOWN}~~` to turn all the formulas into fixed values.

```
{MENUBRANCH menu230}
```

To end this menu option, the macro issues `{WINDOWSON}{PANELON}`, which resume the screen and panel display activities so that you can re-use the custom menu. Then it issues `{MENUBRANCH menu230}` to branch back to the custom menu.

The second menu option is the [**L**ast name] which allows you to extract the last name from the full name. Let's look into the macro code of this menu option.

```
Last name
Extract the LAST NAME from the left column of full names
/RNCtemp2230~~{PANELON}@RIGHT({LEFT},@LENGTH({LEFT})-@IF(@ISERR
    (@FIND(" ",{LEFT},@FIND(" ",{LEFT},1)+1))=1,@FIND(" ",{LEFT}
    ,1),@FIND(" ",{LEFT},@FIND(" ",{LEFT},1)+1))-1)~{PANELOFF}
    {WINDOWSOFF}{RECALC temp2230}
/C~.{LEFT}{END}{DOWN}{RIGHT}~/RNCtemp2230~.{END}{DOWN}~
{RECALC temp2230}/RV.{END}{DOWN}~~/RNDtemp2230~
{WINDOWSON}{PANELON}{MENUBRANCH menu230}
```

The macro first assigns the range name [temp2230] to the current cell using the `/RNC temp2230~~` macro code and issues `{PANELON}` which frees the panel display activity and begins to type a formula into the panel. This time the formula is more complicated than in the previous case of the [**F**irst name]; however the technique is the same. This formula will extract the last name. After typing the formula, the macro recalculates the formula using `{RECALC temp2230}`, even though we did not recalculate the formula in the previous menu option. Depending on the version of Lotus 1-2-3, this command may be needed to update this formula.

Next the macro copies the formula all the way down using the left column length and the previous technique. Then it re-assigns the range name [temp2230] to the range of formulas using `/RNCtemp2230~.{END}{DOWN}~`. Then the macro issues `{RECALC temp2230}` to recalculate this range to update the formulas. Last, the macro issues `/RV.{END}{DOWN}~~` to turn all the formulas into values. Then the macro issues `/RNDtemp2230~` to delete the [temp2230] temporary range name. To end this menu option, the macro issues `{WINDOWSON}{PANELON}` to resume the screen and panel display activities to allow you to use the custom menu again. Then the macro issues `{MENUBRANCH menu230}` to branch back to the custom menu.

```
M.I. & First
Extract the FIRST and M.I. (combined) from the left column of
    full names
/RNCtemp3230~~{PANELON}@LEFT({LEFT},@LENGTH({LEFT})-@LENGTH
    (@RIGHT({LEFT},@LENGTH({LEFT})-@IF(@ISERR(@FIND(" ",{LEFT}
    ,@FIND(" ",{LEFT},1)+1))=1,@FIND(" ",{LEFT},1),@FIND(" ",
    {LEFT},@FIND(" ",{LEFT},1)+1))-1))-1)~{PANELOFF}{WINDOWSOFF}
{RECALC temp3230}/C~.{LEFT}{END}{DOWN}{RIGHT}~/RNCtemp3230~.
{END}{DOWN}~{RECALC temp3230}/RV.{END}{DOWN}~~/RNDtemp3230~
{WINDOWSON}{PANELON}{MENUBRANCH menu230}
```

The [**M**.I. & First] menu option uses almost exactly the same code as the previous one; therefore we will skip the detailed explanation of the code.

```
Break all
Break full names in the left column to three columns of FNAME,
    M.I., LNAME
{PANELON}@LEFT({LEFT},@FIND(" ",{LEFT},1))~{PANELOFF}{WINDOWSOFF}
    {RIGHT}/RNCtemp230~~
{PANELON}@IF(@ISERR(@MID({LEFT 2},@LENGTH({LEFT})+1,@LENGTH(
    {LEFT 2})-@LENGTH({RIGHT})-@LENGTH({LEFT})-2))=1,"",@MID(
    {LEFT 2},@LENGTH({LEFT})+1,@LENGTH({LEFT 2})-@LENGTH({RIGHT})
    -@LENGTH({LEFT})-2))~{PANELOFF}{RECALC temp230}{RIGHT}
/RNCtemp1230~~{PANELON}@RIGHT({LEFT 3},@LENGTH({LEFT 3})-@IF
    (@ISERR(@FIND(" ",{LEFT 3},@FIND(" ",{LEFT 3},1)+1))=1,@FIND
    (" ",{LEFT 3},1),@FIND(" ",{LEFT 3},@FIND(" ",{LEFT 3},1)+1))
    -1)~{PANELOFF}{RECALC temp1230}
{LEFT 2}/C.{END}{RIGHT}~.{LEFT}{END}{DOWN}{RIGHT}~/RNCtemp4230~
    .{END}{DOWN}{RIGHT 2}~{RECALC temp4230}/RV.{END}{DOWN}{END}
    {RIGHT}~~/RNDtemp230~/RNDtemp1230~/RNDtemp4230~
{WINDOWSON}{PANELON}{MENUBRANCH menu230}
```

The [**B**reak all] menu option uses the same technique to create the formulas as the previous menu options, however it creates the three formulas simultaneously: one for the First name the second for the M.I. and the third formula for the Last name. The results of the macro will appear on the three columns adjacent to the columns of the full names.

```
Quit
Quit the macro
```

When you select this menu option, the macro reaches an empty cell and quits.

Here is the full list of cell contents and formulas of the macro.

```
A1: U [W9] '*---A macro to EXTRACT and SPLIT a full name to its components
          , i.e.
A2: U [W9] '    FIRST NAME, M.I., LAST NAME, FIRST+M.I., and ALL THREE
          TOGETHER
A3: [W9] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
        define the
A4: [W9] '    range names in this column (starts with the \Z macro name)
A5: [W9] '*---Point to the cell right to the upper cell of the NAMES list
A6: [W9] '*---Hold the [ALT] key and press [Z] to activate the macro
A7: [W9] '!
A8: [W9] '!
A9: [W9] '!
A10: U [W9] '\Z
B10: [W14] '{BREAKON}
A11: U [W9] 'NAMESPLT
B11: [W14] '{WINDOWSOFF}{PANELOFF}{MENUBRANCH menu230}
A12: [W9] '!
A13: [W9] 'menu230
B13: [W14] 'First name
C13: 'Last name
D13: 'M.I. & First
E13: 'Break all
F13: 'Quit
A14: [W9] '!
B14: [W14] 'Extract the FIRST NAME from the left column of full names
C14: 'Extract the LAST NAME from the left column of full names
D14: 'Extract the FIRST and M.I. (combined) from the left column of full
     names
E14: 'Break full names in the left column to three columns of FNAME, M.I.
     , LNAME
F14: 'Quit the macro
A15: [W9] '!
B15: [W14] '{PANELON}@LEFT({LEFT},@FIND(" ",{LEFT},1))~{PANELOFF}
          {WINDOWSOFF}
C15: '/RNCtemp2230~~{PANELON}@RIGHT({LEFT},@LENGTH({LEFT})-@IF(@ISERR
     (@FIND(" ",{LEFT},@FIND(" ",{LEFT},1)+1))=1,@FIND(" ",{LEFT},1),
     @FIND(" ",{LEFT},@FIND(" ",{LEFT},1)+1))-1)~{PANELOFF}{WINDOWSOFF}
     {RECALC temp2230}
D15: '/RNCtemp3230~~{PANELON}@LEFT({LEFT},@LENGTH({LEFT})-@LENGTH(@RIGHT
     ({LEFT},@LENGTH({LEFT})-@IF(@ISERR(@FIND(" ",{LEFT},@FIND(" ",{LEFT}
     ,1)+1))=1,@FIND(" ",{LEFT},1),@FIND(" ",{LEFT},@FIND(" ",{LEFT},1)
     +1))-1))-1)~{PANELOFF}{WINDOWSOFF}
E15: '{PANELON}@LEFT({LEFT},@FIND(" ",{LEFT},1))~{PANELOFF}{WINDOWSOFF}
     {RIGHT}
     /RNCtemp230~~
A16: [W9] '!
B16: [W14] '/C~.{LEFT}{END}{DOWN}{RIGHT}~/RV.{END}{DOWN}~~
C16: '/C~.{LEFT}{END}{DOWN}{RIGHT}~/RNCtemp2230~.{END}{DOWN}~
     {RECALC temp2230}/RV.{END}{DOWN}~~/RNDtemp2230~
D16: '{RECALC temp3230}
E16: '{PANELON}@IF(@ISERR(@MID({LEFT 2},@LENGTH({LEFT})+1,@LENGTH({LEFT 2}
     )-@LENGTH({RIGHT})-@LENGTH({LEFT})-2))=1,"",@MID({LEFT 2},@LENGTH(
     {LEFT})+1,@LENGTH({LEFT 2})-@LENGTH({RIGHT})-@LENGTH({LEFT})-2))~
     {PANELOFF}{RECALC temp230}{RIGHT}
A17: [W9] '!
B17: [W14] '{WINDOWSON}{PANELON}{MENUBRANCH menu230}
C17: '{WINDOWSON}{PANELON}{MENUBRANCH menu230}
D17: '/C~.{LEFT}{END}{DOWN}{RIGHT}~/RNCtemp3230~.{END}{DOWN}~
     {RECALC temp3230}/RV.{END}{DOWN}~~/RNDtemp3230~
E17: '/RNCtemp1230~~{PANELON}@RIGHT({LEFT 3},@LENGTH({LEFT 3})-@IF(@ISERR
     (@FIND(" ",{LEFT 3},@FIND(" ",{LEFT 3},1)+1))=1,@FIND(" ",{LEFT 3},1)
     ,@FIND(" ",{LEFT 3},@FIND(" ",{LEFT 3},1)+1))-1)~{PANELOFF}
     {RECALC temp1230}
A18: [W9] '!
D18: '{WINDOWSON}{PANELON}{MENUBRANCH menu230}
E18: '{LEFT 2}/C.{END}{RIGHT}~.{LEFT}{END}{DOWN}{RIGHT}~/RNCtemp4230~.
     {END}{DOWN}{RIGHT 2}~{RECALC temp4230}/RV.{END}{DOWN}{END}{RIGHT}~~
```

```
     /RNDtemp230~/RNDtemp1230~/RNDtemp4230~
A19: [W9] '!
E19: '{WINDOWSON}{PANELON}{MENUBRANCH menu230}
```

[3] Underline a Column

```
     A                    B                    C                    D                    E
 1 *---A macro to UNDERLINE the current column to one character less
 2     than the column width
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Place the cell pointer just under the column to underline
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 UNDERLIN        {WINDOWSOFF}{PANELOFF}@REPEAT("-",@CELLPOINTER("width")
                   -1){CALC}{HOME}'~
```

This macro underlines the current column with a group of hyphens, one character shorter than the width of the column. The macro first issues the `{WINDOWSOFF}{PANELOFF}` commands which freeze the screen and panel display activities. Then the macro directly writes the `@REPEAT("-",@CELLPOINTER("width")-1)` formula into the panel and issues `{CALC}`, which is the same as the F9 function key pressed from the keyboard. When the CALC or the F9 key is pressed while a formula is written in the panel, Lotus calculates the formula and replaces it with its result in the panel. Therefore the result of the `@REPEAT` formula, which is a group of hyphens having the length equal to the current column's width less one.

For example: if the current column width is 20, the result of the formula after the `{CALC}` command will be the "`-------------------`" 19 character long string of hyphens. To change it into a label, the macro issues `{HOME}`, which moves the cursor to the beginning of the panel, types an apostrophe "`'`" and finally issues the tilde "`~`" macro command, which is the same as the ENTER key, and writes the hyphen string into the current cell.

[1] Set Titles

	A	B	C	D	E
1	*---A macro to SET TITLES ON				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer at the upper left corner of the titles				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	TITLESET	{RIGHT}{DOWN}/WTB			
12	!	{GOTO}IV8192~{HOME}			

This macro creates both horizontal and vertical titles, and then places the cell pointer on the upper left cell underneath the titles. The macro starts with the `{RIGHT}{DOWN}` commands which move the cell pointer to the location under and right of the titles. Then it issues the `/WTB` (/ **W**orksheet **T**itle **B**oth) macro keys to create the horizontal and the vertical titles. To avoid the common problem of double worksheet views when the HOME key is pressed in a worksheet with both titles set to ON, the macro issues `{GOTO}IV8192~` to display a far screen and then issues `{HOME}`.

[0] Clear Titles

	A	B	C	D	E
1	*---A macro to CLEAR TITLES				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer at the upper left corner of the titles				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	TITLECLR	/WTC			

This is a simple four key macro which clears the titles. The macro issues the / **W**orksheet **T**itle **C**lear standard menu options.

[3] Toggle Titles

```
     A                B                C                D                E
 1 *---A macro to TOGGLE TITLES i.e. to switch between locked and unlocked
 2     titles
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Locate the cell pointer at the upper left corner of the titles
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8 !
 9 !
10 \Z               {BREAKON}
11 TITLETGL         {WINDOWSOFF}{PANELOFF}
12 !                {IF status044="TITLEOFF"}{LET status044,"TITLEON"}
                    {BRANCH set044}
13 !                {IF status044="TITLEON"}{LET status044,"TITLEOFF"}
                    {BRANCH clear044}
14 !
15 set044           {RIGHT}{DOWN}/WTB
16 !                {GOTO}IV8192~{HOME}
17 !
18 clear044         /WTC
19 !
20 status044        TITLEOFF
```

This is a macro toggle between the locked and unlocked titles. The macro records the last title status and uses it to switch to the opposite status. The macro starts with the `{WINDOWSOFF}{PANELOFF}` commands which freeze the unwanted screen and panel display activities. Next the macro issues `{IF status044="TITLEOFF"}` to check if the string in the B20 cell named [status044] is the "`TITLEOFF`" string. If so, the macro issues `{LET status044,"TITLEON"}`, which writes the "`TITLEON`" string in the cell [status044]. Then the macro issues `{BRANCH set044}`, which routes macro control to the [set044] routine which sets both titles on.

If the condition was false, the macro issues `{IF status044="TITLEON"}` to check if the string in B20, [status044], is "`TITLEON`". If so, the macro issues `{LET status044,"TITLEOFF"}` which writes the "`TITLEOFF`" string in cell [status044]. Then the macro issues `{BRANCH clear044}` which routes macro control to the [clear044] routine clearing both titles.

Note: the [set044] routine which locks both titles is exactly the same as the TITLESET.WK1 macro, and the [clear044] routine is identical to the TITLECLR.WK1 macro.

[3] Highlight a Range of Text

```
          A                   B                   C                   D                   E
 1 *---A macro to HIGHLIGHT and UN-HIGHLIGHT a range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 HIGHLIGT       {MENUBRANCH menu123}
12 !
13 menu123        Highlight      Un-highlight      Quit
14 !              Highlight a ranUn-highlight a ranQuit the macro
15 !              {WINDOWSOFF}{pa{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~
16 !              /RUWhich range ?RPWhich range ?~/RNDWhich range ?~{WIND
17 !              {MENUBRANCH men{MENUBRANCH menu123}
```

The macro uses a custom menu with three menu options which are shown here in full because they cannot be displayed clearly on one page without overlapping. If you plan to key this macro into Lotus 1-2-3, you have to key the code in the menu options as it appears here, NOT the code as it appears in the main listing of the macro. You will find a full list of all the cell contents at the end of this section.

```
Highlight
Highlight a range of labels or numbers
{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RNDWhich range ?~
    /RNC{WINDOWSON}{PANELON}Which range ?~{BS}{BS}{?}~
    {WINDOWSOFF}{GOTO}Which range ?~
/RUWhich range ?~/RNDWhich range ?~{WINDOWSON}
{MENUBRANCH menu123}

Un-highlight
Un-highlight a range of labels or numbers
{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RNDWhich range ?~
/RNC{WINDOWSON}{PANELON}Which range ?~{BS}{BS}{?}~{WINDOWSOFF}
    {GOTO}Which range ?~
/RPWhich range ?~/RNDWhich range ?~{WINDOWSON}
{MENUBRANCH menu123}


Quit
Quit the macro
```

To highlight the content of a cell or a range, the macro unprotects the range because Lotus displays unprotected cells in different colors. The macro starts with the `{MENUBRANCH menu123}` macro command which activates the custom menu [menu123]. The first menu option is [**H**ighlight]:

```
Highlight
Highlight a range of labels or numbers
{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RNDWhich range ?~
    /RNC{WINDOWSON}{PANELON}Which range ?~{BS}{BS}{?}~
    {WINDOWSOFF}{GOTO}Which range ?~
/RUWhich range ?~/RNDWhich range ?~{WINDOWSON}
{MENUBRANCH menu123}
```

When you select this menu option, the macro uses the "safe technique" to temporarily assign the [Which range ?] name to the range to highlight, and simultaneously uses the range name as a prompt. Next the macro issues `/RUWhich range ?~` to unprotect [Which range ?], and then

`/RNDWhich range ?` to delete the temporary [Which range ?] name. Next the macro issues `{WINDOWSON}` to resume the screen display activity, and `{MENUBRANCH menu123}` to re-start the [menu123] custom menu. The second menu option works the same way as the previous menu option. The last menu option is [**Q**uit]:

```
Quit
Quit the macro
```

When you select this option, the macro reaches an empty cell and quits.

To help you key this macro into Lotus 1-2-3, here is the full list of the cell contents in this macro.

```
A1: U '*---A macro to HIGHLIGHT and UN-HIGHLIGHT a range
A2: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A3: '    range names in this column (starts with the \Z macro name)
A4: '*---Hold the [ALT] key and press [Z] to activate the macro
A5: '!
A6: '!
A7: '!
A8: '!
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'HIGHLIGT
B11: '{MENUBRANCH menu123}
A12: '!
A13: 'menu123
B13: 'Highlight
C13: 'Un-highlight
D13: 'Quit
A14: '!
B14: 'Highlight a range of labels or numbers
C14: 'Un-highlight a range of labels or numbers
D14: 'Quit the macro
A15: '!
B15: '{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RNDWhich range ?~
     /RNC{WINDOWSON}{PANELON}Which range ?~{BS}{BS}{?}~{WINDOWSOFF}{GOTO}
     Which range ?~
C15: '{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RNDWhich range ?~
     /RNC{WINDOWSON}{PANELON}Which range ?~{BS}{BS}{?}~{WINDOWSOFF}{GOTO}
     Which range ?~
A16: '!
B16: '/RUWhich range ?~/RNDWhich range ?~{WINDOWSON}
C16: '/RPWhich range ?~/RNDWhich range ?~{WINDOWSON}
A17: '!
B17: '{MENUBRANCH menu123}
C17: '{MENUBRANCH menu123}
```

[7] Fill All the Empty Cells with Zeros

```
      A                B                C                D                E
 1 *---A macro to REPLACE all BLANKS with ZEROS in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z             {BREAKON}
11 ZEROBLNK       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~{BS}
                  {BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb050,0}~
12 cont050        {LET hereabs050,@CELLPOINTER("address")}~
                  {LET counter050,0}
13 !              {FOR counter050,0,@COLS(Which range ?)-1,1,labels050}
14 !              {LET rel050,@INFO("release")}~{IF @LEFT(rel050,1)<>"@"}
                  {GOTO}{hereabs050}~{LET counterb050,counterb050+1}~
                  {IF counterb050<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs050}~{BRANCH cont050}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter050           2
18 countera050          5
19 labels050      {RIGHT}{LET here050,@CELLPOINTER("address")}~{LEFT}
                  {FOR countera050,0,@ROWS(Which range ?)-1,1,labelsa050}~
                  {IF counter050<@COLS(Which range ?)-1}{GOTO}{here050}~
                  {LET countera050,0}~
20 !
21 here050        $C$1
22 !
23 labelsa050     {IF @CELLPOINTER("type")="b"}0
24 !              {DOWN}
25 !
26 counterb050          4
27 hereabs050     $A$1
28 !
29 rel050
```

The `@SUM` function does not care if the summed range contains blank cells. However the `@AVG` function will show an error if a cell in the averaged range is blank, because Lotus does not count blank cells; therefore, to get the right average value, all the blank cells must be filled with zeros. This macro will do the job. It will fill all the blank cells in a range with zeros so that any Lotus function will work properly.

```
      A                B                C                D                E
11 ZEROBLNK       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~{BS}
                  {BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb050,0}~
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to process, and simultaneously assigns the [Which range ?] name to the same range, which the macro uses as a prompt. Next the macro issues `{LET counterb050,0}~` to set the content of the cell [counterb050] to zero, which serves as a counter.

```
      A                B                C                D                E
12 cont050        {LET hereabs050,@CELLPOINTER("address")}~
```

```
                {LET counter050,0}
13 !            {FOR counter050,0,@COLS(Which range ?)-1,1,labels050}
14 !            {LET rel050,@INFO("release")}~{IF @LEFT(rel050,1)<>"@"}
                {GOTO}{hereabs050}~{LET counterb050,counterb050+1}~
                {IF counterb050<@SHEETS(Which range ?)}{NS}{GOTO}
                {hereabs050}~{BRANCH cont050}
15 !            {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to its point of origin when the macro is finished, the macro issues `{LET hereabs050,@CELLPOINTER("address")}~`, which stores the current cell pointer address in cell [hereabs050], and then it issues `{LET counter050,0}` to set the content of cell [counter050] to zero, which also serves as a counter. The macro issues `{FOR counter1050,0,@COLS(Which range ?)-1,1,labels050}` to activate the [labels050] routine as many times as the number of columns in the [Which range ?] range. Therefore, before we continue with this code, let's look at the [labels050] routine.

```
     A          B          C          D          E
19 labels050    {RIGHT}{LET here050,@CELLPOINTER("address")}~{LEFT}
                {FOR countera050,0,@ROWS(Which range ?)-1,1,labelsa050}~
                {IF counter050<@COLS(Which range ?)-1}{GOTO}{here050}~
                {LET countera050,0}~
```

The routine uses `{RIGHT}{LET here050,@CELLPOINTER("address")}~{LEFT}` to record the address of the first cell of the next column in cell [here050]. This way, when the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here050]. The `{FOR countera050,0,@ROWS(Which range ?)-1,1,labelsa050}~` macro commands activate the [labelsa050] routine as many times as the number of rows in the [Which range ?] range.

```
     A          B          C          D          E
23 labelsa050   {IF @CELLPOINTER("type")="b"}0
24 !            {DOWN}
```

The [labelsa050] routine issues `{IF @CELLPOINTER("type")="b"}` to check if the current cell is blank. If so, the macro writes the zero "`0`" number to the panel and issues `{DOWN}`, which moves the cell pointer one cell down and simultaneously writes the content of the panel into the current cell. If this is not true, the macro issues `{DOWN}`, which moves the cell pointer to the next cell in the current column. When the [labelsa050] routine ends, the macro returns control to the `{FOR}` command in the [labels050] routine to start the process for the next cell in the current column.

```
     A          B          C          D          E
19 labels050    {RIGHT}{LET here050,@CELLPOINTER("address")}~{LEFT}
                {FOR countera050,0,@ROWS(Which range ?)-1,1,labelsa050}~
                {IF counter050<@COLS(Which range ?)-1}{GOTO}{here050}~
                {LET countera050,0}~
```

When the value in [countera050] reaches the number of rows in the [Which range ?] range minus one, the macro issues `{IF counter050<@COLS(Which range ?)-1}` to check how many columns were processed. If the value in [counter050] is less than the number of columns in [Which range ?], the macro issues the indirect `{GOTO}{here050}~` command, which moves the cell pointer to the first cell of the next column. Next the macro issues `{LET countera050,0}~` to reset the value in [countera050] to zero. When the [labels050] routine is finished, the macro returns control to the [cont050] routine.

	A	B	C	D	E
12	cont050	{LET hereabs050,@CELLPOINTER("address")}~			
		{LET counter050,0}			
13	!	{FOR counter050,0,@COLS(Which range ?)-1,1,labels050}			
14	!	{LET rel050,@INFO("release")}~{IF @LEFT(rel050,1)<>"@"}			
		{GOTO}{hereabs050}~{LET counterb050,counterb050+1}~			
		{IF counterb050<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs050}~{BRANCH cont050}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues `{LET rel050,@INFO("release")}~` to store the result of the `@INFO("release")` 3-D function in [rel050], and then it issues `{IF @LEFT (rel050,1)<>"@"}` to check the first character of the content of [rel050]. If the character is the "`@`" character, you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the `{GOTO}{hereabs050}~` indirect macro command which moves the cell pointer to the origin address of the macro.

The macro continues with `{LET counterb050,counterb050+1}~` to increase the value in cell [counterb050] by one. Now the macro issues `{IF counterb050<@SHEETS(Which range ?)}` to check if the value in [counterb050] is less than the number of sheets in [Which range ?]. If so, there are more sheets to process, and the macro issues `{NS}` to move the cell pointer to the next sheet. Next, the macro issues the indirect `{GOTO}{hereabs050}~` command which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of [Which range ?], but in the new sheet. Last, the macro issues `{BRANCH cont050}` to process the cells of [Which range ?] in the new sheet. When all the sheets are processed, the macro issues `{GOTO}Which range ?~` which moves the cell pointer to the upper left cell of the first sheet of the [Which range ?] range, and finally issues `/RNDWhich range ?~` to delete the temporary [Which range ?] range name and leaves a clean worksheet.

[3] Set Time and File Indicator

```
     A                B                C                D                E
 1 *---A macro to set the date/time/file indicator in the lower left corner
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8                  THIS MACRO WORKS IN LOTUS 2.2 AND UP
 9 !
10 \Z               {BREAKON}
11 CLOCKSET         {MENUBRANCH menu290}
12 !
13 menu290          Standard   International   None    Clock    Filename
14 !                Use Lotus sUse current inteHide thetime and Display fi
15 !                /WGDOCSQ   /WGDOCIQ        /WGDOCNQ/WGDOCCQ /WGDOCFQ
```

This macro is quite a simple one. It saves a few strokes every time you want to change the time/file indicator appearance in the left bottom corner of the screen. This macro uses a custom menu that cannot be seen in full on the page. Therefore we show every menu item separately. The code as it appears in the main listing is not complete because the menu items overlap. The macro starts with the `{MENUBRANCH menu290}` macro command that activates the [menu290] custom menu.

```
Standard
Use Lotus standard date and time
/WGDOCSQ
```

The first menu option is [**S**tandard], which uses the standard `/WGDOCSQ` (/ **W**orksheet **G**lobal **D**efault **O**ther **C**lock **S**tandard **Q**uit) keys. Therefore, every time you use a menu option you save seven key strokes. The rest of the menu options are quite similar, therefore we show them here with no further explanation.

```
International
Use current international date and time setting
/WGDOCIQ

None
Hide the time and date indicator
/WGDOCNQ

Clock
Display the time and date indicator
/WGDOCCQ

Filename
Display file name instead of date and time
/WGDOCFQ
```

[3] Left Align Labels

	A	B	C	D	E
1	*---A macro to LEFT ALIGN all the labels in a range. If a cell				
2	contains a number it WILL NOT be changed				
3	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
4	range names in this column (starts with the \Z macro name)				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	*---Highlight the range to ALIGN, use the direction keys or type the				
7	range address or the range name and press [ENTER]				
8	!				
9	!				
10	\Z	{BREAKON}			
11	LEFTALIN	{WINDOWSOFF}{PANELOFF}/RNCRange to align?~~			
		/RNDRange to align?~			
		/RNC{PANELON}			
11		Range to align?~{BS}{BS}{WINDOWSON}{?}~{WINDOWSOFF}			
		{PANELOFF}			
12	!	{GOTO}Range to align?~/RLLRange to align?~			
		/RNDRange to align?~			

This macro changes the order of the left align process. First, it prompts you to paint the range to align and then aligns all the labels in the range. The macro issues `WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities. Then it uses the "safe technique" to prompt you to paint the range to process, and simultaneously assigns the [Range to align?] name to the same range, which it uses also as a prompt. Next the macro issues `{GOTO}Range to align?~`, which moves the cell pointer to the upper left cell of the range to align, and `/RLLRange to align?~` to left align the range named [Range to align?]. Last, the macro issues the `/RNDRange to align?~` to delete the temporary [Range to align?] range name to leave a clean worksheet.

[3] Parse Data

```
          A                B                C                D
 1 *---A macro to PARSE a range of long labels
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer at the leftmost cell of the range to PARSE
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 PARSE          /DPRFCI{?}~O{?}~Q
12 !              {MENUBRANCH menu025}
13 !
14 menu025        Edit parse line  Accept               Quit
15 !              Edit parse line  Accept parsing line  Quit the macro
16 !              /DPFE{?}~G/WDR~  /DPG/WDR~
```

This macro facilitates the task of parsing data tables, which is a little bit clumsy. It imitates the parsing process, but saves some steps and makes the process more logic. The macro prompts you for the Input range to parse and the Output parsed range using the `{?}` command. The macro begins with the `/DPRFCI` macro keys, which start the data parse process and prompts you for the input range. Then the macro issues `{?}`, which halts execution and allows you to paint the input range (including the suggested parse line) and press ENTER. When you press the ENTER key, the macro issues the tilde "`~`" (which is the same as the ENTER key pressed from the keyboard) and then issues `O{?}` which prompts you for the output range for the parsed data and allows you to point to the location of the output range. When you press ENTER, the macro issues the tilde "`~`" to set the output range and then "`Q`" to return to the READY mode. Then the macro issues `{MENUBRANCH menu025}`, which starts the [menu025] custom menu.

The first menu option is [**E**dit parse line]:

```
Edit parse line
Edit parse line suggested by the LOTUS
/DPFE{?}~G/WDR~
```

When you choose this menu option, the macro issues the `/DPFE` macro keys, which initiate the data parse edit mode, and `{?}`, which halts the macro execution and allows you to edit the suggested parse line. When you press ENTER, the macro issues the tilde "`~`" and then "`G`" to execute the parse process. Next the macro issues the `/WDR~` macro keys to delete the suggested parse line inserted by Lotus.

The second menu option is [**A**ccept]:

```
Accept
Accept parsing line
/DPG/WDR~
```

When you choose this menu option, the macro issues the `/DPG/WDR~` macro keys which execute the parse process and then delete the suggested parse line inserted by Lotus.

The Last menu option is [**Q**uit]:

```
Quit
Quit the macro
```

When you choose this menu option, the macro reaches an empty cell and quits.

[3] Center All the Labels in a Range

```
     A                    B                    C                    D                    E
 1 *---A macro to CENTER all labels in a range. If the cell contains a
 2     number it WILL NOT be changed
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---Highlight the range to ALIGN, use the direction keys or type the
 7     range address or the range name and press [ENTER]
 8 !
 9 !
10 \Z              {BREAKON}
11 CENTER          {WINDOWSOFF}{PANELOFF}/RNCRange to align?~~/RND
                   Range to align?~/RNC{PANELON}Range to align?~{BS}{BS}
                   {WINDOWSON}{?}~{WINDOWSOFF}{PANELOFF}{GOTO}
                   Range to align?~/RLCRange to align?~/RNDRange to align?~
```

This macro centers all the labels in a range. When you start, the macro prompts you to paint the range to center, then press ENTER, and the macro centers the labels in the range. The macro starts with the `{WINDOWSOFF}{PANELOFF}` commands which freeze the screen and panel display activity and then uses the "Safe technique" to assign the [Range to align?] name to the range that you paint, which it uses also as a prompt. When you paint the range and press the ENTER key, the macro issues `/RLCRange to align?~` to center the labels, and then `/RNDRange to align?~`, which deletes the temporary [Range to align?] name and leaves a clean worksheet.

[3] Right Align All the Labels in a Range

	A	B	C	D	E
1	*---A macro to RIGHT ALIGN all labels in a range. Values WILL NOT be				
2	changed.				
3	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
4	range names in this column (starts with the \Z macro name)				
5	*---Hold the <MACRO> key and press Z to activate the macro				
6	*---Highlight the range to RIGHT ALIGN, use the direction keys				
7	or type the range address or the range name.				
8	!				
9	!				
10	\Z	{BREAKON}			
11	RIGHTALN	{WINDOWSOFF}{PANELOFF}/RNCRange to align?~~/RND Range to align?~/RNC{PANELON}Range to align?~{BS}{BS} {WINDOWSON}{?}~{WINDOWSOFF}{PANELOFF}			
12	!	{GOTO}Range to align?~/RLRRange to align?~ /RNDRange to align?~			

This macro right aligns all the labels in a range. When you start, the macro prompts you to paint the range to right align, then press ENTER, and the macro aligns the labels in the range to the right. The macro starts with the `{WINDOWSOFF}{PANELOFF}` commands which freeze the screen and panel display activity and then uses the "Safe technique" to assign the [Range to align?] name to the range that you paint, which it uses also as a prompt. When you paint the range and press ENTER, the macro issues `/RLRRange to align?~`, to align all the labels to the right and `/RNDRange to align?~`, which deletes the temporary [Range to align?] name and leaves a clean worksheet.

[7] Underline All the Labels in a Range

```
     A                B                C                D                E
 1 *---A macro to underline a range. Usually is used for underlining rows
 2     to the column's width minus one, or to the up label/number width -1
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Place the cell pointer under the cell/range/row to be underlined
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 *---Highlight the cells where you want the underline (under the range)
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                      IT WILL WORK IN LOTUS 2.0 AND UP
10 \Z              {BREAKON}
11 RANGUNDL        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{PANELON}{WINDOWSON}Which range ?~
                   {BS}{BS}{?}~{PANELOFF}{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb207,0}~
12 !               {GETLABEL "Enter type of underline (+,-.!,*... etc.) "
                   ,char207}~
13 !               {PANELON} [C]olumn width or upper [L]abel width, [C/L] C
                   {GET key1207}{ESC}
14 cont207         {LET hereabs207,@CELLPOINTER("address")}~
                   {LET counter1207,0}
15 !               {FOR counter1207,0,@COLS(Which range ?)-1,1,labels1207}
16 !               {LET rel207,@INFO("release")}~{IF @LEFT(rel207,1)<>"@"}
                   {GOTO}{hereabs207}~{LET counterb207,counterb207+1}~
                   {IF counterb207<@SHEETS(Which range ?)}{NEXTSHEET}
                   {BRANCH cont207}
17 !               {GOTO}Which range ?~/RNDWhich range ?~{WINDOWSON}
18 !
19 counter1207                 2
20 counter1a207                1
21 labels1207      {RIGHT}{LET here1207,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a207,0,@ROWS(Which range ?)-1,1,
                   labels1a207}~{IF counter1207<@COLS(Which range ?)-1}
                   {GOTO}{here1207}~{LET counter1a207,0}~
22 !
23 here1207        $C$6
24 !
25 labels1a207     {UP}
26 !               {IF @UPPER(key1207)="L"#AND#@CELLPOINTER("type")="v"}
                   {DOWN}@REPEAT("{char207}",@LENGTH(@STRING({UP},0))-1)
                   {CALC}{HOME}"{END}{DOWN}{RETURN}
27 !               {IF @UPPER(key1207)="L"}{DOWN}@REPEAT("{char207}"
                   ,@LENGTH({UP})-1){CALC}{HOME}'{END}{DOWN}{RETURN}
28 !               {DOWN}@REPEAT("{char207}",{wid207}){CALC}{HOME}'{END}
29 !               {DOWN}
30 !
31 char207          +
32 !
33 wid207          3
34 !
35 key1207         ~
36 !
37 counterb207                 3
38 hereabs207      $A$6
39 !
40 rel207          3.00.00
```

The code in cell B33 is the result of the following string formula. If you intend to key this macro into Lotus 1-2-3, you have to key the following formula, NOT the code as it appears in the main listing.

```
33 wid207          @STRING(@CELLPOINTER("width")-1,0)
```

This macro will underline a range/row of labels. Usually it is used to underline rows of labels to

the column's width minus one, or to the up label/number width minus one. You have to place the cell pointer under the row to underline and highlight the row under the row of labels. For example, if you have a worksheet table with a row of fields such as:

	A	B	C	D	E
1	Code	Product name	Store	Sales	
2					

The macro can underline all the fields automatically to look like:

	A	B	C	D	E
1	Code	Product name	Store	Sales	
2	---	----------	----	----	

The macro issues the {WINDOWSOFF}{PANELOFF} commands to freeze the screen and panel display activity. Then the macro uses the "safe technique" to prompt you to paint the range to underline and simultaneously assigns the [Which range ?] name to the same range, which it also uses as a prompt. Next the macro issues {LET counterb207,0}~ to set the content of cell [counterb207] to zero, which serves as a counter.

	A	B	C	D	E
12	!	{GETLABEL "Enter type of underline (+,-.!,*... etc.) " ,char207}~			
13	!	{PANELON} [C]olumn width or upper [L]abel width, [C/L] C {GET key1207}{ESC}			

Now the macro issues

```
{GETLABEL "Enter type of underline (+,-.!,*... etc.) ",char207}~
```

that displays "Enter type of underline (+,-.!,*... etc.) " in the panel. You type the character that you want to underline. Lotus stores your response in cell B31 named [char207]. Next the macro issues {PANELON} and frees the panel activity and types

```
[C]olumn width or upper [L]abel width, [C/L] C
```

to the panel and {GET key1207} to halt the macro execution and wait until you press a key. Then Lotus stores this key in cell [key1207] and the macro immediately issues {ESC} to clear the message from the panel before Lotus writes it into the current cell.

	A	B	C	D	E
14	cont207	{LET hereabs207,@CELLPOINTER("address")}~ {LET counter1207,0}			
15	!	{FOR counter1207,0,@COLS(Which range ?)-1,1,labels1207}			
16	!	{LET rel207,@INFO("release")}~{IF @LEFT(rel207,1)<>"@"} {GOTO}{hereabs207}~{LET counterb207,counterb207+1}~ {IF counterb207<@SHEETS(Which range ?)}{NEXTSHEET} {BRANCH cont207}			
17	!	{GOTO}Which range ?~/RNDWhich range ?~{WINDOWSON}			

To be able to return to its point of origin when the macro is finished, the macro issues {LET hereabs207,@CELLPOINTER("address")}~ storing the current cell pointer address in cell [hereabs207], Then it issues {LET counter1207,0} to set the content of [counter1207] to zero. The macro issues the {FOR counter1207,0,@COLS(Which range ?)-1,1, labels1207} loop command that activates the [labels1207] routine as many times as the number of columns in [Which range ?]. Before we continue with this code, let's look at the

[labels1207] routine.

```
          A                B                C                D                E
21 labels1207      {RIGHT}{LET here1207,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a207,0,@ROWS(Which range ?)-1,1,
                   labels1a207}~{IF counter1207<@COLS(Which range ?)-1}
                   {GOTO}{here1207}~{LET counter1a207,0}~
```

The routine uses `{RIGHT}{LET here1207,@CELLPOINTER("address")}~{LEFT}` to record the address of the first cell of the next column in cell [here1207]. This way, when the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here1207]. The `{FOR counter1a207,0,@ROWS(Which range ?)-1,1,labels1a207}~` commands activate the [labels1a207] routine as many times as the number of rows in [Which range ?]; however, it's pointless to underline more than one row.

```
          A                B                C                D                E
25 labels1a207     {UP}
26 !               {IF @UPPER(key1207)="L"#AND#@CELLPOINTER("type")="v"}
                   {DOWN}@REPEAT("{char207}",@LENGTH(@STRING({UP},0))-1)
                   {CALC}{HOME}"{END}{DOWN}{RETURN}
27 !               {IF @UPPER(key1207)="L"}{DOWN}@REPEAT("{char207}"
                   ,@LENGTH({UP})-1){CALC}{HOME}'{END}{DOWN}{RETURN}
28 !               {DOWN}@REPEAT("{char207}",{wid207}){CALC}{HOME}'{END}
29 !               {DOWN}
```

The [labels1a207] routine starts with `{UP}` that moves the cell pointer to the row of labels. Then it issues `{IF @UPPER(key1207)="L"#AND#@CELLPOINTER("type")="v"}` to check if the character in [key1207] is the "`L`" or "`l`" character, and if the current cell contains a value. If so, the macro issues `{DOWN}` and types a formula to the panel. The macro first types `@REPEAT("` to the panel that displays:

```
@REPEAT("
```

and then issues the `{char207}` routine command which injects the content of [char207] into the panel. The [char207] cell holds the character you chose to form the underlined line. For example, if the character is the "`+`" character, then the panel displays:

```
@REPEAT("+
```

Next the macro continues to type `",@LENGTH(@STRING(` to the panel, therefore the panel displays:

```
@REPEAT("+",@LENGTH(@STRING(
```

If the current cell is the B2 cell, then, when the macro issues `{UP}` to point to the upper cell, the panel displays:

```
@REPEAT("+",@LENGTH(@STRING(B1..B1
```

Now the macro continues to type the `,0))-1)` text to finish the formula, and the panel displays:

```
@REPEAT("+",@LENGTH(@STRING(B1..B1,0))-1)
```

The result of this formula is a string composed of "`+`" characters and the length of the string is

the width of the content of the B1 cell minus 1. To conclude the process, the macro issues {CALC} that turns the formula in the panel into its result. Therefore, if B1 contains the "4.4444444444" value which is twelve characters long, the result of the formula is the "++++++++++" string which is 11 characters long and the panel displays:

```
++++++++++
```

Next the macro issues {HOME} that moves the cursor to the beginning of the panel and then types the double quotes ["] character to align the text to the right. The {END} macro command moves the cursor to the end of the panel and {DOWN} writes the panel to the current cell and creates the underlined line. The {RETURN} macro command routes the macro control back to the mother {FOR} loop routine. The macro used here a @STRING function because the upper cell contained value. In the next piece of code, the macro handles a case where the upper cell contains a label.

	A	B	C	D	E
27	!	{IF @UPPER(key1207)="L"}{DOWN}@REPEAT("{char207}" ,@LENGTH({UP})-1){CALC}{HOME}'{END}{DOWN}{RETURN}			

This is the same code as before, except that this time, the macro does not use the @STRING function because the upper cell already contains a label.

	A	B	C	D	E
28	!	{DOWN}@REPEAT("{char207}",{wid207}){CALC}{HOME}'{END}			
29	!	{DOWN}			

If you choose to underline the labels based on the column width and not the label length, then the macro uses the previous technique, but to type the correct formula to the panel, the macro issues the {wid207} routine command to inject the result of the formula in the [wid207] routine.

```
33 wid207          @STRING(@CELLPOINTER("width")-1,0)
```

The [wid207] routine contains the @STRING(@CELLPOINTER("width")-1,0) formula that returns the current column width minus one as a label. For example, if the column width is 4, the formula returns the label "3". Therefore the panel displays:

```
@REPEAT("+",3)
```

When the macro issues {CALC}{HOME}' the panel changes to:

```
'+++
```

and {DOWN} writes the panel to the current cell.

	A	B	C	D	E
14	cont207	{LET hereabs207,@CELLPOINTER("address")}~ {LET counter1207,0}			
15	!	{FOR counter1207,0,@COLS(Which range ?)-1,1,labels1207}			
16	!	{LET rel207,@INFO("release")}~{IF @LEFT(rel207,1)<>"@"} {GOTO}{hereabs207}~{LET counterb207,counterb207+1}~ {IF counterb207<@SHEETS(Which range ?)}{NEXTSHEET} {BRANCH cont207}			
17	!	{GOTO}Which range ?~/RNDWhich range ?~{WINDOWSON}			

The macro continues with {LET rel207,@INFO("release")}~ that store the result of the @INFO("release") function in [rel207]. Then the macro issues {IF @LEFT(rel207,1)<>"@"} to check if you are using a 2-D or a 3-D Lotus release. If the first character of the content of [rel207] is the "@" character, you are using a 2-D Lotus release, otherwise you are using a Lotus 3-D release. For a 3-D release, the macro uses the {GOTO}{hereabs207}~ indirect command to move to the first cell in the current sheet range, and then issues {LET counterb207,counterb1207+1}~ to increase the counter in [counterb207] by one.

Next the macro uses {IF counterb207<@SHEETS(Which range ?)} to compare the counter value in [counterb207] to the number of sheets in [Which range ?]. If the counter value is still less than the number of sheets in [Which range ?] it means that the macro has to process more sheets before it can quit, therefore the macro issues {NEXTSHEET}{GOTO}{hereabs207}~ to move the cell pointer to the topmost left cell to process the new sheet and issues {BRANCH cont207} to loop back to the beginning of the [cont207] routine.

When the counter in [counterb207] is equal to the number of sheets in [Which range ?], the macro issues {GOTO}Which range ?~ to place the cell pointer back on the origin place, and /RNDWhich range ?~ to delete the [Which range ?] range name to leave a clean worksheet.

[9] Search and Replace for Release 2.0/2.01 and Up

```
     A                    B                    C                    D                    E
 1 *---A macro to SEARCH and REPLACE labels or values in a range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define all
 3     the range names at the first column (starts with the \Z macro name)
 4 *---Place the cell pointer at the upper leftmost cell of the range to be
 5     searched
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 SCRHREPD        {LET lfind243,0}{LET flag243,0}{WINDOWSOFF}{PANELOFF}
                   /RNCWhich range ?~~/RNDWhich range ?~/RNC{WINDOWSON}
                   {PANELON}Which range ?~{BS}{BS}{?}~{GOTO}Which range ?~
                   {prompt243}
12 !               {LET counter1243,0}~
13 !               {FOR counter1243,0,@COLS(Which range ?)-1,1,labels1243}
14 !               {GOTO}Which range ?~/RNDWhich range ?~
15 !
16 counter1243                  2
17 counter1a243                 0
18 labels1243      {IF @UPPER(match1243)="A"}{FOR counter1a243,0,@ROWS
                   (Which range ?)- 1,1,cont2243}~{RIGHT}
                   {UP @ROWS(Which range ?)}{LET counter1a243,0}~
19 !               {IF @UPPER(match1243)="F"}{FOR counter1a243,0,@ROWS
                   (Which range ?)-1,1,search2243}~{RIGHT}
                   {UP @ROWS(Which range ?)}{LET counter1a243,0}~
20 !
21 prompt243       {GETLABEL "Type the OLD string/number to be replaced:"
                   ,dummy1243}~
22 !               {GETLABEL "Type the NEW string/number : ",dummy243}~
                   {IF @CELLPOINTER("type")="b"}{cont1243}
23 !               {GETLABEL "Match (exact) Y/N: ",match243}~
24 !               {GETLABEL "[F]ind & replace / replace [A]ll "
                   ,match1243}~
25 !
26 !
27 cont2243         {WINDOWSOFF}{PANELOFF}{IF @CELLPOINTER("type")="b"}
                   {BRANCH cont1243}
28 !               {IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~
                   {LET flag243,1}~{LET lfind243,0}~{BRANCH cont243}
29 !               {EDIT}{HOME}'~{LET flag243,0}~{LET lfind243,0}~
30 cont243         {PANELOFF}{RECALC dummy2243}~{RECALC dummy3243}~
                   {RECALC dummy4243}~{RECALC err2243}~{RECALC err3243}~
31 !               {IF @ISERR(err2243)<>1#AND#@UPPER(match243)="Y"}
                   {LET dum243,dummy3243}~{PANELON}{LET lfind243,err2243+
                   @LENGTH(dummy243)}~/Cdum243~~{BRANCH cont243}
32 !               {IF @ISERR(err3243)<>1#AND#@UPPER(match243)="N"#AND#
                   @UPPER(dummy243)<>@UPPER(dummy1243)}{LET dum243,
                   dummy4243}~{PANELON}{RECALC err3243}~{LET lfind243,
                   err3243+@LENGTH(dummy243)}~/Cdum243~~{BRANCH cont243}
33 !               {IF @ISERR(err3243)<>1#AND#@UPPER(match243)="N"#AND#
                   @UPPER(dummy243)=@UPPER(dummy1243)}{LET dum243,
                   dummy4243}~{PANELON}{LET lfind243,err3243+1}~/Cdum243~~
                   {BRANCH cont243}
34 !               {IF flag243=1}{EDIT}{HOME}{DEL}{DEL}~{BRANCH cont1243}
35 !               {EDIT}{HOME}{DEL}~
36 cont1243        {DOWN}
37 !
38 lfind243                     9
39 dummy243        666
40 dummy1243       888
41 dummy2243                    0
42 dummy3243                  ERR
43 dummy4243                  ERR
44 flag243                      1
45 match243        n
```

```
match1243       a
err2243                   ERR
err3243                   ERR
findd243                  ERR
findd1243                  54
key243          R
!
search2243      {WINDOWSOFF}{PANELOFF}{RECALC dummy2243}~
                {RECALC dummy3243}~{RECALC dummy4243}~{RECALC err2243}~
                {RECALC err3243}~
!               {LET flag243,0}~{LET lfind243,0}~{LET previous243,
                @CELLPOINTER("address")}~
!               {IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~
                {RECALC dummy2243}~{RECALC dummy3243}~{RECALC dummy4243}
                ~{LET flag243,1}~
loop243         {WINDOWSOFF}{IF @UPPER(match243)="Y"#OR#flag243=1}
                {RECALC err2243}~{RECALC findd243}~{continue1243}
!               {WINDOWSOFF}{IF @UPPER(match243)="N"}{RECALC err3243}~
                {continue2243}
!               {PANELON}{RECALC backk1243}~{back1243}{DOWN}
!
continue2243    {IF @ISERR(err3243)<>1#OR#err3243=0}{RECALC dummy2243}~
                /RVdummy2243~dum243~{PANELON}{GOTO}dum243~{WINDOWSON}
                {RECALC err3243}{EDIT}{HOME}{RIGHT err3243+1}{BEEP}
                {GET key243}~{BRANCH cvv243}
!
continue1243    {IF @ISERR(err2243)<>1#OR#err2243=0}{RECALC dummy2243}~
                /RVdummy2243~dum243~{PANELON}{GOTO}dum243~{WINDOWSON}
                {RECALC findd243}{EDIT}{HOME}{RIGHT findd243+1}{BEEP}
                {GET key243}~{BRANCH cvv243}
!
cvv243          {WINDOWSOFF}{PANELOFF}{RECALC err2243}~{RECALC err3243}~
                {IF @UPPER(key243)="R"#AND#@UPPER(match243)="Y"}
                /RVerr2243~findd1243~{BRANCH loop1243}
!               {WINDOWSOFF}{PANELOFF}{IF @UPPER(key243)="R"#AND#@UPPER
                (match243)="N"}/RVerr3243~findd1243~{BRANCH loop1243}
!               {IF @UPPER(match243)="Y"}{WINDOWSOFF}{PANELOFF}
                /RVerr2243~findd1243~
!               {IF @UPPER(match243)="N"}{WINDOWSOFF}{PANELOFF}
                /RVerr3243~findd1243~
loop1243        {IF @UPPER(key243)="R"}{LET dum243,@LEFT(dum243,
                findd1243)&dummy243&@RIGHT(dum243,@LENGTH(dum243)-
                findd1243-@LENGTH(dummy1243))}~{PANELON}{LET lfind243,
                findd1243+1}~{loop243}
!               {IF @UPPER(key243)<>"R"#AND#@UPPER(key243)<>"Q"}
                {LET lfind243,findd1243+1}~{loop243}
!               {IF @UPPER(key243)="Q"}{RECALC backk1243}~{back1243}
                {QUIT}
!               {RECALC backk243}~{back243}{DOWN}
!
!
back243         {WINDOWSOFF}{PANELOFF}/Cdum243~
backk243        $A$6
!               ~{GOTO}
previous243     $A$6
!               ~
!               {IF flag243=1}{EDIT}{HOME}{DEL}{DEL}~
!
back1243        {WINDOWSOFF}{PANELOFF}{GOTO}
backk1243       $A$6
!               ~
!               {IF flag243=1}{EDIT}{HOME}{DEL}{DEL}~
!
!
!
!
!
!
dum243          '@SIN(666)
!
!
```

```
 94 !
 95 !
 96 !                              Choose one option when you hear the beep
 97 !
 98 !
 99 !                                        Press [R] to Replace
100 !
101 !                                        Press [S] to Skip
102 !
103 !                                        Press [Q] to Quit
104 !
105 !                                           Ctrl Break
```

The code in some cells is the result of the following dynamic string formulas. If you intend to key this macro into Lotus 1-2-3, you have to key the following formulas, NOT the code as it appears in the main listing.

```
 41 dummy2243       @CELLPOINTER("contents")
 42 dummy3243       @LEFT(B41,@FIND(B40,B41,B38))&$B$39&@RIGHT(B41,@LENGTH
                    (B41)-@FIND(B40,B41,B38)-@LENGTH(B40))
 43 dummy4243       @LEFT(B41,@FIND(@UPPER(B40),@UPPER(B41),B38))&$B$39&
                    @RIGHT(B41,@LENGTH(B41)-@FIND(@UPPER(B40),@UPPER(B41)
                    ,B38)-@LENGTH(B40))
 47 err2243         @FIND(B40,B41,B38)
 48 err3243         @FIND(@UPPER(B40),@UPPER(B41),B38)
 49 findd243        @FIND(B40,B41,B38)
 75 backk243        +B77
 82 backk1243       +B77
```

This macro was written to allow users of Lotus 2.0/2.01 to perform search and replace in formulas, values and labels, but it will work in all the Lotus releases from 2.0 and up. You can replace all occurrences in one operation or you can select what to replace and what to skip, or the macro allows you to perform find only.

```
     A              B                C              D              E
 11 SCRHREPD        {LET lfind243,0}{LET flag243,0}{WINDOWSOFF}{PANELOFF}
                    /RNCWhich range ?~~/RNDWhich range ?~/RNC{WINDOWSON}
                    {PANELON}Which range ?~{BS}{BS}{?}~{GOTO}Which range ?~
                    {prompt243}
 12 !               {LET counter1243,0}~
 13 !               {FOR counter1243,0,@COLS(Which range ?)-1,1,labels1243}
 14 !               {GOTO}Which range ?~/RNDWhich range ?~
```

The macro starts with the `{LET lfind243,0}{LET flag243,0}` macro commands to set the values in the [lfind243] and [flag243] cells to zero. Next it issues `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activity. Then the macro uses the "safe technique" to prompt you to paint the range to process and simultaneously assigns the [Which range ?] name to the same range, which it also uses as a prompt. Then it issues the `{prompt243}` routine command, which activates the [prompt243] routine.

```
     A              B                C              D              E
 21 prompt243       {GETLABEL "Type the OLD string/number to be replaced:"
                    ,dummy1243}~
 22 !               {GETLABEL "Type the NEW string/number : ",dummy243}~
                    {IF @CELLPOINTER("type")="b"}{cont1243}
 23 !               {GETLABEL "Match (exact) Y/N: ",match243}~
 24 !               {GETLABEL "[F]ind & replace / replace [A]ll "
                    ,match1243}~
```

The [prompt243] routine starts with

```
{GETLABEL "Type the OLD string/number to be replaced: ",dummy1243}~
```

that display `"Type the OLD string/number to be replaced: "` in the panel. Lotus stores the string or number you type in cell [dummy1243]. Next the macro issues a second

```
{GETLABEL "Type the NEW string/number : ",dummy243}~
```

that display `"Type the NEW string/number : "` in the panel. Lotus stores the string or number you type in cell [dummy243]. Next the macro issues `{IF @CELLPOINTER("type") ="b"}` to check if the current cell is blank. If so, it issues the `{cont1243}` routine command that starts the [cont1243] routine, containing only one macro command, `{DOWN}`, that moves the cell pointer one cell down. Next it issues `{GETLABEL "Match (exact) Y/N: ", match243}~` that display the "`Match (exact) Y/N: `" prompt message in the panel. Lotus stores your response in cell [match243]. Last the macro issues

```
{GETLABEL "[F]ind & replace / replace [A]ll ",match1243}~
```

to display "`[F]ind & replace / replace [A]ll `" in the panel. Lotus stores your response in [match1243]. The macro includes all the options for modern search and replace. Now that the [prompt243] routine is finished, the macro continues with `{LET counter1243 ,0}~` to set the value in [counter1243] to zero, which serves as a counter. Next the macro issues

```
{FOR counter1243,0,@COLS(Which range ?)-1,1,labels1243}
```

that executes the [labels1243] routine as many times as the number of columns in the [Which range ?] range.

```
     A              B                C              D              E
18 labels1243    {IF @UPPER(match1243)="A"}{FOR counter1a243,0,@ROWS
                 (Which range ?)- 1,1,cont2243}~{RIGHT}
                 {UP @ROWS(Which range ?)}{LET counter1a243,0}~
19 !             {IF @UPPER(match1243)="F"}{FOR counter1a243,0,@ROWS
                 (Which range ?)-1,1,search2243}~{RIGHT}
                 {UP @ROWS(Which range ?)}{LET counter1a243,0}~
```

The [labels1243] routine issues `{IF @UPPER(match1243)="A"}` to check if you want to replace all the occurrences of the old string with the new string. If so, the routine continues with

```
{FOR counter1a243,0,@ROWS(Which range ?)-1,1,cont2243}~
```

to execute the [cont2243] routine as many times as the number of rows in [Which range ?]. The cell named [match1243] holds your response to the "`[F]ind & replace / replace [A]ll `" prompt. Therefore, no matter if you use uppercase or lowercase character, the `@UPPER(match1243)` function changes them to uppercase form.

```
     A              B                C              D              E
27 cont2243      {WINDOWSOFF}{PANELOFF}{IF @CELLPOINTER("type")="b"}
                 {BRANCH cont1243}
28 !             {IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~
                 {LET flag243,1}~{LET lfind243,0}~{BRANCH cont243}
29 !             {EDIT}{HOME}'~{LET flag243,0}~{LET lfind243,0}~
30 cont243       {PANELOFF}{RECALC dummy2243}~{RECALC dummy3243}~
                 {RECALC dummy4243}~{RECALC err2243}~{RECALC err3243}~
31 !             {IF @ISERR(err2243)<>1#AND#@UPPER(match243)="Y"}
                 {LET dum243,dummy3243}~{PANELON}{LET lfind243,err2243+
```

```
                  @LENGTH(dummy243)}~/Cdum243~~{BRANCH cont243}
32 !              {IF @ISERR(err3243)<>1#AND#@UPPER(match243)="N"#AND#
                  @UPPER(dummy243)<>@UPPER(dummy1243)}{LET dum243,
                  dummy4243}~{PANELON}{RECALC err3243}~{LET lfind243,
                  err3243+@LENGTH(dummy243)}~/Cdum243~~{BRANCH cont243}
33 !              {IF @ISERR(err3243)<>1#AND#@UPPER(match243)="N"#AND#
                  @UPPER(dummy243)=@UPPER(dummy1243)}{LET dum243,
                  dummy4243}~{PANELON}{LET lfind243,err3243+1}~/Cdum243~~
                  {BRANCH cont243}
34 !              {IF flag243=1}{EDIT}{HOME}{DEL}{DEL}~{BRANCH cont1243}
35 !              {EDIT}{HOME}{DEL}~
36 cont1243       {DOWN}
```

The [cont2243] routine starts with `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel activities. Then the macro issues `{IF @CELLPOINTER("type")="b"}` to check if the current cell is blank. If so, the macro issues `{BRANCH cont1243}` routing the macro execution control to the [cont1243] routine, however this routine contains only the `{DOWN}` macro command that moves the cell pointer to the next cell in the column. If the current cell is not blank, the macro issues `{IF @CELLPOINTER("type")="v"}` to check if the current cell contains a value. If so, the macro issues `{EDIT}` to enter to the EDIT mode, and then `{HOME}` to bring the cursor to the beginning of the panel. Then it types the apostrophe "`'`" character twice to the panel to turn the value into a label (if the cell contains a label the macro types only one apostrophe). To write the changes to the cell, the macro issues the tilde "`~`" command.

Next the macro issues `{LET flag243,1}~{LET lfind243,0}~` to set the flags to the current options you have chosen, and uses `{BRANCH cont243}` to route macro execution to the [cont243] routine.

```
          A               B                C               D               E
30 cont243        {PANELOFF}{RECALC dummy2243}~{RECALC dummy3243}~
                  {RECALC dummy4243}~{RECALC err2243}~{RECALC err3243}~
31 !              {IF @ISERR(err2243)<>1#AND#@UPPER(match243)="Y"}
                  {LET dum243,dummy3243}~{PANELON}{LET lfind243,err2243+
                  @LENGTH(dummy243)}~/Cdum243~~{BRANCH cont243}
```

The [cont243] routine issues `{RECALC dummy2243}~{RECALC dummy3243}~{RECALC dummy4243}~{RECALC err2243}~{RECALC err3243}~` to update the dynamic string formulas in [dummy2243], [dummy3243], [dummy4243], [err2243] and [err3243] respectively.

```
41 dummy2243      @CELLPOINTER("contents")
```

The formulas in the B41 cell named [dummy2243] returns the current cell content.

```
42 dummy3243      @LEFT(B41,@FIND(B40,B41,B38))&$B$39&@RIGHT(B41,@LENGTH
                  (B41)-@FIND(B40,B41,B38)-@LENGTH(B40))
```

The formula in the B42 cell named [dummy3243] returns the updated cell content when an exact match is needed (case sensitive).

```
43 dummy4243      @LEFT(B41,@FIND(@UPPER(B40),@UPPER(B41),B38))&$B$39&
                  @RIGHT(B41,@LENGTH(B41)-@FIND(@UPPER(B40),@UPPER(B41)
                  ,B38)-@LENGTH(B40))
```

The formula in the B43 cell named [dummy4243] returns the updated cell content when an exact match is not needed (case insensitive).

```
47 err2243        @FIND(B40,B41,B38)
```

The formula in the B47 cell named [err2243] returns the location of the string to be replaced inside the cell content text when an exact match is needed (case sensitive).

```
48 err3243          @FIND(@UPPER(B40),@UPPER(B41),B38)
```

The formula in the B48 cell named [err3243] returns the location of the string to be replaced inside the cell content text when an exact match is not needed (case insensitive). If the string is not found in the current cell, both the cell [err3243] and the cell [err2243] will show ERR. Therefore the macro issues `{IF @ISERR(err2243)<>1#AND#@UPPER(match243)="Y"}` to make sure that the cell [err2243] contains a number and not ERR, and you chose the exact match (case sensitive). If the combined conditions are true, the macro issues `{LET dum243 ,dummy3243}` which copies the result of the formula in cell [dummy3243] to cell [dum243] as a label.

Because the text to be replaced can appear more than once in the current cell, the macro uses `{LET lfind243,err2243+@LENGTH(dummy243)}~` to calculate where to start the next search of the old text to be replaced in the same cell again, and to store this value in cell [lfind243]. When [err2243] returns error, the macro stops the search in the current cell. Next the macro issues `/Cdum243~~` to copy the updated text to the current cell, and `{BRANCH cont243}` to route back to the [cont243] routine for the next menu.

```
     A          B          C          D          E
32 !          {IF @ISERR(err3243)<>1#AND#@UPPER(match243)="N"#AND#
              @UPPER(dummy243)<>@UPPER(dummy1243)}{LET dum243,
              dummy4243}~{PANELON}{RECALC err3243}~{LET lfind243,
              err3243+@LENGTH(dummy243)}~/Cdum243~~{BRANCH cont243}
```

If you choose the un-exact match this line of code handles the case. The code here is similar to the previous one, but it also makes sure that the new string is not equal to the old string.

```
     A          B          C          D          E
33 !          {IF @ISERR(err3243)<>1#AND#@UPPER(match243)="N"#AND#
              @UPPER(dummy243)=@UPPER(dummy1243)}{LET dum243,
              dummy4243}~{PANELON}{LET lfind243,err3243+1}~/Cdum243~~
              {BRANCH cont243}
```

This line of code is again for the case of un-exact match (case insensitive), but it handles the case when the new string is also equal to the old string. When the macro finishes the replacement process, it issues `{IF flag243=1}` to check if the cell contained a value/formula before the macro started. If so, the macro issues `{EDIT}{HOME}{DEL} {DEL}~` to delete the two apostrophes and change the label back to value/formula, and `{BRANCH cont1243}` to move the cell pointer to the next cell. If the cell contained a label when the macro started, it issues

```
     A          B          C          D          E
29 !          {EDIT}{HOME}'~{LET flag243,0}~{LET lfind243,0}~
```

which add one apostrophe to the left of the label and then set the content of [flag243] and [lfind243] to zero. The macro uses the content of [flag243] to determine if the current cell contains a value or a label before the macro started. Therefore when the macro finishes the replacement process, it issues `{EDIT}{HOME}{DEL}~` to delete the second apostrophe if the cell originally contained a label.

	A	B	C	D	E
18	labels1243	{IF @UPPER(match1243)="A"}{FOR counter1a243,0,@ROWS			
		(Which range ?)- 1,1,cont2243}~{RIGHT}			
		{UP @ROWS(Which range ?)}{LET counter1a243,0}~			
19	!	{IF @UPPER(match1243)="F"}{FOR counter1a243,0,@ROWS			
		(Which range ?)-1,1,search2243}~{RIGHT}			
		{UP @ROWS(Which range ?)}{LET counter1a243,0}~			

Till now, the macro processed only one column; therefore when the {FOR} loop command is finished, the macro issues {RIGHT}{UP @ROWS(Which range ?)} moving the cell pointer to the right and all the way up to the first cell of the next column. Then it issues {LET counter1a243,0}~ to reset the counter in [counter1a243] to zero. This concluds replacing all occurrences of the searched string.

The second part of the [labels1243] routine starts with {IF @UPPER(match1243)="F"} that checks if you chose the Find & Replace option. If so, the macro issues {FOR counter1a243,0,@ROWS(Which range ?)-1,1,search2243}~ to execute the [search2243] routine as many times as the number of rows in the range named [Which range ?].

	A	B	C	D	E
53	search2243	{WINDOWSOFF}{PANELOFF}{RECALC dummy2243}~			
		{RECALC dummy3243}~{RECALC dummy4243}~{RECALC err2243}~			
		{RECALC err3243}~			
54	!	{LET flag243,0}~{LET lfind243,0}~{LET previous243,			
		@CELLPOINTER("address")}~			
55	!	{IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~			
		{RECALC dummy2243}~{RECALC dummy3243}~{RECALC dummy4243}			
		~{LET flag243,1}~			
56	loop243	{WINDOWSOFF}{IF @UPPER(match243)="Y"#OR#flag243=1}			
		{RECALC err2243}~{RECALC findd243}~{continue1243}			
57	!	{WINDOWSOFF}{IF @UPPER(match243)="N"}{RECALC err3243}~			
		{continue2243}			
58	!	{PANELON}{RECALC backk1243}~{back1243}{DOWN}			

The [search2243] routine issues {WINDOWSOFF}{PANELOFF} which freeze the screen and panel activities, while the macro issues {RECALC dummy2243}~{RECALC dummy3243}~ {RECALC dummy4243}~{RECALC err2243}~{RECALC err3243}~ to update the dynamic formulas in [dummy2243], [dummy3243], [dummy4243], [err2243] and [err3243] respectively. The macro continues with {LET flag243,0}~{LET lfind243,0}~ that set the content of [flag243] and [lfind243] to zero, and then issues {LET previous243 ,@CELLPOINTER("address")}~ to store the current cell address in [previous243]. Lotus uses this address later to return to the current cell.

The macro continues with {IF @CELLPOINTER("type")="v"} to check if the current cell contains a value. If so, the macro issues {EDIT} to enter the EDIT mode, and {HOME} to bring the cursor to the beginning of the panel, then it types the apostrophe "'" character twice to the panel to change the value into a label. To write the changes to the cell, the macro issues tilde "~" (the macro equivalent of ENTER).

Next the macro issues {LET flag243,1}~{LET lfind243,0}~ to set the flags to the current options you have chosen. Next the macro returns and issues the following five {RECALC dummy2243}~{RECALC dummy3243}~{RECALC dummy4243}~{RECALC err2243}~ macro commands to update the dynamic string formulas in [dummy2243], [dummy3243], [dummy4243] and [err2243] respectively. It seems that it is unnecessary to repeat and update these formulas, however our experience with this macro shows that this is needed for the macro to correctly function in all cases. To record that the cell contains a value or a formula, the macro

issues {LET flag243,1}~ that set the content of [flag243] to "1".

Now the macro issues {IF @UPPER(match243)="Y"#OR#flag243=1} to make sure that you chose the un-exact match (case insensitive) or that the cell contains a value or formula. If so, the macro issues {RECALC err2243}~{RECALC findd243}~ to update the formulas in [err2243] and [findd243] respectively. The formula in the B49 cell named [findd243] is:

```
 49 findd243        @FIND(B40,B41,B38)
```

and returns the location of the old string in the current cell. Next it issues the {continue1243} routine command that activates the [continue1243] routine.

	A	B	C	D	E
62	continue1243	{IF @ISERR(err2243)<>1#OR#err2243=0}{RECALC dummy2243}~ /RVdummy2243~dum243~{PANELON}{GOTO}dum243~{WINDOWSON} {RECALC findd243}{EDIT}{HOME}{RIGHT findd243+1}{BEEP} {GET key243}~{BRANCH cvv243}			

The [continue1243] routine issues {IF @ISERR(err2243)<>1#OR#err2243=0} to make sure that the old string exists in the current cell's content. If the condition is true, the macro issues {RECALC dummy2243}~ to update the formula in [dummy2243], and then issues /RVdummy2243~dum243~ to copy the result of the formula in [dummy2243] to [dum243] as a label. Then the macro issues {PANELON}, which resumes the panel activity, and {GOTO} dum243~, which places the cell pointer on the B91 cell named [dum243].

This routine is built to let you find a string occurrence and replace or skip it, therefore you see the text, and the macro points to the string to be replaced in the text when it finds it. The macro uses the panel to display the text and the string to replace. The macro issues {WINDOWSON} that also resumes the screen activity. If you look at the screen when the cell pointer is on B91, [dum243], the following screen is visible with instructions for how to continue.

```
'@SIN(666)
```

	A	B	C	D	E
91	dum243	'@SIN(666)			
92	!				
93	!				
94	!				
95	!				
96	!		Choose one option when you here the beep		
97	!				
98	!				
99	!			Press [R] to Replace	
100	!				
101	!			Press [S] to Skip	
102	!				
103	!			Press [Q] to Quit	
104	!				
105	!			Ctrl Break	

Because the cell pointer is on the B91 cell, the panel displays the content of the cell, @SIN (666) as in this example. Now the macro issues {RECALC findd243} that updates the formula in [findd243]. This formula calculates the location of the old string in the cell content. For example if you choose the "666" as the old string and replace it with the new "77" string, the formula in the cell named [findd243] returns the number "5" because the @FIND function starts to count from zero. Next the macro issues {EDIT}{HOME}{RIGHT findd243+1} {BEEP} putting

the cursor on the first character of the old string. In our example the panel will display:

```
'@SIN(666)
^
```

The "^" shows the location of the cursor, which points to the old string to replace. To hold the text in the panel the macro issues `{GET key243}` that halts the macro execution and waits until you press a key. Then the macro issues `{BRANCH cvv243}` which activates the [cvv243] routine.

	A	B	C	D	E
64	cvv243	{WINDOWSOFF}{PANELOFF}{RECALC err2243}~{RECALC err3243}~ {IF @UPPER(key243)="R"#AND#@UPPER(match243)="Y"} /RVerr2243~findd1243~{BRANCH loop1243}			
65	!	{WINDOWSOFF}{PANELOFF}{IF @UPPER(key243)="R"#AND#@UPPER (match243)="N"}/RVerr3243~findd1243~{BRANCH loop1243}			
66	!	{IF @UPPER(match243)="Y"}{WINDOWSOFF}{PANELOFF} /RVerr2243~findd1243~			
67	!	{IF @UPPER(match243)="N"}{WINDOWSOFF}{PANELOFF} /RVerr3243~findd1243~			
68	loop1243	{IF @UPPER(key243)="R"}{LET dum243,@LEFT(dum243, findd1243)&dummy243&@RIGHT(dum243,@LENGTH(dum243)- findd1243-@LENGTH(dummy1243))}~{PANELON}{LET lfind243, findd1243+1}~{loop243}			
69	!	{IF @UPPER(key243)<>"R"#AND#@UPPER(key243)<>"Q"} {LET lfind243,findd1243+1}~{loop243}			
70	!	{IF @UPPER(key243)="Q"}{RECALC backk1243}~{back1243} {QUIT}			
71	!	{RECALC backk243}~{back243}{DOWN}			

The [cvv243] routine is in charge of replacing the old string with the new string and displaying the text back to the screen, if more than one occurrence of the old string is found in the cell's content. As we mentioned earlier, the whole process takes place in the panel, therefore the macro issues `{WINDOWSOFF}{PANELOFF}` for most of the routine and frees the panel when the macro finds a matched string. The macro continues with `{RECALC err2243}~{RECALC err3243}~` to update the formulas in [err2243] and [err3243] and then issues `{IF @UPPER (key243)="R"#AND#@UPPER(match243)="Y"}`. If you pressed the "`R`" key and also chose the exact match, the macro issues `/RVerr3243~findd1243~` to copy the result of the formula in [err3243] to [findd1243] as a label, and uses `{BRANCH loop1243}` to route the macro execution control to the [loop1243] routine.

	A	B	C	D	E
68	loop1243	{IF @UPPER(key243)="R"}{LET dum243,@LEFT(dum243, findd1243)&dummy243&@RIGHT(dum243,@LENGTH(dum243)- findd1243-@LENGTH(dummy1243))}~{PANELON}{LET lfind243, findd1243+1}~{loop243}			
69	!	{IF @UPPER(key243)<>"R"#AND#@UPPER(key243)<>"Q"} {LET lfind243,findd1243+1}~{loop243}			
70	!	{IF @UPPER(key243)="Q"}{RECALC backk1243}~{back1243} {QUIT}			
71	!	{RECALC backk243}~{back243}{DOWN}			

The [loop1243] routine issues `{IF @UPPER(key243)="R"}` to check if you pressed the "`R`" key. If so, the macro issues the

```
{LET dum243,@LEFT(dum243,findd1243)&dummy243&@RIGHT(dum243,@LENGTH
    (dum243)-findd1243-@LENGTH(dummy1243))}~
```

macro command that stores the result of

```
@LEFT(dum243,findd1243)&dummy243&@RIGHT(dum243,@LENGTH(dum243)
    -findd1243-@LENGTH(dummy1243))
```

in [dum243] which is the current cell (currently located on the upper left cell of the screen) as a label. This formula rebuilds the updated text in the current cell. As we can see,

```
@LEFT(dum243,findd1243)
```

returns the left part of the original text up to the point where the old string has been found. The [dummy243] cell holds the new string and

```
@RIGHT(dum243,@LENGTH(dum243)-findd1243-@LENGTH(dummy1243))
```

holds the right part of the original text in the current cell from the old string (not included) and up to the end. Therefore the full formula returns the updated text in the current cell. Next the macro issues `{PANELON}`, but because the cell pointer is positioned on [dum243], the updated text appears in the panel so you can actually see the result of the replacement operation in the panel. Because the current cell can have more occurrences of the old string, the macro issues `{LET lfind243,findd1243+1}~` which update the value in [lfind243] to the value in [findd1243] plus one.

The macro uses the content of [lfind243] to know where to start looking for the next old string inside the text, but because the macro already found a previous occurrence, there is no point to searching from the beginning of the text. Therefore the macro inserts the location where the last occurrence has been found in [lfind243], so next time the macro will search from that point and not from the beginning. This is also very important when you choose to skip and leave the old string without change. There is no point in processing this string again. Without this command, the macro would enter an endless loop.

Now the macro issues `{loop243}` that starts the [loop243] routine which carries the next search in the current cell, which continues until the macro no longer finds the old string in the current cell's text. The [continue1243] routine ends when the `{IF @ISERR(err3243)<> 1#OR#err3243=0}` condition is false. Then the macro routes back to the second part of B58, [loop243].

	A	B	C	D	E
56	loop243	{WINDOWSOFF}{IF @UPPER(match243)="Y"#OR#flag243=1} {RECALC err2243}~{RECALC findd243}~{continue1243}			
57	!	{WINDOWSOFF}{IF @UPPER(match243)="N"}{RECALC err3243}~ {continue2243}			
58	!	{PANELON}{RECALC backk1243}~{back1243}{DOWN}			

When the [continue1243] routine ends, the macro issues `{WINDOWSOFF}` to freeze the screen activity and `{IF @UPPER(match243)="N"}` to check if you choose the un-exact match (case insensitive) option. If so, the macro continues with a similar code to the previous case, but this time activates the [continue2243] routine instead of the [continue1243] routine. We are not going to investigate this routine any more because it is similar to the [continue1243] routine. If both of the conditions are false, the macro issues `{PANELON}` `{RECALC backk1243}~` to free the panel activity and update the formula in [backk1243].

```
82 backk1243       +B77
```

The formula in cell [backk1243] borrows the content of cell [previous243] which holds the address of the cell in the [Which range ?] range that the macro currently is processing (recall the {LET previous243,@CELLPOINTER("address")}~ macro command in the B54 cell). The cell [backk1243] is the dynamic part of the [back1243] routine.

	A	B	C	D	E
81	back1243	{WINDOWSOFF}{PANELOFF}{GOTO}			
82	backk1243	$A$6			
83	!	~			
84	!	{IF flag243=1}{EDIT}{HOME}{DEL}{DEL}~			

The [back1243] routine issues {WINDOWSOFF}{PANELOFF} to freeze the screen and panel activities, while it issues {GOTO}$A$6~. In this example, A6 is the cell that the macro processes. Till now, all the work was done on the content of [dum243] which is a copy of the original cell's content, therefore the macro moves the cell pointer back to the original cell. Next the macro issues {IF flag243=1} to check if the cell contained a value. If so, it means that the cell contained a value, therefore the macro issues {EDIT}{HOME}{DEL}{DEL}~ to delete the two apostrophes and turn the cell's content back to a value.

This is one of the most complicated macros in ***Super Power***. You should invest some time to follow and understand the code of this macro. This is one of the finest examples of the Lotus macro language potential. Because this macro is so complicated, we show the full list of cell codes in case you want to key it into 1-2-3.

```
A1: U [W13] '*---A macro to SEARCH and REPLACE labels or values in a range
A2: [W13] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define all
A3: [W13] '    the range names at the first column (starts with the \Z
          macro name)
A4: [W13] '*---Place the cell pointer at the upper leftmost cell of the
          range to be
A5: [W13] '    searched
A6: [W13] '*---Hold the [ALT] key and press [Z] to activate the macro
A7: [W13] '!
A8: [W13] '!
A9: [W13] '!
A10: U [W13] '\Z
B10: '{BREAKON}
A11: U [W13] 'SCRHREPD
B11: '{LET lfind243,0}{LET flag243,0}{WINDOWSOFF}{PANELOFF}/RNC
     Which range ?~~/RNDWhich range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
     {BS}{BS}{?}~{GOTO}Which range ?~{prompt243}
A12: [W13] '!
B12: '{LET counter1243,0}~
A13: [W13] '!
B13: '{FOR counter1243,0,@COLS(Which range ?)-1,1,labels1243}
A14: [W13] '!
B14: '{GOTO}Which range ?~/RNDWhich range ?~
A15: [W13] '!
A16: [W13] 'counter1243
B16: 2
A17: [W13] 'counter1a243
B17: 0
A18: [W13] 'labels1243
B18: '{IF @UPPER(match1243)="A"}{FOR counter1a243,0,@ROWS(Which range ?)-1
     ,1,cont2243}~{RIGHT}{UP @ROWS(Which range ?)}{LET counter1a243,0}~
A19: [W13] '!
B19: '{IF @UPPER(match1243)="F"}{FOR counter1a243,0,@ROWS(Which range ?)-1
     ,1,search2243}~{RIGHT}{UP @ROWS(Which range ?)}{LET counter1a243,0}~
A20: [W13] '!
A21: [W13] 'prompt243
B21: '{GETLABEL "Type the OLD string/number to be replaced: ",dummy1243}~
A22: [W13] '!
```

```
B22: '{GETLABEL "Type the NEW string/number : ",dummy243}~{IF @CELLPOINTER
     ("type")="b"}{cont1243}
A23: [W13] '!
B23: '{GETLABEL "Match (exact) Y/N: ",match243}~
A24: [W13] '!
B24: '{GETLABEL "[F]ind & replace / replace [A]ll ",match1243}~
A25: [W13] '!
A26: [W13] '!
A27: [W13] 'cont2243
B27: '{WINDOWSOFF}{PANELOFF}{IF @CELLPOINTER("type")="b"}{BRANCH cont1243}
A28: [W13] '!
B28: '{IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~{LET flag243,1}~
     {LET lfind243,0}~{BRANCH cont243}
A29: [W13] '!
B29: '{EDIT}{HOME}'~{LET flag243,0}~{LET lfind243,0}~
A30: [W13] 'cont243
B30: '{PANELOFF}{RECALC dummy2243}~{RECALC dummy3243}~{RECALC dummy4243}~
     {RECALC err2243}~{RECALC err3243}~
A31: [W13] '!
B31: '{IF @ISERR(err2243)<>1#AND#@UPPER(match243)="Y"}{LET dum243,dummy3243}
     ~{PANELON}{LET lfind243,err2243+@LENGTH(dummy243)}~/Cdum243~~
     {BRANCH cont243}
A32: [W13] '!
B32: '{IF @ISERR(err3243)<>1#AND#@UPPER(match243)="N"#AND#@UPPER(dummy243)
     <>@UPPER(dummy1243)}{LET dum243,dummy4243}~{PANELON} {RECALC err3243}~
     {LET lfind243,err3243+@LENGTH(dummy243)}~/Cdum243~~{BRANCH cont243}
A33: [W13] '!
B33: '{IF @ISERR(err3243)<>1#AND#@UPPER(match243)="N"#AND#@UPPER(dummy243)=
     @UPPER(dummy1243)}{LET dum243,dummy4243}~{PANELON} {LET lfind243,
     err3243+1}~/Cdum243~~{BRANCH cont243}
A34: [W13] '!
B34: '{IF flag243=1}{EDIT}{HOME}{DEL}{DEL}~{BRANCH cont1243}
A35: [W13] '!
B35: '{EDIT}{HOME}{DEL}~
A36: [W13] 'cont1243
B36: '{DOWN}
A37: [W13] '!
A38: [W13] 'lfind243
B38: 9
A39: [W13] 'dummy243
B39: '666
A40: [W13] 'dummy1243
B40: '888
A41: [W13] 'dummy2243
B41: @CELLPOINTER("contents")
A42: [W13] 'dummy3243
B42: @LEFT(B41,@FIND(B40,B41,B38))&$B$39&@RIGHT(B41,@LENGTH(B41)-@FIND(B40,
     B41,B38)-@LENGTH(B40))
A43: [W13] 'dummy4243
B43: @LEFT(B41,@FIND(@UPPER(B40),@UPPER(B41),B38))&$B$39&@RIGHT(B41,@LENGTH
     (B41)-@FIND(@UPPER(B40),@UPPER(B41),B38)-@LENGTH(B40))
A44: [W13] 'flag243
B44: 1
A45: [W13] 'match243
B45: 'n
A46: [W13] 'match1243
B46: 'a
A47: [W13] 'err2243
B47: @FIND(B40,B41,B38)
A48: [W13] 'err3243
B48: @FIND(@UPPER(B40),@UPPER(B41),B38)
A49: [W13] 'findd243
B49: @FIND(B40,B41,B38)
A50: [W13] 'findd1243
B50: 54
A51: [W13] 'key243
B51: 'R
A52: [W13] '!
A53: [W13] 'search2243
B53: '{WINDOWSOFF}{PANELOFF}{RECALC dummy2243}~{RECALC dummy3243}~
     {RECALC dummy4243}~{RECALC err2243}~{RECALC err3243}~
```

```
A54: [W13] '!
B54: '{LET flag243,0}~{LET lfind243,0}~{LET previous243,@CELLPOINTER
     ("address")}~
A55: [W13] '!
B55: '{IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~{RECALC dummy2243}~
     {RECALC dummy3243}~{RECALC dummy4243}~{LET flag243,1}~
A56: [W13] 'loop243
B56: '{WINDOWSOFF}{IF @UPPER(match243)="Y"#OR#flag243=1}{RECALC err2243}~
     {RECALC findd243}~{continue1243}
A57: [W13] '!
B57: '{WINDOWSOFF}{IF @UPPER(match243)="N"}{RECALC err3243}~{continue2243}
A58: [W13] '!
B58: '{PANELON}{RECALC backk1243}~{back1243}{DOWN}
A59: [W13] '!
A60: [W13] 'continue2243
B60: '{IF @ISERR(err3243)<>1#OR#err3243=0}{RECALC dummy2243}~/RVdummy2243~
     dum243~{PANELON}{GOTO}dum243~{WINDOWSON}{RECALC err3243}{EDIT}{HOME}
     {RIGHT err3243+1}{BEEP}{GET key243}~{BRANCH cvv243}
A61: [W13] '!
A62: [W13] 'continue1243
B62: '{IF @ISERR(err2243)<>1#OR#err2243=0}{RECALC dummy2243}~/RVdummy2243~
     dum243~{PANELON}{GOTO}dum243~{WINDOWSON}{RECALC findd243}{EDIT}{HOME}
     {RIGHT findd243+1}{BEEP}{GET key243}~{BRANCH cvv243}
A63: [W13] '!
A64: [W13] 'cvv243
B64: '{WINDOWSOFF}{PANELOFF}{RECALC err2243}~{RECALC err3243}~{IF @UPPER
     (key243)="R"#AND#@UPPER(match243)="Y"}/RVerr2243~findd1243~
     {BRANCH loop1243}
A65: [W13] '!
B65: '{WINDOWSOFF}{PANELOFF}{IF @UPPER(key243)="R"#AND#@UPPER(match243)="N"}
     /RVerr3243~findd1243~{BRANCH loop1243}
A66: [W13] '!
B66: '{IF @UPPER(match243)="Y"}{WINDOWSOFF}{PANELOFF}/RVerr2243~findd1243~
A67: [W13] '!
B67: '{IF @UPPER(match243)="N"}{WINDOWSOFF}{PANELOFF}/RVerr3243~findd1243~
A68: [W13] 'loop1243
B68: '{IF @UPPER(key243)="R"}{LET dum243,@LEFT(dum243,findd1243)&dummy243&
     @RIGHT(dum243,@LENGTH(dum243)-findd1243-@LENGTH(dummy1243))}~{PANELON}
     {LET lfind243,findd1243+1}~{loop243}
A69: [W13] '!
B69: '{IF @UPPER(key243)<>"R"#AND#@UPPER(key243)<>"Q"}{LET lfind243
     ,findd1243+1}~{loop243}
A70: [W13] '!
B70: '{IF @UPPER(key243)="Q"}{RECALC backk1243}~{back1243}{QUIT}
A71: [W13] '!
B71: '{RECALC backk243}~{back243}{DOWN}
A72: [W13] '!
A73: [W13] '!
A74: [W13] 'back243
B74: '{WINDOWSOFF}{PANELOFF}/Cdum243~
A75: [W13] 'backk243
B75: +B77
A76: [W13] '!
B76: '~{GOTO}
A77: [W13] 'previous243
B77: '$A$6
A78: [W13] '!
B78: '~
A79: [W13] '!
B79: '{IF flag243=1}{EDIT}{HOME}{DEL}{DEL}~
A80: [W13] '!
A81: [W13] 'back1243
B81: '{WINDOWSOFF}{PANELOFF}{GOTO}
A82: [W13] 'backk1243
B82: +B77
A83: [W13] '!
B83: '~
A84: [W13] '!
B84: '{IF flag243=1}{EDIT}{HOME}{DEL}{DEL}~
A85: [W13] '!
A86: [W13] '!
```

```
A87: [W13] '!
A88: [W13] '!
A89: [W13] '!
A90: [W13] '!
A91: [W13] 'dum243
B91: ''@SIN(666)
A92: [W13] '!
A93: [W13] '!
A94: [W13] '!
A95: [W13] '!
A96: [W13] '!
D96: 'Choose one option when you here the beep
A97: [W13] '!
A98: [W13] '!
A99: [W13] '!
E99: U 'Press [R] to Replace
A100: [W13] '!
A101: [W13] '!
E101: U 'Press [S] to Skip
A102: [W13] '!
A103: [W13] '!
E103: U 'Press [Q] to Quit
A104: [W13] '!
A105: [W13] '!
E105: U 'Ctrl Break
```

[2] Control Negative Numbers Appearance

```
     A                                   B                                   C
 1 *---A macro to control the negative values display
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 NEGATIVE        {MENUBRANCH menu601}
12 !
13 menu601         Parentheses                         Sign
14 !               Display parentheses around number   Display minus before
15 !               /WGDOIN~QSQ                         /WGDOIN{RIGHT}~QSQ
```

Lotus allows you to define the negative numbers appearance in the worksheet. One option is to show the minus (-) character before the number. The second option is to surround the number by parentheses. This macro saves more than ten key strokes every time you change the negative numbers appearance. The macro starts with the `{MENUBRANCH menu601}` macro command that activates the [menu601] custom menu. The custom menu has two options. The first is the [**P**arentheses] menu option. When you choose this menu option, either press the "P" character or the "p" character or press the ENTER key, the macro issues the `/WGDOIN~QSQ` macro keys that display negative number as surrounded by parentheses. If you choose the second menu option, the macro issues `/WGDOIN{RIGHT}~QSQ` that display negative numbers with the minus (-) character before the number.

[7] Annotate Cell Content

```
     A                    B              C              D              E              F
 1 *---A macro to ANNOTATE cells containing formulas, In release 2.0-2.4
 2     it uses the @IF(1,[cell contents],"Note: [note]") formula that can
 3     annotate both labels and values, limited to 240 characters in
       release
 4     2.0-2.4 or 512 in release 3.0 and up
 5 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 6     range names in this column (starts with the \Z macro name)
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                    IT WILL WORK IN RELEASE 2.0 AND UP
10 \Z              {BREAKON}
11 ANNOTATE        Highlight the cell and press RETURN to annotate or ESC
                   to quit{GET key520}{ESC}
12 !               {LET rel520,@INFO("release")}~
13 !               {IF @LEFT(rel520,1)<>"@"#AND#(key520="{PGDN}"#OR#key520
                   ="{PGUP}"#OR#key520="{DOWN}"#OR#key520="{UP}"#OR#key520
                   ="{LEFT}"#OR#key520="{RIGHT}"#OR#key520="{HOME}"#OR#
                   key520="{NS}"#OR#key520="{PS}")}{key520}~
                   {BRANCH annotate}
14 !               {IF @LEFT(rel520,1)<>"@"#AND#key520="~"}
                   {BRANCH annot3520}
15 !               {IF key520="{PGDN}"#OR#key520="{PGUP}"#OR#key520="
                   {DOWN}"#OR#key520="{UP}"#OR#key520="{LEFT}"#OR#key520=
                   "{RIGHT}"#OR#key520="{HOME}"}{key520}~{BRANCH annotate}
16 !               {IF key520="{ESC}"}{BRANCH rel520}
17 !               {IF key520="~"#AND#@CELLPOINTER("type")="v"}
                   {BRANCH annotv520}
18 !               {IF key520="~"#AND#@CELLPOINTER("type")="l"}
                   {BRANCH annotl520}
19 !               {BRANCH annotate}
20 !
21 annot3520       {GETLABEL "Enter note: ",note520}~
22 !               {EDIT};Note: {note520}~{BRANCH annotate}
23 !               {BRANCH annotate}
24 !
25 annotv520       {GETLABEL "Enter note: ",note520}~
26 !               {EDIT}{HOME}@IF(1,{END},"Note: {note520}")~
27 !               {BRANCH annotate}
28 !
29 annotl520       {GETLABEL "Enter note: ",note520}~
30 !               {EDIT}{HOME}{DEL}@IF(1,"{END}","Note: {note520}")~
31 !               {BRANCH annotate}
32 !
33 note520         Sales of 1992
34 !
35 key520          {ESC}
36 !
37 ret520
38 !
39 rel520          3.00.00
```

Lotus 1-2-3, 2.0-2.4 (2-D releases), does not offer a way to annotate a cell's content. The 3-D releases (3.0/3.1/3.1+ and 123w) of Lotus 1-2-3 include the option to insert a note in the cell. To insert a note, you must type a semicolon between the cell content and the note. For example, if the cell contain the `123456.56` value and the note is "`Sales for 1992`", you should type it as `12345.56;Sales for 1992`. The note is not displayed in the cell, to see the note, you must press the F2 key to get into the EDIT mode. This macro also allows you to embed notes in cells. The macro creates a formula in the cell that returns the same result but includes the note inside the formula.

```
     A              B              C              D              E
11 ANNOTATE        Highlight the cell and press RETURN to annotate or ESC
```

```
                to quit{GET key520}{ESC}
12 !            {LET rel520,@INFO("release")}~
```

The macro starts with "`Highlight the cell and press RETURN to annotate or ESC to quit`". Because Lotus does not find any valid command in the text, it types the text into the panel and then, to keep it there, the macro issues the `{GET key520}` command that halts the macro execution and waits until you press a key. Then it issues `{ESC}` to clear the panel before Lotus writes it into the current cell. Next it issues `{LET rel520,@INFO("release")}~` to store the result of the `@INFO("release")` 3-D function in cell [rel520]. Later Lotus uses this result to determine which Lotus release you are using.

```
     A          B                C          D          E
13 !            {IF @LEFT(rel520,1)<>"@"#AND#(key520="{PGDN}"#OR#key520
                ="{PGUP}"#OR#key520="{DOWN}"#OR#key520="{UP}"#OR#key520
                ="{LEFT}"#OR#key520="{RIGHT}"#OR#key520="{HOME}"#OR#
                key520="{NS}"#OR#key520="{PS}")}{key520}~
                {BRANCH annotate}
14 !            {IF @LEFT(rel520,1)<>"@"#AND#key520="~"}
                {BRANCH annot3520}
15 !            {IF key520="{PGDN}"#OR#key520="{PGUP}"#OR#key520="
                {DOWN}"#OR#key520="{UP}"#OR#key520="{LEFT}"#OR#key520=
                "{RIGHT}"#OR#key520="{HOME}"}{key520}~{BRANCH annotate}
16 !            {IF key520="{ESC}"}{BRANCH rel520}
17 !            {IF key520="~"#AND#@CELLPOINTER("type")="v"}
                {BRANCH annotv520}
18 !            {IF key520="~"#AND#@CELLPOINTER("type")="l"}
                {BRANCH annotl520}
19 !            {BRANCH annotate}
```

Now the macro starts with a series of `{IF}` commands to check which key you pressed and what type of Lotus you are using, a 2-D or 3-D release. The first in the B13 cell checks if you are using a 3-D release and if you pressed one of the following direction keys (`PGDN`, `PGUP`, `DOWN`, `UP`, `LEFT`, `RIGHT`, `HOME`, `NEXTSHEET` or `PREVIOUSSHEET`). If so, the macro issues the `{key520}` routine command that activates the [key520] routine. However because the [key520] routine contains only one cell with the key that you pressed the `{key520}` routine command executes the key that you pressed. You may notice some delay between pressing the key and Lotus' executing it.

This is how the macro monitors and controls your response to the `{GET}` macro command. Next the macro issues `{BRANCH annotate}` and displays the message text again. The second `{IF}` command is the `{IF @LEFT(rel520,1)<>"@"#AND#key520="~"}` which checks if you are using a 3-D release and if you pressed ENTER, if so, the macro issues `{BRANCH annot3520}` which starts the [annot3520] routine.

```
     A          B                C          D          E
21 annot3520    {GETLABEL "Enter note: ",note520}~
22 !            {EDIT};Note: {note520}~{BRANCH annotate}
23 !            {BRANCH annotate}
```

The [annot3520] routine starts with `{GETLABEL "Enter note: ",note520}~` to display the "`Enter note:` " prompt in the panel. When you type the note and press ENTER, Lotus stores the note in cell [note520]. After issuing `{EDIT}` to enter the EDIT mode the macro types the "`;Note:` " text into the panel. To clarify the process, let's assume that the current cell contains the 123456.56 number and the note in cell [note520] is "`Sales for 1992`". When the macro issues the `{EDIT}` command the panel displays:

```
123456.56
```

When the macro issues the ";Note: " text the panel shows:

```
123456.56;Note:
```

Now the macro issues the {note520} routine command while Lotus is still in the EDIT mode. Because the [note520] routine contains only one cell with the "Sales for 1992" note string, the {note520} command injects the note's text into the panel which now displays:

```
123456.56;Note: Sales for 1992
```

Next the macro issues the tilde "~" macro command, which is the same as using the ENTER key from the keyboard, to write the content of the panel into the current cell. It was easy to insert a note because, when you use a 3-D release, all you need to do is type the semicolon before the note's text. The third {IF} command in the B15 cell is the same as the first one except that it is for a 2-D release, therefore the {IF @LEFT(rel520,1)<>"@"} command is omitted. The fourth {IF} command is the {IF key520="{ESC}"} command that checks if you pressed the ESC key. If so, the macro issues {BRANCH ret520} that routes macro control to the [ret520] empty routine and quits. The fifth {IF} command in the B17 cell checks if you pressed ENTER, which is represented in the macro language as the tilde "~" and checks if the cell contains a value. If so, the macro issues {BRANCH annotv520} that routes macro execution to the [annotv520] routine.

	A	B	C	D	E
25	annotv520	{GETLABEL "Enter note: ",note520}~			
26	!	{EDIT}{HOME}@IF(1,{END},"Note: {note520}")~			
27	!	{BRANCH annotate}			

The [annotv520] routine starts with {GETLABEL "Enter note: ",note520}~ to display the "Enter note: " prompt in the panel. When you type the note and press ENTER, Lotus stores the note in [note520]. This routine is similar to the [annot3520] but a little bit more complicated, therefore we will use the same example to clarify the code. Let's again assume that the current cell contains the 123456.56 number and the note is "Sales for 1992". The macro issues {EDIT}{HOME} to enter the EDIT mode and to move the cursor to the beginning of the panel and then types the "@IF(1," text into the panel which now displays:

```
@IF(1,123456.56
```

Then the macro issues {END} that moves the cursor to the end of the text in the panel and continues to type the ","Note: " text into the panel which now displays:

```
@IF(1,123456.56,"Note:
```

Next the macro issues the {note520} routine command while Lotus is still in the EDIT mode. Therefore, because the [note520] routine contains only one cell with the "Sales for 1992" note string, the {note520} routine command injects the note's text into the panel which now displays:

```
@IF(1,123456.56,"Note: Sales for 1992
```

Next the macro types the double quotes and the closing parenthesis [")] to make the panel to

display:

```
@IF(1,123456.56,"Note: Sales for 1992")
```

Then the macro issues the tilde "~" and the result of this formula as displayed in the cell is the 123456.56 number. When you press EDIT, the full formula is displayed in the panel including the note. This is called the enveloping technique where we envelop a cell content by a formula.

The last {IF} command in the B18 cell checks if you pressed the ENTER key, which is represented in macro language as the tilde "~", and at the same time the macro checks if the cell contains a label. If so, the macro issues {BRANCH annotl520} that routes the macro execution to the [annotl520] routine.

	A	B	C	D	E
29	annotl520	{GETLABEL "Enter note: ",note520}~			
30	!	{EDIT}{HOME}{DEL}@IF(1,"{END}","Note: {note520}")~			
31	!	{BRANCH annotate}			

The [annotl520] routine starts with {GETLABEL "Enter note: ",note520}~ which display the "Enter note: " prompt in the panel. When you type the note and press ENTER, Lotus stores the note in [note520]. This routine is similar to the [annotv520], therefore let us assume that the current cell contains the "July" string and the note is "The seventh month of the year". The macro issues {EDIT}{HOME} to enter the EDIT mode and move the cursor to the beginning of the panel and then issues {DEL} to delete the apostrophe prefix. Then the macro types the "@IF(1," text into the panel which now displays:

```
@IF(1,"July
```

and issues {END} to move the cursor to the end of the text in the panel and continues to type the "","Note: " text into the panel which now displays:

```
@IF(1,"July","Note:
```

Next the macro issues the {note520} routine command while Lotus is still in the EDIT mode. Because the [note520] routine contains only one cell with the "The seventh month of the year" note string, the {note520} routine command injects the note's text into the panel which now displays:

```
@IF(1,"July","Note: The seventh month of the year
```

Next the macro types the closing double quotes and the closing parenthesis [")] to make the panel to display:

```
@IF(1,"July","Note: The seventh month of the year")
```

Then the macro issues the tilde "~" and the result of this formula as displayed in the cell is the "July" string. When you press EDIT, the full formula is displayed in the panel including the note. This is called the enveloping technique where we envelop a cell content by a formula.

[6] Turn All the Formulas and the Values into Labels

```
     A                B                C                D                E
 1 *---A macro to turn all formulas and/or numbers in a 3-D or 2-D range
 2     into labels, labels and blank cells will not be changed.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                   IT WILL WORK IN RELEASE 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 DO_LABEL        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~{BS}
                   {BS}{?}~{WINDOWSOFF}
                   {GOTO}Which range ?~{LET counterb407,0}~
12 cont407         {LET hereabs407,@CELLPOINTER("address")}~
                   {LET counter407,0}
13 !               {FOR counter407,0,@COLS(Which range ?)-1,1,labels407}
14 !               {LET rel407,@INFO("release")}~{IF @LEFT(rel407,1)<>"@"}
                   {GOTO}{hereabs407}~{LET counterb407,counterb407+1}~
                   {IF counterb407<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs407}~{BRANCH cont407}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter407                7
18 countera407              18
19 labels407       {RIGHT}{LET here407,@CELLPOINTER("address")}~{LEFT}
                   {FOR countera407,0,@ROWS(Which range ?)-1,1,labelsa407}~
                   {IF counter407<@COLS(Which range ?)-1}{GOTO}{here407}~
                   {LET countera407,0}~
20 !
21 here407         $AM$1
22 !
23 labelsa407      {IF @CELLPOINTER("prefix")="\"#OR#@CELLPOINTER("type")
                   ="l"#OR#@CELLPOINTER("type")="b"}{DOWN}{BRANCH ret407}
24 !               {IF @CELLPOINTER("type")="v"}{EDIT}{HOME}'{DOWN}
25 !
26 counterb407               0
27 hereabs407      $AF$1
28 !
29 rel407          @INFO("release")
30 !
31 ret407
```

One of the basics of the electronic spreadsheet is that when a cell containing a formula is copied, the copied formula refers to new cell but follows the rules of the original formula. Often you need to copy some formulas, but do not want the new formulas to change their reference. It is always possible to define absolute cell addresses, but this is not always the best solution. This macro will turn all the cells containing numbers or formulas (values) into label format by adding the apostrophe prefix to every value in the range. When the macro finds a label in the range it skips and moves to the next cell. You can now move the transformed range to any place without any change in the formulas. A complementary macro to reverse the process is the UN_LABEL.WK1 macro featured next.

The macro starts with `{WINDOWSOFF}{PANELOFF}` to freeze screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to label and also assigns the [Which range ?] name to the same range. Next the macro issues `{LET counterb407,0}~` to set the content of [counterb407] to zero, which serves as a counter.

```
     A                B                C                D                E
12 cont407         {LET hereabs407,@CELLPOINTER("address")}~
```

```
                {LET counter407,0}
13 !            {FOR counter407,0,@COLS(Which range ?)-1,1,labels407}
14 !            {LET rel407,@INFO("release")}~{IF @LEFT(rel407,1)<>"@"}
                {GOTO}{hereabs407}~{LET counterb407,counterb407+1}~
                {IF counterb407<@SHEETS(Which range ?)}{NS}{GOTO}
                {hereabs407}~{BRANCH cont407}
15 !            {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro is finished, the macro issues `{LET hereabs407,@CELLPOINTER("address")}~` to store the current cell pointer address in cell [hereabs407], and then `{LET counter407,0}` to set the content of [counter407] to zero, which also serves as a counter.

The macro issues the `{FOR counter407,0,@COLS(Which range ?)-1,1, labels407}` loop command that activates the [labels407] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels407] routine.

```
     A          B          C          D          E
19 labels407    {RIGHT}{LET here407,@CELLPOINTER("address")}~{LEFT}
                {FOR countera407,0,@ROWS(Which range ?)-1,1,labelsa407}~
                {IF counter407<@COLS(Which range ?)-1}{GOTO}{here407}~
                {LET countera407,0}~
```

This routine uses `{RIGHT}{LET here407,@CELLPOINTER("address")}~{LEFT}` to record the address of the first cell of the next column in cell [here407]. When the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here407]. The `{FOR countera407,0,@ROWS(Which range ?)-1,1,labelsa407}~` commands activate the [labelsa407] routine as many times as the number of rows in the [Which range ?] range.

```
     A          B          C          D          E
23 labelsa407   {IF @CELLPOINTER("prefix")="\"#OR#@CELLPOINTER("type")
                ="l"#OR#@CELLPOINTER("type")="b"}{DOWN}{BRANCH ret407}
24 !            {IF @CELLPOINTER("type")="v"}{EDIT}{HOME}'{DOWN}
```

The [labelsa407] routine issues

```
{IF @CELLPOINTER("prefix")="\"#OR#@CELLPOINTER("type")="l"
    #OR#@CELLPOINTER("type")="b"}
```

to check if the current cell content is a label, the back slash "\" prefix, or blank. If so, the macro issues `{DOWN}`, which moves the cell pointer one cell down and then `{BRANCH ret407}` which routes the macro execution to an empty routine and ends the routine. Next the macro issues `{IF @CELLPOINTER("type")="v"}`, which checks if the cell contains a value. If so, the macro issues `{EDIT}` to enter the EDIT mode, and then `{HOME}` which moves the cursor to the beginning of the panel. Then the macro types the apostrophe "'" and issues `{DOWN}` to move the cell pointer down and, simultaneously writes the panel's content into the current cell. When the [labelsa407] routine ends, the macro returns control to the `{FOR}` loop command in the [labels407] routine to start the process of the next cell in the current column.

```
     A          B          C          D          E
19 labels407    {RIGHT}{LET here407,@CELLPOINTER("address")}~{LEFT}
                {FOR countera407,0,@ROWS(Which range ?)-1,1,labelsa407}~
                {IF counter407<@COLS(Which range ?)-1}{GOTO}{here407}~
                {LET countera407,0}~
```

When the value in [countera407] reaches the number of rows in [Which range ?] minus one, the macro issues {IF counter407<@COLS(Which range ?)-1} to check how many columns were processed. If the value in [counter407] is less than the number of columns in [Which range ?], the macro issues the indirect {GOTO}{here407}~ macro command which moves the cell pointer to the first cell of the next column. Next the macro issues {LET countera407,0}~ to reset the value in [countera407] to zero. When the [labels407] routine is finished the macro returns control back to the [cont407] routine.

	A	B	C	D	E
12	cont407	{LET hereabs407,@CELLPOINTER("address")}~			
		{LET counter407,0}			
13	!	{FOR counter407,0,@COLS(Which range ?)-1,1,labels407}			
14	!	{LET rel407,@INFO("release")}~{IF @LEFT(rel407,1)<>"@"}			
		{GOTO}{hereabs407}~{LET counterb407,counterb407+1}~			
		{IF counterb407<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs407}~{BRANCH cont407}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

When the macro finishes processing the first sheet of [Which range ?] (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues the {LET rel407,@INFO("release")}~ commands to store the result of the @INFO("release") 3-D function in [rel407] and then {IF @LEFT(rel407,1)<>"@"}, which checks the first character of the content of [rel407]. If the character is the "@" character, you are using a 2-D release otherwise you are using a 3-D release, which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the {GOTO} {hereabs407}~ indirect macro command which moves the cell pointer to the origin address of the macro.

The macro continues with {LET counterb407,counterb407+1}~ to increase the value in [counterb407] by one. Now the macro issues {IF counterb407<@SHEETS(Which range ?)} to check if the value in [counterb407] is less than the number of sheets in [Which range ?]. If so, there are more sheets to process, therefore the macro issues {NS} to move the cell pointer to the next sheet. Next the macro issues the indirect {GOTO}{hereabs407}~ macro command which moves the cell pointer to the same address as the upper left cell of the first sheet of [Which range ?], but in the new sheet. Last, the macro issues {BRANCH cont407} to change the cells of the range in the new sheet to labels. When all the sheets are processed, the macro issues {GOTO}Which range ?~ which moves the cell pointer to the upper left cell of the first sheet of [Which range ?] and then issues /RNDWhich range ?~ to delete the temporary [Which range ?] range name.

[6] Turn All the Labels Back into Values and Formulas

```
     A                B                C                D                E
 1 *---A macro to turn all labels in a 3-D or 2-D range to formulas
 2     and/or numbers, values and blank cells will not be changed.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                   IT WILL WORK IN RELEASE 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 UN_LABEL        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb047,0}~
12 cont047         {LET hereabs047,@CELLPOINTER("address")}~
                   {LET counter047,0}
13 !               {FOR counter047,0,@COLS(Which range ?)-1,1,labels047}
14 !               {LET rel047,@INFO("release")}~{IF @LEFT(rel047,1)<>"@"}
                   {GOTO}{hereabs047}~{LET counterb047,counterb047+1}~
                   {IF counterb047<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs047}~{BRANCH cont047}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter047                 2
18 countera047                5
19 labels047       {RIGHT}{LET here047,@CELLPOINTER("address")}~{LEFT}
                   {FOR countera047,0,@ROWS(Which range ?)-1,1,labelsa047}~
                   {IF counter047<@COLS(Which range ?)-1}{GOTO}{here047}~
                   {LET countera047,0}~
20 !
21 here047         $C$1
22 !
23 labelsa047      {IF @CELLPOINTER("type")="l"#AND#@CELLPOINTER("prefix")
                   <>"\"}{EDIT}{HOME}{DEL}{DOWN}{RETURN}
24 !               {DOWN}
25 !
26 counterb047                4
27 hereabs047      $A$1
28 !
29 rel047
```

This macro is a complements to the previous DO_LABEL.WK1 macro. It reverses the process and restores all the previous values back to normal. The main code is essentially the same as the code in the DO_LABEL.WK1 macro except for the [labelsa047] routine which reverses the labels back to values. Therefore we explain only this routine.

```
     A                B                C                D                E
23 labelsa047      {IF @CELLPOINTER("type")="l"#AND#@CELLPOINTER("prefix")
                   <>"\"}{EDIT}{HOME}{DEL}{DOWN}{RETURN}
24 !               {DOWN}
```

The [labelsa047] routine starts with

```
{IF @CELLPOINTER("type")="l"#AND#@CELLPOINTER("prefix ")<>"\"}
```

which checks if the current cell contains a label, but makes sure that it is not preceded by the back slash "\" key if you used the back slash "\" as a prefix. If so, the macro issues the {EDIT} command to enter the EDIT mode and then issues {HOME} to move the cursor to the beginning of the panel. Next the macro issues {DEL} to delete the apostrophe "'" prefix and to change the label in the panel into a value.

The {DOWN} command moves the cell pointer to the next cell to process and simultaneously writes the content of the panel into the current cell. Next the macro issues {RETURN} to pass the macro execution control back to the {FOR} loop in the [labels047] routine. If the condition was not true and the cell is blank or the cell prefix is the back slash "\" character, the macro issues {DOWN} to move the cell pointer to the next cell to process in the current column.

[4] Display a Flashing Message in the Panel

```
     A                B                C                D                E
 1 *---A macro to present a flashing message in the panel area
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Type your message right to the cell MESSAGE
 6 !
 7                  THIS MACRO WORKS WITH LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 MESSAGE2         {BREAKON}
12 !                {RECALC loop660}
12 loop660          {WINDOWSOFF}{PANELOFF}{INDICATE "                 This
                    is our message .... Press any key to Quit           "}
                    {PANELON}
14 !                {EDIT}{WAIT @NOW+@TIME(0,0,1)}{INDICATE}{WAIT @NOW+
                    @TIME(0,0,1)}{ESC}~{LOOK key660}{ESC}
15 !                {IF key660=""}{BRANCH loop660}
16 !                {GET key660}{ESC}~
17 !
18 !
19 message660       This is our message .... Press any key to Quit
20 !
21 key660           {ESC}
```

This macro demonstrates how to display a flashing message in the panel area. You have to enter the message in the B19 cell named [message660]. When you start the macro the message appears flashing in the first row of the panel. You are reminded that the code in B13, [loop660], is the result of the dynamic string formula:

```
13 loop660          +"{WINDOWSOFF}{PANELOFF}{INDICATE """&@REPEAT(" ",(80-
                    @LENGTH(B18))/2)&message660&@REPEAT(" ",(80-@LENGTH
                    (message660))/2)&"""}{PANELON}"
```

If you intend to key this macro into Lotus 1-2-3, you must key this formula, NOT the code as it appears in the main listing. The macro starts with the `{RECALC loop660}` macro command which updates the dynamic formula in cell [loop660] to reflect the changes made in the message. Then the macro issues `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel activities, and then

```
{INDICATE "                 This is our message .... Press
    any key to Quit                   "}
```

This command displays

```
This is our message .... Press any key to Quit
```

centered and highlighted at the first row of the panel. The macro uses the new property of the `{INDICATE "string"}` command which has been incorporated in Lotus 1-2-3 starting with release 2.2. This command is the result of the formula:

```
13 loop660          +"{WINDOWSOFF}{PANELOFF}{INDICATE """&@REPEAT(" ",(80-
                    @LENGTH(B18))/2)&message660&@REPEAT(" ",(80-@LENGTH
                    (message660))/2)&"""}{PANELON}"
```

We can see that this formula uses the `@LENGTH` function to calculate the length of the message

and then it uses the `@REPEAT` function to add leading and trailing spaces to the message to create an 80 character long string with the message centered. It is not enough to add only the leading spaces because the string is pushed to the right and is preceded by a Lotus prompt, therefore the string has to be at least 80 characters long to capture the whole row. Now the macro issues `{PANELON}` to resume the panel activity.

	A	B	C	D	E
14	!	{EDIT}{WAIT @NOW+@TIME(0,0,1)}{INDICATE}{WAIT @NOW+			
		@TIME(0,0,1)}{ESC}~{LOOK key660}{ESC}			
15	!	{IF key660=""}{BRANCH loop660}			
16	!	{GET key660}{ESC}~			

After issuing `{EDIT}` to enter to the EDIT mode and `{WAIT @NOW+@TIME(0,0,1)}`, which keeps the message in the panel for approximately one second, the macro issues `{INDICATE}` without a string which clears the message, and again issues the `{WAIT @NOW+@TIME(0,0,1)}` which keeps the panel clear for approximately one second. The macro continues with `{ESC}` to exit the EDIT mode and issues `{LOOK key660}` which waits until you press a key.

If you press a key, Lotus stores the key in [key660]. Next the macro issues `{IF key660=""}` to check if [key660] is blank. If so, `{BRANCH loop660}` is issued, which loops back and displays the message again. If the condition is false, the macro issues `{GET key660}{ESC}~` to clear the panel from the key you pressed and quits.

[4] Insert a Phrase into a Cell

```
     A                B                C                D                E
 1 *---A macro to ENTER a phrase into a cell
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 PHRASE          Current phrase is: "{phrase1131}"   [A]ccept [C]hange
                   [Q]uit {GET key1131}
12 !               {IF @UPPER(key1131)="A"}{ESC 6}{phrase1131}~
13 !               {IF @UPPER(key1131)="C"}{ESC 6}{phrase1131}{EDIT}{?}~
                   {LET phrase1131,@CELLPOINTER("contents")}~
14 !               {IF @UPPER(key1131)="Q"}{ESC 6}~
15 !               {ESC 6}~
16 !
17 key1131         C
18 !
19 phrase1131      Year to date sales
```

Sometimes we need to insert a phrase of sentence in many cells in the worksheet. Instead of typing it repeatedly, we can let a macro to do the job for us. This macro takes a phrase and writes it into the current cell every time it is activated. It also allow you to change the phrase just before the macro writes it into the current cell. The phrase is located in B19 of the macro code. Before you start the macro, you have to enter the new phrase. This macro comes with the dummy phrase "`Year to date sales`". To clarify the code, let's look at what happens in the panel. When the macro starts, it directly writes the `Current phrase is: "` text into the panel which displays:

```
Current phrase is: "
```

and continues with the `{phrase1131}` routine command which injects the phrase text into the panel following the previous text, therefore the panel displays:

```
Current phrase is: "Year to date sales
```

Next the macro continues to type to the panel the `"  [A]ccept [C]hange [Q]uit` text. The panel now displays

```
Current phrase is: "Year to date sales"   [A]ccept [C]hange [Q]uit
```

To hold this prompt in the panel, the macro issues the `{GET key1131}` command, which halts the macro execution and waits until you press a key. When you press a key, Lotus stores the key in the B17 cell named [key1131]

```
     A                B                C                D                E
12 !               {IF @UPPER(key1131)="A"}{ESC 6}{phrase1131}~
13 !               {IF @UPPER(key1131)="C"}{ESC 6}{phrase1131}{EDIT}{?}~
                   {LET phrase1131,@CELLPOINTER("contents")}~
14 !               {IF @UPPER(key1131)="Q"}{ESC 6}~
15 !               {ESC 6}~
```

Now the macro issues a series of `{IF}` condition commands to check which key you just

pressed. The first is `{IF @UPPER(key1131)="A"}`, which checks if you pressed the "`a`" or the "`A`" keys. If so, the macro issues `{ESC 6}` to clear the panel from the message before Lotus writes it into the current cell, and then issues `{phrase1131}~`, to write the phrase into the panel and then to the current cell.

The second is `{IF @UPPER(key1131)="C"}`, which checks if you pressed the "`c`" or the "`C`" keys. If so, the macro issues `{ESC 6}` to clear the panel from the message before Lotus writes it into the current cell. Then it issues `{phrase1131}~`, which write the phrase into the panel and continues with `{EDIT}{?}`, which enter the EDIT mode and allow you to make changes to the phrase before Lotus writes it to the current cell. When you are finished and press ENTER, the macro issues `{LET phrase1131,@CELLPOINTER("contents")}~` to copy the new phrase to [phrase1131]. Until you change it again, the new phrase is the current phrase.

The third is `{IF @UPPER(key1131)="Q"}`, which checks if you pressed the "`q`" or the "`Q`" keys. If so, the macro issues `{ESC 6}` to clear the panel from the message before Lotus writes it into the current cell and quits. If you press any other key, the macro issues `{ESC 6}` to clear the panel from the message before Lotus writes it into the current cell and quits.

[6] Pad Cells Content with Trailing Periods

```
     A                    B                C                D                E
 1 *---A macro to PAD a cell content with trailing periods [.] to improve
 2     appearance and adjust descriptive cells to column width
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                    IT WILL WORK IN RELEASE 2.0 AND UP
 9 !
10 \Z               {BREAKON}
11 PADDING          {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                    Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                    {LET counterb533,0}~
12 cont533          {LET hereabs533,@CELLPOINTER("address")}~
                    {LET counter1533,0}
13 !                {FOR counter1533,0,@COLS(Which range ?)-1,1,labels1533}
14 !                {LET rel533,@INFO("release")}~{IF @LEFT(rel533,1)<>"@"}
                    {GOTO}{hereabs533}~{LET counterb533,counterb533+1}~
                    {IF counterb533<@SHEETS(Which range ?)}{NS}{GOTO}
                    {hereabs533}~{BRANCH cont533}
15 !                {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1533                  1
18 counter1a533                 0
19 labels1533       {FOR counter1a533,0,@ROWS(Which range ?)-1,1,
                    labels1a533}~{RIGHT}{UP @ROWS(Which range ?)}
                    {LET counter1a533,0}~
20 !
21 labels1a533      {IF @CELLPOINTER("type")="l"#AND#@CELLPOINTER("prefix")
                    <>"\"#AND#@CELLPOINTER("width")>@LENGTH(@CELLPOINTER
                    ("contents"))}{EDIT}{HOME}{DEL}+"{END}"&@REPEAT(".",
                    @CELLPOINTER("width")-@LENGTH(@CELLPOINTER("contents")))
                    {CALC}
22 !                {DOWN}
23 !
24 !
25 counterb533                   0
26 hereabs533       $X$1
27 !
28 rel533           @INFO("release")
```

This macro allows you to pad a label with trailing periods to improve appearance and adjust descriptive cells to column width. For example, if we have the following table:

```
     A                    B                C                D                E
 1 First name           Israel
 2 Last name            Kehaty
 3 Age                  42
 4 Height               168
 5 Occupation           Engineer
```

When you apply this macro to the A1..A5 range the table will look like:

```
     A                    B                C                D                E
 1 First name......... Israel
 2 Last name.......... Kehaty
 3 Age................ 42
 4 Height............. 168
 5 Occupation......... Engineer
```

The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands, which freeze the

screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to pad and simultaneously assigns the [Which range ?] name to the same range and uses it as a prompt. Next the macro issues {LET counterb533,0}~ to set the content of [counterb533] to zero, which serves as a counter.

```
     A              B              C              D              E
12 cont533         {LET hereabs533,@CELLPOINTER("address")}~
                   {LET counter1533,0}
13 !               {FOR counter1533,0,@COLS(Which range ?)-1,1,labels1533}
14 !               {LET rel533,@INFO("release")}~{IF @LEFT(rel533,1)<>"@"}
                   {GOTO}{hereabs533}~{LET counterb533,counterb533+1}~
                   {IF counterb533<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs533}~{BRANCH cont533}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the place of origin when the macro is finished, the macro issues {LET hereabs533,@CELLPOINTER("address")}~, which store the current cell pointer address in [hereabs533], and then issues {LET counter1533,0} to set the content of [counter1533] to zero, which also serves as a counter. The macro issues the {FOR counter1533,0, @COLS(Which range ?)-1,1,labels1533} loop command, which activates the [labels1533] routine as many times as the number of columns in [Which range ?]. Before we continue this code, let's look at the [labels1533] routine.

```
     A              B              C              D              E
19 labels1533      {FOR counter1a533,0,@ROWS(Which range ?)-1,1,labels1a533}~
                   {RIGHT}{UP @ROWS(Which range ?)}{LET counter1a533,0}~
```

The {FOR counter1a533,0,@ROWS(Which range ?)-1,1,labels1a533}~ command activates the [labels1a533] routine as many times as the number of rows in [Which range ?].

```
     A              B              C              D              E
21 labels1a533     {IF @CELLPOINTER("type")="l"#AND#@CELLPOINTER("prefix")
                   <>"\"#AND#@CELLPOINTER("width")>@LENGTH(@CELLPOINTER
                   ("contents"))}{EDIT}{HOME}{DEL}+"{END}"&@REPEAT(".",
                   @CELLPOINTER("width")-@LENGTH(@CELLPOINTER("contents")))
                   {CALC}
22 !               {DOWN}
```

This routine adds the periods to the labels in [Which range ?] and then manipulates the text in the panel and envelops it with a formula which adds the trailing periods to the labels. The macro checks if the current cell has the following three conditions: (1) contains a label, (2) the prefix is not the back slash "\" key and (3) the column width is greater than the label length. If all conditions are true, the macro issues {EDIT}{HOME}{DEL}, which enter Lotus to the EDIT mode and then move the cursor to the beginning of the text in the panel and then delete the apostrophe "'". To make it clearer, let's assume that the current cell contains the "First name" label and that the column width is 20. When the macro issues {EDIT}{HOME}{DEL} the panel displays:

```
First name
```

The macro writes +" directly to the panel and then issues {END} which moves the cursor to the end of the text in the panel. Therefore the panel displays:

```
+"First name_
```

The cursor position is marked by the underscore at the end to the text. Now the macro continues

to type

```
"&@REPEAT(".",@CELLPOINTER("width")-@LENGTH(@CELLPOINTER
    ("content s")))
```

into the panel, which now displays the:

```
+"First name"&@REPEAT(".",@CELLPOINTER("width")-@LENGTH(
    @CELLPOINTER("content s")))
```

formula in the panel. We can see that the formula makes use of the @REPEAT function to add the trailing periods to the "First name" label. To change the formula into its result, the macro issues {CALC} while the formula is still in the panel, therefore the panel displays:

```
First name.........
```

which is the correct result. Now the macro issues {DOWN} which moves the cell pointer to the next cell to process, quits the [labels1a533] routine, and returns control to the {FOR} loop in the [labels1533] routine.

	A	B	C	D	E
19	labels1533	{FOR counter1a533,0,@ROWS(Which range ?)-1,1,labels1a533}~			
		{RIGHT}{UP @ROWS(Which range ?)}{LET counter1a533,0}~			

When the macro finishes processing all the cells in the current column, it issues {RIGHT}{UP @ROWS(Which range ?)} to move to the first cell of the next column in [Which range ?]. Next the macro issues {LET counter1a533,0}~ to reset the counter in [counter1a533] to zero and returns control to the {FOR} loop in the [cont533] routine.

	A	B	C	D	E
12	cont533	{LET hereabs533,@CELLPOINTER("address")}~			
		{LET counter1533,0}			
13	!	{FOR counter1533,0,@COLS(Which range ?)-1,1,labels1533}			
14	!	{LET rel533,@INFO("release")}~{IF @LEFT(rel533,1)<>"@"}			
		{GOTO}{hereabs533}~{LET counterb533,counterb533+1}~			
		{IF counterb533<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs533}~{BRANCH cont533}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

When the macro finishes processing the first sheet of the [Which range ?] range, it issues {LET rel533,@INFO("release")}~ which store the result of the @INFO("release") function in [rel533]. Then the macro issues {IF @LEFT(rel533,1)<>"@"} to check if you are using a 2-D or a 3-D Lotus release. If the "@" is the first character of the content of [rel533], then you are using a 2-D Lotus release, otherwise you are using a Lotus 3-D release. if you are using a 3-D release, the macro issues the {GOTO}{hereabs533}~ indirect macro command to move to the first cell in the current sheet range, and then issues {LET counterb533,counterb533+1}~ to increase the counter in cell [counterb533] by one.

Next the macro uses {IF counterb533<@SHEETS(Which range ?)} to compare the counter value in [counterb533] to the number of sheets in [Which range ?]. If the counter value is still less than the number of sheets in [Which range ?], the macro has to process more sheets before it can quit. Therefore the macro issues {NS}{GOTO}{hereabs533}~ to move the cell pointer to the top left cell to process in the new sheet and issues {BRANCH cont533} to loop

back to the beginning of the [cont533] routine.

When the counter in [counterb533] is equal to the number of sheets in [Which range ?], the macro issues `{GOTO}Which range ?~` to place the cell pointer back on the point of origin just before the macro started, and then uses `/RNDWhich range ?~` to delete the [Which range ?] range name to leave a clean worksheet.

[2] Search and Replace for Release 2.2 and Up

```
     A                    B                C                D                E
 1 *---A macro to replace string occurrences by new string
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6                    THIS MACRO WORKS IN LOTUS 2.2 AND UP
 7 !
 8 !
 9 !
10 \Z               {BREAKON}
11 REPLACE          /RS{?}~
12 LOOP231          {?}~BR{?}~
```

This macro is a search and replace macro which saves you five keystrokes every time you conduct a search and replace. The first `{?}` macro command is issued when Lotus prompts you to insert the range to search. The second `{?}` is issued when Lotus prompts you for the string to search for, and the last `{?}` is issued when Lotus prompts you for the replacement string. The macro assume that the search will be on both labels and formulas and assumes the replace of all occurrences of the old string.

[2] Disable the Undo Feature

```
     A                    B                C                D                E
 1 *---A macro to disable the UNDO feature
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                    THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 UNDO-DIS         /WGDOUDSQQ{ESC}
```

The UNDO option in Lotus is an important feature; however it takes memory. When this option is unnecessary it should be disabled. This macro saves you eight key strokes when you need to disable the UNDO option. The macro uses the exact key sequences as you do from the keyboard; **/ W**orksheet **G**lobal **D**efault **O**ther **U**ndo **D**isable **S**tatus **Q**uit **Q**uit, and then the ESC key.

Notice that when the macro issues the **S**tatus menu option, it halts and displays the status screen. When you press a key, the macro issues **Q**uit **Q**uit and then the ESC key to return to the READY mode. Normally one **Q**uit is enough. However in Lotus 1-2-3 for Windows you need to press the **Q**uit twice. If you press a second "Q" in a 2-D release this "Q" appears in the panel, therefore the `{ESC}` clears the "Q" character from the panel.

[2] Enable the Undo Feature

```
     A                B              C              D              E
 1 *---A macro to enable the UNDO feature
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                    THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 UNDO-ENB         /WGDOUESQQ{ESC}
```

The UNDO option in Lotus is an important feature; however it takes memory. When this option is necessary it should be enabled. This macro saves you eight key strokes when you need to enable the UNDO option. The macro uses the exact key sequences as you do from the keyboard; / **W**orksheet **G**lobal **D**efault **O**ther **U**ndo **E**nable **S**tatus **Q**uit **Q**uit, and then the ESC key.

Notice that when the macro issues the **S**tatus menu option, it halts and displays the status screen. When you press a key, the macro issues **Q**uit **Q**uit and then the ESC key to return to the READY mode. Normally one **Q**uit is enough. However in Lotus 123 for Windows you need to press the **Q**uit twice. If you press a second "Q" in a 2-D release this "Q" appears in the panel, therefore the `{ESC}` clears the "Q" character from the panel.

[3] Toggle the Undo Feature

	A	B	C	D	E
1	*---A macro to TOGGLE UNDO i.e. between UNDO ENABLE and DISABLE				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7		THIS MACRO WORKS IN LOTUS 2.2 AND UP			
8	!				
9	!				
10	\Z	{BREAKON}			
11	UNDOTOGL	{IF status606="UNDODIS"}{LET status606,"UNDOENB"}			
		{BRANCH set606}			
12	!	{IF status606="UNDOENB"}{LET status606,"UNDODIS"}			
		{BRANCH clear606}			
13	!				
14	set606	/WGDOUESQQ{ESC}			
15	!				
16	clear606	/WGDOUDSQQ{ESC}			
17	!				
18	status606	UNDODIS			

This macro is a toggle macro which combines the UNDO-ENB.WK1 and the UNDO-DIS.WK1 macros together. As a default, the first time the macro is activated, it enables the UNDO option. But the macro "remembers" the last operation, therefore next time you use the macro, it disables the UNDO option. Every time you use the macro (without changing the working file) the macro changes the UNDO status.

The macro starts with the `{IF status606="UNDODIS"}` command which checks the content of the B18 cell named [status606]. If the cell contains the "`UNDODIS`" string the macro issues `{LET status606,"UNDOENB"}`, which replaces the "`UNDODIS`" string with the "`UNDOENB`" string in cell [status606]. Next, the macro issues `{BRANCH set606}` which starts the [set606] routine and enables the UNDO option. If the condition was false, the macro issues `{IF status606="UNDOENB"}` which again checks the content of [status606]. If the cell contains the "`UNDOENB`" string, the macro issues `{LET status606,"UNDODIS"}`, which replaces the "`UNDODIS`" string with the "`UNDOENB`" string in [status606]. Next, the macro issues `{BRANCH clear606}`, which starts the [clear606] routine and disables the UNDO option.

[2] Disable the Autoexec Macro from Running

```
      A                  B                C                D                E
 1 *---A macro to disable the AUTOEXEC \0 macros
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                    THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 AUTO-NO          /WGDANSQQ{ESC}
```

The AUTO option in Lotus is an important feature, since it allows you to create automatic running applications. If you assign the [\0] range name to a macro, when you retrieve the worksheet containing it, the macro automatically starts to run. However when you want to edit or debug the macro, you need to disable the AUTO execution option. This macro saves you seven key strokes when you have to disable the AUTO option. The macro uses the exact key sequence as you do from the keyboard; **/ W**orksheet **G**lobal **A**utoexec **N**o **S**tatus **Q**uit **Q**uit, and then the ESC key.

Notice that when the macro issues the **S**tatus menu option, it halts and displays the status screen. When you press a key, the macro issues **Q**uit **Q**uit and then the ESC key to return to the READY mode. Normally one **Q**uit is enough. However in Lotus 1-2-3 for Windows you need to press the **Q**uit twice. If you press a second "Q" in a 2-D release this "Q" appears in the panel, therefore the {ESC} clears the "Q" character from the panel.

[2] Enable the Autoexec Macro from Running

```
      A                B                C                D                E
 1 *---A macro to enable the AUTOEXEC \0 macros
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                   THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 AUTO-YES        /WGDAYSQQ{ESC}
```

The AUTO option in Lotus is an important feature, since it allows you to create automatic running application. If you assign the [\0] range name to a macro, when you retrieve the worksheet containing it, the macro automatically starts to run. However, when you edit or debug the macro, you usually disable the AUTO execution option. When you want to enable the AUTO execution option, this macro saves you seven key strokes. The macro uses the exact key sequence as you do from the keyboard; **/ W**orksheet **G**lobal **A**utoexec **Y**es **S**tatus **Q**uit **Q**uit, and then the ESC key.

Notice that when the macro issues the **S**tatus menu option, it halts and displays the status screen. When you press a key, the macro issues **Q**uit **Q**uit and then the ESC key to return to the READY mode. Normally one **Q**uit is enough. However in Lotus 1-2-3 for Windows you need to press the **Q**uit twice. If you press a second "Q" in a 2-D release this "Q" appears in the panel, therefore the {ESC} clears the "Q" character from the panel.

[3] Toggle Autoexec Macro Execution Mode

```
     A                B                C                D                E
 1 *---A macro to TOGGLE AUTOEXEC \0 macros between YES or NO
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                   THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 AUTOTOGL         {IF status607="AUTO-NO"}{LET status607,"AUTO-YES"}
                    {BRANCH set607}
12 !                {IF status607="AUTO-YES"}{LET status607,"AUTO-NO"}
                    {BRANCH clear607}
13 !
14 set607           /WGDAYSQQ{ESC}
15 !
16 clear607         /WGDANSQQ{ESC}
17 !
18 status607        AUTO-YES
```

This is a toggle macro combining the AUTO-YES.WK1 and the AUTO-NO.WK1 macros. As a default, the first time that the macro is activated, it disables the AUTO option. But the macro "remembers" the last operation, therefore, next time you use the macro it enables the AUTO option. Every time you use the macro (without changing the working file) the macro changes the AUTO status.

The macro starts with the `{IF status607="AUTO-NO"}` macro command which checks the content of the cell named [status607]. If the cell contains the "`AUTO-NO`" string, the macro issues `{LET status606,"AUTO-NO"}`, which replaces the "`AUTO-YES`" string with the "`AUTO-NO`" string in [status607]. Next the macro issues `{BRANCH set607}`, which starts the [set607] routine and enables the AUTO option. If the condition was false, the macro issues `{IF status607="AUTO-YES"}`, which again checks the content of [status607]. If the cell contains the "`AUTO-YES`" string, the macro issues `{LET status607,"AUTO-NO"}`, which replaces the "`AUTO-YES`" string with the "`AUTO-NO`" string in [status607]. Next the macro issues `{BRANCH clear607}` which starts the [clear607] routine and disables the AUTO option.

[2] Disable the Beep Tone

```
     A                    B                C                D                E
 1 *---A macro to disable the BEEP bell in case of error
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                    THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 BEEP-NO          /WGDOBNSQQ{ESC}
```

You can instruct Lotus to disable the BEEP tone or resume it. This macro saves you eight key strokes when you wish to disable the BEEP tone. The macro uses the exact key sequence as you do from the keyboard; **/ W**orksheet **G**lobal **D**efault **O**ther **B**eep **N**o **S**tatus **Q**uit **Q**uit, and then the ESC key.

Notice that when the macro issues the **S**tatus menu option, it halts and displays the status screen. When you press a key, the macro issues **Q**uit **Q**uit and then the ESC key to return to the READY mode. Normally one **Q**uit is enough. However in Lotus 1-2-3 for Windows you need to press the **Q**uit twice. If you press a second "Q" in a 2-D release this "Q" appears in the panel, therefore the `{ESC}` clears the "Q" character from the panel.

[2] Enable the Beep Tone

```
        A                   B               C               D               E
 1 *---A macro to enable the BEEP bell in case of error
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                   THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 BEEP-YES        /WGDOBYSQQ{ESC}
```

You can instruct Lotus to enable the BEEP tone or disable it. This macro saves you eight key strokes when you wish to enable the BEEP tone. The macro uses the exact key sequence as you do from the keyboard; **/ W**orksheet **G**lobal **D**efault **O**ther **B**eep **Y**es **S**tatus **Q**uit **Q**uit, and then the ESC key.

Notice that when the macro issues the **S**tatus menu option, it halts and displays the status screen. When you press a key, the macro issues **Q**uit **Q**uit and then the ESC key to return to the READY mode. Normally one **Q**uit is enough. However in Lotus 1-2-3 for Windows you need to press the **Q**uit twice. If you press a second "Q" in a 2-D release this "Q" appears in the panel, therefore the `{ESC}` clears the "Q" character from the panel.

[3] Toggle the Beep Tone Mode

```
     A              B              C              D              E
 1 *---A macro to TOGGLE BEEP i.e. to switch between BEEP ON and OFF
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                   THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 BEEPTOGL         {IF status605="BEEPOFF"}{LET status605,"BEEPON"}
                    {BRANCH set605}
12 !                {IF status605="BEEPON"}{LET status605,"BEEPOFF"}
                    {BRANCH clear605}
13 !
14 set605           /WGDOBYSQQ{ESC}
15 !
16 clear605         /WGDOBNSQQ{ESC}
17 !
18 status605        BEEPOFF
```

This is a toggle macro combining the BEEP-YES.WK1 and the BEEP-NO.WK1 macros. As a default, the first time that the macro is activated, it enables the BEEP option. But the macro "remembers" the last operation, therefore, next time you use the macro it disables the BEEP option. Every time you use the macro (without changing the working file) the macro changes the BEEP status.

The macro starts with the `{IF status605="BEEPOFF"}` macro command which checks the content of the cell named [status605]. If the cell contains the "`BEEPOFF`" string, the macro issues `{LET status605,"BEEPON"}`, which replaces the "`BEEPOFF`" string with the "`BEEPON`" string in [status605]. Next the macro issues `{BRANCH set605}` which starts the [set605] routine and enables the BEEP option. If the condition was false, the macro issues `{IF status605="BEEPON"}`, which again checks the content of [status605]. If the cell contains the "`BEEPON`" string the macro issues `{LET status605,"BEEPOFF"}`, which replaces the "`BEEPON`" string with the "`BEEPOFF`" string in [status605]. Next the macro issues `{BRANCH clear605}` which starts the [clear605] routine and disables the BEEP option.

[4] Transform All the Formulas in a Worksheet into Values

```
     A                B                C                D                E
 1 *---A macro to change all formulas in 3-D or 2-D worksheet into values.
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 6                  IT WILL WORK IN RELEASE 2.0 AND UP
 7 !
 8      ATTENTION: This macro changes the whole nature of the worksheet
 9                 and all formulas will be turned into values
10 \Z               {BREAKON}
11 VALUEWKS         {LET rel210,@INFO("release")}~
12 !                Are you sure ? [Y/N] N {GET key210}{ESC}{WINDOWSOFF}
                    {PANELOFF}
13 !                {IF @LEFT(rel210,1)="@"#AND#@UPPER(key210)="Y"}{HOME}
                    /RV.{END}{HOME}~~
14 !                {IF @LEFT(rel210,1)<>"@"#AND#@UPPER(key210)="Y"}{FC}
                    /RV.{LC}~~
15 !
16 key210           Y
17 !
18 rel210           3.00.00
```

This macro turns all the formulas in the worksheet into values. The macro starts with the `{LET rel210,@INFO("release")}~` command, which stores the result of the `@INFO("release")` 3-D function in the B18 cell named [rel210]. Later, the macro uses the content of [rel210] to determine whether you are using a 2-D or a 3-D Lotus release. The macro continues and writes the "`Are you sure ? [Y/N] N` " text message directly to the panel as a prompt, issues `{GET key210}` to suspend execution, and waits for your response. When you press a key, the macro issues `{ESC}` to clear the message from the panel before Lotus writes it to the current cell. Next the macro issues `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel activities before starting to alter the worksheet.

Now the macro issues `{IF @LEFT(rel210,1)="@"#AND#@UPPER(key210)="Y"}` which checks if the first character of the content of [rel210] is equal to the "`@`" character, and if you pressed the "`Y`" or the "`y`" character. If so, you are using a 2-D Lotus release, therefore the macro issues `{HOME}/RV.{END}{HOME}~~` to alter all the formulas in the worksheet into values. If the condition is not true, the macro issues `{IF @LEFT(rel210,1)<>"@"#AND#@UPPER(key210)="Y"}`, which checks if the first character of the content of [rel210] is not equal to the "`@`" character, and if you pressed the "`Y`" or the "`y`" character. If so, you are using a 3-D Lotus release, which may contain more than one sheet. Therefore the macro issues `{FC}/RV.{LC}~~` to alter all the formulas in the worksheet into values.

Note: The `{LC}` and the `{LASTCELL}` macro commands are the same command, the `{FC}` and the `{FIRSTCELL}` macro commands are the same command.

[4] Transform the Formulas in a Selected Range into Values

	A	B	C	D	E
1	*---A macro to REPLACE a range with formulas to values				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	VALUFORM	{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~ {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~			
12	!	/RVWhich range ?~~/RNDWhich range ?~			

This macro turns all the formulas in a selected range into values. The macro issues the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and panel activities before it starts to alter the worksheet. Next the macro uses the "Safe technique" to prompt you to paint the range to alter and simultaneously assigns the [Which range ?] name to the same range which it uses as a prompt. When you paint the range and press ENTER, the macro issues `/RVWhich range ?~~` to alter all the formulas in [Which range ?] into values, and then issues `/RNDWhich range ?~` to delete the temporary [Which range ?] range name to leave a clean worksheet.

[6] Strip Leading and Trailing Spaces from Strings

```
     A                B                C                D                E
 1 *---A macro to STRIP leading and trailing spaces from all strings
 2     in a range, can be used to UNDO the CENTER2.WK1 macro operation
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                 IT WILL WORK IN RELEASE 2 AND UP
 9 !
10 \Z              {BREAKON}
11 STRIPSPC        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb531,0}~
12 cont531         {LET hereabs531,@CELLPOINTER("address")}~
                   {LET counter1531,0}
13 !               {FOR counter1531,0,@COLS(Which range ?)-1,1,labels1531}
14 !               {LET rel531,@INFO("release")}~{IF @LEFT(rel531,1)<>"@"}
                   {GOTO}{hereabs531}~{LET counterb531,counterb531+1}~
                   {IF counterb531<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs531}~{BRANCH cont531}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1531                3
18 counter1a531               0
19 labels1531      {FOR counter1a531,0,@ROWS(Which range ?)-1,1,
                   labels1a531}~{RIGHT}{UP @ROWS(Which range ?)}
                   {LET counter1a531,0}~
20 !
21 labels1a531     {IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{DEL}@TRIM
                   ("{END}"){CALC}
22 !               {DOWN}
23 !
24 !
25 counterb531                 0
26 hereabs531      $P$1
27 !
28 rel531          @INFO("release")
```

This macro uses the `@TRIM` function to clean off all the labels/strings in the range from leading or trailing spaces. You are prompted to highlight the range to process and the macro strips all the labels/strings in this range. The macro ignores the values and the empty cells. The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to process and simultaneously assigns the [Which range ?] name to the same range. Next the macro issues `{LET counterb531,0}~`, which set the content of the B25 cell named [counterb531] to zero, which serves as a counter.

```
     A                B                C                D                E
12 cont531         {LET hereabs531,@CELLPOINTER("address")}~
                   {LET counter1531,0}
13 !               {FOR counter1531,0,@COLS(Which range ?)-1,1,labels1531}
14 !               {LET rel531,@INFO("release")}~{IF @LEFT(rel531,1)<>"@"}
                   {GOTO}{hereabs531}~{LET counterb531,counterb531+1}~
                   {IF counterb531<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs531}~{BRANCH cont531}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro is finished, the macro issues `{LET hereabs531,@CELLPOINTER("address")}~`, storing the current cell pointer address in

[hereabs531]. Then it issues {LET counter1531,0} to set the content of [counter1531] to zero, which also serves as a counter. The {FOR counter1531,0,@COLS(Which range ?)-1,1,labels1531} loop command activates the [labels1531] routine as many times as the number of columns in [Which range ?]. Before we continue with this code, let's look at the [labels1531] routine.

	A	B	C	D	E
19	labels1531	{FOR counter1a531,0,@ROWS(Which range ?)-1,1, labels1a531}~{RIGHT}{UP @ROWS(Which range ?)} {LET counter1a531,0}~			

The [labels1531] routine issues {FOR counter1a531,0,@ROWS(Which range ?)-1,1, labels1a531}~ to activate the [labels1a531] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
21	labels1a531	{IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{DEL}@TRIM ("{END}"){CALC}			
22	!	{DOWN}			

The [labels1a531] routine issues {IF @CELLPOINTER("type")="l"} which checks if the cell contains a label. If so, it issues {EDIT}{HOME}{DEL}@TRIM("{END}"){CALC}{DOWN}. To better understand how the macro works, let's assume that the current cell contains the "_____TOTAL_____" label, where the underscores mean spaces. When the macro issues {EDIT}, Lotus enters into the EDIT mode and the panel displays the current cell content:

```
'_____TOTAL_____
```

Next the macro issues {HOME}, which moves the cursor to the beginning of the text in the panel, {DEL} which deletes the apostrophe "'" prefix, and types the [@TRIM("] text directly into the panel, which displays:

```
@TRIM("_____TOTAL_____
```

Now the macro issues {END} moving the cursor to the end of the text in the panel and continues to type the [")] text into the panel, creating an enveloping formula around the label. Now the panel displays:

```
@TRIM("_____TOTAL_____")
```

To change the formula into its label result, the macro issues {CALC} which is exactly the same as the F9 key. Now the panel displays:

```
TOTAL
```

To write the text in the panel and move to the next cell to process, the macro issues {DOWN}. When the [labels1a531] routine ends, the macro returns control to the {FOR} loop command in the [labels1531] routine to process the next cell in the current column.

	A	B	C	D	E
19	labels1531	{FOR counter1a531,0,@ROWS(Which range ?)-1,1, labels1a531}~{RIGHT}{UP @ROWS(Which range ?)} {LET counter1a531,0}~			

When the value in [counter1a531] reaches the number of rows in [Which range ?] minus one, the macro issues {RIGHT}, which moves the cell pointer to the last cell of the next column. Next the macro issues {UP @ROWS(Which range ?)} to move the cell pointer to the first cell of the new column and then issues {LET counter1a531,0}~ to reset the value in [counter1a531] to zero. When the [labels1531] routine is finished, the macro returns control back to the [cont531] routine.

	A	B	C	D	E
12	cont531	{LET hereabs531,@CELLPOINTER("address")}~			
		{LET counter1531,0}			
13	!	{FOR counter1531,0,@COLS(Which range ?)-1,1,labels1531}			
14	!	{LET rel531,@INFO("release")}~{IF @LEFT(rel531,1)<>"@"}			
		{GOTO}{hereabs531}~{LET counterb531,counterb531+1}~			
		{IF counterb531<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs531}~{BRANCH cont531}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

When the macro finishes processing all the columns in the first sheet of [Which range ?] (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues {LET rel531,@INFO("release")}~ which store the result of the @INFO("release") 3-D function in [rel531] and then it issues {IF @LEFT (rel531,1)<>"@"}, which checks the first character of the content of [rel531]. If the character is the "@" character, you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in [Which range ?]. Therefore the macro issues the {GOTO}{hereabs531}~ indirect macro command, which moves the cell pointer to the origin address of the macro.

The macro continues with {LET counterb531,counterb531+1}~ to increase the value in [counterb531] by one. Now the macro issues {IF counterb531<@SHEETS(Which range ?)} to check if the value in [counterb531] is less than the number of sheets in [Which range ?]. If so, there are more sheets to process. Therefore the macro issues {NS} to move the cell pointer to the next sheet. Next the macro issues the indirect {GOTO}{hereabs531}~ command which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of [Which range ?], but in the new sheet.

Last, the macro issues {BRANCH cont531} to process the next sheet in [Which range ?]. When the macro finishes processing all the sheets in [Which range ?], the macro issues {GOTO}Which range ?~, which moves the cell pointer to the upper left cell of the first sheet of [Which range ?] and then /RNDWhich range ?~ to delete the temporary [Which range ?] range name to leave a clean worksheet.

[6] Add Zero Leads to All Numbers

```
     A                B                C                D                E
 1 *---A macro to add zero leads to all numbers in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                    IT WILL WORK IN RELEASE 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 ZEROLEAD        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb212,0}~
12 !               {GETLABEL "How many digits for the numbers ? ",
                   digits212}~{RECALC digits1212}
13 cont212         {LET hereabs212,@CELLPOINTER("address")}~
                   {LET counter1212,0}
14 !               {FOR counter1212,0,@COLS(Which range ?)-1,1,labels1212}
15 !               {LET rel212,@INFO("release")}~{IF @LEFT(rel212,1)<>"@"}
                   {GOTO}{hereabs212}~{LET counterb212,counterb212+1}~
                   {IF counterb212<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs212}~{BRANCH cont212}
16 !               {GOTO}Which range ?~/RNDWhich range ?~
17 !
18 counter1212                3
19 counter1a212               6
20 labels1212      {RIGHT}{LET here1212,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a212,0,@ROWS(Which range ?)-1,1,
                   labels1a212}~{IF counter1212<@COLS(Which range ?)-1}
                   {GOTO}{here1212}~{LET counter1a212,0}~
21 !
22 here1212        $D$1
23 !
24 labels1a212     {IF @CELLPOINTER("type")="v"}{rnd212}
25 !               {DOWN}
26 !
27 !
28 rnd212          {RECALC digits1212}{EDIT}{HOME}'{digits1212}~
29 !
30 !
31 digits212       7
32 !
33 digits1212      000000
34 !
35 counterb212                5
36 hereabs212      $A$1
37 !
38 rel212
```

If you intend to key the macro into Lotus 1-2-3, notice that the code in the cell named [digits1212] is the result of the dynamic string formula:

```
33 digits1212      @REPEAT("0",@VALUE(digits212)-@LENGTH(@STRING
                   (@CELLPOINTER("contents"),0)))
```

You have to key this formula and NOT the code as it appears in the main listing. Sometimes we like code numbers to have a fix width like 000234 or 023675, therefore we need to add leading zeros to the normal code numbers. This macro adds leading zeros to all the numbers is a range.

```
     A                B                C                D                E
11 ZEROLEAD        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
```

```
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                  {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb212,0}~
12 !              {GETLABEL "How many digits for the numbers ? ",
                  digits212}~{RECALC digits1212}
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to process, and also assigns the [Which range ?] name to the same range, which acts as a prompt. Next the macro issues `{LET counterb212,0}~` to set the content of[counterb212] to zero, which serves as a counter. Now the macro issues `{GETLABEL "How many digits for the numbers ? ",digits212}~`, which display the "`How many digits for the numbers ?` " prompt message in the panel. When you respond and insert the number of digits for the numbers, Lotus stores it in [digits212]. Next the macro issues `{RECALC digits1212}` which updates the dynamic formula in [digits1212], which we will analyze later.

```
     A            B                C              D              E
13 cont212        {LET hereabs212,@CELLPOINTER("address")}~
                  {LET counter1212,0}
14 !              {FOR counter1212,0,@COLS(Which range ?)-1,1,labels1212}
15 !              {LET rel212,@INFO("release")}~{IF @LEFT(rel212,1)<>"@"}
                  {GOTO}{hereabs212}~{LET counterb212,counterb212+1}~
                  {IF counterb212<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs212}~{BRANCH cont212}
16 !              {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro finishes, it issues `{LET hereabs212,@CELLPOINTER("address")}~`, which stores the current cell pointer address in [hereabs212], and then issues `{LET counter1212,0}` to set the content of [counter1212] to zero, which also serves as a counter.

The macro issues the `{FOR counter1212,0,@COLS(Which range ?)-1,1, labels1212}` loop command which activates the [labels1212] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels1212] routine.

```
     A            B                C              D              E
20 labels1212     {RIGHT}{LET here1212,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a212,0,@ROWS(Which range ?)-1,1,
                  labels1a212}~{IF counter1212<@COLS(Which range ?)-1}
                  {GOTO}{here1212}~{LET counter1a212,0}~
```

The [labels1212] routine uses `{RIGHT}{LET here1212,@CELLPOINTER("address")}~ {LEFT}` to record the address of the first cell of the next column in [here1212]. When the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here1212]. The `{FOR counter1a212,0, @ROWS(Which range ?)-1,1,labels1a212}~` macro command activates the [labels1a212] routine as many times as the number of rows in the [Which range ?] range.

```
     A            B                C              D              E
24 labels1a212    {IF @CELLPOINTER("type")="v"}{rnd212}
25 !              {DOWN}
26 !
27 !
28 rnd212         {RECALC digits1212}{EDIT}{HOME}'{digits1212}~
```

The [labels1a179] routine issues `{IF @CELLPOINTER("type")="v"}` to check if the current

cell content is a value. If so, the macro issues the {rnd212} routine command which activates the [rnd212] routine. The [rnd212] routine starts with {RECALC digits1212} which updates the formula in[digits1212]. Let's look at the formula:

```
33 digits1212      @REPEAT("0",@VALUE(digits212)-@LENGTH(@STRING
                   (@CELLPOINTER("contents"),0)))
```

To understand the formula, let's assume that the current cell contains the number "123" and that you chose eight digits for the number. The @STRING(@CELLPOINTER("contents"),0) formula returns the number "3". The @VALUE(digits212) formula returns the number "8", therefore the formula in [digits1212] is the same as the @REPEAT("0",5) formula which returns the "00000" string.

Next the macro issues {EDIT}{HOME}' which add the apostrophe "'" to the beginning of the number in the current cell (which is 123 number in our example). Now comes the trick: the macro issues the {digits1212} routine command which injects the content of [digits1212] into the panel following the apostrophe and preceding the 123 number. The result in the panel is '00000123. Next the macro issues the tilde "~" command which is equivalent to ENTER and writes the '00000123 string number to the current cell. When the [rnd212] routine is finished, the macro returns to the [labels1a212] routine and issues {DOWN} to move the cell pointer down to the next cell. When the [labels1a212] routine ends, the macro returns the control to the {FOR} loop command in the [labels1212] routine to start processing the next cell in the current column.

```
     A                B                C                D                E
20 labels1212      {RIGHT}{LET here1212,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a212,0,@ROWS(Which range ?)-1,1,
                   labels1a212}~{IF counter1212<@COLS(Which range ?)-1}
                   {GOTO}{here1212}~{LET counter1a212,0}~
```

When the value in [counter1a212] reaches the number of rows in [Which range ?] minus one, the macro issues {IF counter1212<@COLS(Which range ?)-1} to check how many column were processed. If the value in [counter1212] is less than the number of columns in [Which range ?], the macro issues the indirect {GOTO}{here1212}~ macro command which moves the cell pointer to the first cell of the next column. Next the macro issues {LET counter1a212,0}~ to reset the value in [counter1a212] to zero. When the [labels1212] routine is finished, the macro returns control back to the [cont212] routine.

```
     A                B                C                D                E
13 cont212         {LET hereabs212,@CELLPOINTER("address")}~
                   {LET counter1212,0}
14 !               {FOR counter1212,0,@COLS(Which range ?)-1,1,labels1212}
15 !               {LET rel212,@INFO("release")}~{IF @LEFT(rel212,1)<>"@"}
                   {GOTO}{hereabs212}~{LET counterb212,counterb212+1}~
                   {IF counterb212<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs212}~{BRANCH cont212}
16 !               {GOTO}Which range ?~/RNDWhich range ?~
```

When the macro finishes processing the first sheet of [Which range ?], (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release, First the macro issues {LET rel212,@INFO("release")}~, which store the result of the @INFO("release") 3-D function in [rel212], and then the macro issues {IF @LEFT (rel212,1)<>"@"}, which checks the first character of the content of [rel212]. If the character is a "@", you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in [Which range ?]. Therefore the macro issues the {GOTO}

{hereabs212}~ indirect macro command which moves the cell pointer to the origin address of the macro.

The macro continues with {LET counterb212,counterb212+1}~ which increase the value in [counterb212] by one. Now the macro issues {IF counterb212<@SHEETS(Which range ?)} to check if the value in [counterb212] is less than the number of sheets in [Which range ?]. If so, there are more sheets to process. Therefore the macro issues {NS} to move the cell pointer to the next sheet. Next the macro issues the indirect {GOTO}{hereabs212}~ command, which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of [Which range ?], but in the new sheet. Last, the macro issues {BRANCH cont212} to process the cells of [Which range ?] in the new sheet. When all the sheets are processed, the macro issues {GOTO}Which range ?~, which moves the cell pointer to the upper left cell of the first sheet of [Which range ?], and then /RNDWhich range ?~ to delete the temporary [Which range ?] range name to leave a clean worksheet.

[3] Underline a Label

```
     A                    B                    C                    D                    E
 1 *---A macro to UNDERLINE the current label, it will underline a
 2     label with leading spaces too
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Place the cell pointer just under the label to underline
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 UNDERLN2        {WINDOWSOFF}{PANELOFF}@REPEAT(" ",@LENGTH({UP})-@LENGTH
                   (@TRIM({UP})))&@REPEAT("-",@LENGTH(@TRIM({UP})))){CALC}
                   {HOME}'~
```

This macro allows you to underline a label even if the label has leading spaces. Often, when we want to center a label across the row, we add spaces to push the text to the center. This macro calculates how many spaces are in the text and creates a matching underline. For example, if the worksheet looks like:

```
     A                    B                    C                    D                    E
 1                        The year 1992 car sales
 2
```

and the " `Car sales in the year 1992` " text (including the spaces) is written in the A1 cell, the macro will enter the underline in the A2 cell to look like this:

```
     A                    B                    C                    D                    E
 1                        Car sales in the year 1992
 2                        --------------------------
```

Place the cell pointer on the cell below that contains the label and then activate the macro. The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the screen and panel display activities, while the macro types the "`@REPEAT(" ",@LENGTH(`" text into the panel, the panel displays:

```
@REPEAT(" ",@LENGTH(
```

Next the macro issues `{UP}` which points to the cell containing the label, the A1 cell in our example, therefore the panel displays:

```
@REPEAT(" ",@LENGTH(A1
```

Now the macro types the "`)-@LENGTH(@TRIM(`" text into the panel which displays:

```
@REPEAT(" ",@LENGTH(A1)-@LENGTH(@TRIM(
```

Again, the macro issues `{UP}`, which points to the cell containing the label, the A1 cell in our example, therefore the panel displays:

```
@REPEAT(" ",@LENGTH(A1)-@LENGTH(@TRIM(A1
```

Now the macro types "))) &@REPEAT("-",@LENGTH(@TRIM(" into the panel, which displays:

```
@REPEAT(" ",@LENGTH(A1)-@LENGTH(@TRIM(A1)))&@REPEAT("-",@LENGTH
    (@TRIM(
```

Again, the macro issues {UP}, which points to the cell containing the label, the A1 cell in our example, therefore the panel will display:

```
@REPEAT(" ",@LENGTH(A1)-@LENGTH(@TRIM(A1)))&@REPEAT("-",@LENGTH
    (@TRIM(A1
```

Last the macro types ")))" into the panel to finish the formula. Therefore the panel displays:

```
@REPEAT(" ",@LENGTH(A1)-@LENGTH(@TRIM(A1)))&@REPEAT("-",@LENGTH
    (@TRIM(A1)))
```

For the final touch, the macro issues {CALC}, while the text is still in the panel, which transforms the formula into its result, causing the panel to display the underline line. The macro continues with {HOME}'~ which move the cursor to the beginning of the text in the panel and then type the apostrophe "'" to signal Lotus that this is a label (otherwise Lotus considers the "-" as the beginning of a value). This is a fine example of how to create an enveloping formula in the panel to manipulate the data in the cell.

[6] Un-Indent Label

```
     A                B                C                D                E
 1 *---A macro to UN-INDENT all LABELS in a 3-D or 2-D range a
 2     specified No. of spaces
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                    IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 UNINDENT        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb165,0}~
12 !               {GETNUMBER "How many characters to un-indent ? ",
                   spaces165}~
13 cont165         {LET hereabs165,@CELLPOINTER("address")}~
                   {LET counter1165,0}
14 !               {FOR counter1165,0,@COLS(Which range ?)-1,1,labels1165}
15 !               {LET rel165,@INFO("release")}~{IF @LEFT(rel165,1)<>"@"}
                   {GOTO}{hereabs165}~{LET counterb165,counterb165+1}~
                   {IF counterb165<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs165}~{BRANCH cont165}
16 !               {GOTO}Which range ?~/RNDWhich range ?~
17 !
18 counter1165           1
19 counter1a165          0
20 labels1165      {RIGHT}{LET here165,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a165,0,@ROWS(Which range ?)-1,1,
                   labels1a165}~{IF counter1165<@COLS(Which range ?)-1}
                   {GOTO}{here165}~{LET counter1a165,0}~
21 !
22 here165         $F$1
23 !
24 labels1a165     {IF @CELLPOINTER("type")="l"}{rnd165}
25 !               {DOWN}
26 !
27 rnd165          {IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{RIGHT}
                   {DEL spaces165}{END}~
28 !
29 spaces165            10
30 !
31 counterb165           1
32 hereabs165      $A$1
33 rel165
```

This macro will trim a specified number of characters from any label in a range. It is a complementary macro to the INDENT.WK1 macro. The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to un-indent and simultaneously assigns the [Which range ?] range name to the same range, which the macro uses as a prompt. Next the macro issues `{LET counterb165,0}~` which set the content of the cell named [counterb165] to zero, which serves as a counter. The `{GETNUMBER "How many characters to un-indent ? ",spaces165}` command, displays the "`How many characters to un-indent ?` " prompt message in the panel and waits for your response. When you press ENTER, Lotus stores you response in cell [spaces165].

```
     A                B                C                D                E
13 cont165         {LET hereabs165,@CELLPOINTER("address")}~
                   {LET counter1165,0}
14 !               {FOR counter1165,0,@COLS(Which range ?)-1,1,labels1165}
```

```
15 !              {LET rel165,@INFO("release")}~{IF @LEFT(rel165,1)<>"@"}
                  {GOTO}{hereabs165}~{LET counterb165,counterb165+1}~
                  {IF counterb165<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs165}~{BRANCH cont165}
16 !              {GOTO}Which range ?~/RNDWhich range ?~
```

To return to the point of origin when the macro is finished, it issues `{LET hereabs165, @CELLPOINTER("address")}~`, which store the current cell pointer address in [hereabs165], and then it issues `{LET counter1165,0}` to set the content of [counter1165] to zero, which also serves as a counter. The macro issues the `{FOR counter1165,0,@COLS(Which range ?)-1,1,labels1165}` loop command which activates the [labels1165] routine as many times as the number of columns in [Which range ?]. Before we continue with this code, let's look at the [labels1165] routine.

```
            A                  B              C              D            E
20 labels1165     {RIGHT}{LET here165,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a165,0,@ROWS(Which range ?)-1,1,
                  labels1a165}~{IF counter1165<@COLS(Which range ?)-1}
                  {GOTO}{here165}~{LET counter1a165,0}~
```

This routine uses `{RIGHT}{LET here165,@CELLPOINTER("address")}~{LEFT}` to record the address of the first cell of the next column in cell [here165]. When the current column processing is finished, the macro uses the address stored in [here165] to move the cell pointer directly to the top of the next column. The `{FOR counter1a165,0,@ROWS(Which range ?)-1,1,labels1a165}~` commands activate the [labels1a165] routine as many times as the number of rows in the [Which range ?] range.

```
            A                  B              C              D            E
24 labels1a165    {IF @CELLPOINTER("type")="l"}{rnd165}
25 !              {DOWN}
26 !
27 rnd165         {IF @CELLPOINTER("type")="l"}{EDIT}{HOME}{RIGHT}
                  {DEL spaces165}{END}~
```

The [labels1a165] routine uses `{IF @CELLPOINTER("type")="l"}` to check if the current cell content is a label. If so, the macro issues the `{rnd165}` routine command that activates the [rnd165] routine. The [rnd165] routine uses `{EDIT}{HOME}{RIGHT}{DEL spaces165}{END}~` to delete the specified number of characters from the left side of the label in the cell. Lotus stores the number of characters to delete in cell [spaces165]. The `{DEL spaces165}` command does the actual job of deleting the characters. Using the `{DEL spaces165}` command is quite unusual because [spaces165] contains a value. While it seems logical that it should be a label, Lotus demands that it must be a value. If we want to use it with a label, we need to use the following string formula

```
+"{EDIT}{HOME}{RIGHT}{DEL "&spaces165&"}{END}~"
```

to create the desired code. Here [spaces165] must contains a label and not a value. Let's continue with the [labels1165] routine.

```
            A                  B              C              D            E
20 labels1165     {RIGHT}{LET here165,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a165,0,@ROWS(Which range ?)-1,1,
                  labels1a165}~{IF counter1165<@COLS(Which range ?)-1}
                  {GOTO}{here165}~{LET counter1a165,0}~
```

When this `{FOR}` loop is finished with the first column, the macro uses `{IF counter1165`

<@COLS(Which range ?)-1} to check if there are more columns to process. If so, the macro issues the {GOTO}{here165}~ indirect macro command to move to the first cell of the next column ([here165] holds the address of the first cell in the next column). The last macro command in this routine is {LET counter1a165,0}~ that reset the counter in [counter1a165] to zero. Now we can go back to the [cont165] routine.

	A	B	C	D	E
13	cont165	{LET hereabs165,@CELLPOINTER("address")}~			
		{LET counter1165,0}			
14	!	{FOR counter1165,0,@COLS(Which range ?)-1,1,labels1165}			
15	!	{LET rel165,@INFO("release")}~{IF @LEFT(rel165,1)<>"@"}			
		{GOTO}{hereabs165}~{LET counterb165,counterb165+1}~			
		{IF counterb165<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs165}~{BRANCH cont165}			
16	!	{GOTO}Which range ?~/RNDWhich range ?~			

The macro continues with {LET rel165,@INFO("release")}~ to store the result of the @INFO("release") function in [rel165]. Then the macro issues {IF @LEFT(rel165 ,1)<>"@"} to check if you are using a 2-D or a 3-D Lotus release. If "@" is the first character of the content of [rel165] then you are using a 2-D Lotus release, otherwise you are using a Lotus 3-D release. If you are using a 3-D release the macro uses the {GOTO} {hereabs165}~ indirect command to move to the first cell in the current sheet range, and then issues {LET counterb165,counterb165+1}~ to increase the counter in [counterb165] by one.

Next the macro uses {IF counterb165<@SHEETS(Which range ?)} to compare the counter value in [counterb165] to the number of sheets in [Which range ?]. If the counter value is still less than the number of sheets in [Which range ?], it means that the macro has to process more sheets before it can quit. Therefore the macro issues {NS}{GOTO} {hereabs165}~ to move the cell pointer to the top left cell to process in the new sheet and issues {BRANCH cont165} to loop back to the beginning of the [cont165] routine. When the counter in [counterb165] is equal to the number of sheets in [Which range ?], the macro issues {GOTO}Which range ?~ to place the cell pointer back on the place of origin just before the macro started, and then uses /RNDWhich range ?~ to delete the [Which range ?] range name to leave a clean worksheet.

[6] Indent Text in a Range

```
     A                B                C                D                E
 1 *---A macro to INDENT all LABELS in a 3-D or 2-D range a specified
 2     No. of spaces
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                      IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 INDENT          {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb124,0}~
12 !               {GETNUMBER "How many characters to indent ? ",spaces124}~
                   {RECALC spaces2124}
13 cont124         {LET hereabs124,@CELLPOINTER("address")}~
                   {LET counter1124,0}
14 !               {FOR counter1124,0,@COLS(Which range ?)-1,1,labels1124}
15 !               {LET rel124,@INFO("release")}~{IF @LEFT(rel124,1)<>"@"}
                   {GOTO}{hereabs124}~{LET counterb124,counterb124+1}~
                   {IF counterb124<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs124}~{BRANCH cont124}
16 !               {GOTO}Which range ?~/RNDWhich range ?~
17 !
18 counter1124           1
19 counter1a124          0
20 labels1124      {RIGHT}{LET here124,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a124,0,@ROWS(Which range ?)-1,1,labels1a124}~
                   {IF counter1124<@COLS(Which range ?)-1}{GOTO}{here124}~
                   {LET counter1a124,0}~
21 !
22 here124         $F$1
23 !
24 labels1a124     {IF @CELLPOINTER("type")="l"}{rnd124}~
25 !               {DOWN}
26 ret124
27 !
28 rnd124          {EDIT}{HOME}{RIGHT}{spaces2124}{END}~
29 !
30 spaces124            10
31 !
32 spaces2124          ERR
33 !
34 spaces3124
35 !
36 counterb124           6
37 hereabs124      $A$1
38 !
39 rel124
```

The B32 cell contains a formula, therefore you must key this formula if you intend to key this macro into Lotus 1-2-3.

```
32 spaces2124      @LEFT(B34,B30)
```

The formula in the B32 cell makes use of the B34 cell which must contain a **long string of spaces** at least 80 characters long. Therefore you must key a long string of spaces in the B34 cell, otherwise the formula in the B32 cell will produce an error.

This macro adds a specified number of characters to any label in a range. It is a complementary macro to the previous UN-INDENT.WK1 macro. The macro starts with the

{WINDOWSOFF}{PANELOFF} macro commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to indent and simultaneously assigns the [Which range ?] name to the same range, which the macro then uses as a prompt. Next the macro issues {LET counterb124,0}~ to set the content of the cell named [counterb124] to zero, which serves as a counter. The {GETNUMBER "How many characters to indent ? ",spaces124}~ commands display the "How many characters to indent ? " prompt message in the panel and waits for your response. When you press ENTER, Lotus stores your response in cell [spaces124].

	A	B	C	D	E
13	cont124	{LET hereabs124,@CELLPOINTER("address")}~			
		{LET counter1124,0}			
14	!	{FOR counter1124,0,@COLS(Which range ?)-1,1,labels1124}			
15	!	{LET rel124,@INFO("release")}~{IF @LEFT(rel124,1)<>"@"}			
		{GOTO}{hereabs124}~{LET counterb124,counterb124+1}~			
		{IF counterb124<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs124}~{BRANCH cont124}			
16	!	{GOTO}Which range ?~/RNDWhich range ?~			

To be able to return to the point of origin when the macro is finished, the macro issues {LET hereabs124,@CELLPOINTER("address")}~, which store the current cell pointer address in cell [hereabs124], and then it issues {LET counter1124,0} to set the content of [counter1124] to zero, which also serves as a counter. The macro issues the {FOR counter1124,0,@COLS(Which range ?)-1,1,labels1124} loop command which activates the [labels1124] routine as many times as the number of columns in [Which range ?]. Before we continue with this code, let's look at the [labels1124] routine.

	A	B	C	D	E
20	labels1124	{RIGHT}{LET here124,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a124,0,@ROWS(Which range ?)-1,1,labels1a124}~			
		{IF counter1124<@COLS(Which range ?)-1}{GOTO}{here124}~			
		{LET counter1a124,0}~			

This routine uses {RIGHT}{LET here124,@CELLPOINTER("address")}~{LEFT} to record the address of the first cell of the next column in the cell named [here124]. When the current column processing is finished, the macro uses the address stored in [here124] to move the cell pointer directly to the top of the next column. The {FOR counter1a124,0,@ROWS (Which range ?)-1,1,labels1a124}~ command activates the [labels1a124] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
24	labels1a124	{IF @CELLPOINTER("type")="l"}{rnd124}~			
25	!	{DOWN}			
26	ret124				
27	!				
28	rnd124	{EDIT}{HOME}{RIGHT}{spaces2124}{END}~			

The [labels1a124] routine uses {IF @CELLPOINTER("type")="l"} to check if the current cell's content is a label. If so, the macro issues the {rnd124} routine command which activates the [rnd124] routine. The [rnd124] routine uses {EDIT}{HOME}{RIGHT} to move the cursor to the first character in the panel and then issues {spaces2124}.

32	spaces2124	@LEFT(B34,B30)

The code in the [spaces2124] routine is a string of spaces which is the result of the @LEFT(B34,B30) dynamic string formula. For example, if you insert the number 10 as a

response to the prompt message, the result of this formula is a string of 10 spaces taken from the left side of the long string of spaces in the B34 cell. Therefore, when the macro issues the {spaces2124} routine command, Lotus writes the 10 spaces into the panel which pushes (indents) the content of the panel 10 space characters to the right. Next the routine issues {END}~ which write the content of the panel into the current cell.

Let us continue with the [labels1124] routine.

	A	B	C	D	E
20	labels1124	{RIGHT}{LET here124,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a124,0,@ROWS(Which range ?)-1,1,labels1a124}~			
		{IF counter1124<@COLS(Which range ?)-1}{GOTO}{here124}~			
		{LET counter1a124,0}~			

When the {FOR} loop is finished with the first column, the macro issues {IF counter1124 <@COLS(Which range ?)-1} to check if there are more columns to process. If so, the macro issues the {GOTO}{here124}~ indirect command to move to the first cell of the next column ([here124] holds the address of the first cell in the next column). The last macro command in this routine is {LET counter1a124,0}~ which reset the counter in [counter1a124] to zero. Now we can go back to the [cont124] routine.

	A	B	C	D	E
13	cont124	{LET hereabs124,@CELLPOINTER("address")}~			
		{LET counter1124,0}			
14	!	{FOR counter1124,0,@COLS(Which range ?)-1,1,labels1124}			
15	!	{LET rel124,@INFO("release")}~{IF @LEFT(rel124,1)<>"@"}			
		{GOTO}{hereabs124}~{LET counterb124,counterb124+1}~			
		{IF counterb124<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs124}~{BRANCH cont124}			
16	!	{GOTO}Which range ?~/RNDWhich range ?~			

The macro continues with {LET rel124,@INFO("release")}~, which store the result of the @INFO("release") function in the cell named [rel124]. Then the macro issues {IF @LEFT(rel124,1)<>"@"} to check if you are using a 2-D or a 3-D Lotus release. If "@" is the first character of the content of [rel124], then you are using a 2-D Lotus release, otherwise you are using a 3-D release. If you are using a 3-D release, the macro issues {GOTO} {hereabs124}~ the indirect macro command to move to the first cell in the current sheet of [Which range ?], and {LET counterb124,counterb124+1}~ to increase the counter in [counterb124] by one.

Next the macro issues {IF counterb124<@SHEETS(Which range ?)} to compare the counter value in [counterb124] to the number of sheets in [Which range ?]. If the counter value is still less than the number of sheets in [Which range ?], the macro has to process more sheets before it can quit. Therefore the macro issues {NS}{GOTO}{hereabs124}~ to move the cell pointer to the upper left cell in the new sheet of [Which range ?] and issues {BRANCH cont124} to loop back to the beginning of the [cont124] routine. When the counter in [counterb124] is equal to the number of sheets in [Which range ?], the macro issues {GOTO}Which range ?~ to place the cell pointer back on the place of origin just before the macro started, and finally issues /RNDWhich range ?~ to delete the temporary [Which range ?] range name to leave a clean worksheet.

Graph Macros

[7] Assign Graph Ranges to All the Graphs in One Step

	A	B	C	D
1	*---A macro to assign all graph data ranges (X and A-F) simultaneously.			
2	Simulates the /Graph Group in Lotus 2.2, the first column/row will			
3	become X and the rest A-F.			
4	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the			
5	range names in this column (starts with the \Z macro name)			
6	*---Hold the [ALT] key and press [Z] to activate the macro			
7	*---Expand and highlight the data range and press [ENTER]			
8	!			
9	!			
10	\Z	{BREAKON}		
11	GRPHGRUP	{MENUBRANCH menua120}		
12	cont1120	{WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RND Group range ?~/RNC{WINDOWSON}{PANELON}Group range ?~{BS}{BS}{?}~{WINDOWSOFF}{GOTO}Group range ?~		
13	cont120	{LET counter1120,0}		
14	!	{PANELOFF}{FOR counter1120,1,@COLS(Group range ?),1, labels1120}		
15	!	{GOTO}Group range ?~/RNDGroup range ?~		
16	!			
17	counter1120	5		
18	counter1a120	421		
19	labels1120	{LET loc2120,@CELLPOINTER("address")}~{LET loc1120,@LEFT (loc2120,@FIND("$",loc2120,2)+1)}~{LET loc3120,loc1120&@STRING(@CELLPOINTER("row")+@ROWS(Group range ?)-1,0)}~		
20	!	{LET loc4120,loc2120&".."&loc3120}~{RECALC rec1120}{RECALC rec2120}{IF counter1120>7}{FORBREAK}~{BRANCH ret120}		
21	rec1120	{PANELOFF}/G{RIGHT 5}~{BS}		
22	rec2120	$G$1..$G$10		
23	!	~Q{RIGHT}{PANELON}		
24	!			
25	menua120	Columnwise	Rowwize	Quit
26	!	The 1st column is X	The 1st row is X	Quit the macro
27	!	{BRANCH cont1120}	{BRANCH cont1a120	{BRANCH ret120}
28	!			
29	cont1a120	{WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RND Group range ?~/RNC{WINDOWSON}{PANELON}Group range ?~{BS}{BS}{?}~{WINDOWSOFF}{GOTO}Group range ?~		
30	!	{LET counter120,0}		
31	!	{PANELOFF}{FOR counter120,1,@ROWS(Group range ?),1, labels2120}		
32	!	{GOTO}Group range ?~/RNDGroup range ?~		
33	!			
34	counter120	8		
35	countera120	421		
36	labels2120	{LET loc2120,@CELLPOINTER("address")}~{LET loc1120,@RIGHT (loc2120,@LENGTH(loc2120)-@FIND("$",loc2120,2)-1)}~		
37	!	{LET cll1120,+@IF(@CELLPOINTER("col")+@COLS (Group range ?)-1>26,@CHAR((@CELLPOINTER("col")+@COLS (Group range ?)-@MOD(@CELLPOINTER("col")+@COLS (Group range ?),26))/26+64),"")&@CHAR(@MOD(@CELLPOINTER ("col")+@COLS(Group range ?),26)+63)}~		
38	!	{LET cll120,+"$"&cll1120&"$"}~{LET loc3120,cll120&@STRING (@VALUE(loc1120),0)}~		
39	!	{LET loc4120,loc2120&".."&loc3120}~{RECALC rec3120}{RECALC rec4120}{IF counter120>7}{FORBREAK}~{BRANCH ret120}		
40	rec3120	{PANELOFF}/G{RIGHT 6}~{BS}"		
41	rec4120	$G$1..$G$10		
42	!	~Q{DOWN}{PANELON}		
43	!			
44	cll120	G		
45	cll1120	G		
46	loc1120	G		
47	loc2120	$G$1		
48	loc3120	$G$10		

```
49 loc4120         $G$1..$G$10
50 ret120
```

The macro uses a custom menu with three menu options. Because there is no way to display the custom menu code on this page without overlapping, we show every menu option's code separately.

```
Columnwise
The 1st column is X and the rest are A-F
{BRANCH cont1120}

Rowwizse
The 1st row is X and the rest are A-F
{BRANCH cont1a120}

Quit
Quit the macro
{BRANCH ret120}
```

Notice the code in the B21, B22, B40 and the B41 cells is the result of the following dynamic string formulas.

```
21 rec1120         +"{PANELOFF}/G{RIGHT "&@STRING(B17,0)&"}~{BS}"
22 rec2120         +B49
40 rec3120         +"{PANELOFF}/G{RIGHT "&@STRING(B34,0)&"}~{BS}"
41 rec4120         +B49
```

Therefore if you intend to key the code of this macro into Lotus 1-2-3, you have to key the code of the formulas and the code of the custom menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you, we bring the full cell contents list at the end of this section.

The new releases of Lotus 1-2-3 (2.2 and up) include a new graph option, called /**G**raph **G**roup, which will automatically assign up to seven adjacent data columns or rows to six graphs. The first column or row of data will be the X data range, the second the A data range, the third the B data range, and so on. This macro allows users of Lotus 2.0 and 2.01 to do the same. The macro starts with the `{MENUBRANCH menua120}` macro command which activates the custom menu in the B25 cell named [menua120].

The first menu option is the [**C**olumnwise] menu option:

```
Columnwise
The 1st column is X and the rest are A-F
{BRANCH cont1120}
```

When you choose this option, the macro issues `{BRANCH cont1120}` which routes the macro control to the [cont1120] routine.

```
              A                  B                    C                    D
12 cont1120        {WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RND
                   Group range ?~/RNC{WINDOWSON}{PANELON}Group range ?~{BS}
                   {BS}{?}~{WINDOWSOFF}{GOTO}Group range ?~
13 cont120         {LET counter1120,0}
14 !               {PANELOFF}{FOR counter1120,1,@COLS(Group range ?),1,
                   labels1120}
15 !               {GOTO}Group range ?~/RNDGroup range ?~
```

This routine starts with `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display

activities and then uses the "safe technique" to assign the [Group range ?] name to the selected range and simultaneously uses the [Group range ?] range name as a prompt. Next, the macro issues {LET counter1120,0}, which sets the contents of the B17 cell named [counter1120] to zero, which serves as a counter for one of the {FOR} loop commands. The macro continues with {PANELOFF} which freezes the panel display activity, and then issues {FOR counter1120,1,@COLS(Group range ?),1,labels1120} which activates the [labels1120] routine as many times as the number of columns in the [Group range ?] range.

	A	B	C	D
19	labels1120	{LET loc2120,@CELLPOINTER("address")}~{LET loc1120,@LEFT (loc2120,@FIND("$",loc2120,2)+1)}~{LET loc3120,loc1120& @STRING(@CELLPOINTER("row")+@ROWS(Group range ?)-1,0)}~		
20	!	{LET loc4120,loc2120&".."&loc3120}~{RECALC rec1120} {RECALC rec2120}{IF counter1120>7}{FORBREAK}~ {BRANCH ret120}		
21	rec1120	{PANELOFF}/G{RIGHT 5}~{BS}		
22	rec2120	$G$1..$G$10		
23	!	~Q{RIGHT}{PANELON}		

To assign the X data range and the A to F data ranges, the macro needs to calculate the data range's addresses. First the macro issues {LET loc2120,@CELLPOINTER("address")}~ which calculate the cell pointer address of the first cell of the data range (column) and stores it in the B47 cell named [loc2120]. Next the macro issues {LET loc1120,@LEFT(loc2120 ,@FIND("$",loc2120,2)+1)}~, to extract the left part of the address (the column letter/s) and store it in [loc1120]. The macro continues with {LET loc3120,loc1120& @STRING(@CELLPOINTER("row")+@ROWS(Group range ?)-1,0)}~, to calculate the address of the last cell of the data range (column) and store it in [loc3120]. Next the macro issues {LET loc4120,loc2120&".."&loc3120}~ which combine the address of the first cell of the data range (column) with the last cell of the data range (column) to create the column range address and store it in [loc4120].

If the graph data range is the A1..G10 (7 columns), then [loc1120] will contain the A string, [loc2120] will contain the $A$1 string, and [loc3120] will contain the $A$10 string. Therefore the resultant range address, stored in [loc4120] will be the $A$1..$A$10 range address. The macro continues with {RECALC rec1120}{RECALC rec2120} which recalculate and update the dynamic string formulas in [rec1120] and [rec2120]. The results of the formulas in these cells become part of the code of the macro.

Now the macro issues {IF counter1120>7} to check if the value in [counter1120] is greater than the number seven. If so, the macro understands that all the graph axes (X and A-F) are defined, therefore the macro issues {FORBREAK} to break out of the {FOR} loop, and then issues {BRANCH ret120}, which routes the macro control to the empty [ret120] routine and quits. If the counter value is less or equals seven, then the macro continues with {PANELOFF}which freezes panel display activity, while the macro continues with /G{RIGHT 6}~, which start the graph assignment and assign the columns of data to the A-F axes. The number in the {RIGHT n} is dynamic. It comes from the counter value in [counter1120]. To understand it, let's look at the dynamic string formula in the B21 cell:

```
21 rec1120        +"{PANELOFF}/G{RIGHT "&@STRING(B17,0)&"}~{BS}"
```

The @STRING(B17,0) function returns the value in [counter1120] (B17) in a string format which becomes part of the {RIGHT n} macro command. The number "n" instructs Lotus how many times to press the RIGHT key to choose the correct menu option for the current column.

When you press the **/ G** keys from the keyboard in Lotus 1-2-3, the following menu appears and the highlight is on the [**T**ype] menu option:

```
Type  X  A  B  C  D  E  F  Reset  View  Save  Options  Name  Group  Quit
```

To select the [**F**] menu option, you need to press the RIGHT key seven times, or once for the [**X**] or three for the [**B**]. This is achieved by the dynamic formula that uses the counter value "n" in the B17 cell to build the `{RIGHT n}` macro command to move to the correct menu option.

Next the macro issues `{BS}` to clear any previous range from the panel, followed by the $G$1..$G$10 range address, and then issues the tilde "~", which is the same as the ENTER key. Last the macro issues the [**Q**] macro key to return to the READY mode and then issues `{RIGHT}` which moves the cell pointer to the next column.

The macro executes the [labels1120] routine as many times as the number of columns in the data range. When the `{FOR}` loop is finished, the macro routes control back to the [cont1120] routine and then issues `{GOTO}Group range ?~/RNDGroup range ?~`, which move the cell pointer to the upper left cell of the data range and delete the temporary [Group range ?] range name to leave a <u>clean</u> worksheet.

The second menu option is the [**R**owwizse] menu option:

```
Rowwizse
The 1st row is X and the rest are A-F
{BRANCH cont1a120}
```

When you choose this option, the macro issues `{BRANCH cont1a120}`, which starts the [cont1a120] routine.

	A	B	C	D
29	cont1a120	{WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RND Group range ?~/RNC{WINDOWSON}{PANELON}Group range ?~{BS} {BS}{?}~{WINDOWSOFF}{GOTO}Group range ?~		
30	!	{LET counter120,0}		
31	!	{PANELOFF}{FOR counter120,1,@ROWS(Group range ?),1, labels2120}		
32	!	{GOTO}Group range ?~/RNDGroup range ?~		

This routine is the same as the [cont1120] routine for the previous menu option, except that the counter here is cell [counter120] and the `{FOR}` loop executes the [labels2120] routine as many time as the number of rows in the data range named [Group range ?] instead.

	A	B	C	D
36	labels2120	{LET loc2120,@CELLPOINTER("address")}~{LET loc1120,@RIGHT (loc2120,@LENGTH(loc2120)-@FIND("$",loc2120,2)-1)}~		
37	!	{LET cll1120,+@IF(@CELLPOINTER("col")+@COLS (Group range ?)-1>26,@CHAR((@CELLPOINTER("col")+@COLS (Group range ?)-@MOD(@CELLPOINTER("col")+@COLS (Group range ?),26))/26+64),"")&@CHAR(@MOD(@CELLPOINTER ("col")+@COLS(Group range ?),26)+63)}~		
38	!	{LET cll120,+"$"&cll1120&"$"}~{LET loc3120,cll120&@STRING (@VALUE(loc1120),0)}~		
39	!	{LET loc4120,loc2120&".."&loc3120}~{RECALC rec3120} {RECALC rec4120}{IF counter120>7}{FORBREAK}~ {BRANCH ret120}		
40	rec3120	{PANELOFF}/G{RIGHT 6}~{BS}"		
41	rec4120	$G$1..$G$10		

```
42 !              ~Q{DOWN}{PANELON}
```

This routine works the same way as the [labels1120] routine, however the formulas to create the rows addresses are more complicated. The `{LET loc2120,@CELLPOINTER ("address")}~` macro command stores the current cell pointer address (which is the leftmost cell of every row in the data range) in cell [loc2120]. The

```
{LET loc1120,@RIGHT(loc2120,LENGTH(loc2120)-@FIND("$",loc2120,2)-1)}~
```

macro command extracts the right part of the address (the row number) and stores it in [loc1120]. Then the

```
{LET cll1120,+@IF(@CELLPOINTER("col")+@COLS(Group range ?)-1>26
    ,@CHAR((@CELLPOINTER("col")+@COLS(Group range ?)-@MOD
    (@CELLPOINTER("col")+@COLS(Group range ?),26))/26+64),"")&
    @CHAR(@MOD(@CELLPOINTER("col")+@COLS(Group range ?),26)+63)}~
```

advanced macro command extracts the left part of the address (the column's letter/s) and stores it in the cell named [cll1120]. The `{LET cll120,+"$"&cll1120&"$"}~` macro command adds the "`$`" characters to both sides of the column's letter/s in cell [cll1120] and stores the result in cell [cll120]. The `{LET loc3120,cll120&@STRING(@VALUE (loc1120),0)}` transforms the row number which is stored in [loc1120] into a label form, and then combines it to the column's letter/s which is stored in [cll120], the result, the address of the last cell in the row, is stored in [loc3120]. Last the macro issues `{LET loc4120, loc2120&".."&loc3120}~`, which give the row range address and store it in [loc4120].

To clarify, let's assume that the range of data to graph is the A1..J7 which has seven rows. Let's see the result of each command for the first row. The cell pointer is on the A1 cell, therefore the result of the commands is:

```
The cell named [loc2120] contains $A$1
The cell named [loc1120] contains 1
The cell named [cll1120] contains J
The cell named [cll120]  contains $J$
The cell named [loc3120] contains $J$1 (address of the last cell in the row)
The cell named [loc4120] contains $A$1..$J$1  (the row's range address)
```

From now on the code is the same as the code of the [labels1120] routine except that the cell pointer moves down from row to row.

The last menu option is the [**Q**uit] menu option:

```
Quit
Quit the macro
{BRANCH ret120}
```

When you choose this option, the macro issues `{BRANCH ret120}`, which routes macro control to the empty [ret120] routine and quits.

To make it easier for you to key this macro into Lotus 1-2-3, we show the full list of cell contents and formulas.

```
A1: U [W15] '*---A macro to assign all graph data ranges (X and A-F)
            simultaneously.
A2: U [W15] '    Simulates the /Graph Group in Lotus 2.2, the first column/
            row will
```

```
A3: U [W15] '    become X and the rest A-F.
A4: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A5: [W15] '    range names in this column (starts with the \Z macro name)
A6: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A7: [W15] '*---Expand and highlight the data range and press [ENTER]
A8: [W15] '!
A9: [W15] '!
A10: U [W15] '\Z
B10: '{BREAKON}
A11: U [W15] 'GRPHGRUP
B11: '{MENUBRANCH menua120}
A12: [W15] 'cont1120
B12: '{WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RNDGroup range ?~
     /RNC{WINDOWSON}{PANELON}Group range ?~{BS}{BS}{?}~{WINDOWSOFF}
     {GOTO}Group range ?~
A13: [W15] 'cont120
B13: '{LET counter1120,0}
A14: [W15] '!
B14: '{PANELOFF}{FOR counter1120,1,@COLS(Group range ?),1,labels1120}
A15: [W15] '!
B15: '{GOTO}Group range ?~/RNDGroup range ?~
A16: [W15] '!
A17: [W15] 'counter1120
B17: 5
A18: [W15] 'counter1a120
B18: 421
A19: [W15] 'labels1120
B19: '{LET loc2120,@CELLPOINTER("address")}~{LET loc1120,@LEFT(loc2120
     ,@FIND("$",loc2120,2)+1)}~{LET loc3120,loc1120&@STRING(@CELLPOINTER
     ("row")+@ROWS(Group range ?)-1,0)}~
A20: [W15] '!
B20: '{LET loc4120,loc2120&".."&loc3120}~{RECALC rec1120}{RECALC rec2120}
     {IF counter1120>7}{FORBREAK}~{BRANCH ret120}
A21: [W15] 'rec1120
B21: +"{PANELOFF}/G{RIGHT "&@STRING(B17,0)&"}~{BS}"
A22: [W15] 'rec2120
B22: +B49
A23: [W15] '!
B23: '~q{RIGHT}{PANELON}
A24: [W15] '!
A25: [W15] 'menua120
B25: 'Columnwise
C25: 'Rowwizse
D25: 'Quit
A26: [W15] '!
B26: 'The 1st column is X and the rest are A-F
C26: 'The 1st row is X and the rest are A-F
D26: 'Quit the macro
A27: [W15] '!
B27: '{BRANCH cont1120}
C27: '{BRANCH cont1a120}
D27: '{BRANCH ret120}
A28: [W15] '!
A29: [W15] 'cont1a120
B29: '{WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RNDGroup range ?~
     /RNC{WINDOWSON}{PANELON}Group range ?~{BS}{BS}{?}~{WINDOWSOFF}
     {GOTO}Group range ?~
A30: [W15] '!
B30: '{LET counter120,0}
A31: [W15] '!
B31: '{PANELOFF}{FOR counter120,1,@ROWS(Group range ?),1,labels2120}
A32: [W15] '!
B32: '{GOTO}Group range ?~/RNDGroup range ?~
A33: [W15] '!
A34: [W15] 'counter120
B34: 8
A35: [W15] 'countera120
B35: 421
A36: [W15] 'labels2120
B36: '{LET loc2120,@CELLPOINTER("address")}~{LET loc1120,@RIGHT(loc2120
```

```
     ,@LENGTH(loc2120)-@FIND("$",loc2120,2)-1)}~
A37: [W15] '!
B37: '{LET cll1120,+@IF(@CELLPOINTER("col")+@COLS(Group range ?)-1>26
     ,@CHAR((@CELLPOINTER("col")+@COLS(Group range ?)-@MOD(@CELLPOINTER
     ("col")+@COLS(Group range ?),26))/26+64),"")&@CHAR(@MOD(@CELLPOINTER
     ("col")+@COLS(Group range ?),26)+63)}~
A38: [W15] '!
B38: '{LET cll120,+"$"&cll1120&"$"}~{LET loc3120,cll120&@STRING(@VALUE
     (loc1120),0)}~
A39: [W15] '!
B39: '{LET loc4120,loc2120&".."&loc3120}~{RECALC rec3120}{RECALC rec4120}
     {IF counter120>7}{FORBREAK}~{BRANCH ret120}
A40: [W15] 'rec3120
B40: +"{PANELOFF}/G{RIGHT "&@STRING(B34,0)&"}~{BS}"
A41: [W15] 'rec4120
B41: +B49
A42: [W15] '!
B42: '~q{DOWN}{PANELON}
A43: [W15] '!
A44: [W15] 'cll120
B44: '$G$
A45: [W15] 'cll1120
B45: 'G
A46: [W15] 'loc1120
B46: '$E$
A47: [W15] 'loc2120
B47: '$E$52
A48: [W15] 'loc3120
B48: '$E$62
A49: [W15] 'loc4120
B49: '$E$52..$E$62
A50: [W15] 'ret120
```

[7] Assign Graph Labels to All the Graphs in One Step

```
     A                    B                    C                    D
 1 *---A macro to assign all graph data labels (A-F) in one step
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Expand and highlight the data range and press [ENTER]
 6 *---When you see the prompt menu bar use the  cell pointer to highlight
 7     your choice and press [return] to make your choice
 8             DO NOT USE THE CAPITAL LETTERS TO MAKE YOUR CHOICE!!
 9 !
10 \Z              {BREAKON}
11 GRPHDLBL        {MENUBRANCH menua121}
12 cont1121        {WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RND
                   Group range ?~/RNC{WINDOWSON}{PANELON}Group range ?~{BS}
                   {BS}{?}~{WINDOWSOFF}
                   {GOTO}Group range ?~
13 cont121         {LET counter1121,0}
14 !               {PANELOFF}{FOR counter1121,0,@COLS(Group range ?),1,labels1121}
15 !               {GOTO}Group range ?~/RNDGroup range ?~
16 !
17 counter1121                    6
18 counter1a121                 421
19 labels1121      {LET loc2121,@CELLPOINTER("address")}~{LET loc1121,@LEFT
                   (loc2121,@FIND("$",loc2121,2)+1)}~{LET loc3121,loc1121&
                   @STRING(@CELLPOINTER("row")+@ROWS(Group range ?)-1,0)}~
20 !               {LET loc4121,loc2121&".."&loc3121}~{RECALC rec1121}
                   {RECALC rec2121}{IF counter1121>5}{FORBREAK}~
                   {BRANCH ret121}
21 rec1121         {PANELOFF}/GOD{RIGHT 6}~{BS}
22 rec2121         $A$60..$E$60
23 !               {PANELON}{WINDOWSON}~{?}~{WINDOWSOFF}{PANELOFF}QQQ{RIGHT}
24 !
25 menua121        Columnwise          Rowwizse          Quit
26 !               The 1st column is AThe 1st row is A Quit the macro
27 !               {BRANCH cont1121}  {BRANCH cont1a121{BRANCH ret121}
28 !
29 cont1a121       {WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RND
                   Group range ?~/RNC{WINDOWSON}{PANELON}Group range ?~{BS}
                   {BS}{?}~{WINDOWSOFF}
                   {GOTO}Group range ?~
30 conta121        {LET counter121,0}
31 !               {PANELOFF}{FOR counter121,0,@ROWS(Group range ?),1,
                   labels2121}
32 !               {GOTO}Group range ?~/RNDGroup range ?~
33 !
34 counter121                     6
35 countera121                  421
36 labels2121      {LET loc2121,@CELLPOINTER("address")}~{LET loc1121,@RIGHT
                   (loc2121,@LENGTH(loc2121)-@FIND("$",loc2121,2)-1)}~
37 !               {LET cll1121,+@IF(@CELLPOINTER("col")+@COLS
                   (Group range ?)-1>26,@CHAR((@CELLPOINTER("col")+@COLS
                   (Group range ?)-@MOD(@CELLPOINTER("col")+@COLS
                   (Group range ?),26))/26+64),"")&@CHAR(@MOD(@CELLPOINTER
                   ("col")+@COLS(Group range ?),26)+63)}~
38 !               {LET cll121,+"$"&cll1121&"$"}~{LET loc3121,cll121&
                   @STRING(@VALUE(loc1121),0)}~
39 !               {LET loc4121,loc2121&".."&loc3121}~{RECALC rec3121}
                   {RECALC rec4121}{IF counter121>5}{FORBREAK}~
                   {BRANCH ret121}
40 rec3121         {PANELOFF}/GOD{RIGHT 3}~{BS}"
41 rec4121         $A$60..$E$60
42 !               {PANELON}{WINDOWSON}~{?}~{WINDOWSOFF}{PANELOFF}QQQ{DOWN}
43 !
44 cll121          $E$
45 cll1121         E
46 loc1121         60
47 loc2121         $A$60
```

```
48 loc3121        $E$60
49 loc4121        $A$60..$E$60
50 ret121
```

The macro uses a custom menu with three menu options. Because there is no way to display the custom menu code on this page without overlapping, we show every menu option's code separately.

```
Columnwise
The 1st column is A and the rest are B-F
{BRANCH cont1121}


Rowwizse
The 1st row is A and the rest are B-F
{BRANCH cont1a121}


Quit
Quit the macro
{BRANCH ret121}
```

Notice the code in the B21, B22, B40 and the B41 cells is the result of the following dynamic string formulas.

```
rec1121        +"{PANELOFF}/GOD{RIGHT "&@STRING(B17,0)&"}~{BS}"
rec2121        +B49
rec3121        +"{PANELOFF}/GOD{RIGHT "&@STRING(B34,0)&"}~{BS}"
rec4121        +B49
```

If you intend to key the code of this macro into Lotus 1-2-3, you have to key the code of the formulas and the code of the custom menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you, we show the full cell contents list at the end of this section.

The new releases of Lotus 1-2-3 (2.2 and up) include a new graph option, called / **G**raph **O**ption **D**ata-labels **G**roup, which will automatically assigns up to six data labels to six graphs. This macro allows the users of Lotus 2.0 and 2.01 to do the same. The macro should be used after the graphs have been created using the previous macro GRAPHGRP.WK1 or manually. The range that contains the text for the data labels is usually the same range you used as the A through F data ranges. This range must have the same size as the range containing the A, B, C, D, E and F data range that you label. For example, if the graph has A to F data ranges, each of which has four values, the data label range must have six rows or columns of four cells each.

Because the data range for labels has the same size as the data range for the graphs except the X data range we can use almost the same code as in the previous macro GRAPHGRP.WK1. Therefore we will not repeat the discussion from the pervious macro except the differences.

```
              A                    B                      C                    D
19 labels1121     {LET loc2121,@CELLPOINTER("address")}~{LET loc1121,@LEFT
                  (loc2121,@FIND("$",loc2121,2)+1)}~{LET loc3121,loc1121&
                  @STRING(@CELLPOINTER("row")+@ROWS(Group range ?)-1,0)}~
20 !              {LET loc4121,loc2121&".."&loc3121}~{RECALC rec1121}
                  {RECALC rec2121}{IF counter1121>5}{FORBREAK}~
                  {BRANCH ret121}
21 rec1121        {PANELOFF}/GOD{RIGHT 6}~{BS}
22 rec2121        $A$60..$E$60
23 !              {PANELON}{WINDOWSON}~{?}~{WINDOWSOFF}{PANELOFF}QQQ{RIGHT}
```

The [labels1121] routine is the same as the [labels1120] routine in GRAPHGRP.WK1 except that the counter, [counter1121], counts one column less then the number of columns that the compatible counter counts in the GRAPHGRP.WK1 macro.

	A	B	C	D
36	labels2121	{LET loc2121,@CELLPOINTER("address")}~{LET loc1121,@RIGHT		
		(loc2121,@LENGTH(loc2121)-@FIND("$",loc2121,2)-1)}~		
37	!	{LET cll1121,+@IF(@CELLPOINTER("col")+@COLS		
		(Group range ?)-1>26,@CHAR((@CELLPOINTER("col")+@COLS		
		(Group range ?)-@MOD(@CELLPOINTER("col")+@COLS		
		(Group range ?),26))/26+64),"")&@CHAR(@MOD(@CELLPOINTER		
		("col")+@COLS(Group range ?),26)+63)}~		
38	!	{LET cll121,+"$"&cll1121&"$"}~{LET loc3121,cll121&		
		@STRING(@VALUE(loc1121),0)}~		
39	!	{LET loc4121,loc2121&".."&loc3121}~{RECALC rec3121}		
		{RECALC rec4121}{IF counter121>5}{FORBREAK}~		
		{BRANCH ret121}		
40	rec3121	{PANELOFF}/GOD{RIGHT 3}~{BS}"		
41	rec4121	$A$60..$E$60		
42	!	{PANELON}{WINDOWSON}~{?}~{WINDOWSOFF}{PANELOFF}QQQ{DOWN}		

The [labels2121] routine is the same as the [labels2120] routine in GRAPHGRP.WK1 except that [counter1121] counts one less column then the number of columns that the comparable counter counts in the GRAPHGRP.WK1 macro, and the code in the B40 cell is `/GOD...` instead of the `/G`. The rest of the macro is a repetition of the code in the GRAPHGRP.WK1 macro.

To make it easier for you to key this macro into Lotus 1-2-3, we show the full list of all the cell contents and formulas.

```
A1: U [W14] '*---A macro to assign all graph data labels (A-F) in one step
A2: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W14] '    range names in this column (starts with the \Z macro name)
A4: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W14] '*---Expand and highlight the data range and press [ENTER]
A6: [W14] '*---When you see the prompt menu bar use the  cell pointer to
          highlight
A7: [W14] '    your choice and press [return] to make your choice
A8: U [W14] '         DO NOT USE THE CAPITAL LETTERS TO MAKE YOUR CHOICE!!
A9: [W14] '!
A10: U [W14] '\Z
B10: '{BREAKON}
A11: U [W14] 'GRPHDLBL
B11: '{MENUBRANCH menua121}
A12: [W14] 'cont1121
B12: '{WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RNDGroup range ?~
     /RNC{WINDOWSON}{PANELON}Group range ?~{BS}{BS}{?}~{WINDOWSOFF}
     {GOTO}Group range ?~
A13: [W14] 'cont121
B13: '{LET counter1121,0}
A14: [W14] '!
B14: '{PANELOFF}{FOR counter1121,0,@COLS(Group range ?),1,labels1121}
A15: [W14] '!
B15: '{GOTO}Group range ?~/RNDGroup range ?~
A16: [W14] '!
A17: [W14] 'counter1121
B17: 6
A18: [W14] 'counter1a121
B18: 421
A19: [W14] 'labels1121
B19: '{LET loc2121,@CELLPOINTER("address")}~{LET loc1121,@LEFT(loc2121,
     @FIND("$",loc2121,2)+1)}~{LET loc3121,loc1121&@STRING(@CELLPOINTER
     ("row")+@ROWS(Group range ?)-1,0)}~
A20: [W14] '!
```

```
B20: '{LET loc4121,loc2121&".."&loc3121}~{RECALC rec1121}{RECALC rec2121}
     {IF counter1121>5}{FORBREAK}~{BRANCH ret121}
A21: [W14] 'rec1121
B21: +"{PANELOFF}/GOD{RIGHT "&@STRING(B17,0)&"}~{BS}"
A22: [W14] 'rec2121
B22: +B49
A23: [W14] '!
B23: '{PANELON}{WINDOWSON}~{?}~{WINDOWSOFF}{PANELOFF}QQQ{RIGHT}
A24: [W14] '!
A25: [W14] 'menua121
B25: 'Columnwise
C25: 'Rowwizse
D25: 'Quit
A26: [W14] '!
B26: 'The 1st column is A and the rest are B-F
C26: 'The 1st row is A and the rest are B-F
D26: 'Quit the macro
A27: [W14] '!
B27: '{BRANCH cont1121}
C27: '{BRANCH cont1a121}
D27: '{BRANCH ret121}
A28: [W14] '!
A29: [W14] 'cont1a121
B29: '{WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RNDGroup range ?~
     /RNC{WINDOWSON}{PANELON}Group range ?~{BS}{BS}{?}~{WINDOWSOFF}
     {GOTO}Group range ?~
A30: [W14] 'conta121
B30: '{LET counter121,0}
A31: [W14] '!
B31: '{PANELOFF}{FOR counter121,0,@ROWS(Group range ?),1,labels2121}
A32: [W14] '!
B32: '{GOTO}Group range ?~/RNDGroup range ?~
A33: [W14] '!
A34: [W14] 'counter121
B34: 6
A35: [W14] 'countera121
B35: 421
A36: [W14] 'labels2121
B36: '{LET loc2121,@CELLPOINTER("address")}~{LET loc1121,@RIGHT(loc2121,@LENGTH(loc2121)
     -@FIND("$",loc2121,2)-1)}~
A37: [W14] '!
B37: '{LET cll1121,+@IF(@CELLPOINTER("col")+@COLS(Group range ?)-1>26
     ,@CHAR((@CELLPOINTER("col")+@COLS(Group range ?)-@MOD(@CELLPOINTER
     ("col")+@COLS(Group range ?),26))/26+64),"")&@CHAR(@MOD(@CELLPOINTER
     ("col")+@COLS(Group range ?),26)+63)}~
A38: [W14] '!
B38: '{LET cll121,+"$"&cll1121&"$"}~{LET loc3121,cll121&@STRING(@VALUE(loc1121),0)}~
A39: [W14] '!
B39: '{LET loc4121,loc2121&".."&loc3121}~{RECALC rec3121}{RECALC rec4121}
     {IF counter121>5}{FORBREAK}~{BRANCH ret121}
A40: [W14] 'rec3121
B40: +"{PANELOFF}/GOD{RIGHT "&@STRING(B34,0)&"}~{BS}"
A41: [W14] 'rec4121
B41: +B49
A42: [W14] '!
B42: '{PANELON}{WINDOWSON}~{?}~{WINDOWSOFF}{PANELOFF}QQQ{DOWN}
A43: [W14] '!
A44: [W14] 'cll121
B44: '$E$
A45: [W14] 'cll1121
B45: 'E
A46: [W14] 'loc1121
B46: '60
A47: [W14] 'loc2121
B47: '$A$60
A48: [W14] 'loc3121
B48: '$E$60
A49: [W14] 'loc4121
B49: '$A$60..$E$60
A50: [W14] 'ret121
```

[7] Assign Graph Legends to All the Graphs in One Step

```
     A                    B                    C                    D
 1 *---A macro to assign all graph legends (A-F) in one step
 2     Simulates the /Graph Option Legend Group in Lotus 2.2, the first
 3     column/row will become A and rest B-F.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 *---Expand and highlight the data range and press [ENTER]
 8 !
 9 !
10 \Z              {BREAKON}
11 GRPHLGND        {MENUBRANCH menua122}
12 cont1122        {WINDOWSOFF}{PANELOFF}/RNCLegends range ?~~/RND
                   Legends range ?~/RNC{WINDOWSON}{PANELON}Legends range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Legends range ?~
13 cont122         {LET counter1122,0}
14 !               {PANELOFF}{FOR counter1122,0,@ROWS(Legends range ?)-1,1,
                   labels1122}
15 !               {GOTO}Legends range ?~/RNDLegends range ?~
16 !
17 counter1122                    6
18 counter1a122                 421
19 labels1122      {LET loc2122,@CELLPOINTER("address")}~
20 !               {LET loc4122,+"\"&@CELLPOINTER("address")}~
                   {RECALC rec1122}{RECALC rec2122}{IF counter1122>5}
                   {FORBREAK}~{BRANCH ret122}
21 rec1122         {PANELOFF}/GOL{RIGHT 6}~
22 rec2122         \$G$53
23 !               {PANELON}{WINDOWSON}~{WINDOWSOFF}{PANELOFF}QQ{DOWN}
24 !
25 menua122        Columnwise          Rowwizse            Quit
26 !               The legends must beThe legends must be Quit the macro
27 !               {BRANCH cont1122}   {BRANCH cont1a122}  {BRANCH ret122}
28 !
29 cont1a122       {WINDOWSOFF}{PANELOFF}/RNCLegends range ?~~/RND
                   Legends range ?~/RNC{WINDOWSON}{PANELON}Legends range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Legends range ?~
30 conta122        {LET counter122,0}
31 !               {PANELOFF}{FOR counter122,0,@COLS(Legends range ?)-1,1,
                   labels2122}
32 !               {GOTO}Legends range ?~/RNDLegends range ?~
33 !
34 counter122                     6
35 countera122                  421
36 labels2122      {LET loc2122,@CELLPOINTER("address")}~
37 !               {LET loc4122,+"\"&@CELLPOINTER("address")}~
                   {RECALC rec3122}{RECALC rec4122}{IF counter122>5}
                   {FORBREAK}~{BRANCH ret122}
38 rec3122         {PANELOFF}/GOL{RIGHT 6}~*{ESC}
39 rec4122         \$G$53
40 !               {PANELON}{WINDOWSON}~{WINDOWSOFF}{PANELOFF}QQ{RIGHT}
41 !
42 cll122          $L$
43 cll1122         L
44 loc1122         53
45 loc2122         $G$53
46 loc3122         $L$53
47 loc4122         \$G$53
48 ret122
```

The macro uses a custom menu with three menu options. Because there is no way to display the custom menu code on this page without overlapping, we show every menu option's code separately.

```
Columnwise
```

```
The legends must be organized in a row the 1st for A, the rest
    for B-F
{BRANCH cont1122}

Rowwizse
The legends must be organized in a column, the 1st for A, the
    rest for B-F
{BRANCH cont1a122}

Quit
Quit the macro
BRANCH ret122}
```

Notice the code in the B21, B22, B40 and the B41 cells is the result of the following dynamic string formulas.

```
21 rec1122        +"{PANELOFF}/GOL{RIGHT "&@STRING(B17,0)&"}~"
22 rec2122        +B47
40 rec3122        +"{PANELOFF}/GOL{RIGHT "&@STRING(B34,0)&"}~*{ESC}"
41 rec4122        +B47
```

If you intend to key the code of this macro into Lotus 1-2-3, you have to key the code of the formulas and the code of the custom menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you, we show the full cell contents list at the end of this section.

The new releases of Lotus 1-2-3 (2.2 and up) include a new graph option, that is called / **G**raph **O**ptions **L**egend **R**ange. This option will automatically assign up to six data ranges simultaneously.

The text for the legends must be in a group of adjacent cells. The text in the first cell will become the legend for the A graph and so on. This macro allows the users of Lotus 2.0 and 2.01 to do the same. The macro starts with the `{MENUBRANCH menua122}` macro command which activates the custom menu in the B25 cell named [menua122].

The first menu option is the [**C**olumnwise] menu option:

```
Columnwise
The legends must be organized in a row the 1st for A, the rest
    for B-F
{BRANCH cont1122}
```

When you choose this option the macro issues `{BRANCH cont1122}` which routes the macro control to the [cont1122] routine.

	A	B	C	D
12	cont1122	{WINDOWSOFF}{PANELOFF}/RNCLegends range ?~~/RND Legends range ?~/RNC{WINDOWSON}{PANELON}Legends range ?~ {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Legends range ?~		
13	cont122	{LET counter1122,0}		
14	!	{PANELOFF}{FOR counter1122,0,@ROWS(Legends range ?)-1,1, labels1122}		
15	!	{GOTO}Legends range ?~/RNDLegends range ?~		

This routine starts with `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities and then uses the "safe technique" to assign the [Legends range ?] name to the selected range, and uses it as a prompt to paint the range of the legends. Next the macro issues `{LET counter1122,0}`, which sets the contents of the B17 cell named [counter1122] to zero, which serves as a counter for one of the `{FOR}` loops macro commands. The macro continues with

{PANELOFF} which freezes the panel display activity and issues {FOR counter1112,1,@COLS(Legends range ?),1,labels1122} which activates the [labels1122] routine as many times as the number of columns in the [Legends range ?] range.

	A	B	C	D
19	labels1122	{LET loc2122,@CELLPOINTER("address")}~		
20	!	{LET loc4122,+"\"&@CELLPOINTER("address")}~		
		{RECALC rec1122}{RECALC rec2122}{IF counter1122>5}		
		{FORBREAK}~{BRANCH ret122}		
21	rec1122	{PANELOFF}/GOL{RIGHT 6}~		
22	rec2122	\$G$53		
23	!	{PANELON}{WINDOWSON}~{WINDOWSOFF}{PANELOFF}QQ{DOWN}		

To assign the legends to the graphs, the macro needs to calculates the legend's addresses. First the macro issues {LET loc2122,@CELLPOINTER("address")}~, which calculate the cell pointer address of the first legend in the column and stores it in [loc2122] (the B47 cell). The macro continues with {LET loc4122,+"\"&@CELLPOINTER("address")}~, which store the current cell pointer address preceded by the back slash "\" in cell [loc4122]. Then the macro continues with {RECALC rec1122}{RECALC rec2122} which force Lotus to update the formulas in [rec1122] and [rec2122]. The results of the formulas in these cells become part of the code of the macro.

Now the macro issues {IF counter1120>5} to check if the value in [counter1122] is greater than the number five. If so, the macro understands that all the six legends were assigned to the graphs. Therefore the macro issues {FORBREAK} to break out of the {FOR} loop, and then issues {BRANCH ret122} which routes macro control to the empty [ret122] routine and quits. If the counter value is less than or equal to five, the macro continues with {PANELOFF} which freezes the panel display activity, while it continues with /GOL{RIGHT n}~ which assign the legends to the A-F graphs. The "n" number in the {RIGHT n} is dynamic. It comes from the counter value in [counter1122]. Let's look at the dynamic string formula in the B21 cell:

```
21 rec1122        +"{PANELOFF}/GOL{RIGHT "&@STRING(B17,0)&"}~"
```

the @STRING(B17,0) function returns the value of the counter in [counter1122] in a string format, which becomes part of the {RIGHT n} macro command. The number "n" instructs Lotus how many times to press the RIGHT key to choose the correct menu option for the current graph. When you press the /GOL keys in Lotus 1-2-3, the following menu appears and the highlight is on the [**A**] menu option:

```
A  B  C  D  E  F
```

To select the [**F**] menu option you need to press the RIGHT key five times, or once for the [**B**] graph or three times for the [**D**] graph. This is achieved by the dynamic formula that uses the counter value "n" in the B17 cell to build {RIGHT n} to move to the correct menu option. Next the macro specifies the legend by entering \ (backslash) followed by the address of the cell that contains the text for the legend, and then issues {PANELON}{WINDOWSON} to resume screen and panel display activities, followed by the tilde "~" which is the same as ENTER. Last the macro issues the "QQ" macro keys to return to the READY mode and then issues {DOWN} which moves the cell pointer to the next legend in the column.

The macro executes the [labels1122] routine as many times as the number of legends in the [Legends range ?] range, and when the {FOR} loop is finished, the macro routes control back to

the [cont1122] routine and issues `{GOTO}Legends range ?~/RNDLegends range ?~` which move the cell pointer to the upper cell of the legends range and delete the temporary [Legends range ?] range name to leave a clean worksheet. The second menu option is the [**R**owwizse] menu option:

```
Rowwizse
The legends must be organized in a column, the 1st for A, the
    rest for B-F
{BRANCH cont1a122}
```

When you choose this option, the macro issues `{BRANCH cont1a122}` which starts the [cont1a122] routine.

	A	B	C	D
29	cont1a122	{WINDOWSOFF}{PANELOFF}/RNCLegends range ?~~/RND Legends range ?~/RNC{WINDOWSON}{PANELON}Legends range ?~ {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Legends range ?~		
30	conta122	{LET counter122,0}		
31	!	{PANELOFF}{FOR counter122,0,@COLS(Legends range ?)-1,1, labels2122}		
32	!	{GOTO}Legends range ?~/RNDLegends range ?~		

This routine is the same as the [cont1122] routine for the previous menu option, except that the counter here is [counter122] and the `{FOR}` loop command executes the [labels2122] routine as many time as the number of rows in the data range named [Legends range ?] instead.

	A	B	C	D
36	labels2122	{LET loc2122,@CELLPOINTER("address")}~		
37	!	{LET loc4122,+"\"&@CELLPOINTER("address")}~ {RECALC rec3122}{RECALC rec4122}{IF counter122>5} {FORBREAK}~{BRANCH ret122}		
38	rec3122	{PANELOFF}/GOL{RIGHT 6}~*{ESC}		
39	rec4122	\$G$53		
40	!	{PANELON}{WINDOWSON}~{WINDOWSOFF}{PANELOFF}QQ{RIGHT}		

This routine works the same way as the [labels1122] routine and so does the rest of the code.

The last menu option is the [**Q**uit] menu option:

```
Quit
Quit the macro
{BRANCH ret120}
```

When you choose this option the macro issues `{BRANCH ret122}`, which routes the macro control to the empty [ret120] routine and quits. To make it easier you to key this macro into Lotus 1-2-3, we show the full list of cell contents and formulas.

```
A1: U [W15] '*---A macro to assign all graph legends (A-F) in one step
A2: U [W15] '    Simulates the /Graph Option Legend Group in Lotus 2.2,
           the first
A3: U [W15] '    column/row will become A and rest B-F.
A4: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A5: [W15] '    range names in this column (starts with the \Z macro name)
A6: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A7: [W15] '*---Expand and highlight the data range and press [ENTER]
A8: [W15] '!
A9: [W15] '!
A10: U [W15] '\Z
B10: [W19] '{BREAKON}
A11: U [W15] 'GRPHLGND
```

```
B11: [W19] '{MENUBRANCH menua122}
A12: [W15] 'cont1122
B12: [W19] '{WINDOWSOFF}{PANELOFF}/RNCLegends range ?~~/RNDLegends range ?~
           /RNC{WINDOWSON}{PANELON}Legends range ?~{BS}{BS}{?}~{WINDOWSOFF}
           {GOTO}Legends range ?~
A13: [W15] 'cont122
B13: [W19] '{LET counter1122,0}
A14: [W15] '!
B14: [W19] '{PANELOFF}{FOR counter1122,0,@ROWS(Legends range ?)-1,1,
           labels1122}
A15: [W15] '!
B15: [W19] '{GOTO}Legends range ?~/RNDLegends range ?~
A16: [W15] '!
A17: [W15] 'counter1122
B17: [W19] 6
A18: [W15] 'counter1a122
B18: [W19] 421
A19: [W15] 'labels1122
B19: [W19] '{LET loc2122,@CELLPOINTER("address")}~{LET loc1122,@LEFT
           (loc2122,@FIND("$"  ,loc2122,2)+1)}~{LET loc3122,loc1122&@STRING
           (@CELLPOINTER("row")+@ROWS(Legends range ?)-1,0)}~
A20: [W15] '!
B20: [W19] '{LET loc4122,+"\"&@CELLPOINTER("address")}~{RECALC rec1122}
            {RECALC rec2122}{IF counter1122>5}{FORBREAK}~{BRANCH ret122}
A21: [W15] 'rec1122
B21: U [W19] +"{PANELOFF}/GoL{RIGHT "&@STRING(B17,0)&"}~"
A22: [W15] 'rec2122
B22: U [W19] +B47
A23: [W15] '!
B23: [W19] '{PANELON}{WINDOWSON}~{WINDOWSOFF}{PANELOFF}qq{DOWN}
A24: [W15] '!
A25: [W15] 'menua122
B25: [W19] 'Columnwise
C25: [W20] 'Rowwizse
D25: [W15] 'Quit
A26: [W15] '!
B26: [W19] 'The legends must be organized in a row the 1st for A, the rest
           for B-F
C26: [W20] 'The legends must be organized in a column the 1st for A, the
           rest for B-F
D26: [W15] 'Quit the macro
A27: [W15] '!
B27: [W19] '{BRANCH cont1122}
C27: [W20] '{BRANCH cont1a122}
D27: [W15] '{BRANCH ret122}
A28: [W15] '!
A29: [W15] 'cont1a122
B29: [W19] '{WINDOWSOFF}{PANELOFF}/RNCLegends range ?~~/RNDLegends range ?~
           /RNC{WINDOWSON}{PANELON}Legends range ?~{BS}{BS}{?}~{WINDOWSOFF}
           {GOTO}Legends range ?~
A30: [W15] 'conta122
B30: [W19] '{LET counter122,0}
A31: [W15] '!
B31: [W19] '{PANELOFF}{FOR counter122,0,@COLS(Legends range ?)-1,1,
           labels2122}
A32: [W15] '!
B32: [W19] '{GOTO}Legends range ?~/RNDLegends range ?~
A33: [W15] '!
A34: [W15] 'counter122
B34: [W19] 6
A35: [W15] 'countera122
B35: [W19] 421
A36: [W15] 'labels2122
B36: [W19] '{LET loc2122,@CELLPOINTER("address")}~
A37: [W15] '!
B37: [W19] '{LET loc4122,+"\"&@CELLPOINTER("address")}~{RECALC rec3122}
           {RECALC rec4122}{IF counter122>5}{FORBREAK}~{BRANCH ret122}
A38: [W15] 'rec3122
B38: U [W19] +"{PANELOFF}/GoL{RIGHT "&@STRING(B34,0)&"}~*{ESC}"
A39: [W15] 'rec4122
B39: U [W19] +B47
```

```
A40: [W15] '!
B40: [W19] '{PANELON}{WINDOWSON}~{WINDOWSOFF}{PANELOFF}qq{RIGHT}
A41: [W15] '!
A42: [W15] 'cll122
B42: [W19] '$L$
A43: [W15] 'cll1122
B43: [W19] 'L
A44: [W15] 'loc1122
B44: [W19] '53
A45: [W15] 'loc2122
B45: [W19] '$G$53
A46: [W15] 'loc3122
B46: [W19] '$L$53
A47: [W15] 'loc4122
B47: [W19] '\$G$53
A48: [W15] 'ret122
```

[1] Create a Named Graph

```
     A                B                C                D                E
 1 *---A macro to CREATE a named graph
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 GRAPHMAK       /GNC{?}~Q
```

When you create graphs in Lotus 1-2-3 and want to save them as an active part of the worksheet, not just as PIC files which cannot be processed by the worksheet again, the only way to do it is to assign a name to the graph which can be viewed later or accessed through **/** `Graph` **N**`ame` **U**`se` commands. This macro saves a few keys every time a graph is created and named. The graph starts with the standard macro keys `/GNC`; then it issues the `{?}`, which allows you to write a name for the current graph. When you press ENTER, the macro issues the `~Q` macro keys to save the name and quits to the READY mode.

[3] Delete Named Graphs

```
     A                B                C                D                E
 1 *---A macro to DELETE named graphs, with full screen graphs list.
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Highlight the graph name and press the [ENTER] key
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 GRAPHDEL        {BREAKON}
12 loop019         /GND{NAME}{?}~{ESC 6}
13 !               Delete more graphs ? (Y/N) Y {GET key019}{ESC}~
14 !               {IF @UPPER(key019)="N"}{BRANCH ret019}
15 !               {BRANCH loop019}
16 !
17 key019          N
18 !
19 ret019
```

This macro allows you to easily delete named graphs. It first displays a full screen list of all the named graphs, then you highlight the graph name to delete and press ENTER. The macro starts with `/GND{NAME}{?}` which display a full screen list of all the named graphs and pause for you to highlight the name to delete. When you press ENTER, the macro issues the tilde "`~`" command, which is the same as the ENTER key, and deletes the graph name. Next the macro issues `{ESC 6}` to return to the READY mode. To allow you to delete more graph names, the macro writes "`Delete more graphs ? (Y/N) Y `" directly to the panel, and then issues `{GET key019}` which halts the macro execution and waits for you to read the prompt message and to press a key. Then the macro issues `{ESC}` which clears the message from the panel before Lotus writes it into the current cell.

Next the macro issues `{IF @UPPER(key019)="N"}`, which checks if you pressed one of the "`N`" or the "`n`" keys. If so, the macro issues `{BRANCH ret019}`, which routes the macro control to the empty routine named [ret019] and quits. Otherwise the macro issues `{BRANCH loop019}` which routes the macro control back to the beginning to allow you to delete more graph names.

[1] Use a Named Graph

```
     A              B              C              D              E
 1 *---A macro to make a named graph the current graph
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Highlight the graph name and press the [ENTER] key
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 GRAPHUSE       /GNU{NAME}{?}~Q
```

This macro simplifies the use of named graphs. It displays a full screen list of all the named graphs in the worksheet, then you highlight the graph name to use and press ENTER. The macro starts with the `/GNU` macro keys and issues the `{NAME}` command, which displays a full screen list of all the named graphs in the worksheet. Then the macro follows with `{?}` which halts macro execution while you move the highlight to the name of the graph and press ENTER after which, Lotus displays the graph. When you press any key, the macro issues the `~Q` macro keys to quit to the READY mode. After every `{?}`, the macro should include the tilde "`~`" command because the ENTER key that you just pressed is not enough, it just tells Lotus that the `{?}` command is finished, the tilde is the actual ENTER key which the macro presses.

[0] Reset All the Named Graphs

```
      A                B                C                D                E
 1 *---A macro to DELETE (RESET) all named graphs.
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 GRAPHRST       /GNRQ
```

This is a simple macro which saves three keys using standard macro keys to reset all the named graphs in the worksheet (/ **G**raph **N**ame **R**eset **Q**uit).

[4] Create a Slide Show

```
     A                B                C                D                E
 1 *---A macro to show graphics slide show, the macro includes 5 sample
 2     graphs for demonstration. When you are finished you can reset all
 3     graphs and erase the range of numbers in the next page.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 SLIDESHO        {WINDOWSOFF}{PANELOFF}{LET counter163,-1}~
                   {GETNUMBER "Number of graphs to show in order: ",
                   graphs163}~
12 loopa163        {LET counter163,counter163+1}~{IF counter163<graphs163}
                   {show163}
13 !
14 !
15 graphs163             10
16 show163         /GNU{RIGHT counter163}~Q{BRANCH loopa163}
17 !
18 counter163            10
```

This macro presents a simple slide show from a list of pre-named graphs that you have created and prepared. Let's assume that the worksheet contains 10 graph names. When you start the macro the macro, it uses the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and panel activities, and issues `{LET counter163,-1}~` to set the value in the cell named [counter163] to "-1". Cell [counter163] is uses as a counter for the number of slides. Next the macro issues `{GETNUMBER "Number of graphs to show in order: ",graphs163}` macro code that displays the "`Number of graphs to show in order:` " prompt in the panel. Now insert the number of slides/graphs to display and press ENTER, the number of slides is stored in[graphs163].

The macro begins a loop displaying the number of slides you requested. It issues `{LET counter163,counter163+1}~` to increase the value in [counter163] by one, and uses `{IF counter163<graphs163}` to check that the counter is less than the value stored in [graphs163]. If the counter is still less than the value stored in [graphs163] the macro issues the `{show163}` routine command that activates the [show163] routine.

The [show163] routine starts with the `/GNU` macro keys that display a list of all the graph names in the panel. The macro issues `{RIGHT counter163}`, which moves the highlight to the right as many times as the value of the counter. For example, when the counter is 3, the macro displays the third graph on the list. When you press any key, the macro issues the "`Q`" macro key that returns the worksheet to READY mode, and issues the `{show163}` routine command again. This loop will continue as long as the counter is less than the value stored in the cell named [graphs163].

[5] Graph Group Data Macro

```
     A                B                C                D                E
 1 *---A macro to assign all graph data ranges (X and A-F) simultaneously.
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Expand and highlight the Data range ?and press [ENTER]
 6 !
 7                 THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 GROUPGRA        {WINDOWSOFF}{PANELOFF}/RNCData range ?~~/RND
                   Data range ?~/RNC{WINDOWSON}{PANELON}Data range ?~{BS}
                   {BS}{?}~{WINDOWSOFF}{GOTO}Data range ?~
12 !               {MENUBRANCH menu602}
13 !
14 menu602         Columnwise    Rowwise
15 !               Use columns asUse rows as data ranges
16 !               /GGData range ?GGData range ?~{RIGHT}~QQ{ESC 6}
17 !               /RNDData range/RNDData range ?~
18 !               /GVQ          /GVQ
```

This is the code of the custom menu called [menu602]

```
Columnwise
Use columns as data ranges
/GGData range ?~~QQ{ESC 6}
/RNDData range ?~
/GVQ

Rowwise
Use rows as data ranges
/GGData range ?~{RIGHT}~QQ{ESC 6}
/RNDData range ?~
/GVQ
```

The macro uses a custom menu that cannot be printed in full on one page, therefore we show here every menu item separately. The code as it appears in the main listing is not complete because the menu items overlap.

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands that suppress the screen and panel display activity. Then the macro uses the "safe technique" to paint the data range to graph, and simultaneously assigns the [Data range ?] name to the same range, which it uses as a prompt. When the macro finishes, it issues `{GOTO}Data range ?~` that places the cell pointer on the top left cell of the data range, and `{MENUBRANCH menu602}` to activate the [menu602] custom menu.

The [menu602] custom menu duplicates the standard Lotus menu, and makes you believe you are using the Lotus menu, but you are really using a custom menu. The first menu option is the [**C**olumnwise] menu option

```
Columnwise
Use columns as data ranges
/GGData range ?~~QQ{ESC 6}
/RNDData range ?~
/GVQ
```

The macro issues `/GGData range ?~~QQ{ESC 6}` to assign the range named [Data range ?]

as a graph group and then returns to the READY mode. Next the macro issues `/RNDData range ?~` that deletes the [Data range ?] range name and issues `/GV` macro keys to display the graph. When you press any key, the macro issues the "`Q`" macro key and quits. The second menu option is the [**R**owwise] menu option:

```
Rowwise
Use rows as data ranges
/GGData range ?~{RIGHT}~QQ{ESC 6}
/RNDData range ?~
/GVQ
```

Here the macro uses almost the same code as in the previous menu option except that the `{RIGHT}` macro command points to the Lotus [rowwise] menu option.

[5] Graph Group Labels Macro

```
     A                B                C                D                E
 1 *---A macro to assign all graph data labels (for A-F) simultaneously.
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Expand and highlight the Data range ?and press [ENTER]
 6 !
 7                THIS MACRO WORKS IN LOTUS 2.2 AND UP
 8 !
 9 !
10 \Z             {BREAKON}
11 GROUPLBL       {WINDOWSOFF}{PANELOFF}/RNCData range ?~~/RND
                  Data range ?~/RNC{WINDOWSON}{PANELON}Data range ?~{BS}
                  {BS}{?}~{WINDOWSOFF}{GOTO}Data range ?~
12 !              {MENUBRANCH menu603}
13 !
14 menu603        Columnwise    Rowwise
15 !              Use columns asUse rows as data ranges
16 !              {MENUBRANCH me{MENUBRANCH menua603}
17 !
18 menua603       Center        Left        Above       Right      Below
19 !              Place label onPlace label Place label Place labelPlace
20 !              /GODGdata rang/GODGData ra/GODGData ra/GODGData r/GODG
21 !              {GOTO}Data ranGOTO}Data ra{GOTO}Data r{GOTO}Data {GOTO}
22 !              /GVQ          /GVQ        /GVQ        /GVQ       /GVQ
```

This is the code of the custom menu called [menu603]

```
Columnwise
Use columns as data ranges
{MENUBRANCH menua603}

Rowwise
Use rows as data ranges
{MENUBRANCH menua603}
```

This is the code of the custom menu called [menua603]

```
Center
Place label on data point
/GODGData range ?~~~QQ{ESC 6}
{GOTO}Data range ?~/RNDData range ?~
/GVQ

Left
Place label to the left of data point
/GODGData range ?~~{RIGHT}~QQ{ESC 6}
{GOTO}Data range ?~/RNDData range ?~
/GVQ

Above
Place label above data point
/GODGData range ?~~{R 2}~QQ{ESC 6}
{GOTO}Data range ?~/RNDData range ?~
/GVQ

Right
Place label to the left of data point
/GODGData range ?~~{R 3}~QQ{ESC 6}
{GOTO}Data range ?~/RNDData range ?~
/GVQ

Below
Place label below data point
/GODGData range ?~~{R 4}~QQ{ESC 6}
```

```
{GOTO}Data range ?~/RNDData range ?~
/GVQ
```

The macro uses two custom menus that cannot be printed in full on one page; therefore we show every menu item separately. The code as it appears in the main listing is not complete because the menu items overlap. For this macro to work, a graph must already be defined before the data labels can be assigned.

The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands that suppress the screen and panel display activity. Then the macro uses the "safe technique" to paint the data range to graph, and simultaneously assigns the [Data range ?] name to the same range, which it uses as a prompt. When the macro finishes, it issues {GOTO}Data range ?~ that places the cell pointer on the top left cell of the data range and uses {MENUBRANCH menu603} to activate the [menu603] custom menu.

The [menu603] custom menu duplicates[5] the standard Lotus menu, and makes you believe that you are using it, but you are really using a custom menu. The first menu option in the [menu603] custom menu is the [**C**olumnwise] menu option:

```
Columnwise
Use columns as data ranges
{MENUBRANCH menua603}
```

The macro continues with {MENUBRANCH menua603} that activates the second custom menu named [menua603], which also duplicates the standard Lotus menu, but is a custom menu. The first menu option in the [menua603] custom menu is the [**C**enter] menu option:

```
Center
Place label on data point
/GODGData range ?~~~QQ{ESC 6}
{GOTO}Data range ?~/RNDData range ?~
/GVQ
```

The macro issues /GODGData range ?~~QQ{ESC 6} that assign the range named [Data range ?] as a data labels group and then returns to the READY mode. Next the macro issues /RNDData range ?~ that deletes the [Data range ?] range name and issues the /GV macro keys to display the graph with the data labels. When you press any key, the macro issues the "Q" macro key and quits. The second menu option is the [**L**eft] menu option:

```
Left
Place label to the left of data point
/GODGData range ?~~{RIGHT}~QQ{ESC 6}
{GOTO}Data range ?~/RNDData range ?~
/GVQ
```

Here and in the rest of the options in this custom menu, the macro uses almost the same code as in the previous menu option except that here the macro uses the {RIGHT} or the {R} macro commands to point to the other Lotus menu options.

[4] Graph Group Legends Macro

```
     A          B          C          D          E
 1 *---A macro to assign all graph legends (A-F) simultaneously.
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Expand and highlight the Data range ?and press [ENTER]
 6 !
 7 !
 8                THIS MACRO WORKS IN LOTUS 2.2 AND UP
 9 !
10 \Z             {BREAKON}
11 GROUPLGD       {WINDOWSOFF}{PANELOFF}/RNCGroup range ?~~/RND
                  Group range ?~
                  /RNC{WINDOWSON}{PANELON}Group range ?~{BS}{BS}{?}~
                  {WINDOWSOFF}{GOTO}group range ?~
12 !              /GOLRGroup range ?~QQ
13 !              /RNDGroup range ?~
```

This macro automatically assigns a range of legends to a pre-defined set of graphs. The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands that suppress the screen and panel display activity. Then the macro uses the "safe technique" to paint the range of legend labels, and simultaneously assigns the [Group range ?] range name to the same range, which it uses as a prompt. When the macro finishes, it issues `{GOTO}Group range ?` that places the cell pointer on the top cell of the legend labels range. Now the macro issues `/GOLRGroup range ?~QQ` that assign the legend labels in the [Group range ?] range to the graphs and then uses `/RNDGroup range ?~` to delete the [Group range ?] range name to leave the worksheet as it was before you activated the macro.

This macro does not bring anything new that could not be done from the keyboard. The only difference is that you define the range of legend labels first, and the macro displays the "`Group range ?`" extra prompt.

[0] Create Graph Names Table

	A	B	C	D	E
1	*---A macro to create graph names table				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer where you want the names table, the table				
5	occupies 3 columns and as many rows as necessary				
6	*---Hold the [ALT] key and press [Z] to activate the macro				
7	!				
8		THIS MACRO WORKS IN LOTUS 2.2 AND UP			
9	!				
10	\Z	{BREAKON}			
11	GRAPHTBL	/GNT~Q			

This is a simple six key macro which creates graph names table. The macro issues the **/ G**raph **N**ame **T**able standard menu options.

[5] Display the ASCII Code of a Character

```
     A              B                C              D              E
 1 *---A macro to display the ASCII code for a given character
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 ASCIICOD        {GETLABEL "Enter character : ",char201}~
12 !               {LET cod201,@STRING(@CODE(char201),0)}~
                   {RECALC prompt201}
13 prompt201       The ASCII code for "E" is 69, Press any key to quit.
                   {GET key201}{ESC}
14 !
15 char201         E
16 !
17 cod201          69
18 !
19 key201          l
```

Notice that the code in the B13 cell is the result of a dynamic string formula, therefore if you intend to key this macro into Lotus 1-2-3, you have to key the formula as it appears here, NOT the code as it appears in the main listing.

```
13 prompt201       +"The ASCII code for """&B15&""" is "&B17&", Press any
                   key to quit.{GET key201}{ESC}"
```

This macro displays the ASCII code number for a given character. The macro starts with the `{GETLABEL "Enter character : ",char201}~` macro command which displays the "`Enter character :` " prompt message in the panel and waits for you to type the character. When you type the character and press ENTER, Lotus saves the character in the B15 cell named [char201] and issues `{LET cod201,@STRING(@CODE(char201),0)}` which stores the ASCII code of the character in [cod201].

Before the macro continues, it issues `{RECALC prompt201}` which recalculates the dynamic string formula in the B13 cell named [prompt201]. The result of the formula in the B13 cell serves as part of the macro code, which is one way of creating dynamic code in the Lotus macro language. The formula in the B13 cell is:

```
13 prompt201       +"The ASCII code for """&B15&""" is "&B17&", Press any
                   key to quit. {GET key201}{ESC}"
```

We can see that this formula contains two dynamic parts. The first is the content of the B15 cell named [char201] and the second is the content of the B17 cell named [cod201]. Notice how a double quote is entered in the formula that must create a text which also includes double quotes itself. The result of this formula in the case of the "E" character is:

```
13 prompt201       The ASCII code for "E" is 69, Press any key to quit.
                   {GET key201}{ESC}
```

When the macro processes the code in the B13 cell, it first writes the

```
The ASCII code for "E" is 69, Press any key to quit.
```

message into the panel, and then issues `{GET key201}` which halts the macro execution and waits until you press a key. Then the macro stores the key code in the B19 cell named [key201] and issues `{ESC}`, which clears the message from the panel before Lotus writes it into the current cell. This macro is a fine example of how to create and use a dynamic code string formula combined with the direct writing of the formula's result into the panel to create a sophisticated message for you.

Mathematical Macros

[6] Cross Multiply Two Ranges

```
     A              B                C              D              E
 1 *---A macro to cross multiply (sum products) of two ranges
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer where you want the multiplication result
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 CROSMULT       {BREAKON}
12 cont006        {LET here1006,@CELLPOINTER("address")}~{WINDOWSOFF}
                  {PANELOFF}/RNCFirst range ?~~/RNDFirst range ?~/RNC
                  {PANELON}First range ?{WINDOWSON}~{BS}{BS}{?}~
13 !              {WINDOWSOFF}{PANELOFF}/RNCSecond range ?~~/RND
                  Second range ?~/RNC{PANELON}Second range ?{WINDOWSON}~
                  {BS}{BS}{?}~
14 !              {WINDOWSOFF}{PANELOFF}{IF (@ROWS(First range ?)<>@ROWS
                  (Second range ?))#OR#(@COLS(First range ?)<>@COLS
                  (Second range ?))}{BEEP}{PANELON}{message006}
                  {GET temp006}{ESC}{PANELOFF}{BRANCH cont006}
15 !              {WINDOWSOFF}{PANELOFF}{insert006}{RECALC here006}
16 here006        {LET $D$63,result006}~
17 !              /RNDFirst range ?~/RNDSecond range ?~
18 !
19 message006     The sizes of the two columns are not equal! Press any
                  key to try again.
20 !
21 temp006
22 !
23 insert006      {LET cols006,@COLS(First range ?)}~{LET rows006,@ROWS
                  (First range ?)}~{LET result006,0}~{RECALC loop006}
                  {RECALC loop1006}
24 loop006        {FOR counter1006,0,1,1,loop1006}
25 loop1006       {FOR counter2006,0,2,1,loop2006}
26 !
27 loop2006       {LET result006,+result006+@INDEX(First range ?,
                  counter1006,counter2006)*@INDEX(Second range ?,
                  counter1006,counter2006)}
28 !
29 cols006                 2
30 rows006                 3
31 counter1006             0
32 counter2006
33 result006             190
34 !
35 here1006       $D$6
```

While writing ***Super Power***, I have decided to expand the CROSMLT2.WK1 macro to multiple column ranges and replace it with this macro from the SUPER MACRO LIBRARY.
In this macro we cannot use the Matrix mathematics of Lotus 1-2-3, instead we will make use of the `@INDEX` function. This macro is more general and can also sum products of multi column ranges, however it is slower than the CROSMLT2.WK1 macro when summing products of one column ranges. Place the cell pointer where you want the multiplication result and activate the macro. The macro prompts you to paint the first and second ranges and calculates the result. Notice that this macro uses formulas to create dynamic code. Therefore the code in the B16, B24 and the B26 cell is the result of the following formulas:

```
16 here006         +"{LET "&D6&",result006}~"
24 loop006         +"{FOR counter1006,0,"&@STRING(B29-2,0)&",1,loop1006}"
25 loop1006        +"{FOR counter2006,0,"&@STRING(B30-1,0)&",1,loop2006}"
```

If you intend to key the macro into Lotus 1-2-3, you must key these formulas, NOT the code as it appears in the main listing.

The macro starts with the {LET here1006,@CELLPOINTER("address")}~ macro commands which store the current cell pointer address in the cell named [here1006], where the macro will put the multiplication result. Next the macro issues {WINDOWSOFF} {PANELOFF} which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the first range, and simultaneously assigns the [First range ?] name to the same range, which it uses as a prompt. Next the macro repeats and prompts you to paint the second range, and simultaneously assigns the [Second range ?] name to the same range, which it also uses as a prompt. Now the macro issues:

```
{IF (@ROWS(First range ?)<>@ROWS(Second range ?))#OR#
    (@COLS(First range ?)<>@COLS(Second range  ?))}
```

which checks if the two ranges are not the same size. If so, the macro issues a {BEEP} and {PANELON} to warn you and to free the panel for the warning message. Now the macro issues the {message006} routine command, which contains only text without any Lotus commands. therefore Lotus writes the text of the [message006] routine into the panel which displays:

```
The sizes of the two columns are not equal! Press any key to try again.
```

To hold the message in the panel so you can read it, the macro issues {GET temp006} which stops macro execution and waits for you to press a key. Then the macro resumes control and Lotus stores the key in cell [temp006] and immediately follows with {ESC}, which clears the message from the panel before Lotus writes it into the current cell. Next the macro issues {PANELOFF} to freeze the panel activity and issues {BRANCH cont1006}, which routes macro control back to the beginning so you can start again.

If the two ranges have the same size, the macro issues {insert006} which activates the [insert006] routine:

```
     A              B                C              D              E
23 insert006      {LET cols006,@COLS(First range ?)}~{LET rows006,@ROWS
                  (First range ?)}~{LET result006,0}~{RECALC loop006}
                  {RECALC loop1006}
24 loop006        {FOR counter1006,0,1,1,loop1006}
25 loop1006       {FOR counter2006,0,2,1,loop2006}
26 !
27 loop2006       {LET result006,+result006+@INDEX(First range ?,
                  counter1006,counter2006)*@INDEX(Second range ?,
                  counter1006,counter2006)}
```

The routine starts with {LET cols006,@COLS(First range ?)}~, which store the number of columns of the first range in cell [cols006], and then issues {LET rows006, @ROWS(First range ?)}~, which store the number of rows of the first range in cell [rows006]. Next the macro issues {LET result006,0}~ which reset the value in cell [result006] to zero, which serves as a summation register of all the cell products between the two ranges. Now the macro issues {RECALC loop006}{RECALC loop1006} which update the string formula in [loop006] and [loop1006].

```
24 loop006         +"{FOR counter1006,0,"&@STRING(B29-2,0)&",1,loop1006}"
25 loop1006        +"{FOR counter2006,0,"&@STRING(B30-1,0)&",1,loop2006}"
```

These two formulas make use of the number of rows and the number of columns stored in the B29 cell and the B30 cell named [cols006] and [rows006] respectively. If we would like to cross multiply the A1..B3 range by the D1..E3 range as shown here:

	A	B	C	D	E
1	0	3	9	12	
2	1	4	10	13	
3	2	5	11	14	
4					
5					
6	190				
7					

the two formulas will show:

```
24 loop006          {FOR counter1006,0,0,1,loop1006}
25 loop1006         {FOR counter2006,0,2,1,loop2006}
```

The first `{FOR}` loop activates the [loop1006] routine which contains the second `{FOR}` loop, as many times as the number of the columns in each range. The second `{FOR}` loop activates the [loop2006] routine as many times as the number of rows in each range.

	A	B	C	D	E
27	loop2006	{LET result006,+result006+@INDEX(First range ?, counter1006,counter2006)*@INDEX(Second range ?, counter1006,counter2006)}			

The `{loop2006}` routine, makes use of the `@INDEX` function to multiply all the cells in the two ranges. The result of every multiplication is added to the B33 cell named [result006].

	A	B	C	D	E
15	!	{WINDOWSOFF}{PANELOFF}{insert006}{RECALC here006}			
16	here006	{LET $D$6,result006}~			
17	!	/RNDFirst range ?~/RNDSecond range ?~			

When the `{insert006}` routine is finished, the macro issues `{RECALC here006}` which updates the dynamic formula in cell [here006].

```
16 here006          +"{LET "&D35&",result006}~"
```

This formula makes use of the address stored in the D35 cell named [here1006] to put the multiplication result in the place where you chose (which is the D6 cell in our example). The `{LET $D$6,result006}~` macro command copies the multiplication result from the cell named [result006] to the D6 cell. When the macro is finished, it issues `/RNDFirst range ?~/RNDSecond range ?~` to leave a clean worksheet.

[6] Cross Multiply Two Columns

```
     A              B                C              D              E
 1 *---A macro to cross multiply two columns (the first column can be
 2     the quantities, and the second can be the prices).
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Place the cell pointer where you want the multiplication result
 6     should be above the column ranges
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8 !
 9 !
10 \Z             {BREAKON}
11 CROSMLT2       {BREAKON}
12 cont1006       {WINDOWSOFF}{PANELOFF}/RNCFirst range ?~~/RND
                  First range ?~/RNC{PANELON}First range ?{WINDOWSON}~{BS}
                  {?}~
13 !              {WINDOWSOFF}{PANELOFF}/RNCSecond range ?~~/RND
                  Second range ?~/RNC{PANELON}Second range ?{WINDOWSON}~
                  {BS}{?}~
14 !              {WINDOWSOFF}{PANELOFF}{IF (@ROWS(First range ?)<>@ROWS
                  (Second range ?))#OR#(@COLS(First range ?)<>@COLS
                  (Second range  ?))}{BEEP}{PANELON}{message006}
                  {GET temp006}{ESC}{PANELOFF}{BRANCH cont1006}
15 !              /WIR{UP}~/RTsecond range ?~~
16 !              /DMM{BS}.{END}{RIGHT}~First range ?~{BS}{DOWN 2}~/WDR
                  {DOWN}~
17 !              /RNDFirst range ?~/RNDSecond range ?~
18 !
19 message006     The sizes of the two columns are not equal! Press any
                  key to try again.
20 !
21 temp006        1
```

This macro fills the gap in Lotus 2/2.01/2.2/2.3/2.4 for a missing CROSS MULTIPLY function like the `@SUMPRODUCT` in release 3/3.1/3.1+/123W, which sums the products of two column cells. The program uses the Data Matrix built-in options in Lotus 1-2-3 but is limited to the sum products of two columns. For a more general macro which can sum two general ranges with more than one column each, see the previous CROSMULT.WK1 macro. This macro was replaced in the SUPER MACRO LIBRARY by the CROSMULT.WK1 macro during the writing process of ***Super Power***.

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the first range, and simultaneously assigns the [First range ?] range name to the same range, which acts as a prompt to you. Next the macro repeats and prompts you to paint the second range, and simultaneously assigns the [Second range ?] range name to the same range, which also acts as a prompt to you. Now the macro issues:

```
{IF (@ROWS(First range ?)<>@ROWS(Second range ?))#OR#
    (@COLS(First range ?)<>@COLS(Second range  ?))}
```

which checks if the two ranges are not the same size. If so, the macro issues a `{BEEP}` and `{PANELON}` to warn you and to free the panel for the warning message. Now the macro issues the `{message006}` routine command, which contains only text without any Lotus commands. Therefore Lotus writes the text of the [message006] routine into the panel which displays:

```
The sizes of the two columns are not equal! Press any key to try again.
```

To hold the message in the panel so you can read it, the macro issues {GET temp006} which stops the macro execution and waits for you to press a key. Then the macro resumes control and Lotus stores the key in cell [temp006] and immediately follows with {ESC}, which clears the message from the panel before Lotus writes it into the current cell. Next the macro issues {PANELOFF} to freeze the panel activity and issues {BRANCH cont1006} which routes the macro control back to the beginning so you can start again.

	A	B	C	D	E
15	!	/WIR{UP}~/RTsecond range ?~~			
16	!	/DMM{BS}.{END}{RIGHT}~First range ?~{BS}{DOWN 2}~/WDR {DOWN}~			
17	!	/RNDFirst range ?~/RNDSecond range ?~			

If the ranges have the same size, the macro issues /WIR{UP}~/RTsecond range ?~~ which insert two empty rows above the first column, and then transpose the second column and insert it in the new row. Now we can use Matrix mathematics to multiply the first column by the new row (the transposed second column). Therefore the macro issues /DMM{BS}. {END} {RIGHT}~First range ?~ which multiplies the transposed row by the [First range ?] range. When Lotus prompts for the output range, the macro issues {BS}{DOWN 2}~, placing the result of the multiplication two rows below the transposed row. Next the macro issues /WDR{DOWN}~ to erase the two inserted rows and finally /RNDFirst range ?~/RNDSecond range ?~, which delete the [First range ?] and the [Second range ?] temporary range names to leave a clean worksheet.

[6] Add to Left Column

```
*---A macro to add the column's values to the left column's values. The
    macro assumes two columns of values and adds the right one to the
    left one. Can be used for year to date calculations.
*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
    range names in this column (starts with the \Z macro name)
*---Place the cell pointer on the top cell of the right column
*---Hold the [ALT] key and press [Z] to activate the macro
*---The macro adds the right values to the left and erases the right.
!
\Z              {BREAKON}
ADDLEFT         {BREAKON}
loopa200        Move to top cell of right column and press RETURN
                {GET key200}{ESC}
!               {IF KEY200="{RIGHT}"#OR#key200="{LEFT}"#OR#key200="
                {DOWN}"#OR#key200="{UP}"#OR#key200="{PGUP}"#OR#"{PGDN}"}
                {key200}{BRANCH loopa200}
!               {BRANCH loop200}
!
loop200         {PANELOFF}{LEFT}{IF @CELLPOINTER("type")="b"}{UP}{END}
                {UP}{BRANCH ret200}
!               {EDIT}+{EDIT}{RIGHT}~{EDIT}{CALC}{DOWN}{RIGHT}{UP}/RE~
                {DOWN}
!               {BRANCH loop200}
!
ret200
!
key200          ~
```

This interesting macro adds one column's values to the column to the left of it and clears the column. For example if we have the following two columns of data:

	A	B	C	D	E
1	100	250			
2	200	100			
3	300	300			
4	450	200			
5	600	100			
6					

after using the macro the result will be:

	A	B	C	D	E
1	350				
2	300				
3	600				
4	650				
5	700				
6					

This macro can be used when you need to collect data and only the total interests you. The macro directly writes:

```
Move to top cell of right column and press RETURN
```

into the panel, and issues the `{GET key200}` macro command, which halts the macro execution, and waits for you to use the direction keys, and press ENTER. When you press a key, Lotus stores the key in the cell named [key200], and the macro immediately follows with `{ESC}` which clears the message from the panel before Lotus writes it into the current cell. Next the macro issues:

```
{IF KEY200="{RIGHT}"#OR#key200="{LEFT}"#OR#key200="{DOWN}"#OR#
     key200="{UP}"#OR#key200="{PGUP}"#OR#"{PGDN}"}
```

to checks if you pressed one of the direction keys listed: RIGHT, LEFT, DOWN, UP, PGUP or PGDN. If so, the macro issues the `{ke200}` routine command which starts the [key200] routine. The [key200] routine contains the key that you just pressed; therefore the [key200] routine executes the key as you meant to. For example let's assume that you pressed the DOWN key, which Lotus stores in [key200]. Therefore when the macro issues the `{key200}` routine command, the cell pointer moves one cell down exactly as you wanted when you pressed the DOWN key, but not before the macro monitors it. Then the macro issues `{BRANCH loopa200}`, which routes the macro execution back to the beginning allowing you to continue to use the direction keys to move the cell pointer to the first cell of the column to add (the right column).

If the condition is not true and you pressed any other key, the macro understands that the cell pointer is in the right place and begins to add the values in the column to the column to the left. To add the values to the left column, the macro issues `{BRANCH loop200}` which starts the [loop200] routine.

	A	B	C	D	E
16	loop200	{PANELOFF}{LEFT}{IF @CELLPOINTER("type")="b"}{UP}{END}{UP}{BRANCH ret200}			
17	!	{EDIT}+{EDIT}{RIGHT}~{EDIT}{CALC}{DOWN}{RIGHT}{UP}/RE~{DOWN}			
18	!	{BRANCH loop200}			

The routine starts with `{PANELOFF}` which freezes the panel display activity while the macro issues `{LEFT}` to move the cell pointer to the left and then `{IF @CELLPOINTER("type")="b"}` to check if the cell is empty. If so, the macro understands that it finished adding all the numbers and reached the end of the data in the left column. Therefore it issues `{UP}{END}{UP}` to move the cell pointer to the upper cell of the column and then `{BRANCH ret200}`, which routes the macro control to the [ret200] empty routine and quits. If the cell in the left column is not empty the macro issues `{EDIT}+{EDIT}{RIGHT}~{EDIT}{CALC}{DOWN}{RIGHT}{UP}/RE~{DOWN}` to add the value in the right cell and then erase it. To understand these commands let's look at an example:

	A	B	C	D	E
1	100	250			
2	200	100			
3	300	300			
4	450	200			
5	600	100			
6					

Let's assume that the cell pointer is now located on the A1 cell. After the `{EDIT}` the panel displays:

```
100
```

Then the macro writes the "+" character and the panel displays:

```
100+
```

Then the macro issues `{EDIT}` to enter to POINT mode and then `{RIGHT}` to move the cell

pointer to the B1 cell. Now the panel displays:

```
100+B2
```

To write the content of the panel into the A1 cell, the macro issues the tilde "~". To change the formula in the A1 cell to value, the macro issues {EDIT} followed by {CALC}. Now the panel displays:

```
350
```

To write the result to the A1 cell, the macro issues {DOWN} which moves the cell pointer to the A2 cell and simultaneously writes the content of the panel into the A1 cell. Now the worksheet displays:

	A	B	C	D	E
1	350	250			
2	200	100			
3	300	300			
4	450	200			
5	600	100			
6					

Next the macro issues {RIGHT}{UP} to move to the B1 cell and then erase the B1 cell using /RE~. Last, the macro issues {DOWN} and moves the cell pointer to the B2 cell ready to add the B2 cell to the A2 cell. The worksheet now displays:

	A	B	C	D	E
1	350				
2	200	100			
3	300	300			
4	450	200			
5	600	100			
6					

To continue the addition of the other cells in the B2..B5 range the macro issues {BRANCH loop200}. The process continues until the cell pointer reaches the A6 cell which is empty. For this macro to work, the data in the "A" column of the example must be contiguous. Otherwise the macro stops when it reaches an empty cell. While writing ***Super Power*** I decided to improve the macro and remove this limit. The new macro code is:

	A	B	C	D	E
1	*---A macro to add the column's values to the left column's values. The				
2	macro assumes two columns of values and adds the right one to the				
3	left one. Can be used for year to date calculations.				
4	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
5	range names in this column (starts with the \Z macro name)				
6	*---Place the cell pointer on the top cell of the right column (optional)				
7	*---Hold the [ALT] key and press [Z] to activate the macro				
8	*---The macro adds the right values to the left and erases the right.				
9	!				
10	\Z	{BREAKON}			
11	ADDLEFT	{WINDOWSOFF}{PANELOFF}/RNCPaint column:~~/RND Paint column:~/RNC{PANELON}Paint column:~{BS}{BS} {WINDOWSON}{?}~{LEFT}{LET rows200,@ROWS(Paint column:)}~ {WINDOWSOFF}			
12	!	{FOR counter200,1,rows200,1,loop200}~			
13	!	/RNDPaint column:~			
14	!				
15	loop200	{EDIT}+{DOWN}{UP}{RIGHT}~{EDIT}{CALC}{DOWN}{RIGHT}{UP} /RE~{DOWN}{LEFT}{RETURN}			

```
16 !
17 key200          ~
18 rows200                  18
19 counter200               19
```

The macro starts with `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities. Then the macro uses the "Safe technique" to assign the [Paint column:] range name to the right column (the B1..B5 in our example), while simultaneously using the [Paint column:] name as a prompt. Next the macro issues `{LEFT}`, which moves the cell pointer to the cell to the left (to the A1 in our example) and then the macro issues `{LET rows200,@ROWS(Paint column:)}~` which stores the number of rows of the [Paint column:] range (the B1..B5 range in our example) in the B18 cell named [rows200].

To add all the values in the [Paint column:] range to the left column, the macro issues `{FOR counter200,1,rows200,1,loop200}` which activates the [loop200] as many times as the number of rows in the [Paint column:] range. The [loop200] is the same as in the previous macro code and it adds the value in the right column to the value in the left column. This macro is not limited to contiguous columns because it uses the number of rows which is stored in the [rows200] cell to know how many cells to add. Last the macro issues `/RNDPaint column:~` to leave a clean worksheet.

[7] Sum Multiple Cells

```
     A              B                C                D              E
 1 *---A macro to @SUM MULTIPLE cells and stay where the formula is
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer where you want the @SUM formula to appear
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---Use Ctrl-Break to stop the macro
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                      IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z               {BREAKON}
11 @SUMCELS         {LET rel090,@INFO("release")}~{RECALC loc090}
                    {RECALC form090}
12 form090          {LET here090,@CELLPOINTER("address")}{WINDOWSOFF}
                    {PANELOFF}
13 !                {LET counter090,1}~{RECALC loop1090}{RECALC forma090}
                    {RECALC formc090}
14 loop1090         /RNCCELL6 TO SUM~~
15 forma090         /RNDCELL6 TO SUM~
16 formc090         /RNC{PANELON}{WINDOWSON}CELL6 TO SUM~{BS}{BS}{?}~
                    {WINDOWSOFF}{PANELOFF}
17 !                {LET sum2090,+"CELL"&@STRING(counter090,0)&" TO SUM"}~
18 !                {EDIT}{HOME}'{END}+{sum2090}{HOME}{DEL}~
                    {RECALC formb090}
19 formb090         /RNDCELL6 TO SUM~
20 !                {LET counter090,counter090+1}~{RECALC loop1090}
                    {RECALC forma090}{RECALC formc090}{BRANCH loop1090}
21 !
22 sum1090          CELL5 TO SUM
23 sum2090          CELL5 TO SUM
24 !
25 here090          $B$8
26 !
27 counter090                6
28 !
29 rel090           @INFO("release")
30 !
31 loc090           address
```

Every Lotus 1-2-3 user writes formulas to sum data from scattered cells using the basic arithmetic operation "+" either by typing the cell addresses or more likely by pointing to the cells using the cell pointer. This macro makes the task easier and clearer. You are only prompted to point to the cells and press ENTER, no need to type "+" signs and/or commas "," the macro does the job. Notice that the code in the B12, B14, B15, B16, B19 and the B31 cells is the result of the following dynamic string formulas. Therefore if you intend to key this macro into Lotus 1-2-3, you have to key the formulas as they appear here, NOT the code as it appears in the main listing.

```
12 form090          +"{LET here090,@CELLPOINTER("""&B31&""")}{WINDOWSOFF}
                    {PANELOFF}"
14 loop1090         +"/RNCCELL"&@STRING(B27,0)&" TO SUM~~"
15 forma090         +"/RNDCELL"&@STRING(B27,0)&" TO SUM~"
16 formc090         +"/RNC{PANELON}{WINDOWSON}CELL"&@STRING(B27,0)&" TO
                    SUM~{BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}"
19 formb090         +"/RNDCELL"&@STRING(B27,0)&" TO SUM~"
31 loc090           @IF(@LEFT(B29,1)<>"@","coord","address")
```

```
     A              B                C                D              E
11 @SUMCELS         {LET rel090,@INFO("release")}~{RECALC loc090}
                    {RECALC form090}
12 form090          {LET here090,@CELLPOINTER("address")}{WINDOWSOFF}
                    {PANELOFF}
13 !                {LET counter090,1}~{RECALC loop1090}{RECALC forma090}
```

```
                    {RECALC formc090}
```

The macro starts with the `{LET rel090,@INFO("release")}~` macro command which stores the result of the `@INFO("release")` function in the cell named [rel090]. Later, the macro uses this result to determine if you are using a 2-D or a 3-D Lotus release. Next, the macro issues `{RECALC loc090}{RECALC form090}` to update the result of the formulas in the B31 cell named [loc090] and the B12 cell named [forma090]. The macro continues with `{LET here090,@CELLPOINTER("address")}` which stores the current cell pointer position in cell [here090]. Before we continue, note that the code in the B12 cell is the result of the following formula:

```
12 form090          +"{LET here090,@CELLPOINTER(""&B31&"")}{WINDOWSOFF}
                    {PANELOFF}"
```

We can see that the "dynamic" part of this formula is the content of the B31 cell which contains the following formula:

```
32 loc090           @IF(@LEFT(B29,1)<>"@","coord","address")
```

The formula in B31 checks the first character of the content of B29, the result of the `@INFO("release")` function. If the character is NOT the "`@`" character then you are using a 3-D Lotus release and the result of the formula is the "`coord`" string. The resulting code of the formula in B12 is:

```
{LET here090,@CELLPOINTER("coord")}{WINDOWSOFF}{PANELOFF}
```

If the character is the "`@`" character you are using a 2-D Lotus release and the result of the formula in B31 is the "`address`" string. The resulting code of the formula in B12 is:

```
{LET here090,@CELLPOINTER("address")}{WINDOWSOFF}{PANELOFF}
```

The macro continues with `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities. Then it issues `{LET counter090,1}~`, which set the content of [counter090] to "`1`" and `{RECALC loop1090}{RECALC forma090}{RECALC formc090}` to update the result of the other three dynamic string formulas.

```
     A              B                C              D              E
14 loop1090         /RNCCELL6 TO SUM~~
15 forma090         /RNDCELL6 TO SUM~
16 formc090         /RNC{PANELON}{WINDOWSON}CELL6 TO SUM~{BS}{BS}{?}~
                    {WINDOWSOFF}{PANELOFF}
```

Next the macro uses the "safe technique" to temporarily assign the [CELL(X) TO SUM] range name to the cell that you want to add to the summation formula, and simultaneously uses the range name as a prompt to highlight the cell to sum and press the ENTER key. Then the macro issues `{WINDOWSOFF}{PANELOFF}` to freeze again the screen and panel display activities. The (X) in the range name represents a dynamic variable. To better understand the macro, let's assume that you want to create a formula in the current cell which sums the following six cells: A10, C34, G20, H12, J40 and K10.

When the macro starts, the counter in B27 contains the number "1" therefore the macro assigns the [CELL1 TO SUM] range name to the first cell and displays the "CELL1 TO SUM" prompt in the panel. When you paint the A10 cell and press the ENTER key, the formula in the current

cell becomes +A10 and the A10 cell has the [CELL1 TO SUM] range name. When you later point to the C34 cell, the formula in the current cell becomes the +A10+C34 and the C34 cell has the [CELL2 TO SUM]. The next cell has the [CELL3 TO SUM] and every next cell is assigned a name with an ascending number. You may ask how is it done? The answer is the dynamic formulas. The code in the B14, B15, and the B16 is the result of dynamic string formulas. Let's look at the formula in the B14 cell:

```
+"/RNCCELL"&@STRING(B27,0)&" TO SUM~~"
```

The dynamic part of the formula is the @STRING(B27,0) function, which returns the content of B27 as an alphanumeric string. However, B27 is the counter for the number of cells that you add to the formula in the current cell. Therefore, for the first cell, B27 contains the number "1" and the range name becomes the [CELL1 TO SUM]. For the sixth cell, B27 contains the number "6" and the range name becomes [CELL6 TO SUM] and the formula in the current cell becomes +A10+C34+G20+H12+J40+K10. When you work with the macro, the prompt adapts itself with the ascending number to give you the feeling that it counts the number of cells that you add to the summation formula.

```
     A          B                C          D          E
17 !          {LET sum2090,+"CELL"&@STRING(counter090,0)&" TO SUM"}~
18 !          {EDIT}{HOME}'{END}+{sum2090}{HOME}{DEL}~
              {RECALC formb090}
```

Now the macro issues {LET sum2090,+"CELL"&@STRING(counter090,0)&" TO SUM"}~ which store the result of the +"CELL"&@STRING(counter090,0)&" TO SUM" formula in B23 cell [sum2090] as a pure label. Next, the macro issues {EDIT}{HOME} to start the EDIT mode and move the cursor to the beginning of the formula in the panel. Now the macro writes the apostrophe "'" to the panel to turn the formula into text and the issues {END} which moves the cursor to the end of the text in the panel. The macro continues to write into the panel and writes the "+" character and then issues the {sum2090} routine command, which activates the [sum2090] routine. However, this routine contains the range name of the cell to add to the summation formula, therefore the {sum2090} routine command injects the content of [sum2090] into the panel and adds the cell address to the summation formula. To better understand the process, let's look at an example.

Let's assume that the current cell already contains the +A10+C34 formula and the macro prompts you to point to the third cell (CELL3 TO SUM). If you point to the G20 cell for example, the macro stores the G20 address in [sum2090]. When the macro issues {EDIT} {HOME} the panel displays:

```
+A10+C34
^
```

The cursor is represented by the "^" under the "+" character. Now the macro writes the apostrophe "'" to the panel which displays:

```
'+A10+C34
^
```

Next the macro issues {END}, which moves the cursor to the end of the text in the panel, and then displays:

```
'+A10+C34_
```

Next the macro writes the "+" character and therefore the panel displays:

```
'+A10+C34+_
```

The next {sum2090} routine command injects the content of the cell [sum2090] into the panel which now displays:

```
'+A10+C34+G20_
```

Next the macro issues {HOME}{DEL} which move the cursor to the beginning of the text in the panel and then deletes the apostrophe "'" to change the content of the panel to formula. Last the macro issues the tilde "~" (the same as ENTER), which writes the content of the panel into the current cell, and then {RECALC formb090} to update the dynamic string formula in the cell named [formb090].

	A	B	C	D	E
19	formb090	/RNDCELL6 TO SUM~			
20	!	{LET counter090,counter090+1}~{RECALC loop1090}			
		{RECALC forma090}{RECALC formc090}{BRANCH loop1090}			

As we explained earlier, the main reasons that the macro assigns a range name like the [CELL6 TO SUM] range name is to assist you and give the macro a professional look. However, when the mission is complete, the macro does not need the range name any more, and the macro issues /RNDCELL6 TO SUM~ to delete the range name. This code is the result of the previous {LET sum2090,+"CELL"&@STRING(counter090,0)&" TO SUM"}~ commands. Next the macro issues {LET counter090,counter090+1}, which increases the value in cell [counter090] by one and then issues {RECALC loop1090}{RECALC forma090}{RECALC formc090} to update the dynamic string formulas with respect to the new value in [counter090]. Last, the macro issues {BRANCH loop1090}, which loops back to the B14 cell [loop1090] and starts the process for the next cell to add to the summation formula in the current cell.

You can stop the macro using the [Ctrl-Break] key combination.

[7] Sum Multiple Ranges

```
     A          B          C          D          E
 1 *---A macro to @SUM MULTIPLE ranges
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer where you want the @SUM formula to appear
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---Use Ctrl-Break to stop the macro
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                    IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z          {BREAKON}
11 @SUMRNGS    {LET rel091,@INFO("release")}~{RECALC form091}
12 form091     {LET here091,@CELLPOINTER("address")}{WINDOWSOFF}{PANELOFF}
13 !           {LET counter091,1}{RECALC loop1091}{RECALC loop2091}
               {RECALC loop3091}
               {RECALC loop4091}
14 loop1091    /RNCRANGE6 TO SUM~~
15 loop2091    /RNDRANGE6 TO SUM~
16 loop3091    /RNC{PANELON}{WINDOWSON}RANGE6 TO SUM~{BS}{BS}{?}~
               {WINDOWSOFF}{PANELOFF}
17 !           {LET sum2091,+"@SUM(RANGE"&@STRING(counter091,0)&" TO SUM)"}~
18 !           {EDIT}{HOME}'{END}+{sum2091}{HOME}{DEL}~{GOTO}{here091}~
19 loop4091    /RNDRANGE6 TO SUM~
20 !           {LET counter091,counter091+1}~{RECALC loop1091}
               {RECALC loop2091}{RECALC loop3091}{RECALC loop4091}
               {BRANCH loop1091}
21 !
22 sum2091     @SUM(RANGE5 TO SUM)
23 !
24 here091           ERR
25 !
26 counter091          6
27 !
28 rel091      @INFO("release")
29 !
30 loc091      address
```

Here is the list of formulas in the macro.

```
12 form091          +"{LET here091,@CELLPOINTER("""&B30&""")}{WINDOWSOFF}
                    {PANELOFF}"
14 loop1091         +"/RNCRANGE"&@STRING(B26,0)&" TO SUM~~"
15 loop2091         +"/RNDRANGE"&@STRING(B26,0)&" TO SUM~"
16 loop3091         +"/RNC{PANELON}{WINDOWSON}RANGE"&@STRING(B26,0)&" TO
                    SUM~{BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}"
19 loop4091         +"/RNDRANGE"&@STRING(B26,0)&" TO SUM~"
30 loc091           @IF(@LEFT(B28,1)<>"@","coord","address")
```

This macro is basically the same as the previous @SUMCELS.WK1 macro, but here we can easily sum scattered ranges not just cells. The macro uses the same techniques as the previous @SUMCELS.WK1 macro, except that here the range names are enveloped by the `@SUM()` function as can be seen in the B17 cell. The reason that we have both versions is because the previous macro is more efficient and allows more cells in the formula, because it uses only the "+" operator character, while this macro uses the `+@SUM()` operator, which takes more text. That there is a limit of 240 characters per formula in the 2-D releases of Lotus 1-2-3 and 512 characters in the 3-D releases of Lotus 1-2-3.

[3] Automatically Sum and Underline a Column of Numbers

```
     A          B          C          D          E
 1 *---An AUTOMATIC macro to SUM and UNDERLINE a column of numbers
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Highlight the cell just beneath the column of numbers
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---The column of number must be contiguous
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 SUMUNDRA        {WINDOWSOFF}{PANELOFF}
12 !               @REPEAT("-",@CELLPOINTER("width")-1)~{DOWN}~{ESC}
13 !               @SUM({UP 2}.{END}{UP})~{DOWN}
```

This is one of the classic issues for a macro, the automatic summation of a column of data. Today all the new releases of Lotus 1-2-3 have an icon or botton to simulate this old macro. However, for users who stick with the old versions, this macro will do the job.

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the unwanted screen and panel display activities. Next the macro writes the `@REPEAT("-", @CELLPOINTER"width")-1)` formula into the panel and issues the tilde "`~`", which is the same as the ENTER key, and writes the formula into the current cell. This formula creates a dividing line made of the string of hyphens "`---------`" by the length of the column's width less one. Now the macro issues `{DOWN}`, which moves the cell pointer one cell down and again writes the `@SUM` formula into the panel. The `@SUM` formula building technique is worth a detailed explanation through an example. As an example let us look at the following column of data:

```
     A          B          C          D          E

1       1000
2       2000
3       1500
4       3000
5       ----
6
7
```

The cell pointer is now on the A6 cell. The macro first writes the "`@SUM(`" string into the panel which displays:

```
@SUM(
```

Next the macro issues `{UP 2}` which moves the cell pointer to the A4 cell of our example worksheet. Therefore the panel displays:

```
@SUM(A4..A4
```

Now the macro issues the period "`.`" to anchor the cell pointer to the A4..A4 cell and then issues `{END}{UP}` to paint the A1 range of data, the panel displays now:

```
@SUM(A1..A4
```

To finish the job the macro writes the closing bracket "`)`" so that the panel displays:

```
@SUM(A1..A4)
```

Next the macro issues tilde "`~`", which is the same as the ENTER key, and writes the formula into the A6 cell.

[3] Semi Automatically Sum and Underline a Column of Numbers

	A	B	C	D	E
1	*---A SEMI-AUTOMATIC macro to SUM and UNDERLINE a column of numbers				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell just beneath the column of numbers				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	*---If the column of number is contiguous the whole column will be				
7	highlighted otherwise expand the highlight and press [ENTER]				
8	!				
9	!				
10	\Z	{BREAKON}			
11	SUMUNDRS	@REPEAT("-",@CELLPOINTER("width")-1)~{DOWN}~{ESC}			
12	!	@SUM({UP 2}.{END}{UP}{?})~			

If the SUMUNDRA.WK1 may seem obsolete in the new releases that features an icon or a button that does the same thing, this macro expands the capabilities of the previous macro to handle non-contiguous column of data. The macro contains `{?}`, just before the macro closes the brackets, which halts the macro execution and allows you to manually expand or shrink the range to sum. The rest of the macro is the same as the previous SUMUNDRA.WK1 macro.

[3] Round a Value in a Cell

```
     A              B                C              D              E
 1 *---A macro to @ROUND the value in a cell
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Highlight the cell to be rounded
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 @ROUNDCL        {GETLABEL "How many places: ",places434}~
12 !               {WINDOWSOFF}{PANELOFF}{IF @CELLPOINTER("type")="v"}
                   {rnd434}~
13 !               {ESC}
14 !
15 rnd434          {EDIT}{HOME}@ROUND(@VALUE({END}),{places434})~/RV~~
16 !
17 places434       2
```

This macro uses the `@ROUND` function to round a value in a cell to a specified number of decimal places. When you start the macro, it issues the `{GETLABEL "How many places: ", places434}` macro command which prompts you to insert the number of decimal places. Lotus stores your response as a label in the B17 cell named [places434]. The macro continues with `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities, and then issues `{IF @CELLPOINTER("type")="v"}`, which checks if the current cell contains a value. If so, the macro issues the `{rnd434}` routine command which activates the [rnd434] routine. The [rnd434] routine starts with `{EDIT}` to enter the EDIT mode. To better understand what the macro does, let's assume that the value in the current cell is the 4.44444444, therefore when the macro issues `{EDIT}{HOME}` the panel displays:

```
4.44444444
^
```

The "^" represents the location of the cursor. Next the macro types the "`@ROUND(@VALUE(`" text directly to the panel, therefore the panel displays:

```
@ROUND(@VALUE(4.44444444
```

Now the macro issues `{END}`, which moves the cursor to the end of the text in the panel and continues to type the "`),`" text into the panel which displays

```
@ROUND(@VALUE(4.44444444),
```

Now the macro issues `{places434}` which writes the content of cell [places434] into the panel in the cursor position. Cell [places434] holds the number of decimal places as a label. If you chose to round the numbers to two decimal places, [places434] contains the label "2", therefore the panel now displays:

```
@ROUND(@VALUE(4.44444444),2
```

Last, the macro writes the closing parenthesis and create an enveloping formula around the value. Now the panel displays:

```
@ROUND(@VALUE(4.44444444),2)
```

When the [rnd434] routine is finished, the macro issues the tilde "~" command which is the same as the ENTER key to write the content of the panel into the current cell. Next the macro issues the `/RV~~` macro keys to turn the formula into the 4.44 fixed value.

[7] Round All the Values in a Range

```
     A                B                C                D                E
 1 *---A macro to @ROUND all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 @ROUNDRG        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb435,0}~
12 !               {LET hereabs435,@CELLPOINTER("address")}~
                   {LET counter1435,0}
13 !               {GETLABEL "How many places: ",places435}~
14 cont435         {FOR counter1435,0,@COLS(Which range ?)-1,1,labels1435}
15 !               {LET rel435,@INFO("release")}~{IF @LEFT(rel435,1)<>"@"}
                   {GOTO}{hereabs435}~{LET counterb435,counterb435+1}~
                   {IF counterb435<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs435}~
                   {BRANCH cont435}
16 !               {GOTO}Which range ?~/RVWhich range ?~~/RNDWhich range ?~
17 !
18 counter1435              2
19 counter1a435             5
20 labels1435      {RIGHT}{LET here435,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a435,0,@ROWS(Which range ?)-1,1,
                   labels1a435}~{IF counter1435<@COLS(Which range ?)-1}
                   {GOTO}{here435}~{LET counter1a435,0}~
21 !
22 here435         $C$1
23 !
24 labels1a435     {IF @CELLPOINTER("type")="v"}{rnd435}
25 !               {DOWN}
26 !
27 rnd435          {EDIT}{HOME}@ROUND(@VALUE({END}),{places435})
28 !
29 places435       2
30 !
31 counterb435              4
32 hereabs435      $A$1
33 !
34 rel435
```

This macro uses the @ROUND function to round off all the values in the range to a specified number of decimal places. The macro prompts you to highlight (paint) the range to process and then it rounds all the numbers and formulas in this range into fixed values. The macro ignores the labels and the empty cells. This macro is a little bit different than the ROUNDVAL.WK1 macro, it turns all the rounded formulas and values into fixed values and it uses a different technique to round the values.

```
     A                B                C                D                E
11 @ROUNDRG        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb435,0}~
12 !               {LET hereabs435,@CELLPOINTER("address")}~
                   {LET counter1435,0}
13 !               {GETLABEL "How many places: ",places435}~
```

The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to round, and simultaneously assigns the [Which range ?] name to the same range, to use as a prompt. Next the macro issues {LET counterb435,0}~ while setting the content of the cell named [counterb435] to zero, which serves as a counter.

To be able to return to the point of origin when the macro is finished, the macro issues {LET hereabs435,@CELLPOINTER("address")}~, which store the current cell pointer address in cell [hereabs435], and then it issues {LET counter435,0} to set the content of [counter435] to zero, which also serves as a counter. The macro continues with {GETLABEL "How many places: ",places435}~, which display the "How many places: " prompt message in the panel. When you insert the number of decimal places and press ENTER, Lotus stores it in cell [places435].

	A	B	C	D	E
14	cont435	{FOR counter1435,0,@COLS(Which range ?)-1,1,labels1435}			
15	!	{LET rel435,@INFO("release")}~{IF @LEFT(rel435,1)<>"@"}			
		{GOTO}{hereabs435}~{LET counterb435,counterb435+1}~			
		{IF counterb435<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs435}~			
		{BRANCH cont435}			
16	!	{GOTO}Which range ?~/RVWhich range ?~~/RNDWhich range ?~			

Next it issues the {FOR counter1435,0,@COLS(Which range ?)-1,1,labels1435} loop command, which activates the [labels1435] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels1435] routine.

	A	B	C	D	E
20	labels1435	{RIGHT}{LET here435,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a435,0,@ROWS(Which range ?)-1,1,			
		labels1a435}~{IF counter1435<@COLS(Which range ?)-1}			
		{GOTO}{here435}~{LET counter1a435,0}~			

The [labels1435] routine uses the {RIGHT}{LET here435,@CELLPOINTER("address") }~{LEFT} commands to record the address of the first cell of the next column in cell [here1435]. When the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here435]. The {FOR counter1a435,0,@ROWS(Which range ?)-1,1,labels1a435}~ commands activate the [labels1a435] routine as many times as the number of rows in [Which range ?].

	A	B	C	D	E
24	labels1a435	{IF @CELLPOINTER("type")="v"}{rnd435}			
25	!	{DOWN}			
26	!				
27	rnd435	{EDIT}{HOME}@ROUND(@VALUE({END}),{places435})			

The [labels1a435] routine issues {IF @CELLPOINTER("type")="v"}, which checks if the current cell contains a value. If so, the macro issues the {rnd435} routine command which activates the [rnd435] routine. The [rnd435] routine starts with {EDIT} to enter the EDIT mode. To better understand what the macro does, let's assume that the value in the current cell is 4.44444444, therefore when the macro issues {EDIT}{HOME} the panel displays:

```
4.44444444
^
```

The "^" represents the location of the cursor. Next the macro types "@ROUND(@VALUE(" (without the double quotes) directly to the panel, therefore the panel displays:

```
@ROUND(@VALUE(4.44444444
```

Now the macro issues {END}, which moves the cursor to the end of the text in the panel and continues to type the ")," text into the panel which displays

```
@ROUND(@VALUE(4.44444444),
```

Now the macro issues {places435}, which writes the content of [places435] into the panel in the cursor position. The cell [places435] holds the number of decimal places as a label. If you chose to round the numbers to two decimal places, [places435] contains the label "2", therefore the panel now displays:

```
@ROUND(@VALUE(4.44444444),2
```

Last, the macro writes the closing parenthesis and creates an enveloping formula around the value. Now the panel displays:

```
@ROUND(@VALUE(4.44444444),2)
```

Now that the [rnd435] routine is finished, the macro returns to the [labels1a435] routine and issues {DOWN} to write the formula in the panel into the current cell and moves the cell pointer down to the next cell in the column. When the [labels1a435] routine ends, the macro returns control to the {FOR} loop command in the [labels1435] routine to process the next cell in the current column.

```
     A                B                C                D                E
20 labels1435     {RIGHT}{LET here435,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a435,0,@ROWS(Which range ?)-1,1,
                  labels1a435}~{IF counter1435<@COLS(Which range ?)-1}
                  {GOTO}{here435}~{LET counter1a435,0}~
```

When the value in [counter1a435] reaches the number of rows in the [Which range ?] range minus one, the macro issues {IF counter1435<@cols(Which range ?)-1} to check how many column were processed. If the value in [counter1435] is less than the number of columns in [Which range ?], the macro issues the indirect {GOTO}{here435}~ command which moves the cell pointer to the first cell of the next column. Next the macro issues the {LET counter1a435,0}~ to reset the value in [counter1a435] to zero. When the [labels1435] routine is finished, the macro returns control back to the [cont435] routine.

```
     A                B                C                D                E
14 cont435        {FOR counter1435,0,@COLS(Which range ?)-1,1,labels1435}
15 !              {LET rel435,@INFO("release")}~{IF @LEFT(rel435,1)<>"@"}
                  {GOTO}{hereabs435}~{LET counterb435,counterb435+1}~
                  {IF counterb435<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs435}~
                  {BRANCH cont435}
16 !              {GOTO}Which range ?~/RVWhich range ?~~/RNDWhich range ?~
```

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues {LET rel435,@INFO("release")}~, which store the result

of the `@INFO("release")` 3-D function in [rel435] and then issues `{IF @LEFT (rel435,1)<>"@"}`, which checks the first character of the content of [rel435]. If it is the "`@`" character, you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the `{GOTO}{hereabs435}~` indirect command, which moves the cell pointer to the origin address of the macro.

The macro continues with `{LET counterb435,counterb435+1}~` which increase the value in [counterb435] by one. Now the macro issues `{IF counterb435<@SHEETS (Which range ?)}` to check if the value in [counterb435] is less than the number of sheets in [Which range ?]. If so, there are more sheets to process. Therefore the macro issues `{NS}` to move the cell pointer to the next sheet. Next the macro issues the indirect `{GOTO}{hereabs435}~` macro command, which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of the [Which range ?] range, but in the new sheet. Last, the macro issues `{BRANCH cont435}` to process the next sheet in the [Which range ?] range. When the macro finishes processing all the sheets in the [Which range ?] range, it issues `{GOTO}Which range ?~`, which moves the cell pointer to the upper left cell of the first sheet of the [Which range ?] range. Next the macro issues `/RVWhich range ?~~`, which turns all the formulas in the range into values and then `/RNDWhich range ?~` to delete the temporary [Which range ?] range name and leave a clean worksheet.

[7] Round All the Values in a Range (2)

```
     A                B                C                D                E
 1 *---A macro to ROUND all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 ROUNDVAL        {GETLABEL "Number of places: ",places420}~{WINDOWSOFF}
                   {PANELOFF}/RNCWhich range ?~~/RNDWhich range ?~/RNC
                   {WINDOWSON}{PANELON}Which range ?~{BS}{BS}{?}~
                   {WINDOWSOFF}{GOTO}Which range ?~{LET counterb420,0}~
12 cont420         {LET hereabs420,@CELLPOINTER("address")}~
                   {LET counter1420,0}
13 !               {FOR counter1420,0,@COLS(Which range ?)-1,1,labels1420}
14 !               {LET rel420,@INFO("release")}~{IF @LEFT(rel420,1)<>"@"}
                   {GOTO}{hereabs420}~{LET counterb420,counterb420+1}~
                   {IF counterb420<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs420}~{BRANCH cont420}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1420                3
18 counter1a420               4
19 labels1420      {RIGHT}{LET here1420,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a420,0,@ROWS(Which range ?)-1,1,
                   labels1a420}~{IF counter1420<@COLS(Which range ?)-1}
                   {GOTO}{here1420}~{LET counter1a420,0}~
20 !
21 here1420        $D$1
22 !
23 labels1a420     {WINDOWSOFF}{IF @CELLPOINTER("type")="v"}{rnd420}
24 !               {DOWN}
25 !
26 rnd420          {EDIT},@VALUE(@CELL("contents",places420))){HOME}
                   @ROUND(~
27 !
28 places420       2
29 !
30 counterb420                4
31 hereabs420      $A$1
32 !
33 rel420
```

This macro uses the `@ROUND` function to round off all the values in the range to a specified number of decimal places. The macro prompts you to highlight the range to process and then the macro rounds all the values in this range. The macro ignores the labels and the empty cells. The macro starts with the `{GETLABEL "Number of places: ",places420}~` macro commands, which display "`Number of places:` " in the panel. When you insert the number and press the ENTER key, Lotus stores it in the cell named [places420]. Next the macro issues `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to round and simultaneously assigns the [Which range ?] name to the same range. Next the macro issues `{LET counterb420,0}~`, setting the content of [counterb420] to zero, which serves as a counter.

```
     A                B                C                D                E
12 cont420         {LET hereabs420,@CELLPOINTER("address")}~
                   {LET counter1420,0}
13 !               {FOR counter1420,0,@COLS(Which range ?)-1,1,labels1420}
14 !               {LET rel420,@INFO("release")}~{IF @LEFT(rel420,1)<>"@"}
```

```
                {GOTO}{hereabs420}~{LET counterb420,counterb420+1}~
                {IF counterb420<@SHEETS(Which range ?)}{NS}{GOTO}
                {hereabs420}~{BRANCH cont420}
15 !            {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro is finished, the macro issues `{LET hereabs420,@CELLPOINTER("address")}~`, which store the current cell pointer address in cell [hereabs420], and then `{LET counter420,0}` to set the content of [counter420] to zero, which also serves as a counter. The `{FOR counter1420,0,@COLS(Which range ?)-1,1,labels1420}` loop command activates the [labels1420] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels1420] routine.

```
     A          B          C          D          E
19 labels1420   {RIGHT}{LET here1420,@CELLPOINTER("address")}~{LEFT}
                {FOR counter1a420,0,@ROWS(Which range ?)-1,1,
                labels1a420}~{IF counter1420<@COLS(Which range ?)-1}
                {GOTO}{here1420}~{LET counter1a420,0}~
```

The [labels1420] routine issues `{RIGHT}{LET here1420,@CELLPOINTER("address") }~{LEFT}` to record the address of the first cell of the next column in [here1420]. When the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here1420]. The `{FOR counter1a420,0,@ROWS(Which range ?)1,1,labels1a420}~` macro command activates the [labels1a420] routine as many times as the number of rows in the [Which range ?] range.

```
     A          B          C          D          E
23 labels1a420  {WINDOWSOFF}{IF @CELLPOINTER("type")="v"}{rnd420}
24 !            {DOWN}
25 !
26 rnd420       {EDIT},@VALUE(@CELL("contents",places420))){HOME}
                @ROUND(~
```

The [labelsa420] routine issues `{WINDOWSOFF}` to freeze the screen activity and then `{IF @CELLPOINTER("type")="v"}`, which checks if the cell contains a value. If so, the macro issues the `{rnd420}` routine command, activating the [rnd420] routine. The [rnd420] routine starts with `{EDIT}` to enter the EDIT mode. To better understand what the macro does, let's assume that the value in the current cell is the 4.44444444, therefore when the macro issues `{EDIT}` the panel displays:

```
4.44444444
```

Next the macro types the "`,@VALUE(@CELL("contents",places420)))`" text directly to the panel, therefore the panel displays:

```
4.44444444,@VALUE(@CELL("contents",places420)))
```

Now the macro issues `{HOME}`, which moves the cursor to the beginning of the text in the panel and continues to type the "`@ROUND(`" text into the panel and create an enveloping formula around the value. Now the panel displays:

```
@ROUND(4.44444444,@VALUE(@CELL("contents",places420)))
```

and issues tilde "~" to write the formula into the current cell. If [places420] holds the "`2`" number, the result of this formula is the "`4.44`" number. When the [rnd420] routine is finished

the macro returns to the [labels1a420] routine and issues {DOWN} to move the cell pointer down to the next cell in the column. When the [labels1a420] routine ends, the macro returns the control to the {FOR} loop command in the [labels1420] routine to process the next cell in the current column.

	A	B	C	D	E
19	labels1420	{RIGHT}{LET here1420,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a420,0,@ROWS(Which range ?)-1,1,			
		labels1a420}~{IF counter1420<@COLS(Which range ?)-1}			
		{GOTO}{here1420}~{LET counter1a420,0}~			

When the value in [counter1a420] reaches the number of rows in the [Which range ?] range minus one, the macro issues {IF counter1420<@cols(Which range ?)-1} to check how many columns were processed. If the value in [counter1420] is less than the number of columns in the [Which range ?] range, the macro issues the indirect {GOTO}{here1420}~ macro command which moves the cell pointer to the first cell of the next column. Next the macro issues {LET counter1a420,0}~ to reset the value in [counter1a420] to zero. When the [labels1420] routine is finished, the macro returns control back to the [cont420] routine.

	A	B	C	D	E
12	cont420	{LET hereabs420,@CELLPOINTER("address")}~			
		{LET counter1420,0}			
13	!	{FOR counter1420,0,@COLS(Which range ?)-1,1,labels1420}			
14	!	{LET rel420,@INFO("release")}~{IF @LEFT(rel420,1)<>"@"}			
		{GOTO}{hereabs420}~{LET counterb420,counterb420+1}~			
		{IF counterb420<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs420}~{BRANCH cont420}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release, First the macro issues {LET rel420,@INFO("release")}~, which store the result of the @INFO("release") 3-D function in [rel420], and then {IF @LEFT(rel420,1)<> "@"}, which checks the first character of the content of [rel420]. If it is the "@" character, you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in the [Which range ?]. Therefore the macro issues the {GOTO}{hereabs420}~ indirect macro command which moves the cell pointer to the origin address of the macro.

The macro continues with {LET counterb420,counterb420+1}~ which increase the value in [counterb420] by one. Now the macro issues {IF counterb420<@SHEETS (Which range ?)} to check if the value in [counterb420] is less than the number of sheets in the [Which range ?] range. If so, there are more sheets to process. Therefore the macro issues {NS} to move the cell pointer to the next sheet. Next the macro issues the indirect {GOTO} {hereabs420}~ macro command which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of the [Which range ?] range, but in the new sheet. Last, the macro issues {BRANCH cont420} to process the next sheet in the [Which range ?] range. When the macro finishes processing all the sheets in the [Which range ?] range, it issues {GOTO}Which range ?~, which moves the cell pointer to the upper left cell of the first sheet of the [Which range ?] range and then /RNDWhich range ?~ to delete the temporary [Which range ?] range name.

[7] Change the Sign of All the Values in a Range

```
        A                B                C                D                E
 1 *---A macro to CHANGE THE SIGN of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z             {BREAKON}
11 SIGNCHNG       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                  {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb421,0}~
12 cont421        {LET hereabs421,@CELLPOINTER("address")}~
                  {LET counter1421,0}
13 !              {FOR counter1421,0,@COLS(Which range ?)-1,1,labels1421}
14 !              {LET rel421,@INFO("release")}~{IF @LEFT(rel421,1)<>"@"}
                  {GOTO}{hereabs421}~{LET counterb421,counterb421+1}~
                  {IF counterb421<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs421}~{BRANCH cont421}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1421                 2
18 counter1a421                5
19 labels1421     {RIGHT}{LET here1421,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a421,0,@ROWS(Which range ?)-1,1,type421}~
                  {IF counter1421<@COLS(Which range ?)-1}{GOTO}{here1421}~
                  {LET counter1a421,0}~
20 !
21 here1421       $C$1
22 !
23 labels1a421    {EDIT}{HOME}-({END})
24 value421
25 !
26 type421        {IF @CELLPOINTER("type")="v"}{labels1a421}
27 !              {DOWN}
28 !
29 counterb421                 4
30 hereabs421     $A$1
31 !
32 rel421
```

This macro changes the sign of all the values in a range; the macro ignores labels and blanks. Then the macro prompts you to paint the range to process and then the macro envelopes every value with the `-()` function. For example: if a cell contains the `-1000` number, after the macro is finished, the cell will contain the `-(-1000)` formula.

```
        A                B                C                D                E
11 SIGNCHNG       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                  {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb421,0}~
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the first range, and simultaneously assigns the [Which range ?] range name to the same range, which acts as a prompt[4]. Next the macro issues `{LET counterb421,0}~` setting the content of cell [counterb421] to zero, which serves as a counter.

	A	B	C	D	E
12	cont421	{LET hereabs421,@CELLPOINTER("address")}~			
		{LET counter1421,0}			
13	!	{FOR counter1421,0,@COLS(Which range ?)-1,1,labels1421}			
14	!	{LET rel421,@INFO("release")}~{IF @LEFT(rel421,1)<>"@"}			
		{GOTO}{hereabs421}~{LET counterb421,counterb421+1}~			
		{IF counterb421<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs421}~{BRANCH cont421}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

To be able to return to the point of origin when the macro is finished, the macro issues `{LET hereabs421,@CELLPOINTER("address")}~`, which store the current cell pointer address in [hereabs421], and then issues `{LET counter1421,0}` to set the content of [counter1421] to zero, which also serves as a counter. The macro issues `{FOR counter1421,0,@COLS (Which range ?)-1,1,labels1421}` which activates the [labels1421] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels1421] routine.

	A	B	C	D	E
19	labels1421	{RIGHT}{LET here1421,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a421,0,@ROWS(Which range ?)-1,1,type421}~			
		{IF counter1421<@COLS(Which range ?)-1}{GOTO}{here1421}~			
		{LET counter1a421,0}~			

The [labels1421] routine uses `{RIGHT}{LET here421,@CELLPOINTER("address")}~ {LEFT}` to record the address of the first cell of the next column in cell [here421]. When the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in cell [here421]. The `{FOR counter1a421,0, @ROWS(Which range ?)-1,1,type421}~` macro command activates the [type421] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
26	type421	{IF @CELLPOINTER("type")="v"}{labels1a421}			
27	!	{DOWN}			

The [type421] routine issues `{IF @CELLPOINTER("type")="v"}` to check if the current cell content is a value. If so, the macro issues the `{labels1a421}` routine command which activates the [labels1a421] routine.

	A	B	C	D	E
23	labels1a421	{EDIT}{HOME}-({END})			

The [labels1a421] routine starts with `{EDIT}` to switch to the EDIT mode. Then the macro issues `{HOME}`, which moves the cursor to the first character in the panel and then writes the "`-(`" string to start the enveloping function around the value and then continues to write the closing parenthesis "`)`" character to complete the enveloping . To make it clearer to you, let's see the process step by step. Let's assume that the current cell contains the -1000 value. When the macro issues `{EDIT}` the panel displays:

```
-1000_
```

The underscore represents the cursor. When the macro issues `{HOME}`, the cursor moves to the minus "`-`" character and the panel displays:

```
-1000)
^
```

The "^" represents the cursor. Now the macro types the "-(" string, therefore the panel displays:

```
-(-1000
```

Next the macro issues {END}, which moves the cursor back to the end of the text in the panel and then writes the closing parenthesis ")" character. Now the panel displays:

```
-(-1000)
```

When the [labels1a421] routine ends, the macro returns control to the [type421] routine which issues {DOWN}, which writes the content of the panel into the current cell and then moves the cell pointer to the next cell in the current column.

If the {IF @CELLPOINTER("type")="v"} condition in not true, the macro issues {DOWN}, which moves the cell pointer to the next cell in the current column. When the [type421] routine ends, the macro returns control to the {FOR} loop command in the [labels1421] routine to start the process for the next cell in the current column.

	A	B	C	D	E
19	labels1421	{RIGHT}{LET here1421,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a421,0,@ROWS(Which range ?)-1,1,type421}~			
		{IF counter1421<@COLS(Which range ?)-1}{GOTO}{here1421}~			
		{LET counter1a421,0}~			

When the value in [counter1a421] reaches the number of rows in the [Which range ?] range minus one, the macro issues {IF counter1421<@COLS(Which range ?)-1} to check how many columns were processed. If the value in [counter1421] is less than the number of columns in the [Which range ?] range, the macro issues the indirect {GOTO}{here421}~ macro command, which moves the cell pointer to the first cell of the next column. Next the macro issues {LET counter1a421,0}~ to reset the value in [counter1a421] to zero. When the [labels1421] routine is finished, the macro returns the control back to the [cont421] routine.

	A	B	C	D	E
12	cont421	{LET hereabs421,@CELLPOINTER("address")}~			
		{LET counter1421,0}			
13	!	{FOR counter1421,0,@COLS(Which range ?)-1,1,labels1421}			
14	!	{LET rel421,@INFO("release")}~{IF @LEFT(rel421,1)<>"@"}			
		{GOTO}{hereabs421}~{LET counterb421,counterb421+1}~			
		{IF counterb421<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs421}~{BRANCH cont421}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues {LET rel421,@INFO("release")}~ which store the result of the @INFO("release") 3-D function in the cell named [rel421]. Then the macro issues {IF @LEFT(rel421,1)<>"@"}, which checks the first character of the content of [rel421]. If it is the "@" character, you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the {GOTO}{hereabs421}~ indirect macro command, which moves the cell pointer to the origin address of the macro.

The macro continues with {LET counterb421,counterb421+1}~, which increase the value

in [counterb421] by one. Now the macro issues `{IF counterb421<@SHEETS(Which range ?)}` to check if the value in [counterb421] is less than the number of sheets in the [Which range ?] range. If so, there are more sheets to process. Therefore the macro issues the `{NS}` macro command to move the cell pointer to the next sheet. Next the macro issues the indirect `{GOTO}{hereabs421}~` macro command, which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of the [Which range ?] range, but in the new sheet. Last, the macro issues `{BRANCH cont421}` to process the cells of the [Which range ?] range in the new sheet. When all the sheets are processed, the macro issues `{GOTO}Which range ?~`, which moves the cell pointer to the upper left cell of the first sheet of the [Which range ?] range, and then `/RNDWhich range ?~` which deletes the temporary [Which range ?] range name and leaves a clean worksheet.

[7] Modify All the Values in a Range

```
     A                B                C                D                E
 1 *---A macro to MODIFY all values in a 3-D or 2-D range. For example:
 2     multiply values by 5 or add 9 to all values in the range. Blanks or
 3     labels will not be altered.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                    IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z             {BREAKON}
11 MODIFRNG       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{PANELON}{WINDOWSON}Which range ?~
                  {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~{GETLABEL "
                  Enter desired modification (Example: +5, *4, /3, etc.)"
                  ,value411}{LET counterb411,0}~
12 cont411        {LET hereabs411,@CELLPOINTER("address")}~
                  {LET counter1411,0}
13 !              {FOR counter1411,0,@COLS(Which range ?)-1,1,labels1411}
14 !              {LET rel411,@INFO("release")}~{IF @LEFT(rel411,1)<>"@"}
                  {GOTO}{hereabs411}~{LET counterb411,counterb411+1}~
                  {IF counterb411<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs411}~{BRANCH cont411}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1411           4
18 counter1a411          6
19 labels1411     {RIGHT}{LET here1411,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a411,0,@ROWS(Which range ?)-1,1,type411}~
                  {IF counter1411<@COLS(Which range ?)-1}{GOTO}{here1411}~
                  {LET counter1a411,0}~
20 !
21 here1411       $E$1
22 !
23 labels1a411    {EDIT}{HOME}({END})
24 value411       /10
25 !
26 type411        {IF @CELLPOINTER("type")<>"v"}{DOWN}{RETURN}
27 !              {labels1a411}{DOWN}
28 !
29 counterb411           4
30 hereabs411     $A$1
31 !
32 rel411
```

This macro allows you to MODIFY all the values in a 3-D or 2-D range. For example: you can multiply all the values by 5 or add 9 to all values in the range. Blanks cell or cells which contain labels are ignored.

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to modify and simultaneously assigns the [Which range ?] name to the same range, which acts as a prompt to you and `{GETLABEL "Enter desired modification (Example: +5, *4, /3, etc.) ",value411}` displays:

```
Enter desired modification (Example: +5, *4, /3, etc.)
```

in the panel and waits for your response. When you press ENTER Lotus stores your response in the B24 cell named [value411]. Next the macro issues `{LET counterb411,0}~`, which set the content of the B29 cell named [counterb411] to zero, which serves as a counter.

	A	B	C	D	E
12	cont411	{LET hereabs411,@CELLPOINTER("address")}~			
		{LET counter1411,0}			
13	!	{FOR counter1411,0,@COLS(Which range ?)-1,1,labels1411}			
14	!	{LET rel411,@INFO("release")}~{IF @LEFT(rel411,1)<>"@"}			
		{GOTO}{hereabs411}~{LET counterb411,counterb411+1}~			
		{IF counterb411<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs411}~{BRANCH cont411}			

To be able to return to the point of origin when the macro is finished, the macro issues `{LET hereabs411,@CELLPOINTER("address")}~`, which store the current cell pointer address in cell [hereabs411], and then issues `{LET counter1411,0}` to set the content of [counter1411] to zero, which also serves as a counter. The macro issues `{FOR counter1411,0,@COLS (Which range ?)-1,1,labels1411},` which activates the [labels1411] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels1411] routine.

	A	B	C	D	E
19	labels1411	{RIGHT}{LET here1411,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a411,0,@ROWS(Which range ?)-1,1,type411}~			
		{IF counter1411<@COLS(Which range ?)-1}{GOTO}{here1411}~			
		{LET counter1a411,0}~			

This routine uses `{RIGHT}{LET here1411,@CELLPOINTER("address")}~{LEFT}` to record the address of the first cell of the next column in cell [here1411]. When the current column processing is finished, the macro uses the address stored in the cell named [here1411] to move the cell pointer directly to the top of the next column. The `{FOR counter1a411 ,0,@ROWS(Which range ?)-1,1,type411}~` macro commands activate the [type411] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
26	type411	{IF @CELLPOINTER("type")<>"v"}{DOWN}{RETURN}			
27	!	{labels1a411}{DOWN}			

The [type411] routine uses `{IF @CELLPOINTER("type")="v"}` to check if the current cell's content is a label. If so, the macro issues `{DOWN}{RETURN}`, which move the cell pointer to the next cell in the column and return to the `{FOR}` loop. If the condition is false, the macro issues the `{labels1a411}` routine command which activates the [labels1a411] routine.

	A	B	C	D	E
23	labels1a411	{EDIT}{HOME}({END})			
24	value411	/10			

This routine issues `{EDIT}{HOME}` which enter to EDIT mode and then move the cursor to the beginning of the panel. Now the macro writes the opening parenthesis "`(`" before the content of the panel (a value) and then issues `{END}`, which moves the cursor to the end of the panel, and then writes the closing parenthesis "`)`". Next the macro writes the operation that you entered as a response to the prompt message, and returns to the [type411] routine to execute `{DOWN}`, which writes the content of the panel to the current cell, and moves the cell pointer to the next cell in the column. For example: if you responded with the "`/10`" divide by ten operation, Lotus stores the "`/10`" string in [value411] which becomes part of the code. This is one example of how to use dynamic code in macro language.

Let's continue with the [labels1411] routine.

	A	B	C	D	E
19	labels1411	{RIGHT}{LET here1411,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a411,0,@ROWS(Which range ?)-1,1,type411}~			
		{IF counter1411<@COLS(Which range ?)-1}{GOTO}{here1411}~			
		{LET counter1a411,0}~			

When the {FOR} loop is finished with the first column, the macro issues {IF counter1411 <@COLS(Which range ?)-1} to check if there are more columns to process. If so, the macro issues the {GOTO}{here411}~ indirect macro command to move to the first cell of the next column, the [here411] cell holds the address of the first cell in the next column. The last macro commands in this routine is {LET counter1a411,0}~, which reset the counter in cell [counter1a411] to zero. Now we can go back to the [cont411] routine.

	A	B	C	D	E
12	cont411	{LET hereabs411,@CELLPOINTER("address")}~			
		{LET counter1411,0}			
13	!	{FOR counter1411,0,@COLS(Which range ?)-1,1,labels1411}			
14	!	{LET rel411,@INFO("release")}~{IF @LEFT(rel411,1)<>"@"}			
		{GOTO}{hereabs411}~{LET counterb411,counterb411+1}~			
		{IF counterb411<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs411}~{BRANCH cont411}			

The macro continues with {LET rel411,@INFO("release")}~, which store the result of the @INFO("release") function in [rel411]. Then the macro issues {IF @LEFT(rel411, 1)<>"@"} to check if you are using a 2-D or a 3-D Lotus release. If "@" is the first character of the content of [rel411], then you are using a 2-D Lotus release, otherwise you use a Lotus 3-D release. If you are using a 3-D release, the macro issues the {GOTO}{hereabs411}~ indirect macro command to move to the first cell in the current sheet of the [Which range ?] range, and then issues {LET counterb411,counterb411+1}~ to increase the counter in [counterb411] by one.

Next the macro issues {IF counterb411<@SHEETS(Which range ?)} to compare the counter value in [counterb411] to the number of sheets in the [Which range ?] range. If the counter value is still less than the number of sheets in the [Which range ?] range, the macro has to process more sheets before it can quit. Therefore the macro issues {NS}{GOTO} {hereabs411}~ to move the cell pointer to the upper left cell in the new sheet of the [Which range ?] range and issues {BRANCH cont411} to loop back to the beginning of the [cont411] routine. When the counter in [counterb411] is equal to the number of sheets in the [Which range ?] range, the macro issues {GOTO}Which range ?~ to place the cell pointer back on the place of origin just before the macro started, and then issues /RNDWhich range ?~ to delete the temporary [Which range ?] range name to leave a clean worksheet.

[7] Calculate the @ABS of All the Values in a Range

```
     A              B                C                D              E
 1 *---A macro to CALCULATE the @ABS of all values in a 3-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 @ABSRANG         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                    Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                    {LET counterb422,0}~
12 cont422          {LET hereabs422,@CELLPOINTER("address")}~
                    {LET counter1422,0}
13 !                {FOR counter1422,0,@COLS(Which range ?)-1,1,labels1422}
14 !                {LET rel422,@INFO("release")}~{IF @LEFT(rel422,1)<>"@"}
                    {GOTO}{hereabs422}~{LET counterb422,counterb422+1}~
                    {IF counterb422<@SHEETS(Which range ?)}{NS}{GOTO}
                    {hereabs422}~{BRANCH cont422}
15 !                {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1422               4
18 counter1a422              6
19 labels1422       {RIGHT}{LET here422,@CELLPOINTER("address")}~{LEFT}
                    {FOR counter1a422,0,@ROWS(Which range ?)-1,1,
                    labels1a422}~{IF counter1422<@COLS(Which range ?)-1}
                    {GOTO}{here422}~{LET counter1a422,0}~
20 !
21 here422          $E$34
22 !
23 labels1a422      {IF @CELLPOINTER("type")="v"}{rnd422}
24 !                {DOWN}
25 !
26 rnd422           {EDIT}){HOME}@ABS({END}
27 !
28 counterb422               1
29 hereabs422       $A$34
30 !
31 rel422
```

This macro applies the `@ABS` function on all the values in a range. It ignores labels and blanks. The macro prompts you to paint the range to process, and then it envelopes every value with the `@ABS` function. For example: if a cell contains the `-1000` number, after the macro is finished the cell will contain the `@ABS(-1000)` formula.

```
     A              B                C                D              E
11 @ABSRANG         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                    Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                    {LET counterb422,0}~
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the first range, and simultaneously assigns the [Which range ?] name to the same range, to act as a prompt. Next the macro issues `{LET counterb422,0}~`, which set the content of cell [counterb422] to zero, which serves as a counter.

```
     A              B                C                D              E
```

```
12 cont422          {LET hereabs422,@CELLPOINTER("address")}~
                    {LET counter1422,0}
13 !                {FOR counter1422,0,@COLS(Which range ?)-1,1,labels1422}
14 !                {LET rel422,@INFO("release")}~{IF @LEFT(rel422,1)<>"@"}
                    {GOTO}{hereabs422}~{LET counterb422,counterb422+1}~
                    {IF counterb422<@SHEETS(Which range ?)}{NS}{GOTO}
                    {hereabs422}~{BRANCH cont422}
15 !                {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the place of origin when the macro is finished, the macro issues `{LET hereabs422,@CELLPOINTER("address")}~`, which store the current cell pointer address in the B29 cell named [hereabs422], and then issues `{LET counter1422,0}` to set the content of [counter1422] to zero, which also serves as a counter. The macro issues `{FOR counter1422,0,@COLS(Which range ?)-1,1,labels1422}` which activates the [labels1422] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels1422] routine.

```
     A              B                C              D              E
19 labels1422       {RIGHT}{LET here422,@CELLPOINTER("address")}~{LEFT}
                    {FOR counter1a422,0,@ROWS(Which range ?)-1,1,
                    labels1a422}~{IF counter1422<@COLS(Which range ?)-1}
                    {GOTO}{here422}~{LET counter1a422,0}~
```

The [labels1422] routine uses `{RIGHT}{LET here422,@CELLPOINTER("address")}~{LEFT}` to record the address of the first cell of the next column in cell [here422]. When the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here422]. The `{FOR counter1a422,0,@ROWS(Which range ?)-1,1,labels1a422}~` macro commands activate the [labels1a422] routine as many times as the number of rows in the [Which range ?] range.

```
     A              B                C              D              E
23 labels1a422      {IF @CELLPOINTER("type")="v"}{rnd422}
24 !                {DOWN}
25 !
26 rnd422           {EDIT}){HOME}@ABS({END}
```

The [labels1a422] routine issues `{IF @CELLPOINTER("type")="v"}` to check if the current cell content is a value. If so, the macro issues the `{rnd422}` routine command which activates the [rnd422] routine. The [rnd422] routine starts with `{EDIT}` to switch to the EDIT mode and then writes the closing parenthesis "`)`" character. Then the macro issues `{HOME}`, which moves the cursor to the first character in the panel and continues to write the "`@ABS(`" string to complete the enveloping function around the value. To make it clearer, let's see the process step by step. Let's assume that the current cell contains the `-1000` value. When the macro issues `{EDIT}` the panel displays:

```
-1000_
```

The underscore represents the cursor. When the macro writes the "`)`" character, the panel displays:

```
-1000)
```

When the macro issues `{HOME}`, the cursor moves to the minus "`-`" character and the panel displays:

```
-1000)
```

```
^
```

The "^" represents the cursor. Now the macro types the "@ABS(" string and the panel displays the complete formula:

```
@ABS(-1000)
```

Note: You can create a macro like this for all the functions in Lotus 1-2-3. However, because the SUPER MACRO LIBRARY already contains all these macros we include them too. When you will key them into 1-2-3, you will be able to enjoy them with the MACRO MANAGER. For an explanation of how each macro works, please see this macro. You may ask: Why not create a custom menu in this macro which will allow us to choose any Lotus function? The answer is simple: Such a macro will need to have more then one menu because there are more than eight functions, and the Lotus custom menu is limited to eight menu options, therefore we will have to create nested menus. This is one approach when you use the macros manually. But we can keep every macro as a separate file because we have created the MACRO MANAGER which allows you to activate any macro in ***Super Power*** (and in the SUPER MACRO LIBRARY) directly from the disk using point and shoot without the need to create custom menus. The custom menu is the main menu of the MACRO MANAGER.

Next the macro returns to the [labels1a422] routine and issues {DOWN}, which writes the @ABS(-1000) formula into the current cell and moves the cell pointer to the next cell in the current column. If the {IF @CELLPOINTER("type")="v"} condition in not true, the macro issues {DOWN}, which moves the cell pointer to the next cell in the current column. When the [labels1a422] routine ends the macro returns the control to the {FOR} loop command in the [labels1422] routine to start the process for the next cell in the current column.

```
     A              B                C              D              E
19 labels1422     {RIGHT}{LET here422,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a422,0,@ROWS(Which range ?)-1,1,
                  labels1a422}~{IF counter1422<@COLS(Which range ?)-1}
                  {GOTO}{here422}~{LET counter1a422,0}~
```

When the value in [counter1a422] reaches the number of rows in the [Which range ?] range minus one, the macro issues {IF counter1422<@COLS(Which range ?)-1} to check how many columns were processed. If the value [counter1422] is less than the number of columns in the [Which range ?] range, the macro issues the indirect {GOTO}{here422}~ macro command, which moves the cell pointer to the first cell of the next column. Next the macro issues {LET counter1a422,0}~ to reset the value in [counter1a422] to zero. When the [labels1422] routine is finished, the macro returns control back to the [cont422] routine.

```
     A              B                C              D              E
12 cont422        {LET hereabs422,@CELLPOINTER("address")}~
                  {LET counter1422,0}
13 !              {FOR counter1422,0,@COLS(Which range ?)-1,1,labels1422}
14 !              {LET rel422,@INFO("release")}~{IF @LEFT(rel422,1)<>"@"}
                  {GOTO}{hereabs422}~{LET counterb422,counterb422+1}~
                  {IF counterb422<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs422}~{BRANCH cont422}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
```

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First the macro issues the {LET rel422,@INFO("release")}~, which store the

result of the `@INFO("release")` 3-D function in the cell named [rel422], and then the macro issues `{IF @LEFT(rel422,1)<>"@"}`, which checks the first character of the content of [rel422]. If "`@`" is the character, you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the `{GOTO}{hereabs422}~` indirect macro command, which moves the cell pointer to the origin address of the macro.

The macro continues with `{LET counterb422,counterb422+1}~`, which increments the value in [counterb422] by one. Now the macro issues `{IF counterb422<@SHEETS(Which range ?)}` to check if the value in [counterb422] is less than the number of sheets in the [Which range ?] range. If so, there are more sheets to process. Therefore the macro issues `{NS}` to move the cell pointer to the next sheet. Next the macro issues the indirect `{GOTO}{hereabs422}~` macro command which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of the [Which range ?] range, but in the new sheet. Last, the macro issues `{BRANCH cont422}` to process the cells of the [Which range ?] range in the new sheet. When all the sheets are processed, the macro issues `{GOTO}Which range ?~`, which moves the cell pointer to the upper left cell of the first sheet of the [Which range ?] range, and then `/RNDWhich range ?~` which deletes the temporary [Which range ?] range name and leaves a clean worksheet.

[7] Calculate the @ACOS of All the Values In a Range

```
     A                B                C                D                E
 1 *---A macro to CALCULATE the @ACOS of all values in a 3-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 @ACOSRNG        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb423,0}~
12 cont423         {LET hereabs423,@CELLPOINTER("address")}~
                   {LET counter1423,0}
13 !               {FOR counter1423,0,@COLS(Which range ?)-1,1,labels1423}
14 !               {LET rel423,@INFO("release")}~{IF @LEFT(rel423,1)<>"@"}
                   {GOTO}{hereabs423}~{LET counterb423,counterb423+1}~
                   {IF counterb423<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs423}~{BRANCH cont423}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1423              2
18 counter1a423             5
19 labels1423      {RIGHT}{LET here423,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a423,0,@ROWS(Which range ?)-1,1,
                   labels1a423}~{IF counter1423<@COLS(Which range ?)-1}
                   {GOTO}{here423}~{LET counter1a423,0}~
20 !
21 here423         $C$1
22 !
23 labels1a423     {IF @CELLPOINTER("type")="v"}{rnd423}
24 !               {DOWN}
25 !
26 rnd423          {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@ACOS({END}
27 !
28 counterb423                4
29 hereabs423      $A$1
30 !
31 rel423
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @ASIN of All the Values In a Range

```
     A                B                C                D                E
 1 *---A macro to CALCULATE the @ASIN of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 @ASINRNG        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb424,0}~
12 cont424         {LET hereabs424,@CELLPOINTER("address")}~
                   {LET counter1424,0}
13 !               {FOR counter1424,0,@COLS(Which range ?)-1,1,labels1424}
14 !               {LET rel424,@INFO("release")}~{IF @LEFT(rel424,1)<>"@"}
                   {GOTO}{hereabs424}~{LET counterb424,counterb424+1}~
                   {IF counterb424<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs424}~{BRANCH cont424}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1424              2
18 counter1a424             5
19 labels1424      {RIGHT}{LET here424,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a424,0,@ROWS(Which range ?)-1,1,
                   labels1a424}~{IF counter1424<@COLS(Which range ?)-1}
                   {GOTO}{here424}~{LET counter1a424,0}~
20 !
21 here424         $C$1
22 !
23 labels1a424     {IF @CELLPOINTER("type")="v"}{rnd424}
24 !               {DOWN}
25 !
26 rnd424          {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@ASIN({END}
27 !
28 counterb424                4
29 hereabs424      $A$1
30 !
31 rel424
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @ATAN of All the Values In a Range

```
     A              B                C              D              E
 1 *---A macro to CALCULATE the @ATAN of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 @ATANRNG         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                    Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                    {LET counterb425,0}~
12 cont425          {LET hereabs425,@CELLPOINTER("address")}~
                    {LET counter1425,0}
13 !                {FOR counter1425,0,@COLS(Which range ?)-1,1,labels1425}
14 !                {LET rel425,@INFO("release")}~{IF @LEFT(rel425,1)<>"@"}
                    {GOTO}{hereabs425}~{LET counterb425,counterb425+1}~
                    {IF counterb425<@SHEETS(Which range ?)}{NS}{GOTO}
                    {hereabs425}~{BRANCH cont425}
15 !                {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1425               2
18 counter1a425              5
19 labels1425       {RIGHT}{LET here425,@CELLPOINTER("address")}~{LEFT}
                    {FOR counter1a425,0,@ROWS(Which range ?)-1,1,
                    labels1a425}~{IF counter1425<@COLS(Which range ?)-1}
                    {GOTO}{here425}~{LET counter1a425,0}~
20 !
21 here425          $C$1
22 !
23 labels1a425      {IF @CELLPOINTER("type")="v"}{rnd425}
24 !                {DOWN}
25 !
26 rnd425           {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@ATAN({END}
27 !
28 counterb425              4
29 hereabs425       $A$1
30 !
31 rel425
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @COS of All the Values In a Range

```
     A                B                C                D                E
 1 *---A macro to CALCULATE the @COS of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 @COSRANG        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb426,0}~
12 cont426         {LET hereabs426,@CELLPOINTER("address")}~
                   {LET counter1426,0}
13 !               {FOR counter1426,0,@COLS(Which range ?)-1,1,labels1426}
14 !               {LET rel426,@INFO("release")}~{IF @LEFT(rel426,1)<>"@"}
                   {GOTO}{hereabs426}~{LET counterb426,counterb426+1}~
                   {IF counterb426<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs426}~{BRANCH cont426}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1426              2
18 counter1a426             5
19 labels1426      {RIGHT}{LET here426,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a426,0,@ROWS(Which range ?)-1,1,
                   labels1a426}~{IF counter1426<@COLS(Which range ?)-1}
                   {GOTO}{here426}~{LET counter1a426,0}~
20 !
21 here426         $C$1
22 !
23 labels1a426     {IF @CELLPOINTER("type")="v"}{rnd426}
24 !               {DOWN}
25 !
26 rnd426          {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@COS({END}
27 !
28 counterb426                4
29 hereabs426      $A$1
30 !
31 rel426
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @EXP of All the Values In a Range

```
     A              B                C              D              E
 1 *---A macro to CALCULATE the @EXP of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 @EXPRANG         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                    Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                    {LET counterb427,0}~
12 cont427          {LET hereabs427,@CELLPOINTER("address")}~
                    {LET counter1427,0}
13 !                {FOR counter1427,0,@COLS(Which range ?)-1,1,labels1427}
14 !                {LET rel427,@INFO("release")}~{IF @LEFT(rel427,1)<>"@"}
                    {GOTO}{hereabs427}~{LET counterb427,counterb427+1}~
                    {IF counterb427<@SHEETS(Which range ?)}{NS}{GOTO}
                    {hereabs427}~{BRANCH cont427}
15 !                {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1427               2
18 counter1a427              5
19 labels1427       {RIGHT}{LET here427,@CELLPOINTER("address")}~{LEFT}
                    {FOR counter1a427,0,@ROWS(Which range ?)-1,1,
                    labels1a427}~{IF counter1427<@COLS(Which range ?)-1}
                    {GOTO}{here427}~{LET counter1a427,0}~
20 !
21 here427          $C$1
22 !
23 labels1a427      {IF @CELLPOINTER("type")="v"}{rnd427}
24 !                {DOWN}
25 !
26 rnd427           {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@EXP({END}
27 !
28 counterb427                4
29 hereabs427       $A$1
30 !
31 rel427
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @INT of All the Values In a Range

```
     A                B                C                D                E
 1 *---A macro to CALCULATE the @INT of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 @INTRANG        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb428,0}~
12 cont428         {LET hereabs428,@CELLPOINTER("address")}~
                   {LET counter1428,0}
13 !               {FOR counter1428,0,@COLS(Which range ?)-1,1,labels1428}
14 !               {LET rel428,@INFO("release")}~{IF @LEFT(rel428,1)<>"@"}
                   {GOTO}{hereabs428}~{LET counterb428,counterb428+1}~
                   {IF counterb428<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs428}~{BRANCH cont428}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1428              2
18 counter1a428             5
19 labels1428      {RIGHT}{LET here428,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a428,0,@ROWS(Which range ?)-1,1,
                   labels1a428}~{IF counter1428<@COLS(Which range ?)-1}
                   {GOTO}{here428}~{LET counter1a428,0}~
20 !
21 here428         $C$1
22 !
23 labels1a428     {IF @CELLPOINTER("type")="v"}{rnd428}
24 !               {DOWN}
25 !
26 rnd428          {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@INT({END}
27 !
28 counterb428              4
29 hereabs428      $A$1
30 !
31 rel428
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @LN of All the Values In a Range

```
     A                B                C                D                E
 1 *---A macro to CALCULATE the @LN of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 @LNRANGE        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb429,0}~
12 cont429         {LET hereabs429,@CELLPOINTER("address")}~
                   {LET counter1429,0}
13 !               {FOR counter1429,0,@COLS(Which range ?)-1,1,labels1429}
14 !               {LET rel429,@INFO("release")}~{IF @LEFT(rel429,1)<>"@"}
                   {GOTO}{hereabs429}~{LET counterb429,counterb429+1}~
                   {IF counterb429<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs429}~{BRANCH cont429}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1429              2
18 counter1a429             5
19 labels1429      {RIGHT}{LET here429,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a429,0,@ROWS(Which range ?)-1,1,
                   labels1a429}~{IF counter1429<@COLS(Which range ?)-1}
                   {GOTO}{here429}~{LET counter1a429,0}~
20 !
21 here429         $C$1
22 !
23 labels1a429     {IF @CELLPOINTER("type")="v"}{rnd429}
24 !               {DOWN}
25 !
26 rnd429          {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@LN({END}
27 !
28 counterb429                4
29 hereabs429      $A$1
30 !
31 rel429
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @LOG of All the Values In a Range

```
     A                B                C                D                E
 1 *---A macro to CALCULATE the @LOG of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 @LOGRANG        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb430,0}~
12 cont430         {LET hereabs430,@CELLPOINTER("address")}~
                   {LET counter1430,0}
13 !               {FOR counter1430,0,@COLS(Which range ?)-1,1,labels1430}
14 !               {LET rel430,@INFO("release")}~{IF @LEFT(rel430,1)<>"@"}
                   {GOTO}{hereabs430}~{LET counterb430,counterb430+1}~
                   {IF counterb430<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs430}~{BRANCH cont430}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1430              2
18 counter1a430             5
19 labels1430      {RIGHT}{LET here430,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a430,0,@ROWS(Which range ?)-1,1,
                   labels1a430}~{IF counter1430<@COLS(Which range ?)-1}
                   {GOTO}{here430}~{LET counter1a430,0}~
20 !
21 here430         $C$1
22 !
23 labels1a430     {IF @CELLPOINTER("type")="v"}{rnd430}
24 !               {DOWN}
25 !
26 rnd430          {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@LOG({END}
27 !
28 counterb430                4
29 hereabs430      $A$1
30 !
31 rel430
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @SIN of All the Values In a Range

```
     A                B                C                D                E
 1 *---A macro to CALCULATE the @SIN of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 @SINRANG        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb431,0}~
12 cont431         {LET hereabs431,@CELLPOINTER("address")}~
                   {LET counter1431,0}
13 !               {FOR counter1431,0,@COLS(Which range ?)-1,1,labels1431}
14 !               {LET rel431,@INFO("release")}~{IF @LEFT(rel431,1)<>"@"}
                   {GOTO}{hereabs431}~{LET counterb431,counterb431+1}~
                   {IF counterb431<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs431}~{BRANCH cont431}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1431              5
18 counter1a431             4
19 labels1431      {RIGHT}{LET here431,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a431,0,@ROWS(Which range ?)-1,1,
                   labels1a431}~{IF counter1431<@COLS(Which range ?)-1}
                   {GOTO}{here431}~{LET counter1a431,0}~
20 !
21 here431         $F$1
22 !
23 labels1a431     {IF @CELLPOINTER("type")="v"}{rnd431}
24 !               {DOWN}
25 !
26 rnd431          {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@SIN({END}
27 !
28 counterb431              4
29 hereabs431      $A$1
30 !
31 rel431
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @SQRT of All the Values In a Range

```
     A                B                C                D                E
 1 *---A macro to CALCULATE the @SQRT of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 @SQRTRNG         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                    Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                    {LET counterb432,0}~
12 cont432          {LET hereabs432,@CELLPOINTER("address")}~
                    {LET counter1432,0}
13 !                {FOR counter1432,0,@COLS(Which range ?)-1,1,labels1432}
14 !                {LET rel432,@INFO("release")}~{IF @LEFT(rel432,1)<>"@"}
                    {GOTO}{hereabs432}~{LET counterb432,counterb432+1}~
                    {IF counterb432<@SHEETS(Which range ?)}{NS}{GOTO}
                    {hereabs432}~
                    {BRANCH cont432}
15 !                {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1432               5
18 counter1a432              4
19 labels1432       {RIGHT}{LET here432,@CELLPOINTER("address")}~{LEFT}
                    {FOR counter1a432,0,@ROWS(Which range ?)-1,1,
                    labels1a432}~{IF counter1432<@COLS(Which range ?)-1}
                    {GOTO}{here432}~{LET counter1a432,0}~
20 !
21 here432          $F$1
22 !
23 labels1a432      {IF @CELLPOINTER("type")="v"}{rnd432}
24 !                {DOWN}
25 !
26 rnd432           {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@SQRT({END}
27 !
28 counterb432             4
29 hereabs432       $A$1
30 !
31 rel432
```

See **Calculate the @ABS of All the Numbers In a Range**.

[7] Calculate the @TAN of All the Values In a Range

```
     A                B                C                D                E
 1 *---A macro to CALCULATE the @TAN of all values in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 @TANRANG        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb433,0}~
12 cont433         {LET hereabs433,@CELLPOINTER("address")}~
                   {LET counter1433,0}
13 !               {FOR counter1433,0,@COLS(Which range ?)-1,1,labels1433}
14 !               {LET rel433,@INFO("release")}~{IF @LEFT(rel433,1)<>"@"}
                   {GOTO}{hereabs433}~{LET counterb433,counterb433+1}~
                   {IF counterb433<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs433}~{BRANCH cont433}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1433              5
18 counter1a433             4
19 labels1433      {RIGHT}{LET here433,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a433,0,@ROWS(Which range ?)-1,1,
                   labels1a433}~{IF counter1433<@COLS(Which range ?)-1}
                   {GOTO}{here433}~{LET counter1a433,0}~
20 !
21 here433         $F$1
22 !
23 labels1a433     {IF @CELLPOINTER("type")="v"}{rnd433}
24 !               {DOWN}
25 !
26 rnd433          {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@TAN({END}
27 !
28 counterb433              4
29 hereabs433      $A$1
30 !
31 rel433
```

See **Calculate the @ABS of All the Numbers In a Range**.

[3] Calculate the @ABS of a Value In a Cell

```
        A                B                C                D                E
 1 *---A macro to CALCULATE the @ABS of the value in a cell
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer on the cell to be calculated
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 @ABSCELL        {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@ABS(~
12 !               {ESC}
```

This macro applies the `@ABS` function to the value in the current cell. For example, if the cell contains the `-1000` value, the result of the macro is the `@ABS(-1000)` which returns the `1000` value. The macro leaves the formula in the cell so that you can see what it did. The macro starts with the `{IF @CELLPOINTER("type")="v"}` macro command to make sure the current cell contains a value. If so, the macro issues the `{EDIT}` command to switch to the EDIT mode and then writes the closing parenthesis "`)`" character. Then the macro issues `{HOME}`, which moves the cursor to the first character in the panel and then continue to write the "`@ABS(`" string to complete the enveloping function around the value and issues the tilde "`~`" command (the same as the ENTER key) to write the `@ABS(-1000)` formula into the current cell.

To clarify, let's see the process step by step. Let's assume that the current cell contains the `-1000` value. When the macro issues `{EDIT}` the panel displays:

```
-1000_
```

The underscore represents the cursor. When the macro writes the "`)`" character, the panel displays:

```
-1000)
```

When the macro issues `{HOME}`, the cursor moves to the minus "`-`" character and the panel displays:

```
-1000)
^
```

The "^" represents the cursor. Now the macro types the "`@ABS(`" string and the panel displays the complete formula:

```
@ABS(-1000)
```

Note: You can create a macro like this for all the functions in Lotus 1-2-3. However because the SUPER MACRO LIBRARY already contains all these macros we bring them here. For explanation how each macro works you are referred to this macro. You may ask why not to create a custom menu in this macro which will allow us to choose any Lotus function? The answer is simple: such a macro will need to have more then one menu because there are more than eight functions and the Lotus custom menu is limited to eight menu options, therefore you must create nested menus. This is one approach when you use the macros manually. But we can keep every macro as a separate file because we have created the MACRO MANAGER, which

allows us to activate any macro in ***Super Power*** (and in the SUPER MACRO LIBRARY) directly from the disk using point and shoot without the need to create custom menus. The custom menu is the main menu of the MACRO MANAGER.

[3] Calculate the @ACOS of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @ACOS of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@ACOSCEL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@ACOS(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @ASIN of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @ASIN of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@ASINCEL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@ASIN(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @ATAN of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @ATAN of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@ATANCEL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@ATAN(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @COS of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @COS of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@COSCELL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@COS(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @EXP of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @EXP of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@EXPCELL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@EXP(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @INT of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @INT of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@INTCELL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@INT(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @LN of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @LN of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@LNCELL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@LN(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @LOG of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @LOG of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@LOGCELL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@LOG(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @SIN of a Value In a Cell

```
     A              B                C              D              E
 1 *---a macro to calculate the @SIN of the value in a cell
 2 *---use the /range name label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \z macro name)
 4 *---highlight the cell to be calculated
 5 *---hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \z              {BREAKON}
11 @SINCELL        {IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@SIN(~
12 !               {ESC}
```

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @SQRT of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @SQRT of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@SQRTCEL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@SQRT(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

[3] Calculate the @TAN of a Value In a Cell

	A	B	C	D	E
1	*---A macro to CALCULATE the @TAN of the value in a cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Highlight the cell to be calculated				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	@TANCELL	{IF @CELLPOINTER("type")="v"}{EDIT}){HOME}@TAN(~			
12	!	{ESC}			

See **Calculate the @ABS of a Value in a Cell**.

Utilities for *.MLB Macros

[4] Add-In Attach Macro

```
     A              B                C              D              E
 1 *---A macro to attach add-ins, The macro assumes that the DOS default
 2     directory is the directory where LOTUS and the add-ins are located
 3     the ALT-F10 key combination is assigned to the attached add-in
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8                 THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4
 9 !
10 \Z              {ESC}
11 ADDINATH        {GETLABEL "Enter add-in name: ",addname270}~
                   {RECALC form1270}{RECALC form2270}
12 form1270        {IF @ISAPP("D:\MACROMGR")=1}{BEEP}The D:\MACROMGR add
                   in already attached! press any key...{GET key270}{ESC}
                   {BRANCH ret270}
13 form2270        /AA{ESC}D:\MACROMGR~{LEFT}~Q{BRANCH ret270}
14 !
15 addname270      D:\MACROMGR
16 !
17 !
18 key270          j
19 ret270
```

Here are the two string formulas in cells [form1270] and [form2270] (B12 and B13). You must type the following code to these cells instead of the code as it appears in the main listing of the macro.

```
12 form1270        +"{IF @ISAPP("""&B15&""")=1}{BEEP}The "&@UPPER(C16)
                   &" add in already attached! press any key...
                   {GET key270}{ESC}{BRANCH ret270}"
13 form2270        +"/AA{ESC}"&B15&"~{LEFT}~Q{BRANCH ret270}"
```

This macro makes the process of an add-in attaching to Lotus easier. Instead of using the `/AA....` keys, the macro issues the `{GETLABEL "Enter add-in name: ",addname270 }~` macro command which displays the "`Enter add-in name: `" prompt in the panel. Your response is stored in the B15 cell named [addname270]. Next the macro uses `{RECALC form1270}` to update the formula in cell [form1270]. The code in [form1270] is the result of the following dynamic string formula:

```
12 form1270        +"{IF @ISAPP("""&B15&""")=1}{BEEP}The "&@UPPER(C16)
                   &" add in already attached! press any key...
                   {GET key270}{ESC}{BRANCH ret270}"
```

which uses [addname270] that holds the add-in name to create the correct code.

```
     A              B                C              D              E
12 form1270        {IF @ISAPP("D:\MACROMGR")=1}{BEEP}The D:\MACROMGR add
                   in already attached! press any key...{GET key270}{ESC}
                   {BRANCH ret270}
13 form2270        /AA{ESC}D:\MACROMGR~{LEFT}~Q{BRANCH ret270}
```

The [form1270] routine issues the `{IF @ISAPP("D:\MACROMGR")=1}` command to check if the [D:\MACROMGR] add-in is already in memory. If so, the macro uses the `{BEEP}` to sound a beep and types

```
The D:\MACROMGR add in already attached! press any key...
```

prompt to the panel. Notice that this prompt is also the result of a dynamic string formula that takes the "D:\MACROMGR" string from [addname270]. Following the prompt, the macro issues {GET key270} that halts the macro and waits for you to press any key. The key that you press is stored in [key270]. When you press any key, the macro immediately issues {ESC} to clear the panel before the prompt will be written to the current cell, and issues {BRANCH ret270} that routes the macro execution to an empty routine and quits.

If the add-in is not attached to the worksheet, the macro issues /AA{ESC}D:\MACROMGR~{LEFT}~ to attach the D:\MACROMGR add-in to the worksheet, and issues {BRANCH ret270} that routes the macro execution to an empty routine and quits.

[1] Add-In Detach Macro

```
     A                B                C                D                E
 1 *---A macro to Detach add-ins
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Use Ctrl-Break to quit the macro
 6 !
 7                 THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4
 8 !
 9 !
10 \Z              {BREAKON}
11 ADDINDTH        /AD{NAME}{?}~Q
```

This is a very simple macro to detach an add-in from the worksheet. Its advantage is that it displays a full screen list of all the add-ins available, instead of only one line of add-in names. The macro issues the `/AD` macro keys and then issue the `{NAME}` macro command that creates the full screen list of all the add-ins available. Then the macro follows with `{?}` that allows you to use the direction keys to point to the add-in, or to type the add-in name. When you press the ENTER key the macro issues the "`~`" macro key and returns to the READY mode.

[1] Create *.MLB Macros from *.WK1 Macros

```
     A              B              C              D              E
 1 *---A macro to create library macros (*.MLB) from a (*.WK1) macro/file
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                 THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4
 8 !
 9 !
10 \Z              {BREAKON}
11 LIBMAKE2        /AIMACROMGR~S{NAME}{?}~{?}~
```

This macro allows you to save a WK1 macro as an MLB macro. Before you can use this macro, you have to attach the MACROMGR.ADN add-in to the worksheet. The macro issues the `/AIMACROMGR~` macro keys invoking the MACROMGR.ADN add-in. Next the macro issues the "`S`" macro key to activate the Lotus **S**ave menu option. Then it issues `{NAME}`, which displays a full screen list of all the MLB macros in the memory, and then `{?}` input command. You can now accept one of the macro names and replace the old macro with the new one, or type a new name and press the ENTER key. After you press ENTER, the macro issues a second `{?}` that allows you to paint the range that contains the text of the macro. When you press ENTER, the text of the WK1 macro is erased and a new MLB macro appears in the memory and in the disk.

Note: We used two consecutive `{?}` macro commands to create input options. This is not a completely safe macro technique, but it works fine as long as you use the direction keys to point to the Lotus menu options, and then press the ENTER key. If you press the capital first letter of the menu titles, the macro may not work properly. If you want to improve this macro you can duplicate the Lotus menus using custom menus.

[4] Edit and Save an *.MLB Library

```
          A                   B                   C                   D
 1 *---A macro to edit and save a library macro (*.MLB)
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the ,ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                 THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4
 8 !
 9 !
10 \Z              {BREAKON}
11 LIBEDIT         {MENUBRANCH menu150}
12 !
13 menu150         Edit          Save                  Quit
14 !               Edit macro libSave the edited librarQuit the macro
15 !               /AIMACROMGR~  /AIMACROMGR~S{NAME}{?}~{?}~
16 !               E{NAME}{?}~o~ {?}~~
```

This is the code of the custom menu called [menu150]

```
Edit
Edit macro library (copy to worksheet for editing)
/AIMACROMGR~
E{NAME}{?}~o~

Save
Save the edited library macro
/AIMACROMGR~S{NAME}{?}~{?}~
{?}~~

Quit
Quit the macro
```

This macro simplifies the process to create (save) an MLB macro from an existing WK1 macro or to edit an existing MLB macro. Before you can use this macro, you have to attach the MACROMGR.ADN add-in to the worksheet, invoke it, and load the MLB macros to edit. The macro uses a custom menu that cannot be printed in full on one page. Therefore, we show every menu item separately. The code as it appears in the main listing is not complete because the menu items overlap. The macro starts with `{MENUBRANCH menu150}` that starts the [menu150] custom menu. The first menu option is [**E**dit]:

```
Edit
Edit macro library (copy to worksheet for editing)
/AIMACROMGR~
E{NAME}{?}~O~
```

the macro issues `/AIMACROMGR~` that invokes the MACROMGR.ADN add-in that comes with the new versions of Lotus 1-2-3 starting with Release 2.2. Next the macro issues the "`E`" macro key that enters the edit mode. Then the macro issues `{NAME}` that displays a full screen list of all the MLB macros in the memory, and then issues `{?}` to allow you to use the direction keys to point to the name of the macro to edit. When you point to the name of the macro and press ENTER, the macro issues the "`O`" macro key to overwrite any existing and conflicting range names in the worksheet. When the macro is finished, a copy of the MLB macro text will appear in the worksheet ready to be edited. When you are finished with the editing process you can restart the macro and use the [**S**ave] menu option. The second menu option is [**S**ave]:

```
Save
Save the edited library macro
/AIMACROMGR~S{NAME}{?}~{?}~
{?}~~
```

The macro issues `/AIMACROMGR~` that invokes the MACROMGR.ADN add-in. Next the macro issues the "S" macro key activating the save mode. Then the macro issues `{NAME}`, which displays a full screen list of all the MLB macros in the worksheet, and `{?}` to allow you to use the direction keys to point to the name of the macro to edit. When you point to the name of the macro or type a new name for the MLB macro and press ENTER, the macro issues a `{?}` that allows you to paint the range containing the text of the macro. Last the macro issues the third `{?}`, which allows you assign a password to the macro or not. When you press ENTER, the text of the macro is erased and a new MLB macro appears in memory and in the disk.

Note: We have used three consecutive `{?}` macro commands to create input options for you. This is not a completely safe macro technique, but it works fine as long as you use the direction keys to point to the Lotus menu options and then press ENTER. If you press the capital first letter of the menu titles, the macro may not work properly. If you want to improve this macro, you can duplicate the Lotus menus using custom menus.

The third menu option is [**Q**uit]:

```
Quit
Quit the macro
```

When you choose this menu option, the macro meets an empty cell without any macro command; therefore it quits.

[2] Load *.MLB Macros to the Memory

```
     A                B                C                D                E
 1 *---A macro to continuously load a library macros (*.MLB) from the disk
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press <[Z] to activate the macro
 5 *---Use the Ctrl-Break to Quit
 6 !
 7                THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4
 8 !
 9 !
10 \Z              {BREAKON}
11 LIBLOAD         /AIMACROMGR~
12 load151         L{NAME}{?}~
13 !               {BRANCH load151}
```

This macro simplifies the process to load an MLB macro into memory. The macro starts with the `/AIMACROMGR~` macro keys invoking the MACROMGR.ADN add-in that comes with the new versions of Lotus 1-2-3, starting with Release 2.2. Next the macro issues the "`L`" macro key to enter the load mode and then issues `{NAME}` that displays a full screen list of all the MLB macros in the worksheet. Next the macro issues `{?}` to allow you to use the direction keys to point to the name of the macro that you want to load. When you point to the name of the macro and press ENTER, the macro loads it into the memory. To allow you to load more MLB macros, the macro issues the macro command that routes the macro execution to the [load151] routine activating the **L**oad option again. The macro stops when you press the `Ctrl-Break` key combination or when you try to load more than ten MLB macros to the memory, because Lotus allows no more that ten MLB macros in the memory.

[4] Load *.MLB Macros to the Memory (2)

```
     A          B          C          D          E
 1 *---A macro to load *.MLB macros to memory
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Use Ctrl-Break to quit
 6 !
 7                THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4
 8 !
 9 !
10 \Z             {BREAKON}
11 LOAD_MLB       {BREAKON}
12 !              {ONERROR err651}
13 cont651        /AIMACROMGR~L{NAME}{?}~Q
14 !              {BRANCH cont651}
15 err651
```

To use an *.MLB macro you must attach the MACROMGR.ADN add-in, which comes with Lotus 2.2/2.3 and 2.4, and then invoke it. This macro assumes that the MACROMGR.ADN is already attached and makes the process of loading an *.MLB macro easier. First, it invokes the MACROMGR.ADN. Then it displays a full screen list of all the macros in the disk. When you point to the file and press ENTER, the macro loads the *.MLB macro and re-displays the screen. This way you can load macros continuously.

The macro starts with the `{ONERROR err651}` macro command which makes sure that if an error occurs, the macro branches to the empty B15 cell named [err651] and quits. Such an error can occur when you try to load more than ten *.MLB macros to memory simultaneously. This is a limit set by Lotus 1-2-3, which allows up to ten *.MLB macros in memory simultaneously. The macro continues with the `/AIMACROMGR~` macro code which invokes the MACROMGR.ADN and then continues with "`L`" which issues the [**L**oad] menu option.

Then the macro issues `{NAME}`, which tells Lotus to display a full screen list of all the *.MLB macros in the default directory and continues with `{?}`, which allows you to move and point to the macro to load. When you press ENTER, the macro loads the *.MLB macro that you selected and issues the "`Q`" macro key to return to READY mode. Next the macro issues `{BRANCH cont651}`, which loops back to the beginning and allows you to load another macro. To quit the macro you must use the Ctrl-Break key combination.

[2] Remove *.MLB Macros from the Memory

```
     A              B              C              D              E
 1 *---A macro to remove *.MLB macros from memory
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8                 THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4
 9 !
10 \Z              {BREAKON}
11 REMOVMLB        {BREAKON}
12 !               {ONERROR err650}
13 cont650         /AIMACROMGR~R{NAME}{?}~
14 !               {BRANCH cont650}
15 err650
```

To use an *.MLB macro you must attach the MACROMGR.ADN add-in which comes with Lotus 2.2/2.3 and 2.4 and then invoke it. This macro assumes that the MACROMGR.ADN is already attached and it makes the process of removing an *.MLB macro easier. First, it invokes the MACROMGR.ADN and then it displays a full screen list of all the macros in the disk. When you point to the file name and press ENTER to remove the macro, it removes the MLB macro and re-displays the full screen list. This way you can remove macros from memory continuously.

The macro starts with the `{ONERROR err650}` macro command which makes sure that if an error occurs, the macro branches to the empty B15 cell named [err650] and quits. The macro continues with the `/AIMACROMGR~` macro code, which invokes the MACROMGR.ADN and then continues with the "`R`" macro key, which issues the [**R**emove] menu option. Then the macro issues `{NAME}`, which tells Lotus to display a full screen list of all the *.MLB macros in the default directory and continues with `{?}`, which allows you to move and point to the macro to remove. When you press ENTER, the macro removes the *.MLB macro you selected and issues the "`Q`" macro key to return to READY mode. Next the macro issues `{BRANCH cont650}`, which loops back to the beginning and allows you to remove another macro. To quit the macro, you must use the Ctrl-Break key combination.

[4] Remove *.MLB Macros from the Memory (2)

```
     A              B                C              D              E
 1 *---A macro to remove library macros (*.MLB) from the memory
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7                 THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4
 8 !
 9 !
10 \Z              {BREAKON}
11 LIBRMOVE        {BREAKON}
12 loop152         /AIMACROMGR~R{NAME}{?}~
13 !               More to remove? [Y/N] Y {GET key152}{ESC}~
14 !               {IF @UPPER(key152)="N"}{CALC}{BRANCH ret152}
15 !               {BRANCH Loop152}
16 !
17 key152          ~
18 !
19 ret152
```

This macro automates the process to remove several MLB macros from the memory and it is an improved version of the REMOVMLB.WK1 macro. The macro starts with the `/AI MACROMGR~R` macro keys invoking the MACROMGR.ADN add-in manager and initiates the Lotus **R**emove menu option. Next the macro issues `{NAME}` that displays a full screen list of all the MLB macros in the memory, and then issues `{?}` input command, which allows you to use the direction keys to point to the file to remove. When you press the ENTER key, the macro types the

```
More to remove? [Y/N] Y
```

prompt message to the panel (because Lotus cannot find a valid Lotus command in this text) and then issues `{GET key152}` that halts macro execution and waits for your response. When you press ENTER, the macro stores your response in the B17 cell named [`key152`] and immediately issues `{ESC}` to clear the panel from the message before Lotus writes the message to the current cell. To check your response, the macro issues `{IF @UPPER(key152) ="N"}` that makes use of the `@UPPER(key152)` Lotus function to determine if you want to continue or to quit.

If you press the "N" or the "n" key the macro issues `{CALC}` and then `{BRANCH ret152}` that routes the macro execution to an empty routine and quits. If you press the "Y" or the "y" key or any other key, the macro issues `{BRANCH Loop152}` that routes the macro execution back to the beginning and you can remove the next macro

[8] Create *.MLB Macro Libraries from *.WK1 and *.MLB Macros

	A	B	C	D	E
1	*---A macro to create library macros (*.MLB) from the macros (*.WK1) in				
2	the default directory to be used with the MACROMGR.ADN add-in. The				
3	macros must follow the rules given in ***Super Power*** for creating macros				
4	compatible with the SUPER MACROS LIBRARY.				
5	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
6	range names in this column (starts with the \X macro name)				
7	*---Hold the [ALT] key and press [X] to activate the macro				
8		THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4			
9					
10	\X	{BREAKON}			
11	LIBMAKE	{GOTO}counter@#~{LEFT}{DOWN 2}{LET counter@#,-1}~			
		{MENUBRANCH menu@#}			
12	!				
13	menu@#	**A**uto **O**ne_at_a_time **C**ombine *.WK1 **L**ib_Combine **Q**uit			
14	!	Create Create library Combine *.WK1 Combine sma Quit			
15	!	{GETNUM{LET flag@#,2}~{{combine@#} {combinea@#}			
16	!	{LET flag@#,1}~{auto@#}			
17	!				
18	semiauto@#	/FCCE{NAME}{?}~{WINDOWSOFF}{PANELOFF}{BRANCH general@#}			
19	!				
20	auto@#	{RESTART}{LET counter@#,counter@#+1}~{IF counter@#=			
		num@#}{QUIT}			
21	auto1@#	/FCCE{RIGHT 2}~{WINDOWSOFF}{PANELOFF}			
		{BRANCH general@#}"			
22	!				
23	general@#	{ONERROR err2@#}/WDR{DOWN 8}~{DOWN}/C~nam@#~/C~nam2@#~			
		{EDIT}{HOME}{DEL}@CHAR(2)&"{END}"{CALC}~{UP}/RNLR{END}			
		{DOWN}~/RND\Z~/RNC!~~/RND!~			
24	!	{ONERROR err1@#}/AIMACROMGR~S*{ESC}			
25	nam@#	DIRFILE			
26	!	{PANELON}~.{END}{DOWN}{BIGRIGHT 2}~~			
27	remove@#	/AIMACROMGR~R			
28	nam2@#	DIRFILE			
29	!	~			
30	!	{IF flag@#=1}{WINDOWSON}{PANELON}{BRANCH auto@#}			
31	!	{IF flag@#=2}{WINDOWSON}{PANELON}{BRANCH semiauto@#}			
32	!				
33	flag@#	1			
34	!				
35	err1@#	{BEEP}{PANELON}File already exists! or illegal macro...			
		Press any key {GET key@#}{ESC}{IF flag@#=1}			
		{BRANCH remove@#}			
36	!	File already exists! or illegal macro... Press any key			
		{GET key@#}{ESC}{IF flag@#=2}{BRANCH remove@#}			
37	!				
38	err2@#	{BEEP}{PANELON}File already exists! or illegal macro...			
		Press any key GET key@#}{ESC}{IF flag@#=1}			
		{BRANCH auto@#}			
39	!	File already exists! or illegal macro... Press any key			
		{GET key@#}{ESC}{IF flag@#=2}{BRANCH auto@#}			
40	!				
41	num@#	3			
42	!				
43	combine@#	{GOTO}counter@#~{LEFT}{DOWN 2}			
44	loop1@#	/FCCE{NAME}{?}~/WDR.{DOWN 8}~{DOWN}{EDIT}{HOME}{DEL}			
		@CHAR(2)&"{END}"{CALC}~{UP}{END}{DOWN}{DOWN}!{DOWN}			
45	loop2@#	[C]ombine more macros [M]ake library [Q]uit			
		{GET key@#}{ESC}			
46	!	{IF @UPPER(key@#)="Q"}{QUIT}			
47	!	{IF @UPPER(key@#)="C"}{BRANCH loop1@#}			
48	!	{IF @UPPER(key@#)="M"}{GOTO}counter@#~{DOWN 2}{LEFT}			
		/RNLR{END}{DOWN}~/RND\Z~/RND!~/AIMACROMGR~S{?}~.{END}			
		{DOWN}{RIGHT 8}~~{BRANCH libmake}			
49	!	{BRANCH loop2@#}			

```
50 !
51 key@#            M
52 !
53 combinea@#       {GOTO}counter@#~{LEFT}{DOWN 2}
54 loop1a@#         /AIMACROMGR~L{NAME}{?}~E~O~{END}{DOWN}{DOWN}!~{DOWN}
55 loop2a@#          [C]ombine more macros  [M]ake library  [Q]uit
                    {GET key@#}{ESC}
56 !                {IF @UPPER(key@#)="Q"}{QUIT}
57 !                {IF @UPPER(key@#)="C"}/AIMACROMGR~R~{BRANCH loop1a@#}
58 !                {IF @UPPER(key@#)="M"}{GOTO}counter@#~{DOWN 2}{LEFT}
                    /AIMACROMGR~S{?}~.{END}{DOWN}{RIGHT 8}~~{BRANCH libmake}
59 !                {BRANCH loop2a@#}
60 !
61 counter@#                  2
```

The code in the cell named [auto1@#] is the result of the following dynamic formula:

```
21 !                +"/FCCE{RIGHT "&@STRING(B61,0)&"}~{WINDOWSOFF}
                    {PANELOFF}{BRANCH general@#}"
```

If you plan to key this macro into 1-2-3, you need to key this formula, NOT the code as it appears in the main listing.

This is the code of the [menu@#] custom menu:

```
Auto
Create library macros (*.MLB) from macros (*.WK1) in one run
{GETNUMBER "How many macros to create? ",num@#}~
{LET flag@#,1}~{auto@#}

One_at_a_time
Create library macros from macros in the default directory
   one at a time
{LET flag@#,2}~{semiauto@#}

Combine *.WK1
Combine *.WK1 macros to create one large *.MLB library
{combine@#}

Lib_Combine
Combine small *.MLB macros to create one large *.MLB library
{combinea@#}

Quit
Quit the macro
```

This macro automates the process to create (save) MLB macros from existing WK1 macros or to create libraries. The macro uses a custom menu which cannot be printed on one page without overlapping. Therefore we show every menu item separately. The code as it appears in the main listing is not complete because the menu items overlap. The macro starts with the `{GOTO}counter@#~{LEFT}{DOWN 2}` macro commands that move the cell pointer two rows below the macro code. Later, the macro will combine the WK1 macros to that location and will save them as MLB macros. The macro continues with `{LET counter@#,-1}~` to set the counter for the number of files that it processes to minus one, and issues `{MENUBRANCH menu@#}` starting the [menu@#] custom menu. The first menu option in the [menu@#] custom menu is [**A**uto]:

```
Auto
Create library macros (*.MLB) from macros (*.WK1) in one run
{GETNUMBER "How many macros to create? ",num@#}~
{LET flag@#,1}~{auto@#}
```

The macro issues {GETNUMBER "How many macros to create? ",num@#}~ to prompt you to enter the number of macros in the directory that you need to transform into MLB macros. Your response is stored in the cell named [num@#]. The macro uses the {LET flag@#,1}~ to set the value in the B33 cell named [flag@#] to one (raise a flag). Next the macro uses the {auto@#} routine command to activate the [auto@#] routine.

	A	B	C	D	E
20	auto@#	{RESTART}{LET counter@#,counter@#+1}~{IF counter@#=num@#}{QUIT}			
21	auto1@#	/FCCE{RIGHT 2}~{WINDOWSOFF}{PANELOFF}{BRANCH general@#}"			

The {RESTART} macro command clears the subroutine stack, so the macro starts fresh from this point. The {LET counter@#,counter@#+1}~ macro commands increase the value in cell [counter@#] by one. If the value in [counter@#] is equal to the value stored in [num@#], the macro issues {QUIT} and quits. The code in [auto1@#] is the result of a dynamic formula:

```
21 auto1@#          +"/FCCE{RIGHT "&@STRING(B61,0)&"}~{WINDOWSOFF}
                    {PANELOFF}{BRANCH general@#}"
```

This formula uses the value in [counter@#] to calculate how many times to use {RIGHT} to point to the file name in the full screen list of file names. For example, let's assume that the default directory contains 100 WK1 macros. If the value in [counter@#] is 50, the macro will process the 50th file in the list of files. This way the macro can automatically process all the macros in the default directory if you enter the number 100. If you enter the number 50, the macro creates 50 MLB macros from the first 50 files in the directory. In fact, we have used this menu option to automatically create approximately 200 MLB macros from the 250 WK* macros in the SUPER MACRO LIBRARY. The macro issues the /FCCE macro keys to combines the next file to process, and issues {WINDOWSOFF}{PANELOFF} to suspend screen and panel activities while it uses {BRANCH general@#} to activate the [general@#] routine.

	A	B	C	D	E
23	general@#	{ONERROR err2@#}/WDR{DOWN 8}~{DOWN}/C~nam@#~/C~nam2@#~{EDIT}{HOME}{DEL}@CHAR(2)&"{END}"{CALC}~{UP}/RNLR{END}{DOWN}~/RND\Z~/RNC!~~/RND!~			

The macro starts with {ONERROR err2@#} activating the [err2@#] error handling routine. If an error occurs, the [err2@#] error handling routine takes care of it and resumes macro execution. Next the macro uses /WDR{DOWN 8}~{DOWN} to delete the first nine rows of the combined macro. Remember that the macros in ***Super Power*** reserve the first nine rows for instructions. Now the cell pointer stands on the cell that holds the macro or file name. Therefore the macro uses /C~nam@#~/C~nam2@#~ to copy the macro name to cells [nam@#] and [nam2@#].

When an MLB macro is loaded into memory and you press the ALT-F3 key combination to display the macro name, Lotus displays all the range names in the MLB macros, and the range names in the worksheet. If you work on a large Lotus file with macros, it is quite difficult to locate the macro name from a full screen list of range names. Therefore this macro adds the @CHAR(2) character to the beginning of the macro name. Because Lotus sorts the range names that are displayed on the screen, the macro name will be placed at the upper left corner part of the screen . The @CHAR(2) character appears as a hyphen preceding the macro name.

The macro issues {EDIT}{HOME}{DEL} to delete the apostrophe preceding the macro name

and then types the `@CHAR(2)&"` text into the panel. Lotus types the text into the panel because it does not find any valid command in the text. The macro issues `{END}`to move the cursor to the end of text in the panel, and then types the closing double quotes [`"`], and issues `{CALC}~` transforming the formula in the panel into a text. The best way to understand the macro code is to use an example. Let's assume that the macro name is [`FILEDIR`]. Therefore, when the macro issues `{EDIT}` the panel displays:

```
'FILEDIR
```

When the macro issues the `{HOME}{DEL}` macro commands the panel displays:

```
FILEDIR
```

After the macro issues the `@CHAR(2)&"` text the panel displays:

```
@CHAR(2)&"FILEDIR
```

After the macro issues `{END}`, the cursor moves to the end of the text in the panel. When the macro issues the double quotes [`"`] the panel displays:

```
@CHAR(2)&"FILEDIR"
```

When the macro issues the `{CALC}` macro commands the panel displays:

```
-FILEDIR
```

The macro continues with `{UP}/RNLR{END}{DOWN}~` to assign all the range names in the first column of the combined macro. Because all the macros in ***Super Power*** contain the [\Z] range name, the macro uses `/RND\Z` to delete the [\Z] range name. When the macro assigns the range names, there is a possibility that the macro assigns the exclamation mark "`!`" range name which Lotus cannot use as a legal range name. Therefore, to be on the safe side the macro first uses the `/RNC!~~` to assign the exclamation mark "`!`" range name and then uses `/RND!~` to delete the exclamation mark "`!`" range name. This procedure assures no error occurs if the macro does not initially contain the exclamation mark "`!`" range name.

	A	B	C	D	E
24	!	{ONERROR err1@#}/AIMACROMGR~S*{ESC}			
25	nam@#	DIRFILE			
26	!	{PANELON}~.{END}{DOWN}{BIGRIGHT 2}~~			
27	remove@#	/AIMACROMGR~R			
28	nam2@#	DIRFILE			
29	!	~			
30	!	{IF flag@#=1}{WINDOWSON}{PANELON}{BRANCH auto@#}			
31	!	{IF flag@#=2}{WINDOWSON}{PANELON}{BRANCH semiauto@#}			

Now the macro issues `{ONERROR err1@#}` that assigns a second error handling routine, the [err1@#] routine, in case that the MACROMGR.ADN add-in is missing. The macro issues `/AIMACROMGR~S` invoking the MACROMGR.ADN add-in, and starting the Lotus **S**ave menu option. To safely clear the panel from any previous file, the macro uses `*{ESC}`. Next the macro types the file [DIRFILE] name into the panel, and uses `.{END}{DOWN}{BIGRIGHT 2}~~` to paint the combined macro code and save it as an MLB macro. When Lotus creates an MLB macro, it writes the macro to the disk and simultaneously stores it in the memory.

To clear the memory, the macro issues `/AIMACROMGR~RDIRFILE~` and types the name of the last created file [DIRFILE] in our example. Now the macro is ready to process another file. The value the macro stored in [flag@#] tells the macro if you use the [**A**uto] menu option from the custom menu or the [**O**ne_at_a_time] Semi auto option. Therefore, if the value in the cell named [flag@#] is equal to "`1`", the macro issues `{BRANCH auto@#}` routing the macro control to the [auto@#] routine. If the value in [flag@#] is equal to "`2`", the macro issues `{BRANCH semiauto@#}` to route the macro control to the [semiauto@#] routine. The second menu option is [**O**ne_at_a_time]:

```
One_at_a_time
Create library macros from macros in the default directory
   one at a time
{LET flag@#,2}~{semiauto@#}
```

When you choose this option you can manually pick the file that you want to translate and save as an MLB macro from a full screen list of the file names. The macro sets the flag in cell [flag@#] to "2" and issues the `{semiauto@#}` routine command that starts the [semiauto@#] routine.

	A	B	C	D	E
18	semiauto@#	/FCCE{NAME}{?}~{WINDOWSOFF}{PANELOFF}{BRANCH general@#}			

The [semiauto@#] routine starts with `/FCCE` and issues `{NAME}`, which displays a full screen list of all the files in the default directory, and `{?}`, which allows you to use the direction keys to point to the file name. When you press ENTER, the macro issues `{WINDOWSOFF}{PANELOFF}` suppressing screen and panel activities while the macro uses `{BRANCH general@#}` to start the [general@#] routine. We already studied the [general@#] routine; therefore we can continue with the next menu option. The third menu option is [**C**ombine *.WK1]:

```
Combine *.WK1
Combine *.WK1 macros to create one large *.MLB library
{combine@#}
```

When you choose this menu option, you can combine a few WK1 macros together and create one large combined MLB macro. Lotus allows no more than ten MLB macros in memory. Using this menu option, you can combine and create MLB macros that each contain several MLB macros and overcome the limitations set by Lotus. This menu option contains only one command, the `{combine@#}` routine command that starts the [combine@#] routine.

	A	B	C	D	E
43	combine@#	{GOTO}counter@#~{LEFT}{DOWN 2}			
44	loop1@#	/FCCE{NAME}{?}~/WDR.{DOWN 8}~{DOWN}{EDIT}{HOME}{DEL}			
		@CHAR(2)&"{END}"{CALC}~{UP}{END}{DOWN}{DOWN}!{DOWN}			
45	loop2@#	[C]ombine more macros [M]ake library [Q]uit			
		{GET key@#}{ESC}			
46	!	{IF @UPPER(key@#)="Q"}{QUIT}			
47	!	{IF @UPPER(key@#)="C"}{BRANCH loop1@#}			
48	!	{IF @UPPER(key@#)="M"}{GOTO}counter@#~{DOWN 2}{LEFT}			
		/RNLR{END}{DOWN}~/RND\Z~/RND!~/AIMACROMGR~S{?}~.{END}			
		{DOWN}{RIGHT 8}~~{BRANCH libmake}			
49	!	{BRANCH loop2@#}			

The routine starts with `{GOTO}counter@#~` that moves the cell pointer to the end of this macro and then issues `{LEFT}{DOWN 2}` to move the cell pointer two rows below this macro code. Now the macro issues `/FCCE` to start the file combine procedure, `{NAME}{?}` to display a

full screen list of all the files in the default directory, and allow you to use the direction keys to point to the file name. When you press the ENTER key, the macro issues `/WDR.{DOWN 8}~` deleting the first nine rows of the newly combined macro. Then it uses

```
{DOWN}{EDIT}{HOME}{DEL}@CHAR(2)&"{END}"{CALC}~
```

to add the `@CHAR(2)` character to the macro name. We have studied this technique earlier in the [general@#] routine; therefore you are referred to the [general@#] routine in this macro for detailed information. Next the macro issues `{UP}{END}{DOWN}{DOWN}!` to move the cell pointer to the end of the combined macro and then one more cell down, and enters the exclamation mark "`!`" to the cell to create continuity between this combined macro and the next combined macro. The macro, then, uses `{DOWN}` to move the cell pointer one cell down to the location for the next macro to be combined. Now the macro types the following

```
[C]ombine more macros  [M]ake library  [Q]uit
```

to prompt you, and issues `{GET key@#}` halting macro execution and keeping the message in the panel until you press a key. Lotus, then stores the key in [key@#]. Now the macro starts a series of `{IF}` conditions to check the key you pressed. The `{IF @UPPER(key@#)="Q"}` macro command uses the `@UPPER(key@#)` Lotus function to check if you pressed the "q" or the "Q" key. If so, the macro issues `{QUIT}` and quits. The `{IF @UPPER(key@#)="C"}` macro command uses the `@UPPER(key@#)` Lotus function to check if you pressed the "c" or the "C" key. If so, the macro issues `{BRANCH loop1@#}` to allow you to combine a file again. The `{IF @UPPER(key@#)="M"}` macro command uses the `@UPPER(key@#)` Lotus function to check if you pressed the "m" or the "M" key. If so, the macro issues `{GOTO}counter@# ~{DOWN 2}{LEFT}` positioning the cell pointer on the first cell of the first combined macro, and then

```
/RNLR{END}{DOWN}~/RND\Z~/RND!~/AIMACROMGR~S{?}~.{END}{DOWN}{RIGHT 8}~~
```

to save the group of combined macros as one MLB macro. We have seen this same code when we have studied the [general@#] routine. Next the macro issues `{BRANCH libmake}` to return to the beginning to allow you to create more MLB macros. If none of the previous conditions is true, the macro issues `{BRANCH loop2@#}` and displays the message again. This is another example of how we can use a macro to control your input.

Summary: The macro combines the macros in the default directory and inserts the exclamation mark "!" between them to create a contiguous column that allows the macro to use `/RNLR.{END}{DOWN}~` to assign all the range names in the first column in one session. Notice that this macro works only with macros that conform to the structure of macros used in ***Super Power***.

The fourth menu option is [**L**ib_Combine]:

```
Lib_Combine
Combine small *.MLB macros to create one large *.MLB library
{combinea@#}
```

This menu option allows you to combine MLB macros into one MLB macro thereby overcoming the limitation of ten macros in memory imposed by Lotus. The code in this menu option contains only the `{combinea@#}` routine command that activates the [combinea@#] routine.

```
     A              B                C              D              E
53 combinea@#     {GOTO}counter@#~{LEFT}{DOWN 2}
54 loop1a@#       /AIMACROMGR~L{NAME}{?}~E~O~{END}{DOWN}{DOWN}!~{DOWN}
55 loop2a@#        [C]ombine more macros  [M]ake library  [Q]uit
                  {GET key@#}{ESC}
56 !              {IF @UPPER(key@#)="Q"}{QUIT}
57 !              {IF @UPPER(key@#)="C"}/AIMACROMGR~R~{BRANCH loop1a@#}
58 !              {IF @UPPER(key@#)="M"}{GOTO}counter@#~{DOWN 2}{LEFT}
                  /AIMACROMGR~S{?}~.{END}{DOWN}{RIGHT 8}~~{BRANCH libmake}
59 !              {BRANCH loop2a@#}
```

The [combinea@#] routine starts with `{GOTO}counter@#~{LEFT}{DOWN 2}` that move the cell pointer two rows below the code of this macro. The macro issues `/AIMACROMGR~L` invoking the MACROMGR.ADN add-in and initiating the **L**oad Lotus menu option. Next the macro issues `{NAME}{?}` that display a full screen list of all the MLB macros in the default directory and waits. You use the direction keys to point to the file to load, and then ENTER. Then the macro types:

```
[C]ombine more macros  [M]ake library  [Q]uit
```

to the panel to prompt you, and issues `{GET key@#}` that halts macro execution and keeps the message in the panel until you press a key. When you press a key, Lotus stores the key in [key@#]. Now the macro starts a series of `{IF}` conditions to check the key you pressed. The `{IF @UPPER(key@#)="Q"}` macro command uses the `@UPPER(key@#)` Lotus function to check if you pressed the "q" or the "Q" key. If so, the macro issues `{QUIT}` and quits. The `{IF @UPPER(key@#)="C"}` macro command uses the `@UPPER(key@#)` Lotus function to check if you pressed the "c" or the "C" key. If so, the macro issues `/AIMACROMGR~R~{BRANCH loop1a@#}` to invoke the MACROMGR.ADN add-in, remove the macro from the memory, and branch to the [loop1a@#] routine to allow you to load another MLB macro. The `{IF @UPPER(key@#)="M"}` macro command uses the `@UPPER(key@#)` Lotus function to check if you pressed the "m" or the "M" key. If so, the macro issues `{GOTO}counter@#~{DOWN 2}{LEFT}` that position the cell pointer on the first cell of the first combined macro, and then issues:

```
/AIMACROMGR~S{?}~.{END}{DOWN}{RIGHT 8}~~
```

to save the group of combined macros as one MLB macro. We have seen this same code when we have studied the [general@#] routine. Next the macro issues `{BRANCH libmake}` to return to the beginning, allowing you to create more MLB macros. If none of the previous conditions is true the macro issues `{BRANCH loop2a@#}` and displays the message again. This is another example of how we can use a macro to control your input. We have not, yet, looked into the error handling routine; therefore let's begin with the [err1@#] routine

```
     A              B                C              D              E
35 err1@#         {BEEP}{PANELON}File already exists! or illegal macro...
                  Press any key {GET key@#}{ESC}{IF flag@#=1}
                  {BRANCH remove@#}
36 !              File already exists! or illegal macro... Press any key
                  {GET key@#}{ESC}{IF flag@#=2}{BRANCH remove@#}
37 !
```

The [err1@#] routine issues a `{BEEP}` when an error occurs, and issues `{PANELON}` to free the panel activity. Then the macro writes:

```
File already exists! or illegal macro... Press any key
```

to the screen and issues `{GET key@#}` that halts the macro and waits until you press a key. When you presses a key, the macro issues `{ESC}` to clear the message from the panel before it can be written to the cell. Then the macro uses `{IF flag@#=1}` to check the value in [flag@#]. If the value is "`1`", the macro issues `{BRANCH remove@#}` starting the [remove@#] routine, we studied earlier, removing the MLB file from the memory. If the value in [flag@#] is not "`1`", the macro uses the same code again and displays the same message again. Something is wrong with this code. It seem like we can just use `{IF flag@#=1#OR#flag@#=2}{BRANCH remove@#}` to do the same job. However, if we do not repeat the code as it appears here, the macro does not work correctly. We encourage the interested reader to research this macro further. The code, as it appears here, should display the message twice if the value in [flag@#] is "`2`", but when we used this macro, it did not happen. It seems that Lotus does not execute the first `{ONERROR}` macro command.

	A	B	C	D	E
38	err2@#	{BEEP}{PANELON}File already exists! or illegal macro... Press any key GET key@#}{ESC}{IF flag@#=1} {BRANCH auto@#}			
39	!	File already exists! or illegal macro... Press any key {GET key@#}{ESC}{IF flag@#=2}{BRANCH auto@#}			

The [err2@#] error handling routine uses similar code to the [err1@#] error handling routine; therefore we will not go into it.

Because this macro is complicated we show a full list of all the cell contents.

```
A1: U '*---A macro to create library macros (*.MLB) from the macros (*.WK1)
      in
A2: U '    the default directory to be used with the MACROMGR.ADN add-in.
      The
A3: U '    macros must follow the rules given in Super Power for creating
      macros
A4: U '    compatible with the SUPER MACROS LIBRARY.
A5: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A6: '    range names in this column (starts with the \X macro name)
A7: '*---Hold the [ALT] key and press [X] to activate the macro
A8: U '                 THIS MACRO WORKS ONLY WITH LOTUS 2.2 to 2.4
A9: U '
A10: U '\X
B10: '{BREAKON}
A11: U 'LIBMAKE
B11: '{GOTO}counter@#~{LEFT}{DOWN 2}{LET counter@#,-1}~{MENUBRANCH menu@#}
A12: '!
A13: 'menu@#
B13: 'Auto
C13: 'One_at_a_time
D13: 'Combine *.wk1
E13: 'Lib_Combine
F13: 'Quit
A14: '!
B14: 'Create library macros (*.MLB) from macros (*.WK1) in one run
C14: 'Create library macros from macros in the default directory one at a
     time
D14: 'Combine *.WK1 macros to create one large *.MLB library
E14: 'Combine small *.MLB macros to create one large *.MLB library
F14: 'Quit the macro
A15: '!
B15: '{GETNUMBER "How many macros to create? ",num@#}~
C15: '{LET flag@#,2}~{semiauto@#}
D15: '{combine@#}
E15: '{combinea@#}
```

```
A16: '!
B16: '{LET flag@#,1}~{auto@#}
A17: '!
A18: 'semiauto@#
B18: '/FCCE{NAME}{?}~{WINDOWSOFF}{PANELOFF}{BRANCH general@#}
A19: '!
A20: 'auto@#
B20: '{RESTART}{LET counter@#,counter@#+1}~{RECALC auto1@#}{IF counter@#=
     num@#}
     {QUIT}
A21: 'auto1@#
B21: U +"/FCCE{RIGHT "&@STRING(B61,0)&"}~{WINDOWSOFF}{PANELOFF}
       {BRANCH general@#}"
A22: '!
A23: U 'general@#
B23: '{ONERROR err2@#}/WDR{DOWN 8}~{DOWN}/C~nam@#~/C~nam2@#~{EDIT}{HOME}
     {DEL}@CHAR(2)&"{END}"{CALC}~{UP}/RNLR{END}{DOWN}~/RND\Z~/RNC!~~/RND!~
A24: '!
B24: '{ONERROR err1@#}/AIMACROMGR~S*{ESC}
A25: 'nam@#
B25: U 'LIBMAKE
A26: '!
B26: '{PANELON}~.{END}{DOWN}{BIGRIGHT 2}~~
A27: 'remove@#
B27: '/AIMACROMGR~R
A28: 'nam2@#
B28: U 'LIBMAKE
A29: '!
B29: '~
A30: '!
B30: '{IF flag@#=1}{WINDOWSON}{PANELON}{BRANCH auto@#}
A31: '!
B31: '{IF flag@#=2}{WINDOWSON}{PANELON}{BRANCH semiauto@#}
A32: '!
A33: 'flag@#
B33: 1
A34: '!
A35: 'err1@#
B35: U '{BEEP}{PANELON}File already exists! or illegal macro... Press any
       key{GET key@#}{ESC}{IF flag@#=1}{BRANCH remove@#}
A36: '!
B36: U 'File already exists! or illegal macro... Press any key {GET key@#}
       {ESC}{IF flag@#=2}{BRANCH remove@#}
A37: '!
A38: 'err2@#
B38: U '{BEEP}{PANELON}File already exists! or illegal macro... Press any
       key{GET key@#}{ESC}{IF flag@#=1}{BRANCH auto@#}
A39: '!
B39: U 'File already exists! or illegal macro... Press any key {GET key@#}
       {ESC}{IF flag@#=2}{BRANCH auto@#}
A40: '!
A41: 'num@#
B41: 3
A42: '!
A43: 'combine@#
B43: '{GOTO}counter@#~{LEFT}{DOWN 2}
A44: 'loop1@#
B44: '/FCCE{NAME}{?}~/WDR.{DOWN 8}~{DOWN}{EDIT}{HOME}{DEL}@CHAR(2)&"{END}"
     {CALC}~{UP}{END}{DOWN}{DOWN}!{DOWN}
A45: 'loop2@#
B45: ' [C]ombine more macros  [M]ake library  [Q]uit {GET key@#}{ESC}
A46: '!
B46: '{IF @UPPER(key@#)="Q"}{QUIT}
A47: '!
B47: '{IF @UPPER(key@#)="C"}{BRANCH loop1@#}
A48: '!
B48: '{IF @UPPER(key@#)="M"}{GOTO}counter@#~{DOWN 2}{LEFT}/RNLR{END}{DOWN}~
      /RND\Z~/RND!~/AIMACROMGR~S{?}~.{END}{DOWN}{RIGHT 8}~~{BRANCH libmake}
A49: '!
B49: '{BRANCH loop2@#}
A50: '!
```

```
A51: 'key@#
B51: 'M
A52: '!
A53: 'combinea@#
B53: '{GOTO}counter@#~{LEFT}{DOWN 2}
A54: 'loop1a@#
B54: '/AIMACROMGR~L{NAME}{?}~E~O~{END}{DOWN}{DOWN}!~{DOWN}
A55: 'loop2a@#
B55: ' [C]ombine more macros  [M]ake library  [Q]uit {GET key@#}{ESC}
A56: '!
B56: '{IF @UPPER(key@#)="Q"}{QUIT}
A57: '!
B57: '{IF @UPPER(key@#)="C"}/AIMACROMGR~R~{BRANCH loop1a@#}
A58: '!
B58: '{IF @UPPER(key@#)="M"}{GOTO}counter@#~{DOWN 2}{LEFT}/AIMACROMGR~S{?}~
     .{END}{DOWN}{RIGHT 8}~~{BRANCH libmake}
A59: '!
B59: '{BRANCH loop2a@#}
A60: '!
A61: 'counter@#
B61: U 2
```

Modifying Macros

[6] Replace All ERR Values or NA Values with Zeros

```
     A              B                C              D              E
 1 *---A macro to REPLACE all ERR or NA in a 3-D or 2-D range with ZEROS
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                    IT WILL WORK IN RELEASE 2.0 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 ZEROERR          {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                    Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                    {LET counterb179,0}~
12 cont179          {LET hereabs179,@CELLPOINTER("address")}~
                    {LET counter1179,0}
13 !                {FOR counter1179,0,@COLS(Which range ?)-1,1,labels1179}
14 !                {LET rel179,@INFO("release")}~{IF @LEFT(rel179,1)<>"@"}
                    {GOTO}{hereabs179}~{LET counterb179,counterb179+1}~
                    {IF counterb179<@SHEETS(Which range ?)}{NS}{GOTO}
                    {hereabs179}~{BRANCH cont179}
15 !                {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter1179                 4
18 counter1a179                0
19 labels1179       {RIGHT}{LET here179,@CELLPOINTER("address")}~{LEFT}
                    {FOR counter1a179,0,@ROWS(Which range ?)-1,1,
                    labels1a179}~{IF counter1179<@COLS(Which range ?)-1}
                    {GOTO}{here179}~{LET counter1a179,0}~
20 !
21 here179          $F$1
22 !
23 labels1a179      {IF @ISERR(@CELLPOINTER("contents"))=1}0~
24 !                {IF @ISNA(@CELLPOINTER("contents"))=1}0~
25 !                {DOWN}
26 !
27 counterb179                  5
28 hereabs179       $A$1
29 !
30 rel179
```

This macro turns all the cells containing ERR or NA into zero. The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to process, and simultaneously assigns the [Which range ?] name to the same range, which the macro uses as a prompt. Next the macro issues {LET counterb179,0}~ setting the content of the B27 cell named [counterb179] to zero, which serves as a counter.

```
     A              B                C              D              E
12 cont179          {LET hereabs179,@CELLPOINTER("address")}~
                    {LET counter1179,0}
13 !                {FOR counter1179,0,@COLS(Which range ?)-1,1,labels1179}
14 !                {LET rel179,@INFO("release")}~{IF @LEFT(rel179,1)<>"@"}
                    {GOTO}{hereabs179}~{LET counterb179,counterb179+1}~
                    {IF counterb179<@SHEETS(Which range ?)}{NS}{GOTO}
                    {hereabs179}~{BRANCH cont179}
15 !                {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro is finished, the macro issues {LET hereabs179,@CELLPOINTER("address")}~, which store the current cell pointer address in cell [hereabs179], and then it issues {LET counter1179,0} to set the content of

[counter1179] to zero, which also serves as a counter.

The macro issues the {FOR counter1179,0,@COLS(Which range ?)-1,1,labels1179} loop command to activate the [labels1179] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels1179] routine.

	A	B	C	D	E
19	labels1179	{RIGHT}{LET here179,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a179,0,@ROWS(Which range ?)-1,1,labels1a179}~			
		{IF counter1179<@COLS(Which range ?)-1}{GOTO}{here179}~			
		{LET counter1a179,0}~			

The routine uses {RIGHT}{LET here179,@CELLPOINTER("address")}~{LEFT} to record the address of the first cell of the next column in [here179]. This way, when the current column processing is finished, the macro moves the cell pointer directly to the top of the next column using the address stored in [here179]. The {FOR counter1a179,0, @ROWS(Which range ?)-1,1,labels1a179}~ commands activate the [labels1a179] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
23	labels1a179	{IF @ISERR(@CELLPOINTER("contents"))=1}0~			
24	!	{IF @ISNA(@CELLPOINTER("contents"))=1}0~			
25	!	{DOWN}			

The [labels1a179] routine issues {IF @ISERR(@CELLPOINTER("contents"))=1}0~ to check if the current cell content is ERR value. If so, the macro types a zero "0" into the panel and issues the tilde "~", which is the same as ENTER from the keyboard. Next the macro issues {IF @ISNA(@CELLPOINTER("contents"))=1}0~ to check if the cell contains the NA value. If so, the macro again types a zero "0" into the panel and issues the tilde "~". Next the macro issues {DOWN} to move the cell pointer down to the next cell. When the [labels1a179] routine ends, the macro returns the control to the {FOR} loop command in the [labels1179] routine to start the process of the next cell in the current column.

	A	B	C	D	E
19	labels1179	{RIGHT}{LET here179,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a179,0,@ROWS(Which range ?)-1,1,			
		labels1a179}~{IF counter1179<@COLS(Which range ?)-1}			
		{GOTO}{here179}~{LET counter1a179,0}~			

When the value in [counter1a179] reaches the number of rows in the [Which range ?] range minus one, the macro issues {IF counter1179<@cols(Which range ?)-1} to check how many column were processed. If the value in [counter407] is less than the number of columns in the [Which range ?] range, the macro issues the indirect {GOTO}{here179}~ macro command to move the cell pointer to the first cell of the next column. Next the macro issues {LET counter1a179,0}~ to reset the value in [counter1a179] to zero. When the [labels1179] routine is finished the macro returns the control back to the [cont179] routine.

	A	B	C	D	E
12	cont179	{LET hereabs179,@CELLPOINTER("address")}~			
		{LET counter1179,0}			
13	!	{FOR counter1179,0,@COLS(Which range ?)-1,1,labels1179}			
14	!	{LET rel179,@INFO("release")}~{IF @LEFT(rel179,1)<>"@"}			
		{GOTO}{hereabs179}~{LET counterb179,counterb179+1}~			
		{IF counterb179<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs179}~{BRANCH cont179}			

```
15 !              {GOTO}Which range ?~/RNDWhich range ?~
```

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First, the macro issues `{LET rel179,@INFO("release")}~`, which stores the result of the `@INFO("release")` 3-D function in cell [rel179], and then `{IF @LEFT (rel179,1)<>"@"}`, which checks the first character of the content of [rel179]. If "`@`" is the character, then you are using a 2-D release, otherwise you are using a 3-D release which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the `{GOTO}{hereabs179}~` indirect command which moves the cell pointer to the origin address of the macro.

The macro continues with `{LET counterb179,counterb179+1}~`, which increase the value in [counterb179] by one. Now the macro issues `{IF counterb179<@SHEETS (Which range ?)}` to check if the value in [counterb179] is less than the number of sheets in [Which range ?]. If so, there are more sheets to process. Therefore the macro issues `{NS}` to move the cell pointer to the next sheet. Next the macro issues the indirect `{GOTO}{hereabs179}~` command, which moves the cell pointer to the same address as the address of the upper left cell of the first sheet of the [Which range ?] range, but in the new sheet. Last, the macro issues `{BRANCH cont179}` to process the cells of [Which range ?] in the new sheet. When all the sheets are processed, the macro issues `{GOTO}Which range ?~` moving the cell pointer to the upper left cell of the first sheet of the [Which range ?] range, and then `/RNDWhich range ?~` to delete the temporary [Which range ?] range name to leave a clean worksheet.

[3] RECALC a Range

	A	B	C	D	E
1	*---A macro to RECALC a range				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	RECLCRNG	{WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND			
		Which range ?~/RNC{PANELON}{WINDOWSON}Which range ?~			
		{BS}{?}~			
12	!	{PANELOFF}{RECALC Which range ?}			
13	!	/RNDWhich range ?~			

When a worksheet application grows, it slows down all operations, We suggest you set the default global recalculation to manual, which avoids the recalculation of the worksheet after every operation. Many times a range must be recalculated to reflect the changes made to the worksheet. Changing the recalculation to automatic is not the best solution, because pressing the F9 (CALC) key calculates the whole worksheet, which may take long time. This macro uses the `{RECALC range}` macro command to recalculate only a selected range, even if the worksheet is in the manual recalculation mode.

The macro starts with the `{WINDOWSOFF}{PANELOFF}` commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to recalculate, and simultaneously assigns the [Which range ?] name to the same range, which acts as a prompt. Next the macro issues `{PANELOFF}{RECALC Which range ?}` updating all the formulas in the [Which range ?] range. Last, the macro issues `/RNDWhich range ?~` to delete the temporary [Which range ?] range name and leave a clean worksheet.

[6] Erase Only the Numbers in a Range

```
     A                B                C                D                E
 1 *---A macro to ERASE all NUMBERS in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z             {BREAKON}
11 DELNUMBR       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{PANELON}Which range ?~{WINDOWSON}
                  {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb403,0}~
12 cont403        {LET hereabs403,@CELLPOINTER("address")}~
                  {LET counter403,0}
13 !              {FOR counter403,0,@COLS(Which range ?)-1,1,labels403}
14 !              {LET rel403,@INFO("release")}~{IF @LEFT(rel403,1)<>"@"}
                  {GOTO}{hereabs403}~{LET counterb403,counterb403+1}~
                  {IF counterb403<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs403}~{BRANCH cont403}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter403           3
18 countera403          5
19 labels403      {RIGHT}{LET here403,@CELLPOINTER("address")}~{LEFT}
                  {FOR countera403,0,@ROWS(Which range ?)-1,1,labelsa403}~
                  {IF counter403<@COLS(Which range ?)-1}{GOTO}{here403}~
                  {LET countera403,0}~
20 !
21 here403        $D$1
22 !
23 labelsa403     {PANELON}{IF @CELLPOINTER("type")<>"v"}{DOWN}{RETURN}
24 !              /RE~{DOWN}
25 !
26 counterb403          4
27 hereabs403     $A$1
28 !
29 rel403
```

This macro will erase all the cells in a range which contain numbers. When the macro finds a label in the range, it skips and moves to the next cell. The macro starts with the `{PANELOFF}{WINDOWSOFF}` macro commands to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to process and simultaneously assigns the [Which range ?] name to the same range, which acts as a prompt. Next the macro issues `{LET counterb403,0}~` which set the content of the B26 cell named [counterb403] to zero, which serves as a counter.

```
     A                B                C                D                E
12 cont403        {LET hereabs403,@CELLPOINTER("address")}~
                  {LET counter403,0}
13 !              {FOR counter403,0,@COLS(Which range ?)-1,1,labels403}
14 !              {LET rel403,@INFO("release")}~{IF @LEFT(rel403,1)<>"@"}
                  {GOTO}{hereabs403}~{LET counterb403,counterb403+1}~
                  {IF counterb403<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs403}~{BRANCH cont403}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro is finished, the macro issues `{LET hereabs403,@CELLPOINTER("address")}~`, which store the current cell pointer address in cell [hereabs403], and then it issues `{LET counter403,0}` to set the content of [counter403]

to zero, which also serves as a counter.

The macro issues the {FOR counter403,0,@COLS(Which range ?)-1,1,labels403} loop command which activates the [labels403] routine as many times as the number of columns in the [Which range ?] range. Therefore before we continue with this code, let's look at the [labels403] routine.

	A	B	C	D	E
19	labels403	{RIGHT}{LET here403,@CELLPOINTER("address")}~{LEFT}			
		{FOR countera403,0,@ROWS(Which range ?)-1,1,labelsa403}~			
		{IF counter403<@COLS(Which range ?)-1}{GOTO}{here403}~			
		{LET countera403,0}~			

The routine uses {RIGHT}{LET here403,@CELLPOINTER("address")}~{LEFT} to record the address of the first cell of the next column in cell [here403]. This way, when the current column processing is finished, the macro uses the address stored in [here403] to move the cell pointer directly to the top of the next column. The {FOR countera403,0,@ROWS (Which range ?)-1,1,labelsa403}~ macro command activates the [labelsa403] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
23	labelsa403	{PANELON}{IF @CELLPOINTER("type")<>"v"}{DOWN}{RETURN}			
24	!	/RE~{DOWN}			

The [labelsa403] routine issues {IF @CELLPOINTER("type")<>"v"} to check if the current cell content is not a value. If so, the macro issues {DOWN}, which moves the cell pointer one cell down, and then {RETURN}, which routes the macro execution back to the {FOR} loop. If the condition is false, the macro issues /RE~ to erase the content of the cell (a value) and then {DOWN} to move the cell pointer down to the next cell to process. When the [labelsa403] routine ends, the macro returns the control to the {FOR} loop command in the [labels403] routine to start the process of the next cell in the current column.

	A	B	C	D	E
19	labels403	{RIGHT}{LET here403,@CELLPOINTER("address")}~{LEFT}			
		{FOR countera403,0,@ROWS(Which range ?)-1,1,labelsa403}~			
		{IF counter403<@COLS(Which range ?)-1}{GOTO}{here403}~			
		{LET countera403,0}~			

When the value in [countera403] reaches the number of rows in the [Which range ?] range minus one, the macro issues{IF counter403<@COLS(Which range ?)-1} to check how many columns were processed. If the value in [counter403] is less than the number of columns in [Which range ?], the macro issues the indirect {GOTO}{here403}~ command, which moves the cell pointer to the first cell of the next column. Next the macro issues {LET countera403,0}~ to reset the value in [countera403] to zero. When the [labels403] routine is finished, the macro returns control back to the [cont403] routine.

	A	B	C	D	E
12	cont403	{LET hereabs403,@CELLPOINTER("address")}~			
		{LET counter403,0}			
13	!	{FOR counter403,0,@COLS(Which range ?)-1,1,labels403}			
14	!	{LET rel403,@INFO("release")}~{IF @LEFT(rel403,1)<>"@"}			
		{GOTO}{hereabs403}~{LET counterb403,counterb403+1}~			
		{IF counterb403<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs403}~{BRANCH cont403}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First, the macro issues `{LET rel403,@INFO("release")}~`, which stores the result of the `@INFO("release")` 3-D function in cell [rel403] and then `{IF @LEFT (rel403,1)<>"@"}`, which checks the first character of the content of [rel403]. If "`@`" is the character it means that you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the `{GOTO}{hereabs403}~` indirect command which moves the cell pointer to the origin address of the macro.

The macro continues with `{LET counterb403,counterb403+1}~` which increase the value in [counterb403] by one. Now the macro issues `{IF counterb403<@SHEETS(Which range ?)}` to check if the value in [counterb403] is less than the number of sheets in the [Which range ?] range. If so, there are more sheets to process. Therefore the macro issues `{NS}` to move the cell pointer to the next sheet. Next the macro issues the indirect `{GOTO} {hereabs403}~` macro command, which moves the cell pointer to the same address as the upper left cell of the first sheet of the [Which range ?] range but in the new sheet. Last, the macro issues `{BRANCH cont403}` to process the cells in the new sheet of [Which range ?]. When all the sheets are processed, the macro issues `{GOTO}Which range ?~`, which moves the cell pointer to the upper left cell of the first sheet of [Which range ?] and then `/RND Which range ?~` macro command to delete the temporary [Which range ?] range name and leave a clean worksheet.

[6] Erase Only the Labels Inside a Range

```
     A                B                C                D                E
 1 *---A macro to ERASE all LABELS in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z             {BREAKON}
11 DELLABEL       {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                  {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb405,0}~
12 cont405        {LET hereabs405,@CELLPOINTER("address")}~
                  {LET counter405,0}
13 !              {FOR counter405,0,@COLS(Which range ?)-1,1,labels405}
14 !              {LET rel405,@INFO("release")}~{IF @LEFT(rel405,1)<>"@"}
                  {GOTO}{hereabs405}~{LET counterb405,counterb405+1}~
                  {IF counterb405<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs405}~{BRANCH cont405}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter405            3
18 countera405           5
19 labels405      {RIGHT}{LET here405,@CELLPOINTER("address")}~{LEFT}
                  {FOR countera405,0,@ROWS(Which range ?)-1,1,labelsa405}~
                  {IF counter405<@COLS(Which range ?)-1}{GOTO}{here405}~
                  {LET countera405,0}~
20 !
21 here405        $D$1
22 !
23 labelsa405     {PANELON}{IF @CELLPOINTER("type")<>"l"#OR#@CELLPOINTER
                  ("prefix")="\"#OR#@CELLPOINTER("prefix")="|"}{DOWN}
                  {RETURN}
24 !              {PANELOFF}/RE~{DOWN}
25 !
26 counterb405            4
27 hereabs405     $A$1
28 !
29 rel405
```

This macro will erase all the cells in a range which contains labels. When the macro finds a cell containing a value or a blank cell, it skips and moves to the next cell. The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to process and simultaneously assigns the [Which range ?] name to the same range, which acts as a prompt. Next it issues `{LET counterb405,0}~`, setting the content of the B26 cell named [counterb405] to zero, which serves as a counter.

```
     A                B                C                D                E
12 cont405        {LET hereabs405,@CELLPOINTER("address")}~
                  {LET counter405,0}
13 !              {FOR counter405,0,@COLS(Which range ?)-1,1,labels405}
14 !              {LET rel405,@INFO("release")}~{IF @LEFT(rel405,1)<>"@"}
                  {GOTO}{hereabs405}~{LET counterb405,counterb405+1}~
                  {IF counterb405<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs405}~{BRANCH cont405}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro is finished, the macro issues `{LET`

hereabs405,@CELLPOINTER("address")}~, which store the current cell pointer address in cell [hereabs405], and then issues {LET counter405,0} to set the content of [counter405] to zero, which also serves as a counter. The macro issues the {FOR counter405,0,@COLS(Which range ?)-1,1,labels405} loop command which activates the [labels405] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels405] routine.

	A	B	C	D	E
19	labels405	{RIGHT}{LET here405,@CELLPOINTER("address")}~{LEFT} {FOR countera405,0,@ROWS(Which range ?)-1,1,labelsa405}~ {IF counter405<@COLS(Which range ?)-1}{GOTO}{here405}~ {LET countera405,0}~			

The routine uses {RIGHT}{LET here405,@CELLPOINTER("address")}~{LEFT} to record the address of the first cell of the next column in cell [here405]. This way, when the current column processing is finished, the macro uses the address stored in [here405] to move the cell pointer directly to the top of the next column. The {FOR countera405,0,@ROWS (Which range ?)-1,1,labelsa405}~ macro commands activate the [labelsa405] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
23	labelsa405	{PANELON}{IF @CELLPOINTER("type")<>"l"#OR#@CELLPOINTER ("prefix")="\"#OR#@CELLPOINTER("prefix")="|"}{DOWN} {RETURN}			
24	!	{PANELOFF}/RE~{DOWN}			

The [labelsa405] routine issues

```
{IF @CELLPOINTER("type")<>"l"}#OR#@CELLPOINTER("prefix")="\"#OR#
    @CELLPOINTER("prefix")="|"}
```

to check if the current cell content is not a label and the prefix of the content of the label is not the back slash "\" character or the split bar "|". The back slash is mostly used to produce separating lines due to its unique property to repeat the characters that follows it to all the width of the column, but Lotus recognizes it as a label prefix. The split bar is used as an embedded printer command, but again is recognized by Lotus as a label prefix. If so, the macro issues {DOWN}, which moves the cell pointer one cell down and then issues {RETURN}, which routes the macro execution back to the {FOR} loop. If the condition is false, the macro issues /RE~ to erase the content of the cell (a label) and then issues {DOWN} to move the cell pointer down to the next cell to process. When the [labelsa405] routine ends, the macro returns the control to the {FOR} loop command in the [labels405] routine to start the process of the next cell in the current column.

	A	B	C	D	E
19	labels405	{RIGHT}{LET here405,@CELLPOINTER("address")}~{LEFT} {FOR countera405,0,@ROWS(Which range ?)-1,1,labelsa405}~ {IF counter405<@COLS(Which range ?)-1}{GOTO}{here405}~ {LET countera405,0}~			

When the value in cell [countera405] reaches the number of rows in the [Which range ?] range minus one, the macro issues {IF counter405<@COLS(Which range ?)-1} to check how many columns were processed. If the value in[counter405] is less than the number of columns in [Which range ?], the macro issues the indirect {GOTO}{here405}~ macro command, which moves the cell pointer to the first cell of the next column. Next the macro issues {LET

countera405,0}~ to reset the value in cell [countera405] to zero. When the [labels405] routine is finished, the macro returns the control back to the [cont405] routine.

	A	B	C	D	E
12	cont405	{LET hereabs405,@CELLPOINTER("address")}~			
		{LET counter405,0}			
13	!	{FOR counter405,0,@COLS(Which range ?)-1,1,labels405}			
14	!	{LET rel405,@INFO("release")}~{IF @LEFT(rel405,1)<>"@"}			
		{GOTO}{hereabs405}~{LET counterb405,counterb405+1}~			
		{IF counterb405<@SHEETS(Which range ?)}{NS}{GOTO}			
		{hereabs405}~{BRANCH cont405}			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First, the macro issues {LET rel405,@INFO("release")}~, which store the result of the @INFO("release") 3-D function in cell [rel405] and then {IF @LEFT (rel405,1)<>"@"}, which checks the first character of the content of [rel405]. If "@" is the character, you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in the [Which range ?] range. Therefore the macro issues the {GOTO} {hereabs405}~ indirect command, which moves the cell pointer to the origin address of the macro.

The macro continues with {LET counterb405,counterb405+1}~ which increase the value in cell [counterb405] by one. Now the macro issues {IF counterb405<@SHEETS (Which range ?)} to check if the value in [counterb405] is less than the number of sheets in the [Which range ?] range. If so, there are more sheets to process. Therefore the macro issues {NS} to move the cell pointer to the next sheet. Next the macro issues the indirect {GOTO} {hereabs405}~ command, which moves the cell pointer to the same address as the upper left cell of the first sheet of [Which range ?], but in the new sheet. Last, the macro issues {BRANCH cont405} to process the cells in the new sheet of [Which range ?]. When all the sheets are processed, the macro issues {GOTO}Which range ?~, which moves the cell pointer to the upper left cell of the first sheet of [Which range ?] and then issues /RND Which range ?~ to delete the temporary [Which range ?] range name and leave a clean worksheet.

[6] Erase Only the Zeros in a Range

```
     A              B                C                D              E
 1 *---A macro to ERASE all ZEROS in a 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 DELZEROS        {WINDOWSOFF} {PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb406,0}~
12 cont406         {LET hereabs406,@CELLPOINTER("address")}~
                   {LET counter406,0}
13 !               {FOR counter406,0,@COLS(Which range ?)-1,1,labels406}
14 !               {LET rel406,@INFO("release")}~{IF @LEFT(rel406,1)<>"@"}
                   {GOTO}{hereabs406}~{LET counterb406,counterb406+1}~
                   {IF counterb406<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs406}~{BRANCH cont406}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter406             3
18 countera406            5
19 labels406       {RIGHT}{LET here406,@CELLPOINTER("address")}~{LEFT}
                   {FOR countera406,0,@ROWS(Which range ?)-1,1,labelsa406}~
                   {IF counter406<@COLS(Which range ?)-1}{GOTO}{here406}~
                   {LET countera406,0}~
20 !
21 here406         $D$1
22 !
23 labelsa406      {PANELON}{IF @CELLPOINTER("contents")=0#AND#
                   @CELLPOINTER("type")<>"l"}{PANELOFF}/RE~
24 !               {DOWN}
25 !
26 counterb406            4
27 hereabs406      $A$1
28 !
29 rel406          3.00.00
```

This macro will erase only the ZERO values in a range. The range can contain any type of data but only the ZERO values will be erased leaving the labels and other values intact. The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to process and simultaneously assigns the [Which range ?] range name to the same range, which acts as a prompt. Next the macro issues `{LET counterb406,0}~` setting the content of the B26 cell named [counterb406] to zero, which serves as a counter.

```
     A              B                C                D              E
12 cont406         {LET hereabs406,@CELLPOINTER("address")}~
                   {LET counter406,0}
13 !               {FOR counter406,0,@COLS(Which range ?)-1,1,labels406}
14 !               {LET rel406,@INFO("release")}~{IF @LEFT(rel406,1)<>"@"}
                   {GOTO}{hereabs406}~{LET counterb406,counterb406+1}~
                   {IF counterb406<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs406}~{BRANCH cont406}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro is finished, the macro issues `{LET hereabs406,@CELLPOINTER("address")}~`, which store the current cell pointer address in

cell [hereabs406], and then {LET counter406,0} to set the content of cell [counter406] to zero, which also serves as a counter.

The macro issues the {FOR counter406,0,@COLS(Which range ?)-1,1,labels406} loop command which activates the [labels406] routine as many times as the number of columns in the [Which range ?] range. Before we continue with this code, let's look at the [labels406] routine.

	A	B	C	D	E
19	labels406	{RIGHT}{LET here406,@CELLPOINTER("address")}~{LEFT}			
		{FOR countera406,0,@ROWS(Which range ?)-1,1,labelsa406}~			
		{IF counter406<@COLS(Which range ?)-1}{GOTO}{here406}~			
		{LET countera406,0}~			

The routine uses {RIGHT}{LET here406,@CELLPOINTER("address")}~{LEFT} to record the address of the first cell of the next column in cell [here406]. This way, when the current column processing is finished, the macro uses the address stored in [here406] to move the cell pointer directly to the top of the next column. The {FOR countera406,0,@ROWS (Which range ?)-1,1,labelsa406}~ commands, activate the [labelsa406] routine as many times as the number of rows in the [Which range ?] range.

	A	B	C	D	E
23	labelsa406	{PANELON}{IF @CELLPOINTER("contents")=0#AND#			
		@CELLPOINTER("type")<>"l"}{PANELOFF}/RE~			
24	!	{DOWN}			

The [labelsa406] routine issues

{IF @CELLPOINTER("contents")=0#AND#{IF @CELLPOINTER("type")<>"l"}

to check if the current cell content is not a label and it contains a zero. If so, the macro issues /RE~ to erase the content of the cell (a zero) and then issues {DOWN} to move the cell pointer down to the next cell to process. If the condition is false, the macro issues {DOWN} to move the cell pointer down to the next cell to process. When the [labelsa406] routine ends, the macro returns control to the {FOR} loop command in the [labels406] routine to process the next cell in the current column.

	A	B	C	D	E
19	labels406	{RIGHT}{LET here406,@CELLPOINTER("address")}~{LEFT}			
		{FOR countera406,0,@ROWS(Which range ?)-1,1,labelsa406}~			
		{IF counter406<@COLS(Which range ?)-1}{GOTO}{here406}~			
		{LET countera406,0}~			

When the value in [countera406] reaches the number of rows in the [Which range ?] range minus one, the macro issues {IF counter406<@COLS(Which range ?)-1} to check how many columns were processed. If the value in [counter406] is less than the number of columns in [Which range ?], the macro issues the indirect {GOTO}{here406}~ macro command, which moves the cell pointer to the first cell of the next column. Next the macro issues {LET countera406,0}~ to reset the value in [countera406] to zero. When the [labels406] routine is finished. the macro returns control back to the [cont406] routine.

	A	B	C	D	E
12	cont406	{LET hereabs406,@CELLPOINTER("address")}~			
		{LET counter406,0}			
13	!	{FOR counter406,0,@COLS(Which range ?)-1,1,labels406}			

```
14 !             {LET rel406,@INFO("release")}~{IF @LEFT(rel406,1)<>"@"}
                 {GOTO}{hereabs406}~{LET counterb406,counterb406+1}~
                 {IF counterb406<@SHEETS(Which range ?)}{NS}{GOTO}
                 {hereabs406}~{BRANCH cont406}
15 !             {GOTO}Which range ?~/RNDWhich range ?~
```

When the macro finishes processing the first sheet of the [Which range ?] range (in a 2-D release this is the only sheet), it starts a process to check if you are using a 2-D or a 3-D Lotus release. First, the macro issues `{LET rel406,@INFO("release")}~`, which stores the result of the `@INFO("release")` 3-D function in cell [rel406] and then it issues `{IF @LEFT(rel406,1)<>"@"}`, which checks the first character of the content of [rel406]. If "@" is the character, you are using a 2-D release, otherwise you are using a 3-D release, which may have more than one sheet in [Which range ?]. Therefore the macro issues the `{GOTO}{hereabs406}~` indirect command, which moves the cell pointer to the origin address of the macro.

The macro continues with `{LET counterb406,counterb406+1}~` which increase the value in [counterb406] by one. Now the macro issues `{IF counterb406<@SHEETS (Which range ?)}` to check if the value in [counterb406] is less than the number of sheets in the [Which range ?] range. If so, there are more sheets to process. Therefore the macro issues `{NS}` to move the cell pointer to the next sheet. Next the macro issues the indirect `{GOTO}{hereabs406}~` command, which moves the cell pointer to the same address as the upper left cell of the first sheet of [Which range ?], but in the new sheet. Last, the macro issues `{BRANCH cont406}` to process the cells in the new sheet of [Which range ?]. When all the sheets are processed, the macro issues `{GOTO}Which range ?~`, which moves the cell pointer to the upper left cell of the first sheet of [Which range ?] and then `/RNDWhich range ?~` to delete the temporary [Which range ?] range name and leave a clean worksheet.

[6] Change All the Labels in a Range to UPPERCASE Form

```
     A                B                C                D                E
 1 *---A macro to turn all labels in a 3-D or 2-D range to UPPERCASE form
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 UPERCASE        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb048,0}~
12 cont048         {LET hereabs048,@CELLPOINTER("address")}~
                   {LET counter048,0}
13 !               {FOR counter048,0,@COLS(Which range ?)-1,1,labels048}
14 !               {LET rel048,@INFO("release")}~{IF @LEFT(rel048,1)<>"@"}
                   {GOTO}{hereabs048}~{LET counterb048,counterb048+1}~
                   {IF counterb048<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs048}~{BRANCH cont048}
15 !               {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter048             3
18 countera048            5
19 labels048       {RIGHT}{LET here048,@CELLPOINTER("address")}~{LEFT}
                   {FOR countera048,0,@ROWS(Which range ?)-1,1,labelsa048}~
                   {IF counter048<@COLS(Which range ?)-1}{GOTO}{here048}~
                   {LET countera048,0}~
20 !
21 here048         $D$1
22 !
23 labelsa048      {PANELON}{RECALC prop048}{IF @CELLPOINTER("type")<>"b"
                   #AND#@CELLPOINTER("type")<>"v"#AND#@CELLPOINTER("prefix")
                   <>"\"#AND#@CELLPOINTER("prefix")<>"|"}{PANELOFF}
                   /RVprop048~~{EDIT}~
24 !               {DOWN}
25 !
26 prop048         \Z <----- This is the result of @UPPER(@CELLPOINTER
                                                     ("contents"))
27 !
28 counterb048           4
29 hereabs048      $A$1
30 !
31 rel048
```

See **Change All the Labels in a Range to LOWERCASE Form**.

[6] Change All the Labels in a Range to Proper Form

```
     A              B                C                D              E
 1 *---A macro to turn all labels in a 3-D or 2-D range to PROPER form
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                     IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z             {BREAKON}
11 PROPER         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                  Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                  {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                  {LET counterb412,0}~
12 cont412        {LET hereabs412,@CELLPOINTER("address")}~
                  {LET counter412,0}
13 !              {FOR counter412,0,@COLS(Which range ?)-1,1,labels412}
14 !              {LET rel412,@INFO("release")}~{IF @LEFT(rel412,1)<>"@"}
                  {GOTO}{hereabs412}~{LET counterb412,counterb412+1}~
                  {IF counterb412<@SHEETS(Which range ?)}{NS}{GOTO}
                  {hereabs412}~{BRANCH cont412}
15 !              {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 counter412            3
18 countera412           6
19 labels412      {RIGHT}{LET here412,@CELLPOINTER("address")}~{LEFT}
                  {FOR countera412,0,@ROWS(Which range ?)-1,1,labelsa412}~
                  {IF counter412<@COLS(Which range ?)-1}{GOTO}{here412}~
                  {LET countera412,0}~
20 !
21 here412        $D$1
22 !
23 labelsa412     {PANELON}{RECALC prop412}{IF @CELLPOINTER("type")<>"b"
                  #AND#@CELLPOINTER("type")<>"v"#AND#@CELLPOINTER("prefix")
                  <>"\"#AND#@CELLPOINTER("prefix")<>"|"}{PANELOFF}
                  /RVprop412~~{EDIT}~
24 !              {DOWN}
25 !
26 prop412        \Z <----- This is the result of @PROPER(@CELLPOINTER
                                                    ("contents"))
27 !
28 counterb412           2
29 hereabs412     $A$1
30 !
31 rel412
```

See **Change All the Labels in a Range to LOWERCASE Form**.

Cells, Ranges and Cell Pointer Moving Macros

[4] Jump to Any Column and Stay at the Same Screen Position

	A	B	C	D	E
1	*---A macro to jump to other column and stay at the same screen position				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	JUMPCOL	{GETLABEL "Enter column letter (A, B, AA, etc. ...) :" ,coll204}~			
12	!	{WINDOWSOFF}{PANELOFF}{LET rowhere204,@STRING (@CELLPOINTER("row"),0)}~			
13	!	{LET there204,coll204&rowhere204}~			
14	!	/WTB{RESTART}{GOTO}{there204}~/WTC{QUIT}			
15	!				
16	coll204	AA			
17	!				
18	rowhere204	14			
19	!				
20	there204	AA14			

This macro allows you to jump to any column and stay at the same screen position. Normally, when you issue the GOTO command, the target cell is positioned on the upper left cell of the screen. This macro places it on the same screen position, for example: if the cell pointer is in the D14 cell, which in our example is on the 10th row and the 4th column of the screen, and you want to move the cell pointer to the "AA" column, the macro will place the cell pointer in the AA14 cell and on the 10th row and the 4th of the screen.

The macro starts with the `{GETLABEL "Enter column letter (A, B, AA, etc. ...) : ",coll204}` macro command, which displays "`Enter column letter (A, B, AA, etc. ... ):`" in the panel and waits for you to the insert the column's letter and press ENTER. Lotus stores the column's letter in the B16 cell named [coll204]. Next the macro issues `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities while the macro issues `{LET rowhere204,@STRING(@CELLPOINTER("row"),0)}~`, which store the current row number as a string in cell [rowhere204].

Now the macro issues `{LET there204,coll204&rowhere204}~`, which combine the column's letter in cell [coll204] with the row number string in cell [rowhere204] together to create the target address, and store it in cell [there204]. To make sure that the target cell will be in the same row position within the screen, the macro issues the `/WTB` macro keys, which create horizontal and vertical titles. Then the macro issues `{RESTART}`, which clears the subroutine stack to end the macro when the current subroutine ends. This command is placed here in case the macro is used from a macro manager. Next the macro issues the `{GOTO} {there204}~` indirect macro command which moves the cell pointer to the calculated target cell address. Last the macro issues `/WTC{QUIT}` which clear the titles and quits the macro.

[1] Scroll the Cell to the Top of the Screen

```
     A                B                C                D                E
 1 *---A macro to move the current cell to the top of the screen
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 TOPSCREN        {WINDOWSOFF}{PANELOFF}{DOWN 19}{UP 19}
```

Often, when you work on a worksheet, you want to scroll a cell to the first row of the screen, this macro will do the job. The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands that freeze the screen and panel display activities while it uses `{DOWN 19}{UP 19}` to move the cell to the top row of the screen. This technique is good for an 80X25 Lotus screen. If you use different number of rows in the screen, the number 19 must be changed accordingly.

[5] Visit a Set of Addresses/Stations in Order

```
     A              B              C              D              E
 1 *---A macro to move the cell pointer around the worksheet to predefined
 2     set of locations. Every time the macro is activated the cell pointer
 3     moves to the next location (station). The macro already includes a
 4     list of stations A1, B1. . . G1, insert your stations instead. The
 5     list is not limited in size but has to be contiguous.
 6 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 7     range names in this column (starts with the \Z macro name)
 8 *---Hold the [ALT] key and press [Z] to activate the macro
 9 !
10 \Z              {BREAKON}
11 STATIONS        {WINDOWSOFF}{PANELOFF}{GOTO}first208~{DOWN}{LEFT}!~
                   /C~.{RIGHT}{END}{DOWN}{LEFT}~{RIGHT}{UP}/M~{END}{DOWN}
                   {DOWN}~
12 !               {DOWN}/M.{END}{DOWN}~{UP}~
13 !               {UP}{LEFT}first208~/RNLR~{END}{DOWN}last#208~/RNLR~
14 !               {GOTO}{last#208}~/RNDlast#208~
15 !
16 !               LIST
17 !               ~~~~
18 first208        A1
19 !               B1
20 !               C1
21 !               D1
22 !               E1
23 !               F1
24 !               G1
```

Let's assume a worksheet with many tables. When you frequently have to move from one table to another to make changes or view results, instead of remembering the order of movements and the addresses to move to, this macro will do the job accurately and with no effort. All you have to do is insert the list of addresses (stations) in the proper order, starting from the B18 cell named [first208] and downward. There is no limit on the number of stations as long as the list is contiguous. This macro already contains a list of addresses A1, B1, C1, D1, E1, F1 and G1, therefore just override it with your addresses.

When you start the macro, it uses the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and panel activities, and issues `{GOTO}first208~` to move the cell pointer to cell [first208], which contains the first station in the list to move to. Because we cannot know how long the list will be, the macro uses the length of the list of stations to fill the "A" column with the exclamation mark "`!`" character. The reason is that when this macro is used from the macro manager, these exclamation mark "`!`" characters are essential to the stage where the manager erases the macro when the macro is finished. To fill the column with the list of the exclamation mark "`!`" characters, the macro issues:

```
{DOWN}{LEFT}!~/C~.{RIGHT}{END}{DOWN}{LEFT}~
```

The macro makes use of the length of the adjacent stations list in the "B" column, to paint a list of exclamation mark "`!`" characters with the same length in the "A" column. The macro starts with `{DOWN}{LEFT}` that move the cell pointer to the cell below the cell containing the "first208" string, and types the "`!`" character to the panel, and issues the "`~`" macro key to write the content of the panel into the current cell. Now the macro uses the `/C~` macro keys and then issues the period key "`.`" to anchor the cell pointer. Then the macro uses `{RIGHT} {END} {DOWN}` to paint (highlight) the "A" and the "B" columns. Now the macro uses `{LEFT}` to shrink the highlighted area to the "A" column alone, and issues the tilde "`~`" (same as ENTER

from the keyboard).

Next the macro uses `/M~{END}{DOWN}{DOWN}~` to move the first address to the end of the list (in our example the A1 address is placed after the G1 address). To close the gap, the macro uses `{DOWN}/M.{END}{DOWN}~{UP}~` to push the whole list one cell up. Now the list will look like this:

	A	B	C	D	E
16	!	LIST			
17	!	~~~~			
18	first208	B1			
19	!	C1			
20	!	D1			
21	!	E1			
22	!	F1			
23	!	G1			
24	!	A1			

	A	B	C	D	E
13	!	{UP}{LEFT}first208~/RNLR~{END}{DOWN}last#208~/RNLR~			
14	!	{GOTO}{last#208}~/RNDlast#208~			

When the macro moved the first cell to the end of the list the [first208] range name also moved; therefore the macro continues with `{UP}{LEFT}/RNLR~` that move the cell pointer to the A18 cell and re-create the [first208] range name. Now the macro uses `{END}{DOWN}` to move the cell pointer to last occupied cell in the "A" column and types the "`last#208`" string in that cell. In our example it is the A24 cell, and the list will look like this:

	A	B	C	D	E
16	!	LIST			
17	!	~~~~			
18	first208	B1			
19	!	C1			
20	!	D1			
21	!	E1			
22	!	F1			
23	!	G1			
24	LAST#208	A1			

To assign this name to the B24..C24 cell, the macro issues the `/RNLR~`. Now the macro issues the indirect `{GOTO}{last#208}~` macro command that moves the cell pointer to the first station, in our example the A1..A1 cell. Next time you activate the macro, the cell pointer will move to the B1 cell because it is now the first address in the list. Every time you activate the macro, the next cell in the list is pushed to the head of the list and the first address to the bottom of the list.

[7] Go to the Last Cell Pointer Location

```
     A                B                C                D                E
 1 *---A macro that REMEMBERS last pointer position including the sheet
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 GOTOBACK        {LET rel408,@INFO("release")}~{RECALC loc408}
12 !               {WINDOWSOFF}{PANELOFF}{LET dummy408,@CELLPOINTER
                   ("address")}~
13 !               {PANELON}{GOTO}{previous408}{GET input408}{PANELOFF}
14 !               {IF input408="{ESC}"}{ESC 3}{WINDOWSON}{PANELON}
                   {BRANCH ret408}
15 !               {WINDOWSON}{IF input408="{R}"#OR#input408="{L}"#OR#
                   input408="{U}"#OR#input408="{D}"#OR#input408="{HOME}"
                   #OR#input408="{PGDN}"#OR#input408="{PGUP}"}{ESC 6}
                   {input408}{LET previous408,dummy408}~{WINDOWSON}{PANELON}
                   {GOTO}{?}~~{BRANCH ret408}
16 !               {IF @LEFT(rel408,1)<>"@"#AND#(input408="{NS}"#OR#
                   input408="{PS}")}{ESC 6}{WINDOWSON}{input408}
                   {LET previous408,dummy408}~{WINDOWSON}{PANELON}{GOTO}
                   {?}~~{BRANCH ret408}
17 !               {IF input408<>"~"}{ESC 6}{PANELON}{GOTO}{input408}{?}
                   {LET previous408,dummy408}~~{BRANCH ret408}
18 !               {WINDOWSOFF}{ESC 6}{LET previousa408,previous408}~
                   {LET previous408,dummy408}~{WINDOWSON}{PANELON}{GOTO}
                   {previousa408}~{BRANCH ret408}
19 !
20 previous408     $A$1
21 !
22 previousa408    $B$33
23 !
24 !
25 dummy408        $A$1
26 !
27 input408        ~
28 !
29 ret408
30 !
31 rel408          @INFO("release")
32 !
33 loc408          address
```

This macro allows you to move from place to place in the worksheet and the macro always remembers the last cell pointer position and offers it as the default place to move to. Every time this macro is activated, it records the current cell pointer position and offers the previous position to move to. This macro also works in a 3-D release in which it records the coordination instead of the address.

Before we continue, notice that the code in the B12 cell and the B33 cell named [loc408] is the result of dynamic string formulas, which are updated according to the type of worksheet that you use. If you intend to key the macro into Lotus 1-2-3, you have to key the formulas as they appear here, NOT the code as it appears in the main listing of the macro.

```
12 !               +"{WINDOWSOFF}{PANELOFF}{LET dummy408,@CELLPOINTER
                   (""""&B33&""""")}~"
33 loc408          @IF(@LEFT(B31,1)<>"@","coord","address")
```

	A	B	C	D	E
11	GOTOBACK	{LET rel408,@INFO("release")}~{RECALC loc408}			
12	!	{WINDOWSOFF}{PANELOFF}{LET dummy408,@CELLPOINTER ("address")}~			
13	!	{PANELON}{GOTO}{previous408}{GET input408}{PANELOFF}			

The macro starts with the `{LET rel408,@INFO("release")}~` macro commands which store the result of the `@INFO("release")` function in the B31 cell named [rel408]. Later, the macro uses this result to check if you are using a 2-D or a 3-D Lotus release. Next the macro issues `{RECALC loc408}` to update the dynamic formula in cell [loc408]. Now the macro issues `{WINDOWSOFF}{PANELOFF}`, which freeze the screen and panel display activities. To record the current cell pointer position, the macro uses `{LET dummy408 ,@CELLPOINTER("address")}~`, which store the current cell pointer position in cell [dummy408]. Recall that the code we have just explained is the result of a dynamic string formula, therefore let us look at the formula:

```
12 !            +"{WINDOWSOFF}{PANELOFF}{LET dummy408,@CELLPOINTER
                ("""&B33&""")}~"
```

The "dynamic" part of this formula is the content of the B33 cell, which, itself, contains a formula:

```
33 loc408       @IF(@LEFT(B31,1)<>"@","coord","address")
```

The formula in the B33 cell named [loc408] returns the "coord" string if the first character of the content of the B31 cell is not the "@" character, which means that you are using a 3-D Lotus release. If "@" is the character, you are using a 2-D Lotus release; therefore the formula returns the "address" string. Now let's go back to the formula in the B12 cell. If you are using a 3-D release, the code in the B12 cell looks like:

```
{WINDOWSOFF}{PANELOFF}{LET dummy408,@CELLPOINTER("coord")}~
```

If you are using a 2-D release the code in the B12 cell looks like:

```
{WINDOWSOFF}{PANELOFF}{LET dummy408,@CELLPOINTER("address")}~
```

Now the macro issues `{PANELON}` which resumes the panel display activity, and then the indirect `{GOTO}{previous408}` macro command allowing you to move the cell pointer to the address stored in cell [previous408]. If this is the first time that you are using the macro, the cell named [previous408] contains the A1 cell address, otherwise it contains the previous location of the cell pointer. The macro follows with `{GET input408}`, which waits until you press a key. Lotus stores the key in cell [input408].

	A	B	C	D	E
14	!	{IF input408="{ESC}"}{ESC 3}{WINDOWSON}{PANELON} {BRANCH ret408}			

Now the macro starts with a series of `{IF}` commands to check what key you pressed (the key is stored in [input408]). First the macro issues `{IF input408="{ESC}"}` to check if you pressed the ESC key. If so, the macro issues `{ESC 3}` to exit to the READY mode, and then issues `{WINDOWSON}{PANELON}` to resume the screen and panel display activities, and then issues `{BRANCH ret408}`, which routes the macro control to the empty [ret408] routine and quits.

	A	B	C	D	E
15	!	{WINDOWSON}{IF input408="{R}"#OR#input408="{L}"#OR# input408="{U}"#OR#input408="{D}"#OR#input408="{HOME}" #OR#input408="{PGDN}"#OR#input408="{PGUP}"}{ESC 6} {input408}{LET previous408,dummy408}~{WINDOWSON}{PANELON} {GOTO}{?}~~{BRANCH ret408}			

Next the macro issues the code in the B13 cell to check if you pressed one of the direction keys. If so, the macro issues `{ESC 6}` to exit to the READY mode, and then issues the `{input408}` routine command. When Lotus encounters a routine command, it executes the command in the cell with the same name as the name in the routine command, meaning the B27 cell with the [input408] range name. However, this cell contains only the last key that you pressed, therefore Lotus carries the same key as you meant to, with a little delay. As long as you press the keys that the macro allows, the macro executes these keys. Now, because you moved the cell pointer, the cell location before the move becomes the "previous" cell. Therefore the macro issues `{LET previous408,dummy408}~` which copies the address to cell [previous408].

Now the macro issues `{WINDOWSON}{PANELON}` to resume the screen and panel display activities, and then issues `{GOTO}{?}` to start the GOTO process and allow you to do one of the following: 1. To press the ENTER key 2. To type the address 3. To point to the cell to go and then press the ENTER key. When you press ENTER, the macro issues the tilde "`~`", which is the same as ENTER, and moves the cell pointer. Then the macro issues `{BRANCH ret408}`, which routes macro control to the empty [ret408] routine and quits.

	A	B	C	D	E
16	!	{IF @LEFT(rel408,1)<>"@"#AND#(input408="{NS}"#OR# input408="{PS}")}{ESC 6}{WINDOWSON}{input408} {LET previous408,dummy408}~{WINDOWSON}{PANELON}{GOTO} {?}~~{BRANCH ret408}			

Next, the macro issues `{IF @LEFT(rel408,1)<>"@"#AND#....}` to check if you are using a 3-D Lotus release. If the first character of the content of the cell named [rel408] is different from the "`@`" character, you are using a 3-D Lotus release, and the macro allows you to use the NEXT-SHEET and other direction keys which are allowed only in the 3-D releases. If so, the macro issues `{ESC 6}` to exit to the READY mode, and then issues the `{input408}` routine command. When Lotus encounters the `{input408}` routine command, it executes the command in the cell with the [input408] name. However, this cell contains only the last key you pressed, therefore Lotus carries the same key as you meant to, with a little delay. As long as you press the direction keys of the 3-D releases which the macro allows, the macro executes these keys. Now, because you moved the cell pointer, the cell location before the move becomes the "previous" cell. Therefore the macro issues `{LET previous408,dummy408}~` to copy the address to [previous408].

Now the macro issues `{WINDOWSON}{PANELON}` to resume the screen and panel display activities and then issues `{GOTO}{?}` to allow you to insert the target cell address. When you press ENTER, the macro issues the tilde "`~`" and moves the cell pointer. Then the macro issues `{BRANCH ret408}`, which routes the macro control to the empty [ret408] routine and quits.

	A	B	C	D	E
17	!	{IF input408<>"~"}{ESC 6}{PANELON}{GOTO}{input408}{?} {LET previous408,dummy408}~~{BRANCH ret408}			

The last `{IF}` command is `{IF input408<>"~"}`, which checks if you pressed any key but ENTER. If so, (such as the "A" character) the macro issues `{ESC 6}` to return to the READY

mode and then {PANELON} to resume the panel display activity. Then the macro issues the indirect {GOTO}{input408} macro command, which injects the content of cell [input408] into the panel as the response to {GOTO}.

Recall that the cell named [input408] holds the key that you pressed, therefore the {input408} routine command injects this key into the panel. The macro follows immediately with {? to allow you to complete the address entry. When you press ENTER, the macro first issues {LET previous408,dummy408}, which copies the content of [dummy408] to [previous408], updating the content of [previous408] to include the last cell pointer position before the current move. The next time you start the macro, it will offer this address as the default address to go to.

	A	B	C	D	E
18	!	{WINDOWSOFF}{ESC 6}{LET previousa408,previous408}~ {LET previous408,dummy408}~{WINDOWSON}{PANELON}{GOTO} {previousa408}~{BRANCH ret408}			

If you just press ENTER, it means that you want to return to the suggested previous location; however there is a small problem, the cell named [previous408] holds the previous location but now we need to store there the current location before you move the cell pointer. Therefore, the macro issues {WINDOWSOFF} which freezes the screen display activity, and then issues {ESC 6} to exit to the READY mode. Then the macro issues {LET previousa408 ,previous408}~ which temporarily copy the content of [previous408] to [previousa408]. Now the macro issues {LET previous408,dummy408}~, which store the current location in [previous408]. Next the macro issues {WINDOWSON}{PANELON} to resume the screen and panel display activities and then issues the indirect {GOTO}{previousa408}~ macro command, sending the cell pointer to the address which is written in [previousa408].

[6] Jump to the End of a Row, a Column or a Sheet

```
     A          B          C          D          E          F          G    H
 1 *---A macro to jump with the cell pointer to ends of row, columns, sheets
 2     even if they are not empty
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                     IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z             {BREAKON}
11 GOTOENDS       {LET rel503,@INFO("release")}~{MENUBRANCH menu1503}
12 !
13 menu1503       Top    Bottom    Left    Right    In     Out     Quit
14 !              Jump toJump to toJump totJump to tJump toJump to Quit t
15 !              {RECALC{RECALC bo{RECALC {RECALC R{IF @LE{IF @LEFT{INDI
16 !              {MENUBR{MENUBRANC{MENUBRA{MENUBRAN{IF @LEF{MENUBRANCH m
17 !                                                 {MENUBRANCH menu1503}
18 ret503
19 !
20 rel503         @INFO("release")
21 !
22 top503         $C$1
23 !
24 bottom503      $C$8192
25 !
26 left503        A26
27 !
28 right503       IV28
29 !
30 out503         A:$C$30
31 !
32 inA503         $AE:$C$2
33 !
34 dum503         $A:$C$2
```

The macro uses a custom menu with seven menu options. Because there is no way to display the custom menu code on this page without overlapping, we show every menu option's code separately.

```
Top
Jump to the top of the column
{RECALC top503}{GOTO}{top503}~
{MENUBRANCH menu1503}
Bottom

Jump to the bottom of the column
{RECALC bottom503}{GOTO}{bottom503}~
{MENUBRANCH menu1503}

Left
Jump to the most first cell of the row
{RECALC left503}{GOTO}{left503}~
{MENUBRANCH menu1503}

Right
Jump to the most last cell of the row
{RECALC right503}{GOTO}{right503}~
{MENUBRANCH menu1503}

In
Jump to the last sheet and stay in the same row and column
{IF @LEFT(rel503,1)<>"@"}{LET dum503,@CELLPOINTER("coord")}~{LC}
    {END}{NS}
```

```
{IF @LEFT(rel503,1)<>"@"}{LET ina503,@LEFT(@CELLPOINTER("coord")
    ,@FIND(":",@CELLPOINTER("coord"),0))&@RIGHT(dum503,@LENGTH
    (dum503)-@FIND(":",dum503,0))}~{GOTO}{ina503}~
{MENUBRANCH menu1503}

Out
Jump to the first sheet and stay in the same row and column
{IF @LEFT(rel503,1)<>"@"}{RECALC out503}{GOTO}{out503}~
{MENUBRANCH menu1503}

Quit
Quit the macro
{INDICATE}{BRANCH ret503}
```

Also notice that the code in the B12, B24, B26, B28 and the B30 cells is the result of the following dynamic string formulas:

```
12 top503          @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER
                   ("address"),1)+1)&"1"
24 bottom503       @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER
                   ("address"),1)+1)&"8192"
26 left503         +"A"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER
                   ("address"))-@FIND("$",@CELLPOINTER("address"),1)-1)
28 right503        +"IV"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER
                   ("address"))-@FIND("$",@CELLPOINTER("address"),1)-1)
30 out503          +"A:"&@CELLPOINTER("address")
```

If you intend to key the code of this macro into Lotus 1-2-3, you have to key the code of the formulas and the code of the custom menu options as they appear here, NOT the code as it appears in the main listing of the macro. To further assist you, we show the full cell contents at the end of this section.

There are times when you want to move the cell pointer to the end of the worksheet and stay at the same row or the same column or the same address in the last sheet of the worksheet (Z-direction). This macro uses a custom menu which allows you to choose what direction to jump to using one key press. The macro starts with the `{LET rel503,@INFO("release")}~` macro commands, which store the result of the `@INFO("release")` function in the B20 cell named [rel503]. Later the macro uses this result to determine if you are using a 2-D or a 3-D Lotus release. Then the macro issues `{MENUBRANCH menu1503}`, which activates the custom menu in the B13 cell named [menu1503]. The first menu option is [**T**op]:

```
Top
Jump to the top of the column
{RECALC top503}{GOTO}{top503}~
{MENUBRANCH menu1503}
```

When you choose this option, the macro issues `{RECALC top503}`, which updates the dynamic string formula in cell [top503] which is:

```
@LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER("address"),1)+1)&"1"
```

to calculate the address of the first cell in the current column. The formula extracts the column letter using `@FIND`, the `@CELLPOINTER("address")` and `@LEFT` Lotus functions, and then adds the number "`1`" to the letter. For example, let's assume that the current cell is F53, the macro extracts the "`F`" column letter from the address and adds the number "1" as a string to this. The result is the F1 cell address string which is the top cell in the "F" column. When the top cell address is calculated, the macro uses the indirect macro command `{GOTO}{top503}~` to move the cell pointer to the first cell in the current column. Next the macro returns to the menu

using {MENUBRANCH menu1503}. The second menu option is [**B**ottom]:

```
Bottom
Jump to the bottom of the column
{RECALC bottom503}{GOTO}{bottom503}~
{MENUBRANCH menu1503}
```

The [**B**ottom] menu option uses the same code. The only difference is that the number "8192" is added to the extracted column letter instead of the number "1". The third menu option is [**L**eft]:

```
Left
Jump to the most first cell of the row
{RECALC left503}{GOTO}{left503}~
{MENUBRANCH menu1503}
```

The [**L**eft] menu option is almost the same as the previous two except that the formula in the B26 cell [left503] extracts the row number of the current address and adds the "A" character to it to create the leftmost cell address in the current row. The formula is:

```
+"A"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER("address"))
    -@FIND("$",@CELLPOINTER("address"),1)-1)
```

The fourth menu option is [**R**ight]:

```
Right
Jump to the most last cell of the row
{RECALC right503}{GOTO}{right503}~
{MENUBRANCH menu1503}
```

The [**R**ight] menu option uses the same code as the [Left] menu option, except that the "IV" characters are added to the extracted row number instead of the "A" character. The fifth menu option is [**I**n]:

```
In
Jump to the last sheet and stay in the same row and column
{IF @LEFT(rel503,1)<>"@"}{LET dum503,@CELLPOINTER("coord")}~{LC}
    {END}{NS}
{IF @LEFT(rel503,1)<>"@"}{LET ina503,@LEFT(@CELLPOINTER("coord")
    ,@FIND(":",@CELLPOINTER("coord"),0))&@RIGHT(dum503,@LENGTH
    (dum503)-@FIND(":",dum503,0))}~{GOTO}{ina503}~
{MENUBRANCH menu1503}
```

When you choose the [**I**n] menu option, the macro issues {IF @LEFT(rel503,1)<>"@"}, which makes sure that you are using a 3-D release of Lotus 1-2-3. Then the macro continues with {LET dum503,@CELLPOINTER("coord")}~, which store the current cell pointer location in the B34 cell named [dum503], and then the macro issues {LC}{END}{NS}, which move the cell pointer first to the last occupied cell in the worksheet, then to the last sheet in the worksheet. Again the macro issues {IF @LEFT(rel503,1)<>"@"} to make sure that you use a 3-D release. If so, the macro issues:

```
{LET ina503,@LEFT(@CELLPOINTER("coord"),@FIND(":",@CELLPOINTER
    ("coord"),0))&@RIGHT(dum503,@LENGTH(dum503)-@FIND(":",dum503
    ,0))}~
```

which calculates the location of the desired cell in the last sheet (the current sheet). The sheet identification character is taken from the current sheet, which is the last sheet in the worksheet using the

```
@LEFT(@CELLPOINTER("coord"),@FIND(":",@CELLPOINTER("coord"),0))
```

string formula. The rest of the target cell coordinate is extracted from the cell coordinate previously stored in the B34 cell named [dum503], using the

```
@RIGHT(dum503,@LENGTH(dum503)-@FIND(":",dum503,0))
```

string formula. The {LET ina503, ...} command stores the result of the target cell coordinate in the B32 cell named [ina503] and the indirect {GOTO}{ina503}~ macro command sends the cell pointer to the target cell location.

Note: The reason that we sliced the code and used the {IF @LEFT(rel503,1)<>"@"} twice instead of combining the two lines of code together is that we wanted the code to be compatible to 2-D and 3-D releases. In a 2-D release a cell can contain a maximum of 240 characters, while in a 3-D release the cell can contain a maximum of 512 characters in a single cell.

The sixth menu option is [**O**ut]:

```
Out
Jump to the first sheet and stay in the same row and column
{IF @LEFT(rel503,1)<>"@"}{RECALC out503}{GOTO}{out503}~
{MENUBRANCH menu1503}
```

When you choose [**O**ut], the macro issues {IF @LEFT(REL503,1)<>"@"}, which checks if you are using a 3-D release. If so, the macro re-calculates the formula in the B30 cell named [out503], which is a much simpler formula than the formula in [**I**n] because the sheet letter is known, it is the first sheet in the worksheet represented by the "A" character. The formula in cell [out503] is:

```
+"A:"&@CELLPOINTER("address")
```

The seventh and last menu option is [Quit]:

```
Quit
Quit the macro
{INDICATE}{BRANCH ret503}
```

When you choose this option, the macro issues {INDICATE}, which clears any custom label in the indicate area and then issues {BRANCH ret503} macro, to route macro control to the empty [ret503] routine and quits.

Note: In the new releases of Lotus 1-2-3, you can use the {INDICATE "long string"} commands to display long messages in the first row of the screen. The {INDICATE} macro command without the "long string" resets the indicate area to the standard Lotus of five characters in the upper right side of the screen

---

To make it easier for you to key this macro into Lotus 1-2-3, we show the full list of cell contents and formulas.

```
A1: U [W15] '*---A macro to jump with the cell pointer to ends of row,
            columns, sheets
A2: U [W15] '    even if they are not empty
A3: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
```

```
          define the
A4: [W15] '    range names in this column (starts with the \Z macro name)
A5: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A6: [W15] '!
A7: U [W15] '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3
           RELEASE
A8: U [W15] '                    IT WILL WORK IN LOTUS 2.0 AND UP
A9: [W15] '!
A10: U [W15] '\Z
B10: '{BREAKON}
A11: U [W15] 'GOTOENDS
B11: '{LET rel503,@INFO("release")}~{MENUBRANCH menu1503}
A12: [W15] '!
A13: [W15] 'menu1503
B13: 'Top
C13: 'Bottom
D13: 'Left
E13: 'Right
F13: 'In
G13: 'Out
H13: 'Quit
A14: [W15] '!
B14: 'Jump to the top of the column
C14: 'Jump to the bottom of the column
D14: 'Jump to the most first cell of the row
E14: 'Jump to the most last cell of the row
F14: 'Jump to the last sheet and stay in the same row and column
G14: 'Jump to the first sheet and stay in the same row and column
H14: 'Quit the macro
A15: [W15] '!
B15: '{RECALC top503}{GOTO}{top503}~
C15: '{RECALC bottom503}{GOTO}{bottom503}~
D15: '{RECALC left503}{GOTO}{left503}~
E15: '{RECALC right503}{GOTO}{right503}~
F15: '{IF @LEFT(rel503,1)<>"@"}{LET dum503,@CELLPOINTER("coord")}~{LC}{END}
     {NS}
G15: '{IF @LEFT(rel503,1)<>"@"}{RECALC out503}{GOTO}{out503}~
H15: '{INDICATE}{BRANCH ret503}
A16: [W15] '!
B16: '{MENUBRANCH menu1503}
C16: '{MENUBRANCH menu1503}
D16: '{MENUBRANCH menu1503}
E16: '{MENUBRANCH menu1503}
F16: '{IF @LEFT(rel503,1)<>"@"}{LET ina503,@LEFT(@CELLPOINTER("coord"),
     @FIND(":",@CELLPOINTER("coord"),0))&@RIGHT(dum503,@LENGTH(dum503)-
     @FIND(":",dum503,0))}~{GOTO}{ina503}~
G16: '{MENUBRANCH menu1503}
A17: [W15] '!
F17: '{MENUBRANCH menu1503}
A18: [W15] 'ret503
A19: [W15] '!
A20: [W15] 'rel503
B20: '@INFO("release")
A21: [W15] '!
A22: [W15] 'top503
B22: U @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER("address"),1)+
       1)&"1"
A23: [W15] '!
A24: [W15] 'bottom503
B24: U @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER("address"),1)+1)
       &"8192"
A25: [W15] '!
A26: [W15] 'left503
B26: U +"A"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER("address"))
       -@FIND("$",@CELLPOINTER("address"),1)-1)
A27: [W15] '!
A28: [W15] 'right503
B28: U +"IV"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER("address"))
       -@FIND("$",@CELLPOINTER("address"),1)-1)
A29: [W15] '!
A30: [W15] 'out503
```

```
B30: U +"A:"&@CELLPOINTER("address")
A31: [W15] '!
A32: [W15] 'inA503
B32: '$AE:$C$2
A33: [W15] '!
A34: [W15] 'dum503
B34: '$A:$C$2
```

[6] Jump to the End of the Worksheet

```
     A              B          C       D       E       F       G       H
 1 *---A macro to JUMP to the ends of rows, columns, sheets (last occupied
   cell)
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                     IT WILL WORK IN RELEASE 2.0 AND UP
 8 !
 9 !
10 \Z               {BREAKON}
11 JUMPENDS         {LET rel524,@INFO("release")}~{MENUBRANCH menu1524}
12 !
13 menu1524         Right    Left    Up    Down    In    Out    Move   Quit
14 !                Jump to tJump to Jump tJump to Erase Erase aMove thQuit
15 !                {rightaa5{leftaa5{topaa{bottoma{IF(@L{IF(@LE{move52{BR
16 !                {MENUBRAN{MENUBRA{MENUB{MENUBRA{MENUB{MENUBR{MENUBRANC
17 !
18 ret524
19 !
20 rel524           3.00.00
21 !
22 top524           $A$1
23 !
24 bottom524        $A$8192
25 !
26 left524          A1
27 !
28 right524         IV1
29 !
30 out524           A:$A$1
31 !
32 inA524           $U:$H$38
33 !
34 dum524           $D$10
35 !
36 ERATOP524        A:$D$10..$J:$D$10
37 !
38 dumc524          $J:$D$10
39 !
40 topaa524         {WINDOWSOFF}{RECALC top524}{GOTO}{top524}~
                    {IF @CELLPOINTER("type")="b"}{END}{DOWN}
41 !                {WINDOWSON}
42 !
43 bottomaa524       {WINDOWSOFF}{RECALC bottom524}{GOTO}{bottom524}~
                    {IF @CELLPOINTER("type")="b"}{END}{UP}
44 !                {WINDOWSON}
45 !
46 leftaa524         {WINDOWSOFF}{RECALC left524}{GOTO}{left524}~
                    {IF @CELLPOINTER("type")="b"}{END}{RIGHT}
47 !                {WINDOWSON}
48 !
49 rightaa524        {WINDOWSOFF}{RECALC right524}{GOTO}{right524}~
                    {IF @CELLPOINTER("type")="b"}{END}{LEFT}
50 !                {WINDOWSON}
51 !
52 inaa524           {WINDOWSOFF}{IF @LEFT(rel524,1)<>"@"}
                    {LET dum524,@CELLPOINTER("address")}~{LC}{END}{NS}{GOTO}
                    {dum524}~{IF @CELLPOINTER("type")="b"}{END}{PS}
53 !                {WINDOWSON}
54 !
55 outaa524         {WINDOWSOFF}{IF @LEFT(rel524,1)<>"@"}{RECALC out524}
                    {GOTO}{out524}~{IF @CELLPOINTER("type")="b"}{END}{NS}
56 !                {WINDOWSON}
57 !
58 move524          Use the arrow keys to move the cell pointer and press
```

```
                ENTER: {GET key524}{ESC}
59 !            {IF key524="~"}{RETURN}
60 !            {IF key524="{PGDN}"#OR#key524="{PGUP}"#OR#key524="
                {DOWN}"#OR#key524="{UP}"#OR#key524="{LEFT}"#OR#key524=
                "{RIGHT}"}{key524}
61 !            {IF @LEFT(rel524,1)<>"@"#AND#(key524="{NS}"#OR#key524=
                "{PS}")}{key524}
62 !            {BRANCH move524}
63 !
64 key524        ~
```

This macro allows you to jump to the end of a column, a row or in the Z direction to the last sheet same address or to the first sheet same address. This macro is different from the previous GOTOENDS.WK1, it moves the cell pointer to the last occupied cell in the column, row or sheet. For example: let's assume that the cell pointer is on the F35 cell and the last occupied cell in the current column is the F321 cell. Therefore the macro will move the cell pointer to the F321 cell. This macro uses a custom menu with eight menu options. There is no way they can fit on one page without overlapping, so we show the code of every menu option separately. If you intend to key this macro into Lotus, you must use the code as it appears here, and NOT the code as it appears in the main listing.

```
Right
Jump to the rightmost occupied cell in this row
{rightaa524}
{MENUBRANCH menu1524}

Left
Jump to the leftmost occupied cell in this row
{leftaa524}
{MENUBRANCH menu1524}

Up
Jump to the upper occupied cell in this column
{topaa524}
{MENUBRANCH menu1524}

Down
Jump to the lower occupied cell in this column
{bottomaa524}
{MENUBRANCH menu1524}

In
Jump to the last sheet with a cell at the same address
{IF @LEFT(rel524,1)<>"@"}{inaa524}
{MENUBRANCH menu1524}

Out
Jump to the first sheet with a cell at the same address
{IF @LEFT(rel524,1)<>"@"}{outaa524}
{MENUBRANCH menu1524}

Move
Move the cell pointer to other location
{move524}
{MENUBRANCH menu1524}

Quit
Quit the macro
{BRANCH ret524}
```

Notice that the code in some of the cells in the macro is the result of dynamic string formulas. If you intend to key the code into Lotus 1-2-3, you must key in the following formulas, NOT the code as it appears in the main listing.

```
22 top524       @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER
```

```
                    ("address"),1)+1)&"1"
 24 bottom524       @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER
                    ("address"),1)+1)&"8192"
 26 left524         +"A"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER
                    ("address"))-@FIND("$",@CELLPOINTER("address"),1)-1)
 28 right524        +"IV"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER
                    ("address"))-@FIND("$",@CELLPOINTER("address"),1)-1)
 30 out524          +"A:"&@CELLPOINTER("address")
```

```
     A              B          C       D       E       F       G       H
 11 JUMPENDS        {LET rel524,@INFO("release")}~{MENUBRANCH menu1524}
```

The macro starts with the `{LET rel524,@INFO("release")}~` macro commands, which store the result of the `@INFO("release")` function in the cell named [rel524]. Later the macro uses the stored result to check if you are using a 3-D or a 2-D release. Next the macro issues `{MENUBRANCH menu1524}` starting the [menu1524] custom menu. The first menu option is [**R**ight]:

```
Right
Jump to the rightmost occupied cell in this row
{rightaa524}
{MENUBRANCH menu1524}
```

When you choose this menu option, the macro issues the `{rightaa524}` routine command, which moves the cell pointer to the last occupied cell right of the current row. When the routine is finished, the macro control returns to `{MENUBRANCH menu1524}`, which re-starts the custom menu.

```
     A              B          C       D       E       F       G       H
 49 rightaa524      {WINDOWSOFF}{RECALC right524}{GOTO}{right524}~
                    {IF @CELLPOINTER("type")="b"}{END}{LEFT}
 50 !               {WINDOWSON}
```

The `{rightaa524}` routine starts with `{WINDOWSOFF}`, which freezes the screen activity. Then the macro issues `{RECALC right524}`, to update the dynamic string formula in cell [right524] and the `{GOTO}{right524}~` indirect macro command to move the cell pointer to the last cell in the row (in the "IV") column.

```
 28 right524        +"IV"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER
                    ("address"))-@FIND("$",@CELLPOINTER("address"),1)-1)
```

The result of the formula in [right524] is the last cell in this row which is the IV35 cell if the current cell is the F35 cell. Next the macro issues `{IF @CELLPOINTER("type")="b"}`, which checks if the cell is blank. If so, the macro issues `{END}{LEFT}`, which move the cell pointer to the last occupied cell in the row. The Second menu option is [**L**eft]:

```
Left
Jump to the leftmost occupied cell in this row
{leftaa524}
{MENUBRANCH menu1524}
```

When you choose this menu option, the macro issues the `{leftaa524}` routine command, which moves the cell pointer to the last occupied cell left of the current cell. When the routine is finished, macro control returns to `{MENUBRANCH menu1524}`, which re-starts the custom menu.

```
          A          B          C          D          E          F          G          H
46 leftaa524         {WINDOWSOFF}{RECALC left524}{GOTO}{left524}~
                     {IF @CELLPOINTER("type")="b"}{END}{RIGHT}
47 !                 {WINDOWSON}
```

The {leftaa524} routine starts with {WINDOWSOFF} which freezes the screen activity. Then the macro issues {RECALC left524}, which updates the dynamic string formula in [left524] and issues the {GOTO}{left524}~ indirect macro command to move the cell pointer to the first cell in the current row.

```
26 left524           +"A"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER
                     ("address"))-@FIND("$",@CELLPOINTER("address"),1)-1)
```

The result of the formula in [left524] is the first cell in this row, which is the A35 cell if the current cell is the F35 cell. Next the macro issues {IF @CELLPOINTER("type")="b"} which checks if the cell is blank. If so, the macro issues {END}{RIGHT}, which move the cell pointer to the first occupied cell in the row. The third menu option is [**U**p]:

```
Up
Jump to the upper occupied cell in this column
{topaa524}
{MENUBRANCH menu1524}
```

When you choose this menu option, the macro issues the {topaa524} routine command, which moves the cell pointer to first occupied cell in the column and above to the current cell. When the routine is finished, macro control returns to {MENUBRANCH menu1524}, which re-starts the custom menu.

```
          A          B          C          D          E          F          G          H
40 topaa524          {WINDOWSOFF}{RECALC top524}{GOTO}{top524}~
                     {IF @CELLPOINTER("type")="b"}{END}{DOWN}
41 !                 {WINDOWSON}
```

The {topaa524} routine starts with {WINDOWSOFF} which freezes the screen activity. Then the macro issues {RECALC top524}, to update the dynamic string formula in [top524] and issues the {GOTO}{top524}~ indirect macro command to move the cell pointer to the first cell in the current column.

```
22 top524            @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER
                     ("address"),1)+1)&"1"
```

The result of the formula in [top524] is the first cell in this column, which is the F1 cell, if the current cell is the F35 cell. Next the macro issues {IF @CELLPOINTER("type")="b"}, which checks if the cell is blank. If so, the macro issues {END}{DOWN}, which move the cell pointer to the first occupied cell in the current column. The fourth menu option is [**D**own]:

```
Down
Jump to the lower occupied cell in this column
{bottomaa524}
{MENUBRANCH menu1524}
```

When you choose this menu option, the macro issues the {bottomaa524} routine command, which moves the cell pointer to last occupied cell in the current column and below to the current cell. When the routine is finished, macro control returns to {MENUBRANCH menu1524}, which re-starts the custom menu.

	A	B	C	D	E	F	G	H
43	bottomaa524	{WINDOWSOFF}{RECALC bottom524}{GOTO}{bottom524}~						
		{IF @CELLPOINTER("type")="b"}{END}{UP}						
44	!	{WINDOWSON}						

The {bottomaa524} routine starts with {WINDOWSOFF} which freezes the screen activity. Then the macro issues {RECALC bottom524}, which updates the dynamic string formula in [top524] and issues the {GOTO}{bottom524}~ indirect macro command to move the cell pointer to the last cell in the current column.

24	bottom524	@LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER ("address"),1)+1)&"8192"

The result of the formula in [bottom524] is the last cell in the current column, which is the F8192 cell, if the current cell is the F35 cell. Next the macro uses {IF @CELLPOINTER ("type")="b"} to check if the cell is blank. If so, the macro issues {END}{UP}, which move the cell pointer to the last occupied cell in the current column. The fifth menu option is [**I**n]:

```
In
Jump to the last sheet with a cell at the same address
{IF @LEFT(rel524,1)<>"@"}{inaa524}
{MENUBRANCH menu1524}
```

When you choose this menu option, the macro issues {IF @LEFT(rel524,1)<>"@"}, which checks if the first character of the content of [rel524] is the "@" character. If so, you are using a 3-D release and the macro issues the {inaa524} routine command. When the routine is finished, macro control returns to {MENUBRANCH menu1504}, which re-starts the custom menu. If the condition is not true, you are using a 2-D release and the macro issues {MENUBRANCH menu1504} which re-starts the custom menu.

	A	B	C	D	E	F	G	H
52	inaa524	{WINDOWSOFF}{IF @LEFT(rel524,1)<>"@"}						
		{LET dum524,@CELLPOINTER("address")}~{LC}{END}{NS}{GOTO}						
		{dum524}~{IF @CELLPOINTER("type")="b"}{END}{PS}						
53	!	{WINDOWSON}						

The {inaa524} routine is more complicated from the previous four routines because the number of sheets in the current worksheet is not a fixed number like the number of columns or the number of rows. The routine starts with {WINDOWSOFF}, which freezes the screen activity. Then the macro issues {IF @LEFT(rel504,1)<>"@"}, which checks if the first character of the content of [rel504] is the "@" character. If so, you are using a 3-D Lotus release; therefore the macro issues {LET dum504,@CELLPOINTER("address")}~, which store the current cell's address in [dum524]. Next the routine issues {LC}, which moves the cell pointer to last cell in the last occupied sheet, and {END}{NS} move the cell pointer to the last sheet in the worksheet. Now the cell pointer is on the last sheet in the worksheet.

To bring the cell pointer to the same address, the macro issues the indirect {GOTO}{dum524}~ macro command and then {IF @CELLPOINTER("type")="b"}, which checks if the cell is blank. If so, the macro issues {END}{PS}, which move the cell pointer to the last sheet which contained an occupied cell at the same address (F35 in our example). The sixth menu option is [**O**ut]:

```
Out
Jump to the first sheet with a cell at the same address
{IF @LEFT(rel524,1)<>"@"}{outaa524}
```

```
{MENUBRANCH menu1524}
```

When you choose this menu option, the macro issues `{IF @LEFT(rel524,1)<>"@"}`, which checks if the first character of the content of [rel524] is the "`@`" character. If so, you are using a 3-D release and the macro issues the `{outaa524}` routine command. When the routine is finished, macro control returns to `{MENUBRANCH menu1504}` which re-starts the custom menu. If the condition is false, you are using a 2-D release and the macro issues `{MENUBRANCH menu1504}` which re-starts the custom menu.

	A	B	C	D	E	F	G	H
55	outaa524	{WINDOWSOFF}{IF @LEFT(rel524,1)<>"@"}{RECALC out524}						
		{GOTO}{out524}~{IF @CELLPOINTER("type")="b"}{END}{NS}						
56	!	{WINDOWSON}						

The `{outaa524}` routine starts with `{WINDOWSOFF}`, which freezes the screen activity. Then the macro issues `{IF @LEFT(rel524,1)<>"@"}` and `{RECALC out524}` which updates the dynamic string formula in cell [out504] and `{GOTO}{out524}~`.

```
30 out524          +"A:"&@CELLPOINTER("address")
```

The result of the formula in [out524] is the cell coordinate in the first sheet (the "`A`" sheet) which is the `A:F35` cell, if the current cell is the F35 cell. Next the macro issues `{IF @CELLPOINTER("type")="b"}` which checks if the cell is blank. If so, the macro issues `{END}{NS}`, which move the cell pointer to the first sheet with an occupied cell at the same address. The seventh menu option is [**M**ove]:

```
Move
Move the cell pointer to other location
{move524}
{MENUBRANCH menu1524}
```

When you choose this menu option, the macro issues the `{move524}` routine command which starts the [move524] routine.

	A	B	C	D	E	F	G	H
58	move524	Use the arrow keys to move the cell pointer and press						
		ENTER: {GET key524}{ESC}						
59	!	{IF key524="~"}{RETURN}						
60	!	{IF key524="{PGDN}"#OR#key524="{PGUP}"#OR#key524="						
		{DOWN}"#OR#key524="{UP}"#OR#key524="{LEFT}"#OR#key524=						
		"{RIGHT}"}{key524}						
61	!	{IF @LEFT(rel524,1)<>"@"#AND#(key524="{NS}"#OR#key524=						
		"{PS}")}{key524}						
62	!	{BRANCH move524}						

The [move524] routine allows you to move the cell pointer to any cell in the worksheet using the direction keys. When the cell pointer is in place, you can use the rest of the menu options in the custom menu. The routine starts with "`Use the arrow keys to move the cell pointer and press ENTER:`" which is displayed in the panel. Lotus writes the message into the panel because it does not contain any valid Lotus command. Then the routine continues with `{GET key524}`, which stops the macro execution and waits until you read the message and press a key. When you press a key, Lotus stores the key in the cell named [key524] and the macro issues `{ESC}` to clear the message from the panel before Lotus writes the message to the current cell.

The routine continues with series of `{IF}` commands to check the key that you pressed. The

first is {IF key524="~"}, which checks if you pressed ENTER. If so, the macro issues {RETURN} to return to the custom menu. The second is

```
{IF key524="{PGDN}"#OR#key524="{PGUP}"#OR#key524="{DOWN}"#OR#
    key524="{UP}"#OR#key524="{LEFT}"#OR#key524="{RIGHT}"}
```

which checks if you pressed one of the direction keys in the numeric key pad. If so, the macro issues the {key504} routine command, which starts the [key504] routine. Recall that [key504] contains the key that you pressed, therefore when the macro issues the {key504} routine command, the key is executed as though it was pressed from the keyboard. The third is

```
{IF @LEFT(rel524,1)<>"@"#AND#(key524="{NS}"#OR#key524="{PS}")}
```

which checks if you are using a 3-D Lotus release (and if, at the same time, you pressed the NEXT SHEET (CTRL-PGUP) or the PREVIOUS SHEET (CTRL-PGDN) commands. If so, the macro issues the {key524} routine command, which starts the [key524] routine. Recall that [key524] contains the key that you pressed; therefore when the macro issues the {key524} routine command, the key is executed as though it was pressed from the keyboard. If none of the {IF} commands is true, the macro issues {BRANCH move524} which loops back to the beginning of the [move524] routine and displays the message prompt. This routine is a fine example of how we can monitor and control the keys which are allowed from a macro. The tenth menu option is [**Q**uit]:

```
Quit
Quit the macro
{BRANCH ret524}
```

When you choose this menu option, the macro issues {BRANCH ret524} which routes macro control to the empty [ret524] routine and quits .

Because this macro is quite complicated, we shoe a full list of all the cell contents and formulas:

```
A1: U '*---A macro to JUMP to the ends of row, columns, sheets
A2: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A3: '    range names in this column (starts with the \Z macro name)
A4: '*---Hold the [ALT] key and press [Z] to activate the macro
A5: '!
A6: U '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
A7: U '                   IT WILL WORK IN RELEASE 2.0 AND UP
A8: '!
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'JUMPENDS
B11: '{LET rel524,@INFO("release")}~{MENUBRANCH menu1524}
A12: '!
A13: 'menu1524
B13: 'Right
C13: 'Left
D13: 'Up
E13: 'Down
F13: 'In
G13: 'Out
H13: 'Move
I13: 'Quit
A14: '!
B14: 'Jump to the rightmost occupied cell in this row
C14: 'Jump to the leftmost occupied cell in this row
D14: 'Jump to the upper occupied cell in this column
E14: 'Jump to the lower occupied cell in this column
```

```
F14: 'Jump to the last sheet with a cell at the same address
G14: 'Jump to the first sheet with a cell at the same address
H14: 'Move the cell pointer to other location
I14: 'Quit the macro
A15: '!
B15: '{rightaa524}
C15: '{leftaa524}
D15: '{topaa524}
E15: '{bottomaa524}
F15: '{IF @LEFT(rel524,1)<>"@"}{inaa524}
G15: '{IF @LEFT(rel524,1)<>"@"}{outaa524}
H15: '{move524}
I15: '{BRANCH ret524}
A16: '!
B16: '{MENUBRANCH menu1524}
C16: '{MENUBRANCH menu1524}
D16: '{MENUBRANCH menu1524}
E16: '{MENUBRANCH menu1524}
F16: '{MENUBRANCH menu1524}
G16: '{MENUBRANCH menu1524}
H16: '{MENUBRANCH menu1524}
A17: '!
A18: 'ret524
A19: '!
A20: 'rel524
B20: '3.00.00
A21: '!
A22: 'top524
B22: U @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER("address"),1)
       +1)&"1"
A23: '!
A24: 'bottom524
B24: U @LEFT(@CELLPOINTER("address"),@FIND("$",@CELLPOINTER("address"),1)
       +1)&"8192"
A25: '!
A26: 'left524
B26: U +"A"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER("address"))
       -@FIND("$",@CELLPOINTER("address"),1)-1)
A27: '!
A28: 'right524
B28: U +"IV"&@RIGHT(@CELLPOINTER("address"),@LENGTH(@CELLPOINTER("address"))
       -@FIND("$",@CELLPOINTER("address"),1)-1)
A29: '!
A30: 'out524
B30: U +"A:"&@CELLPOINTER("address")
A31: '!
A32: 'inA524
B32: '$U:$H$38
A33: '!
A34: 'dum524
B34: '$D$10
A35: '!
A36: 'ERATOP524
B36: 'A:$D$10..$J:$D$10
A37: '!
A38: 'dumc524
B38: '$J:$D$10
A39: '!
A40: 'topaa524
B40: '{WINDOWSOFF}{RECALC top524}{GOTO}{top524}~{IF @CELLPOINTER("type")=
     "b"}{END}{DOWN}
A41: '!
B41: '{WINDOWSON}
A42: '!
A43: 'bottomaa524
B43: '{WINDOWSOFF}{RECALC bottom524}{GOTO}{bottom524}~{IF @CELLPOINTER
     ("type")="b"}{END}{UP}
A44: '!
B44: '{WINDOWSON}
A45: '!
A46: 'leftaa524
```

```
B46: '{WINDOWSOFF}{RECALC left524}{GOTO}{left524}~{IF @CELLPOINTER("type")
     ="b"}{END}{RIGHT}
A47: '!
B47: '{WINDOWSON}
A48: '!
A49: 'rightaa524
B49: '{WINDOWSOFF}{RECALC right524}{GOTO}{right524}~{IF @CELLPOINTER("type")
     ="b"}{END}{LEFT}
A50: '!
B50: '{WINDOWSON}
A51: '!
A52: 'inaa524
B52: '{WINDOWSOFF}{IF @LEFT(rel524,1)<>"@"}{LET dum524,@CELLPOINTER
     ("address")}~{LC}{END}{NS}{GOTO}{dum524}~{IF @CELLPOINTER("type")="b"}
     {END}{PS}
A53: '!
B53: '{WINDOWSON}
A54: '!
A55: 'outaa524
B55: '{WINDOWSOFF}{IF @LEFT(rel524,1)<>"@"}{RECALC out524}{GOTO}{out524}~
     {IF @CELLPOINTER("type")="b"}{END}{NS}
A56: '!
B56: '{WINDOWSON}
A57: '!
A58: 'move524
B58: 'Use the arrow keys to move the cell pointer and press ENTER:
     {GET key524}{ESC}
A59: '!
B59: '{IF key524="~"}{RETURN}
A60: '!
B60: '{IF key524="{PGDN}"#OR#key524="{PGUP}"#OR#key524="{DOWN}"#OR#key524=
     "{UP}"#OR#key524="{LEFT}"#OR#key524="{RIGHT}"}{key524}
A61: '!
B61: '{IF @LEFT(rel524,1)<>"@"#AND#(key524="{NS}"#OR#key524="{PS}")}{key524}
A62: '!
B62: '{BRANCH move524}
A63: '!
A64: 'key524
B64: '~
```

[6] Extra Cell and Range Move Options

```
                    A                B                  C              D              E       F
 1 *---A macro with standard and extra SPECIAL MOVE options :
 2     1) Standard move  2) Move to target cell and stay there 3) Move from
 3     there to here  4) Repeat last move
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8 !
 9 !
10 \Z               {BREAKON}
11 MOVESPCL         {LET rel205,@INFO("release")}~{RECALC froma205}
                    {RECALC fromb205}{MENUBRANCH menu205}
12 !
13 menu205          Standard_Move   Move_&_go  From_move   Repeat  Quit
14 !                Standard 1-2-3 MMove the ceMove from otRepeat LQuit th
15 !                {LET flag205,1}~{LET flag20{froma205}  {LET fla{IF fla
16 !                /MRange to move /MRange to {MENUBRANCH {GOTO}wh{LET fl
17 !                {MENUBRANCH menu{MENUBRANCHmenu205}    {MENUBRA{BRANCH
18 !
19 key205           {RIGHT}
20 flag205                         0
21 flaga205                        0
22 here205          $D:$E$3
23 !
24 rel205           3.00.00
25 !
26 loc205           coord
27 !
28 froma205         Move the cell pointer to the target location and press
                    [ENTER] {GET key205}{ESC}~{LET here205,@CELLPOINTER
                    ("coord")}~
29 !                {IF key205="~"}{LET flag205,1}{WINDOWSOFF}{PANELOFF}
                    /RNCRange to move ?~~/RNDRange to move ?~/RNC{PANELON}
                    Range to move ?~{BS}{BS}{WINDOWSON}{?}~/MRange to move ?~
                    {here205}~
                    {MENUBRANCH menu205}
30 fromb205         {key205}{?}~{LET here205,@CELLPOINTER("coord")}~
                    {LET flag205,1}{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~
                    ~/RNDRange to move ?~/RNC{PANELON}Range to move ?~{BS}
                    {BS}{WINDOWSON}{?}~/MRange to move ?~{here205}~{RETURN}
31 !
32 ret205
```

The macro uses a custom menu of five items, and each of them in a different column, therefore the menu items code in the B13..F17 range cannot fit on one page without overlapping, so we show each of the menu option codes separately. If you intend to key this macro into 1-2-3, you need to key the code as it appears here for every menu item, NOT the code as it appears in the main listing.

```
Standard_Move
Standard 1-2-3 Move
{LET flag205,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~~
/RNDRange to move ?~/RNC{PANELON}Range to move ?~{BS}{BS}
{WINDOWSON}{?}~/MRange to move ?~{?}~
{MENUBRANCH menu205}

Move_&_go
Move the cell and the cell pointer to the target cell
{LET flag205,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~~
/RNDRange to move ?~/RNC{PANELON}Range to move ?~{BS}{BS}
{WINDOWSON}{?}~{GOTO}{?}~~/MRange to move ?~~
{MENUBRANCH menu205}

From_move
```

```
Move from other place to here
{froma205}
{MENUBRANCH menu205}

Repeat
Repeat Last move at the Current Cell.
{LET flag205,1}{LET flaga205,1}{WINDOWSOFF}{PANELOFF}/RNC
    Where to ?~~
/RNDwhere to ?~/RNC{PANELON}Where to ?~{BS}{BS}{WINDOWSON}{?}~
{GOTO}where to ?~/MRange to move ?~~
{MENUBRANCH menu205}

Quit
Quit the macro
{IF flag205=1}/RNDRange to move ?~
{LET flag205,0}~
{BRANCH ret205}
```

Notice that the code in the B26, B28 and the B30 cells is the result of dynamic string formulas. If you intend to key this macro into 1-2-3, you need to key the formulas as they appear here, NOT the code which appears in the main macro text.

```
26 loc205          @IF(@LEFT(B24,1)<>"@","coord","address")
28 froma205        +"Move the cell pointer to the target location and
                   press [ENTER] {GET key205}{ESC}~{LET here205,
                   @CELLPOINTER(""&B26&"")}~"
30 fromb205        +"{key205}{?}~{LET here205,@CELLPOINTER(""&B26&"")}~
                   {LET flag205,1}{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~~
                   /RNDRange to move ?~/RNC{PANELON}Range to move ?~{BS}{BS}
                   {WINDOWSON}{?}~/MRange to move ?~{here205}~{RETURN}"
```

This macro adds the following move options: 1. Move to target cell and stay there 2. Move from remote address to here 3. Repeat last move. The third is like the standard move, but saves you the re-definition of the origin range.

	A	B	C	D	E	F
11	MOVESPCL	{LET rel205,@INFO("release")}~{RECALC froma205}				
		{RECALC fromb205}{MENUBRANCH menu205}				

The `{LET rel205,@INFO("release")}~` store the result of the `@INFO("release")` function in the B24 cell named [rel205], which the macro uses later to decide if you are using a 3-D release or 2-D Lotus release of Lotus 1-2-3. Then the macro recalculates the formula in the [froma205] and the [fromb205] cells, which are the dynamic part of the macro code. The result of these two formulas is based on the formula in cell [loc205], which is the formula that determines if you are using a 2-D or a 3-D Lotus release. If you are using a 2-D release, the result of the formula in [loc205] will be the "address" string. If you are using a 3-D release, the result of the formula in [loc205] will be the "coord" string. When the macro finishes recalculating the two formulas, it issues `{MENUBRANCH menu205}` to branch to the [menu205] custom menu.

```
Standard_Move
Standard 1-2-3 Move
{LET flag205,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~~
/RNDRange to move ?~/RNC{PANELON}Range to move ?~{BS}{BS}
{WINDOWSON}{?}~/MRange to move ?~{?}~
{MENUBRANCH menu205}
```

The first menu option is the [**S**tandard_Move] which is the same move as the standard Lotus move option. First the macro sets the value in cell [flag205] to "1" using `{LET flag205,1}~`, then the macro freezes the screen and panel activities using `{WINDOWSOFF}{PANELOFF}` and starts a process to create a range name using:

```
/RNCRange to move ?~~/RNDRange to move ?~
    /RNC{PANELON}Range to move ?~{BS}{BS}{WINDOWSON}{?}~
```

Here the macro combines two techniques. It creates the [Range to move ?] range name for the range to be moved and simultaneously uses the range name as a prompt to you. When you paint the range and press ENTER, the macro issues `/MRange to move ?~{?}~` to start the move procedure. The `{?}` macro command pauses the macro and waits for you to point to the target location and to press ENTER. When you have pressed ENTER, the macro branches back to the custom menu using `{MENUBRANCH menu205}`.

```
Move_&_go
Move the cell and the cell pointer to the target cell
{LET flag205,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~~
/RNDRange to move ?~/RNC{PANELON}Range to move ?~{BS}{BS}
{WINDOWSON}{?}~{GOTO}{?}~~/MRange to move ?~~
{MENUBRANCH menu205}
```

The second menu option is [**M**ove_&_go]. You can use this to move a range from any place to a new target place. When the move process ends, the cell pointer will be positioned on the target location instead of the origin location. The macro uses the same code as in the previous menu option except `{GOTO}{?}~~` which prompts you to move the cell pointer to the target location. When the cell pointer is on the target location the macro issues `/MRange to move ?~~` that moves the range named [Range to move ?] to the target location, but this time, the cell pointer will stay at the target location.

```
From_move
Move from other place to here
{froma205}
{MENUBRANCH menu205}
```

The third menu option is [**F**rom_move]. You can use this to move a range from any place to the current cell position. When you select this option, the macro issues the `{froma205}` routine command which activates the [froma205] routine.

	A	B	C	D	E	F
28	froma205	Move the cell pointer to the target location and press [ENTER] {GET key205}{ESC}~{LET here205,@CELLPOINTER ("coord")}~				

Because Lotus cannot find a valid command in the following code it writes it to the panel which displays:

```
Move the cell pointer to the target location and press [ENTER]
```

and immediately issues `{GET key205}`, stopping the macro execution and waiting until you press a key. The macro stores this key in cell [key205] and issues `{ESC}` to clear the message prompt from the panel. Now the macro stores the cell pointer position using:

```
{LET here205,@CELLPOINTER("coord")}~
```

The code in the [froma205] routine is the result of the string formula:

```
+"Move the cell pointer to the target location and press [ENTER]
    {GET key205}{ESC}~{LET here205,@CELLPOINTER(""&B26&"")}~"
```

	A	B	C	D	E	F

```
29 !                {IF key205="~"}{LET flag205,1}{WINDOWSOFF}{PANELOFF}
                    /RNCRange to move ?~~/RNDRange to move ?~/RNC{PANELON}
                    Range to move ?~{BS}{BS}{WINDOWSON}{?}~/MRange to move ?~
                    {here205}~
                    {MENUBRANCH menu205}
```

Next the macro issues `{IF key205="~"}` to check if you pressed ENTER (represented by the tilde "~" in the Lotus macro language). If so, it means that you want to copy the range into the current cell position. The macro inserts the value of "1" into cell [flag205] using `{LET flag205,1}` and freezes the screen and panel activities using `{WINDOWSOFF}{PANELOFF}` and starts a process to create a range name using:

```
/RNCRange to move ?~~/RNDRange to move ?~/RNC{PANELON}
    Range to move ?~{BS}{BS}{WINDOWSON}{?}~
```

The macro combines two techniques: it creates the [Range to move ?] range name for the range to be moved, and simultaneously uses the range name as a prompt. Move the cell pointer to the range to be moved, paint the range, and press ENTER. When you paint the range and press ENTER the macro starts the move procedure using `/MRange to move ?~{here205}~` and then branches back to the custom menu using `{MENUBRANCH menu205}`. The `{here205}~` routine command injects the content of [here205] into the panel as a response to the prompt to enter the target cell.

	A	B	C	D	E	F
30	fromb205	{key205}{?}~{LET here205,@CELLPOINTER("coord")}~ {LET flag205,1}{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~ ~/RNDRange to move ?~/RNC{PANELON}Range to move ?~{BS} {BS}{WINDOWSON}{?}~/MRange to move ?~{here205}~{RETURN}				

If you press any other key, the macro issues the `{key205}` routine command, activating the key that you have just pressed, and then issues `{?}`. Move the cell pointer to the target location and press ENTER. The macro records the address of the target cell in [here205] and stores the value of "1" in [flag205], which is used as a flag for the macro when you choose the **Q**uit option. Next the macro freezes the screen and panel activities using `{WINDOWSOFF} {PANELOFF}` and again uses the "safe technique" to assign the [Range to move ?] range name to the range to be moved and simultaneously uses the range name as a prompt. Last, the macro moves the [Range to move ?] range to the target cell and ends the routine using `/MRange to move ?~{here205}~{RETURN}`.

```
Repeat
Repeat Last move at the Current Cell.
{LET flag205,1}{WINDOWSOFF}{PANELOFF}/RNCwhere to ?~~
/RNDwhere to ?~/RNC{PANELON}Where to ?~{BS}{BS}{WINDOWSON}{?}~
{GOTO}where to ?~/MRange to move ?~~
{MENUBRANCH menu205}
```

This time the macro issues `{LET flag205,1}`, which writes the "1" label in cell [flag205], and then `{WINDOWSOFF}{PANELOFF}`, which freeze the screen and panel display activities. Next the macro assigns the [Where to ?] range name for the target range and simultaneously uses the range name as a prompt. Move the cell pointer to the target range and press ENTER. The macro then issues `{GOTO}where to?~/MRange to move ?~~` and moves the range. Last, the macro issues `{MENUBRANCH menu205}` to route macro control back to the main menu.

```
Quit
Quit the macro
```

```
{IF flag205=1}/RNDRange to move ?~
{LET flag205,0}~
{BRANCH ret205}
```

In the [**Q**uit] menu option, the macro issues `{IF flag205=1}` to check the content of cell [flag205]. If it contains the number "1", the worksheet contains a range with the [Range to move ?] range name; therefore the macro issues `/RNDRange to move ?~`, which deletes this name and issues `{LET flag205,0}~` which resets the content of [flag205] to "0" before it quits, thereby leaving a worksheet that is free from unnecessary range names. To further assist you to key this macro into Lotus 1-2-3 we bring here the complete cell content list of the macro.

```
A1: U [W10] '*---A macro with standard and extra SPECIAL MOVE options :
A2: U [W10] '    1) Standard move  2) Move to target cell and stay there
           3) Move from
A3: U [W10] '    there to here  4) Repeat last move
A4: [W10] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A5: [W10] '    range names in this column (starts with the \Z macro name)
A6: [W10] '*---Hold the [ALT] key and press [Z] to activate the macro
A7: [W10] '!
A8: [W10] '!
A9: [W10] '!
A10: U [W10] '\Z
B10: [W15] '{BREAKON}
A11: U [W10] 'MOVESPCL
B11: [W15] '{LET rel205,@INFO("release")}~{RECALC froma205}{RECALC fromb205}
           {MENUBRANCH menu205}
A12: [W10] '!
A13: [W10] 'menu205
B13: [W15] 'Standard_Move
C13: 'Move_&_go
D13: 'From_move
E13: 'Repeat
F13: 'Quit
A14: [W10] '!
B14: [W15] 'Standard 1-2-3 Move
C14: 'Move the cell and the cell pointer to the target cell
D14: 'Move from other place to here
E14: 'Repeat Last move at the Current Cell.
F14: 'Quit the macro
A15: [W10] '!
B15: [W15] '{LET flag205,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~~
           /RNDRange to move ?~/RNC{PANELON}Range to move ?~{BS}{BS}{WINDOWSON}{?}~
C15: '{LET flag205,1}~{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~~/RND
     Range to move ?~/RNC{PANELON}Range to move ?~{BS}{BS}{WINDOWSON}{?}~
     {GOTO}{?}~~
D15: '{froma205}
E15: '{LET flag205,1}{LET flaga205,1}{WINDOWSOFF}{PANELOFF}/RNCWhere to ?~~
     /RNDWhere to ?~/RNC{PANELON}Where to ?~{BS}{BS}{WINDOWSON}{?}~
F15: '{IF flag205=1}/RNDRange to move ?~
A16: [W10] '!
B16: [W15] '/MRange to move ?~{?}~
C16: '/MRange to move ?~~
D16: '{MENUBRANCH menu205}
E16: '{GOTO}Where to ?~/MRange to move ?~~{LET flaga205,0}~
F16: '{LET flag205,0}{LET flaga205,0}~
A17: [W10] '!
B17: [W15] '{MENUBRANCH menu205}
C17: '{MENUBRANCH menu205}
E17: '{MENUBRANCH menu205}
F17: '{BRANCH ret205}
A18: [W10] '!
A19: [W10] 'key205
B19: [W15] '{RIGHT}
A20: [W10] 'flag205
B20: [W15] 0
A21: [W10] 'flaga205
B21: [W15] 0
```

```
A22: [W10] 'here205
B22: [W15] '$D:$E$3
A23: [W10] '!
A24: [W10] 'rel205
B24: [W15] '3.00.00
A25: [W10] '!
A26: [W10] 'loc205
B26: U [W15] @IF(@LEFT(B24,1)<>"@","coord","address")
A27: [W10] '!
A28: [W10] 'froma205
B28: U [W15] +"Move the cell pointer to the target location and press
             [ENTER] {GET key205}{ESC}~{LET here205,@CELLPOINTER
             (""""&B26&""")}~"
A29: [W10] '!
B29: [W15] '{IF key205="~"}{LET flag205,1}{WINDOWSOFF}{PANELOFF}/RNC
           Range to move ?~~/RNDRange to move ?~/RNC{PANELON}Range to move ?~
           {BS}{BS}{WINDOWSON}{?}~/MRange to move ?~{here205}~
           {MENUBRANCH menu205}
A30: [W10] 'fromb205
B30: U [W15] +"{key205}{?}~{LET here205,@CELLPOINTER(""""&B26&""")}~
             {LET flag205,1}{WINDOWSOFF}{PANELOFF}/RNCRange to move ?~~/RND
             Range to move ?~/RNC{PANELON}Range to move ?~{BS}{BS}
             {WINDOWSON}{?}~/MRange to move ?~{here205}~{RETURN}"
A31: [W10] '!
A32: [W10] 'ret205
```

[1] Use Point and Shoot to Jump to a Range

```
     A              B              C              D              E
 1 *---A macro to GO TO any range in the worksheet
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 GOTORANG        {GOTO}{NAME}{NAME}{?}~{INDICATE}{RESTART}
```

This is a simple macro to move to any named range in the macro using point and shoot. The macro starts with the `{GOTO}` macro command which is like pressing the F5 function key from the keyboard and then issues `{NAME}` twice to display a full screen list of all the range names in the worksheet. Next the macro issues `{?}` to pause and wait for you to point (highlight) the desired range name and press ENTER. Then, the macro issues the tilde "`~`" which is the same as the ENTER key and moves the cell pointer to the named range.

The `{INDICATE}` and the `{RESTART}` commands are used here so that the macro will work properly with the macro manager. When this macro is used from the macro manager, the manager uses the `{INDICATE "message......."}` macro command to display a message. When we use this macro, we want the cell pointer to move to a named range and to exit to READY mode; therefore we also need to stop the manager. Using `{RESTART}` causing the macro to lose control. However we also need to reset the `{INDICATE "message....."}`. The `{INDICATE}` macro command, without a parameter, erases the message and returns the control over the INDICATE area to Lotus 1-2-3.

[6] Move the Cell Pointer Between the Worksheet Corners

```
     A              B                C              D              E
 1 *---A macro to move the cell pointer around the worksheet corners
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 CORNERS         {LET rel202,@INFO("release")}~{RECALC loc202}
                   {RECALC form202}
12 form202         {LET here1#202,@CELLPOINTER("coord")}
13 loop#202        {WINDOWSON}{PANELON}[.]corners [N]next sheet
                   [P]revious sheet [S]tay, any other key to return
                   {GET key1#202}{ESC}{IF key1#202="."}{BRANCH main#202}
14 !               {IF @UPPER(key1#202)="S"}{QUIT}
15 !               {IF @LEFT(rel202,1)<>"@"#AND#@UPPER(key1#202)="N"}{NS}
                   {last#202}{RECALC form202}{BRANCH form202}
16 !               {IF @LEFT(rel202,1)="@"#AND#@UPPER(key1#202)="N"}
                   {BRANCH form202}
17 !               {IF @LEFT(rel202,1)<>"@"#AND#@UPPER(key1#202)="P"}{PS}
                   {last#202}{RECALC form202}{BRANCH form202}
18 !               {IF @LEFT(rel202,1)="@"#AND#@UPPER(key1#202)="P"}
                   {BRANCH form202}
19 !               {GOTO}{here1#202}~
20 !
21 main#202        {WINDOWSOFF}{PANELOFF}{GOTO}first#202~/M~{END}{DOWN}
                   {DOWN}~
22 !               {DOWN}/M.{END}{DOWN}~{UP}~
23 !               {UP}{LEFT}first#202~/RNLR~{END}{DOWN}last#202~/RNLR~
24 !               {GOTO}{here1#202}~{last#202}{BRANCH loop#202}
25 !
26 ret#202
27 !
28 here1#202       $C:$D$1
29 !
30 key1#202        S
31 !
32 rel202          3.00.00
33 !
34 loc202          coord
35 first#202       {HOME}
36 !               {END}{HOME}{DOWN}{END}{LEFT}{UP}
37 !               {END}{HOME}
38 last#202        {END}{HOME}{RIGHT}{END}{UP}{LEFT}
```

This macro allows you to jump with the cell pointer to the four remote corners of the occupied worksheet in a cyclic direction. Before we continue, notice that the codes in B12 and B34 are the result of the following two dynamic string formulas. If you intend to key this macro into 1-2-3, you have to key the formula codes, NOT the code as it appears in main macro text.

```
12 form202         +"{LET here1#202,@CELLPOINTER("""&B34&""")}"
34 loc202          @IF(@LEFT(B32,1)<>"@","coord","address")
```

```
     A              B                C              D              E
11 CORNERS         {LET rel202,@INFO("release")}~{RECALC loc202}
                   {RECALC form202}
12 form202         {LET here1#202,@CELLPOINTER("coord")}
```

The macro starts with the `{LET rel202,@INFO("release")}~` macro commands which store the result of the `@INFO("release")` function in the B32 cell named [rel202] for later

use, to check if you are using a 3-D or a 2-D release. Next the macro issues {RECALC loc202}{RECALC form202} to update the two dynamic formulas in the cells [loc202] and [form202]. The {LET here1#202,@CELLPOINTER("coord")} command stores the current cell position in cell [here1#202]. The code in [form202] is the result of:

```
+"{LET here1#202,@CELLPOINTER("""&B34&""")}"
```

dynamic formula. We can see that the "dynamic" part of this formula is the content of B34 cell [loc202]. However the content of [loc202] is a result of the:

```
@IF(@LEFT(B32,1)<>"@","coord","address")
```

dynamic formula. The "dynamic" part of this formula is the content of the B32 cell named [rel202]. Recall [rel202] holds the result of the @INFO("release") function. The formula in [form202] can have two results. If you are using a 3-D release, the result of the @LEFT(rel202,1) function is unequal to the "@" character and the result is the "coord" string. If you are using Lotus 3.0/3.1/3.1+/3.4 the result is "3". If you are using Lotus for Windows 1.0/1.0a/1.1 the result is "1". If you are using a 2-D release the result of the @LEFT(rel202,1) function is equal to the "@" character and the result of the formula in [form202] is the "address" string. Therefore the formula in [loc202] can have two forms. If you are using a 2-D release the formula accepts the following form:

```
{LET here1#202,@CELLPOINTER("address")}
```

If you are using a 3-D release the formula accepts the following form:

```
{LET here1#202,@CELLPOINTER("coord")}
```

	A	B	C	D	E
13	loop#202	{WINDOWSON}{PANELON}[.]corners [N]next sheet [P]revious sheet [S]tay, any other key to return {GET key1#202}{ESC}{IF key1#202="."}{BRANCH main#202}			

Now the macro issues {WINDOWSON}{PANELON} to free the screen and panel activities, and types

```
[.]corners [N]next sheet [P]revious sheet [S]tay, any other key to return
```

message to the panel and issues the {GET key1#202} macro command that stop the macro execution and waits for your response. The message is typed into the panel because Lotus can find a valid Lotus command in the text. When you press a key the macro stores the key in cell [key1#202] and immediately issues {ESC} to clear the message from the panel before Lotus writes it into the current cell.. If you press the period [.] key, the macro issues {BRANCH main#202} to start the [main#202] routine.

	A	B	C	D	E
21	main#202	{WINDOWSOFF}{PANELOFF}{GOTO}first#202~/M~{END}{DOWN} {DOWN}~			
22	!	{DOWN}/M.{END}{DOWN}~{UP}~			
23	!	{UP}{LEFT}first#202~/RNLR~{END}{DOWN}last#202~/RNLR~			
24	!	{GOTO}{here1#202}~{last#202}{BRANCH loop#202}			

The [main#202] routine issues {WINDOWSOFF}{PANELOFF} to freeze the screen and panel activities, and then issues {GOTO}first#202~, which moves the cell pointer to cell [first#202].

To understand the next macro code, let's look at the following range of the macro:

	A	B	C	D	E
35	first#202	{HOME}			
36	!	{END}{HOME}{DOWN}{END}{LEFT}{UP}			
37	!	{END}{HOME}			
38	last#202	{END}{HOME}{RIGHT}{END}{UP}{LEFT}			
39	!				

Cell [first#202] contains {HOME}; the second cell below contains {END}{HOME}{DOWN} {END}{LEFT}{UP} that move the cell pointer to the lowest left corner of the worksheet. The next cell contains {END}{HOME} that move the cell pointer to the lowest right corner of the worksheet and the content of cell [last#202] is {END}{HOME}{RIGHT}{END}{UP}{LEFT} that move the cell pointer to the highest right corner of the worksheet. The cell pointer is now on [first#202] containing {HOME}. The next code in the [main#202] routine is /M~{END} {DOWN}{DOWN}~, move the content of [first#202] to the cell bellow [last#202]. Now the worksheet looks like:

	A	B	C	D	E
36	first#202				
37	!	{END}{HOME}{DOWN}{END}{LEFT}{UP}			
38	!	{END}{HOME}			
39	last#202	{END}{HOME}{RIGHT}{END}{UP}{LEFT}			
25	!	{HOME}			

When the macro issues {DOWN}/M.{END}{DOWN}~{UP}~ that push the four cell up one row, the worksheet looks like:

	A	B	C	D	E
36	first#202	{END}{HOME}{DOWN}{END}{LEFT}{UP}			
37	!	{END}{HOME}			
38	!	{END}{HOME}{RIGHT}{END}{UP}{LEFT}			
39	last#202	{HOME}			
25	!				

Next the [main#202] routine issues {UP}{LEFT}first#202~/RNLR~ to re-assign the [first#202] range name. When the macro moved the content of [first#202], the name moved with it. Therefore the macro needs to re-assign the name. By issuing {END}{DOWN} last#202~/RNLR~ the [last#202] range name is re-assigned. Now the routine issues the {GOTO}{here1#202}~ indirect macro command which moves the cell pointer back to its origin cell and then {last#202} executes the macro commands in [last#202]. In this case, [last#202] contains {HOME}; therefore the cell pointer moves to the upper left corner of the worksheet. Every time you press the period [.] key, the content of [first#202] is moved to the bottom of the list and the list is pushed one cell up. Then the macro executes the command in [last#202]. This way a cyclic movement of the cell pointer is achieved every time you press the period [.] key. Till now, we have covered pressing the period key, only. The next series of {IF} commands deal with all the other cases.

	A	B	C	D	E
14	!	{IF @UPPER(key1#202)="S"}{QUIT}			
15	!	{IF @LEFT(rel202,1)<>"@"#AND#@UPPER(key1#202)="N"}{NS} {last#202}{RECALC form202}{BRANCH form202}			
16	!	{IF @LEFT(rel202,1)="@"#AND#@UPPER(key1#202)="N"} {BRANCH form202}			
17	!	{IF @LEFT(rel202,1)<>"@"#AND#@UPPER(key1#202)="P"}{PS} {last#202}{RECALC form202}{BRANCH form202}			
18	!	{IF @LEFT(rel202,1)="@"#AND#@UPPER(key1#202)="P"} {BRANCH form202}			

```
19 !              {GOTO}{here1#202}~
```

The first {IF @UPPER(key1#202)="S"} command checks if you pressed the "S" or the "s" characters. If so, the macro issues {QUIT} and quits. This option lets you stop the macro when you want to stay on the current corner.

The second {IF @LEFT(rel202,1)<>"@"#AND#@UPPER(key1#202)="N"} command checks if you are using a 3-D Lotus release and that you pressed the "N" or the "n" character to move to the next sheet. If so, the macro issues {NS}, which moves the cell pointer to the next sheet and then {last#202} activating the [last#202] routine. For example, if the [last#202] routine contains {HOME}, the cell pointer moves to the upper left corner of the new sheet. Because the cell pointer moved to a new sheet, the macro issues {RECALC form202} to update the formula in [form202]. Then it issues {BRANCH form202} to display the message and start all over again.

The third {IF @LEFT(rel202,1)="@"#AND#@UPPER(key1#202)="N"} command checks if you are using a 2-D Lotus release, even though you pressed the "N" or the "n" character to move to the next sheet. If so, the macro issues {BRANCH form202} to display the message and to start all over again.

The fourth {IF @LEFT(rel202,1)<>"@"#AND#@UPPER(key1#202)="P"} command checks if you are using a 3-D Lotus release and that you pressed the "P" or the "p" character to move to the previous sheet. If so, the macro issues {PS}, which moves the cell pointer to the previous sheet and then {last#202} activating the [last#202] routine. For example, if the [last#202] routine contains {HOME}, the cell pointer moves to the upper left corner of the new sheet. Because the cell pointer moved to a new sheet, the macro issues {RECALC form202} to update the formula in [form202]. Then it issues {BRANCH form202} to display the message and to start all over again.

The fifth {IF @LEFT(rel202,1)="@"#AND#@UPPER(key1#202)="P"} command checks if you are using a 2-D Lotus release, even though you pressed the "P" or the "p" character to move to the previous sheet. If so, the macro issues {BRANCH form202} to display the message and to start all over again. If you press any other key the macro issues {GOTO} {here1#202}~ to move the cell pointer back to its origin address and quit.

Macros Affecting Range Names

[0] Name Cells to the Right

	A	B	C	D	E
1	*---A macro to name the cells RIGHT to the current cells				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer on the uppermost name				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	*---If there is ONLY one name just press [ENTER] otherwise "PAINT"				
7	the NAMES range using the direction keys and press [ENTER]				
8	!				
9	!				
10	\Z	{BREAKON}			
11	NAMRIGHT	/RNLR{?}~			

For macro users, naming cells is one of the busy operations they do. When you use or build a macro or a macro based application, you need to name ranges in the worksheet. The most efficient way is to use the / **R**ange **N**ame **L**abel method where the name text is to the left, right, up or down to the range to be named.

This simple macro saves few key strokes every time you need to assign a range name. The macro starts with the `/RNLR` macro keys and then issues `{?}`, which halts macro execution and pauses until you paint the cell/range to the left of the cell/s to be named. When you press ENTER, the macro issues the tilde "`~`", which is the same as the ENTER key and finishes the operation.

In the next page you can see the same macro for naming to the left, up and down.

[0] Name Cells to the Left

	A	B	C	D	E
1	*---A macro to name the cell LEFT to the current cell				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer on the upper most name				
5	*---Hold the [ALT] key and press Z to activate the macro				
6	*---If there is ONLY one name just press [ENTER] otherwise "PAINT"				
7	the NAMES range using the direction keys and press [ENTER]				
8	!				
9	!				
10	\Z	{BREAKON}			
11	NAMELEFT	/RNLL{?}~			

See the NAMRIGHT.WK1 macro.

[0] Name Cells Down

	A	B	C	D	E
1	*---A macro to name the cells DOWN to the current cells				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer on the left most name				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	*---If there is ONLY one name just press [ENTER] otherwise "PAINT"				
7	the NAMES range using the direction keys and press [ENTER]				
8	!				
9	!				
10	\Z	{BREAKON}			
11	NAMEDOWN	/RNLD{?}~			

See the NAMRIGHT.WK1 macro.

[0] Name Cells Up

	A	B	C	D	E
1	*---A macro to name the cells UP to the current cells				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer on the leftmost name				
5	*---Hold the [ALT] key and press Z to activate the macro				
6	*---If there is ONLY one name just press [ENTER] otherwise "PAINT"				
7	the NAMES range using the direction keys and press [ENTER]				
8	!				
9	!				
10	\Z	{BREAKON}			
11	NAME_UP	/RNLU{?}~			

See the NAMRIGHT.WK1 macro.

[4] The Range Name Create Macro

```
     A              B                C                D              E
 1 *---A macro to CREATE range names
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer at the cell to be named
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 RANGECRT       {LET key029,""}~
12 loopa029       {PANELOFF}/RN{PANELON}C*{BS}{key029}{?}~~
13 loop029        {PANELON}Move using arrows, Enter name and press [ENTER]
                  or [ESC] to quit
                  {GET key029}{ESC}{PANELOFF}
14 !              {IF key029="{ESC}"}{BRANCH ret029}
15 !              {IF key029="{RIGHT}"#OR#key029="{LEFT}"#OR#key029="{UP}
                  "#OR#key029="{DOWN}"#OR#key029="{PGDN}"#OR#key029="
                  {PGUP}"}{key029}{BRANCH loop029}
16 !              {BRANCH loopa029}
17 !
18 key029         {ESC}
19 !
20 ret029
```

This macro makes the task of range name creating much easier when you need to create large number of range names simultaneously. All you need to do is move the cell pointer to the desired location, type the new range name, paint the range and press ENTER. Then the macro prompts you back to use the direction keys to move and type the next range name. This way, you are freed from the boring task of pressing the / **R**ange **N**ame **C**reate again and again and can concentrate only on the job at hand.

The macro starts with the `{LET key029,""}~` macro commands to write the null string to the cell named [key029]. We could also use `{BLANK key029}~`. Next the macro issues `{PANELOFF}/RN{PANELON}C`, suppressing part of the Lotus prompt in the panel during the / **R**ange **N**ame **C**reate ... process. Then the macro writes the astrix "`*`" character to the panel and issues `{BS}`, which safely clears the panel from any previous range names created earlier.

Now the macro issues the `{key029}` routine command which injects the content of cell [key029] into the panel. Recall that this cell contains the null string at the beginning, therefore it writes nothing to the panel. Later the macro uses the same cell to hold the keys that you press, therefore for the first time, we had to use the null string. Next the macro issues `{?}`, which allows you to type the range name to create. When you press ENTER, the macro issues the tilde "`~`" which is the same as ENTER twice to assign the range name to the current cell.

```
13 loop029        {PANELON}Move using arrows, Enter name and press [ENTER]
                  or [ESC] to quit {GET key029}{ESC}{PANELOFF}
```

Next the macro issues `{PANELON}`, which releases the panel activity and then types the:

```
Move using arrows, Enter name and press [ENTER] or [ESC] to quit
```

prompt message to the panel. To hold the message in the panel until you respond, the macro issues `{GET key029}` which halts the macro execution and waits until you press a key. Lotus

stores the key's macro code in cell [key029] and immediately issues {ESC} to clear the message prompt from the panel, before Lotus writes the message into the current cell.

	A	B	C	D	E
14	!	{IF key029="{ESC}"}{BRANCH ret029}			
15	!	{IF key029="{RIGHT}"#OR#key029="{LEFT}"#OR#key029="{UP}"#OR#key029="{DOWN}"#OR#key029="{PGDN}"#OR#key029="{PGUP}"}{key029}{BRANCH loop029}			

Now the macro starts a series of {IF} conditions to check which key you pressed. We want to allow you to use the direction keys to move the cell pointer or the ESC key to quit. Therefore, the macro issues {IF key029="{ESC}"} to check if you pressed ESC. If so, the macro issues {BRANCH ret029}, which routes macro execution to the [ret029] empty routine and quits. Next the macro issues:

```
{IF key029="{RIGHT}"#OR#key029="{LEFT}"#OR#key029="{UP}"#OR#
    key029="{DOWN}"#OR#key029="{PGDN}"#OR#key029="{PGUP}"}
    {key029}{BRANCH loop029}
```

to check if you pressed one of the direction keys. If so, the macro issues {key029}, which executes the direction command in cell [key029]. For example, if you pressed the RIGHT arrow key, then the cell named [key029] should contain the {RIGHT} string which the macro executes. After the macro executes the direction command, it issues {BRANCH loop029}, which routes the macro control back to the beginning and re-displays the prompt message allowing you to use the direction keys again and again. When you press any other key, the macro issues {BRANCH loopa029}, which loops back to the beginning of the macro and allows you to assign a new range name to the current cell.

[4] Verify Range Name Existence

```
     A                B                C                D                E
 1 *---A macro to VERIFY a range name existence, list all range names
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---When you get the list of the range names use the direction keys to
 6     page through the range names and press [ENTER] to finish.
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 VERIFY          {LET here167,@CELLPOINTER("address")}~
12 !               {PANELOFF}{GOTO}{PANELON}{NAME}{NAME}{?}{ESC 6}~
13 !               {GOTO}{here167}~
14 !
15 here167         $B$12
```

This macro displays a full screen list of all the pre-assigned range names in the worksheet. The macro starts with the `{LET here167,@CELLPOINTER("address")}~` macro commands to store the current cell pointer address in the B15 cell named [here167]. Later the macro uses this address to return to its origin. Next the macro issues `{PANELOFF}` to suppressany panel activity and `{GOTO}` followed by `{PANELON}`, which resumes panel activity. Then it issues `{NAME}{NAME}{?}` which display a full screen list of all the range names in the worksheet. Now you can scroll and use the direction keys to look for any range name. When you press ENTER the macro issues `{ESC 6}` to return to the READY mode and then the `{GOTO}{here167}~` indirect macro command to return the cell pointer to its origin location before the macro started.

[6] Edit Formulas with Range Names

```
     A                B                C                D                E
 1 *---A macro to EDIT a formula using the range names instead of addresses
 2     when the addresses has names but are not shown in the formula
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Place the cell pointer on the formula to be edited
 6 *---When you get the formula as literal, edit it and press [ENTER]
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                    IT WILL WORK IN RELEASE 2.0 AND UP
10 \Z              {BREAKON}
11 EDITFORM        {LET rel521,@INFO("release")}~{IF @LEFT(rel521,1)<>"@"}
                   {BRANCH editfrm3521}
12 !               {WINDOWSOFF}{PANELOFF}/RNChere521~~/RNDhere521~
                    /RNChere521~~
13 !               {CONTENTS here521,here521,240,117}{LET here521,@TRIM
                   (here521)}~{WINDOWSON}{PANELON}{EDIT}{HOME}{DEL}{END}{?}~
14 !               /RNDhere521~
15 !
16 rel521          @INFO("release")
17 !
18 editfrm3521     {PANELOFF}{WINDOWSOFF}{RECALC formula521}
19 !               {CONTENTS formula521,$C$34,240,117}
20 !               {LET formula521,@TRIM(formula521)}
21 !               {change521}{DOWN}{UP}{change521}{PANELON}{WINDOWSON}{?}~
22 !
23 counter521                  9
24 !
25 absabs521       {RIGHT}{ABS}{ABS}
26 !
27 formula521      +A34+B34
28 !
29 change521       /WGDOBNQ{EDIT}{HOME}{FOR counter521,1,@LENGTH
                   (formula521),1,absabs521}
```

When a formula contains addresses of cells that have a range name, and you press the F2 function key to enter the EDIT mode, the formula in the panel sometimes shows cell addresses instead of the range names. This macro displays the formula in the panel always with range names instead of cell addresses. Before we continue, note that the code in the B19 cell is the result of the following dynamic formula:

```
19 !         +"{CONTENTS formula521,"&@CELLPOINTER("address")&",512,117}"
```

Therefore if you want to key this macro into Lotus 1-2-3 you have to key this formula in the B19 cell of the macro.

```
     A                B                C                D                E
11 EDITFORM        {LET rel521,@INFO("release")}~{IF @LEFT(rel521,1)<>"@"}
                   {BRANCH editfrm3521}
12 !               {WINDOWSOFF}{PANELOFF}/RNChere521~~/RNDhere521~
                    /RNChere521~~
13 !               {CONTENTS here521,here521,240,117}{LET here521,@TRIM
                   (here521)}~{WINDOWSON}{PANELON}{EDIT}{HOME}{DEL}{END}{?}~
14 !               /RNDhere521~
```

The macro starts with the `{LET rel521,@INFO("release")}~` macro command which stores the result of the `@INFO("release")` function in the B16 cell named [rel202]. Later the macro uses the stored result to check if you are using a 3-D or a 2-D release. Next the macro issues `{IF @LEFT(rel521,1)<>"@"}` to check if "`@`" is not the first character in the content of [`rel521`]. If so, you are using a 3-D release; therefore, the macro issues `{BRANCH`

{editfrm3521} to activate the [editfrm3521] routine.

	A	B	C	D	E
18	editfrm3521	{PANELOFF}{WINDOWSOFF}{RECALC formula521}			
19	!	{CONTENTS formula521,$C$34,511,117}			
20	!	{LET formula521,@TRIM(formula521)}			
21	!	{change521}{DOWN}{UP}{change521}{PANELON}{WINDOWSON}{?}~			

The [editfrm3521] routine starts with {PANELOFF}{WINDOWSOFF} which freeze the screen and panel activities. Next the macro issues {RECALC formula521} to update the formula in cell [formula521]. Let's assume that the current cell is the $C$34 cell, containing the +A34 +B34 formula, and simultaneously A34 contains the SALE1 name and B34 contains the SALE2 name. The {CONTENTS formula521,$C$34,511,117} command inserts the "+A34+B34" formula as a label in cell [formula521]. The 117 tells Lotus that the format is label and 511 makes it 511 characters long. To make it 511 characters long, Lotus adds spaces to the "+A34+B34" formula. Therefore, {LET formula521,@TRIM(formula521)} uses @TRIM(formula521) to strip all the spaces and leave only the "+A34+B34" text. We have used 511 to allow the macro to work with formulas long up to 511 characters. The Lotus manual and Lotus Help have more details on the {CONTENTS} macro command. The code of {CONTENTS formula521,$C$34,511,117} macro command is the result of the dynamic string formula:

```
19 !         +"{CONTENTS formula521,"&@CELLPOINTER("address")&",511,117}"
```

The "dynamic" part of the formula is the @CELLPOINTER("address") function which enters the address of the current cell into the code. In our example, it is the $C$34 cell address. Now that the cell [formula521] contains the text form of the formula, the macro issues {change521}, which starts the [change521] routine.

	A	B	C	D	E
29	change521	/WGDOBNQ{EDIT}{HOME}{FOR counter521,1,@LENGTH (formula521),1,absabs521}			

To avoid beeps during the work of this routine the macro issues /WGDOBNQ and {EDIT} {HOME} to move the cursor to the beginning of the formula in the panel. Next the macro issues {FOR counter521,1,@LENGTH(formula521),1,absabs521} activating the [absabs521] routine as many times as the length of the formula text in [formula521]. The purpose of using the {CONTENTS} macro command was to create a text form of the current formula so that the macro can use the @LENGTH function to calculate how many times to activate the [absabs521] routine.

	A	B	C	D	E
25	absabs521	{RIGHT}{ABS}{ABS}			

The [absabs521] routine issues {RIGHT}, which moves the cursor to the next character in the panel and issues {ABS} twice which is the equivalent of the F4 function key. When a formula is displayed in the panel after pressing the F2 (EDIT) key, the cursor is on one of the cell addresses, and you press the F4 key, Lotus changes the address from relative address to absolute address, if it was relative address before, and from absolute address to relative address if it was absolute before. When this cell also has a range name, the F4 key also changes the address to the range name and it stays that way. The next F4 key does not change the range name back to address but continues to toggle from absolute to relative. Therefore, pressing the F4 twice turns the cell address to a range name and simultaneously returns it to the correct reference (relative or absolute).

```
     A              B                C              D              E
18 editfrm3521    {PANELOFF}{WINDOWSOFF}{RECALC formula521}
19 !              {CONTENTS formula521,$C$34,240,117}
20 !              {LET formula521,@TRIM(formula521)}
21 !              {change521}{DOWN}{UP}{change521}{PANELON}{WINDOWSON}{?}~
```

Now the [editfrm3521] routine issues `{DOWN}{UP}` which write the transformed formula into the current cell and `{change521}` again, and then `{PANELON}{WINDOWSON}{?}`, which frees the panel and screen activities so that you can view the transformed formula with the range names in the panel and make changes. It seems that the macro does not need to use `{change521}` a second time, but we have found that some versions of 3-D Lotus will not work correctly unless we use this routine twice.

This is another example of the different behavior of different Lotus releases. There is a different routine for a 3-D and 2-D releases because the [editfrm3521] routine is needed only for 3-D Lotus release spreadsheets, even if only one sheet is used. This is another incompatibility between the 3-D releases and the 2-D releases. To see the difference let's continue with the main macro:

```
     A              B                C              D              E
11 EDITFORM       {LET rel521,@INFO("release")}~{IF @LEFT(rel521,1)<>"@"}
                  {BRANCH editfrm3521}
12 !              {WINDOWSOFF}{PANELOFF}/RNChere521~~/RNDhere521~
                   /RNChere521~~
13 !              {CONTENTS here521,here521,240,117}{LET here521,@TRIM
                  (here521)}~{WINDOWSON}{PANELON}{EDIT}{HOME}{DEL}{END}{?}~
14 !              /RNDhere521~
```

If you are using a 2-D lotus release, the macro continues with `{WINDOWSOFF}{PANELOFF}` and then uses the "safe technique" to assign the [here521] range name to the current cell. Now it issues `{CONTENTS here521,here521,240,117}` and `{LET here521,@TRIM (here521)}~` as in the case of a 3-D release; but this time `{CONTENTS}` and the `@TRIM` function operate on the current cell, itself. The interesting result is the formula in the cell automatically transformed to the range names without any need to use `{ABS}` as we have did in the 3-D Lotus release. This is another example of incompatibility between 2-D and 3-D Lotus releases. Last the macro issues `{EDIT}{HOME}{DEL}{END}{?}~` to change the text in the panel back into a formula and allow you to edit it. When you finish, the macro issues `/RNDhere521~` to delete the temporary [here521] range name and leave a clean worksheet.

[5] Create Range Names Table

```
     A              B              C              D              E
 1 *---A macro to create a TABLE of all range names, the table is created
 2     outside the worksheet area using the {END}{HOME} key combination.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 RANGETBL       {WINDOWSOFF}{PANELOFF}{MENUBRANCH menu154}
12 !
13 menu154        Worksheet  Printer  Both     Quit
14 !              Create the Print thePrint andQuit the macro
15 !              .          .        .
16 !                         .        .
17 !
```

One of the most important things to do when you develop a macro based application or an application with many range names, is to create a range name table in a remote area of the worksheet, which contains the full list of the range names in the worksheet and their addresses. This table can save many hours of work when, by accident, you reset all the range name instead of create a new name. This macro documents all the range names in a range outside the working area using the `{END}{HOME}{RIGHT}{END}{UP}` macro commands which put the table in the first cell of the last occupied column. Then the macro uses the `/RNT` macro keys.

The macro uses the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and panel display activities and then `{MENUBRANCH menu154}` to start the [menu154] custom menu. Because this macro uses a custom menu, the full code of the menus cannot be displayed on one page. Therefore, we show the menu code for every menu option separately. If you intend to key this macro into 1-2-3, you need to key the code as it appears here, NOT the partial code shown in the main macro text. The first menu option is [**W**orksheet], which occupies the B13..B15 range.

```
Worksheet
Create the names table in the worksheet
/RNT{BS}{END}{HOME}{RIGHT}{END}{UP}~{END}{HOME}{RIGHT}{END}{UP}
    {LEFT 2}
```

The macro begins with the `/RNT` macro keys and then issues `{BS}` to free the cell pointer and `{END}{HOME}{RIGHT}{END}{UP}~` to enter the range names table. Now the macro moves the cell pointer to the table using `{END}{HOME}{RIGHT}{END}{UP}{LEFT 2}`.

The second menu option is [**P**rinter], which occupies the C13..C16 range.

```
Printer
Print the range names table
/RNT{BS}{END}{HOME}{RIGHT}{END}{UP}~{END}{HOME}{RIGHT}{END}{UP}
{LEFT 2}/RNCtemp~~/RNDtemp~/RNCtemp~.{END}{DOWN}{RIGHT}~
     /PPCRARtemp~gq
/REtemp~/RNDtemp~
```

This menu option prints the table to the printer. Again the macro uses the `/RNT` macro keys and

then uses {BS} to free the cell pointer. Next the macro uses {END}{HOME}{RIGHT} {END}{UP}~ to insert the table in a range outside the working area, and moves the cell pointer to the table using {END}{HOME}{RIGHT}{END}{UP}{LEFT 2}. Next the macro issues

```
/RNCtemp~~/RNDtemp~/RNCtemp~.{END}{DOWN}{RIGHT}~
```

using the "safe technique" to assign the [temp] range name to the table of range names. The macro continues with /PPCRARtemp~gq to print the [temp] range and last erases the table using /REtemp~ and deletes the [temp] name using /RNDtemp~ to leave the worksheet the way it was before the macro started.

The third menu option is [**B**oth], which occupies the D13..D17 range.

```
Both
Print and create
/RNT{BS}{END}{HOME}{RIGHT}{END}{UP}~{END}{HOME}{RIGHT}{END}{UP}
{LEFT 2}/RNCtemp~~/RNDtemp~/RNCtemp~.{END}{DOWN}{RIGHT}~
     /PPCRARtemp~gq
/RNDtemp~
```

This menu option is exactly the same as the [**P**rinter] menu option except that here the macro does not erase the [temp] range which remains in the worksheet.

The fourth menu option is [**Q**uit], which occupies the E13..E14 range.

```
Quit
Quit the macro
```

When you select this option, the macro reaches an empty cell and quits.

To assist you to key this macro into 1-2-3, here is the cell contents list of the macro.

```
A1: U '*---A macro to create a TABLE of all range names, the table is created
A2: U '    outside the worksheet area using the {End}{Home} key combination.
A3: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A4: '    range names in this column (starts with the \Z macro name)
A5: '*---Hold the [ALT] key and press [Z] to activate the macro
A6: '!
A7: '!
A8: '!
A9: '!
A10: U '\Z
B10: '{BREAKON}
A11: U 'RANGETBL
B11: '{WINDOWSOFF}{PANELOFF}{MENUBRANCH menu154}
A12: '!
A13: 'menu154
B13: 'Worksheet
C13: 'Printer
D13: 'Both
E13: 'Quit
A14: '!
B14: 'Create the names table in the worksheet
C14: 'Print the range names table
D14: 'Print and create
E14: 'Quit the macro
A15: '!
B15: '/RNT{BS}{END}{HOME}{RIGHT}{END}{UP}~{END}{HOME}{RIGHT}{END}{UP}{LEFT 2}
C15: '/RNT{BS}{END}{HOME}{RIGHT}{END}{UP}~{END}{HOME}{RIGHT}{END}{UP}{LEFT 2}
```

```
     /RNCtemp~~/RNDtemp~/RNCtemp~.{END}{DOWN}{RIGHT}~/PPCRARtemp~GQ
D15: '/RNT{BS}{END}{HOME}{RIGHT}{END}{UP}~{END}{HOME}{RIGHT}{END}{UP}{LEFT 2}
     /RNCtemp~~/RNDtemp~/RNCtemp~.{END}{DOWN}{RIGHT}~/PPCRARtemp~GQ
A16: '!
C16: '/REtemp~/RNDtemp~
D16: '/RNDtemp~
```

[4] Restore Range Names

```
     A                B                C                D                E
 1 *---A macro to RESTORE all range names after issuing /RNR accidentally.
 2     The worksheet must contain a range names table previously created
 3     using /RNT. It is a good practice to set aside a place for range
 4     names table and using the /RNT to update the table any time new
 5     range names are created.
 6 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 7     range names in this column (starts with the \Z macro name)
 8 *---Place the cell pointer on the upper left cell of the names table.
 9 *---Hold the [ALT] key and press [Z] to activate the macro
10 \Z              {BREAKON}
11 RESTORE         {LET name506,@CELLPOINTER("contents")}~{RIGHT}
                   {LET range506,@CELLPOINTER("contents")}~
                   {RECALC create506}
12 create506       /RNCBUDGET~B100~
13 !               {LEFT}{DOWN}{BRANCH RESTORE}
14 !
15 name506         BUDGET
16 !
17 range506        B100
```

The code in the B12 cell named [create506] is the result of a formula, therefore if you intend to key this macro into 1-2-3, you must key the following formula in the B12 cell, NOT the code as it appears in the main macro text.

```
12 create506       +"/RNC"B15&"~"&B17&"~"
```

If your worksheet contains an updated range table and your worksheet lost part or all of its range names, this macro will re-assign all the range names, using the range names table. A range names table is built of two columns, the first column includes the range names and the second column includes the addresses. Place the cell pointer on the first cell of the first column and activate the macro.

The macro starts with the `{LET name506,@CELLPOINTER("contents")}~` macro command to store the range name in the current cell in the cell named [name506]. Then the macro issues `{RIGHT}` to moves the cell pointer to the addresses column and issues `{LET range506,@CELLPOINTER("contents")}~` to store the range address in the current column in cell [range506]. The code in the B12 cell is the result of the

```
+"/RNC"B15&"~"&B17&"~"
```

formula which uses the content of the B15 cell named [range506] and B17 cell named [name506] to create the code for the macro. Before the macro reaches the code in the B12 cell it issues `{RECALC create506}` to update the formula in [create506]. If the name is BUDGET and the address is B100, the code in cell [create506] is `/RNCBUDGET~B100~` which re-creates the [BUDGET] range name in B100. Next the macro moves the cell pointer to the next row in the table using `{LEFT}{DOWN}` and branches back to the beginning to process the next range name using `{BRANCH RESTORE}`.

[6] Delete a Group of Range Names

```
     A                B                C                D                E
 1 *---A macro to DELETE a LISTED GROUP of range names, the macro checks
 2     the validity of the range names too
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                    IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z             {BREAKON}
11 RANGDELG       {WINDOWSOFF}{PANELOFF}/RNCNames range ?~~/RND
                  Names range ?~/RNC{WINDOWSON}{PANELON}Names range ?~
                  {BS}{BS}{?}~{GOTO}Names range ?~{LET counterb153,0}~
12 cont153        {LET hereabs153,@CELLPOINTER("address")}~
                  {LET counter1153,0}
13 !              {FOR counter1153,0,@COLS(Names range ?)-1,1,labels1153}
14 !              {LET rel153,@INFO("release")}~{IF @LEFT(rel153,1)<>"@"}
                  {GOTO}{hereabs153}~{LET counterb153,counterb153+1}~
                  {IF counterb153<@SHEETS(Names range ?)}{NS}{GOTO}
                  {hereabs153}~{BRANCH cont153}
15 !              {GOTO}Names range ?~/RNDNames range ?~
16 !
17 counter1153           1
18 counter1a153          0
19 labels1153     {RIGHT}{LET here153,@CELLPOINTER("address")}~{LEFT}
                  {FOR counter1a153,0,@ROWS(Names RANGE ?)-1,1,labels1a153}~
                  {IF counter1153<@COLS(Names range ?)-1}{GOTO}{here153}~
                  {LET counter1a153,0}~
20 !
21 here153        $F$1
22 !
23 labels1a153    {cancel1153}
24 !              {DOWN}
25 ret153
26 !
27 cancel1153     IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER("type")="v"
                  #OR#@LENGTH(@CELLPOINTER("contents"))>15}{BRANCH ret1153}
28 !              /RNLR~{IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER
                  ("type")="v"}{BRANCH ret1153}
29 !              {LET nam153,@CELLPOINTER("contents")}~
30 !              /RND{nam153}~{ESC 6}
31 ret1153        {RETURN}
32 !
33 !
34 !
35 nam153         uuuu
36 !
37 counterb153           2
38 !
39 hereabs153     $A$1
40 !
41 rel153
```

This macro can delete a list of range names in a worksheet. If you keep a record of the range names in the macro or create a range names table, or build macro the way we build the in ***Super Power*** where the range names description is in the first column of the macro, all you need to do is paint the range where these description names are when the macro asks for range of names. The macro will read every cell content and check if a particular range name exists, if so, the macro deletes the range name.

```
     A                B                C                D                E
11 RANGDELG       {WINDOWSOFF}{PANELOFF}/RNCNames range ?~~/RND
                  Names range ?~/RNC{WINDOWSON}{PANELON}Names range ?~
```

```
{BS}{BS}{?}~{GOTO}Names range ?~{LET counterb153,0}~
```

The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands to freeze unwanted screen and panel activities and uses the "safe technique" to assign the [Names range ?] name to the range containing the names of the range names, while simultaneously using [Names range ?] as a prompt. Last, the macro sets the content of the B37 cell named [counterb153], which serves as a counter, to zero "0" using {LET counterb153,0}~.

	A	B	C	D	E
12	cont153	{LET hereabs153,@CELLPOINTER("address")}~			
		{LET counter1153,0}			

Now the macro uses {LET hereabs153,@CELLPOINTER("address")}~ to store the current cell pointer position in cell [hereabs153] and {LET counter1153,0}~ to set the content of [counter1153], which serves as a second counter, to zero "0" .

	A	B	C	D	E
13	!	{FOR counter1153,0,@COLS(Names range ?)-1,1,labels1153}			

The following {FOR} loop macro command executes the [labels1153] routine as many times as the number of columns in the range named [Names range ?]. Before continuing let's look in the [labels1153] routine:

	A	B	C	D	E
19	labels1153	{RIGHT}{LET here153,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a153,0,@ROWS(Names RANGE ?)-1,1,labels1a153}~			
		{IF counter1153<@COLS(Names range ?)-1}{GOTO}{here153}~			
		{LET counter1a153,0}~			

The [labels1153] routine controls the number of column to process. It moves the cell pointer to the first cell of the next column using {RIGHT} and then issues {LET here153, @CELLPOINTER("address")}~ to record the cell pointer address in cell [here153]. Next the macro moves the cell pointer back using {LEFT}. Now the routine issues a second {FOR} loop command:

```
{FOR counter1a153,0,@ROWS(Names RANGE ?)-1,1,labels1a153}~
```

which executes the [labels1a153] routine as many times as the number of rows in the range named [Names range ?]. Before continuing let's look in the [labels1a153] routine:

	A	B	C	D	E
23	labels1a153	{cancel1153}			
24	!	{DOWN}			

which activates another routine, [cancel1153], that actually deletes the range name whose name string appears in the current cell position.

	A	B	C	D	E
27	cancel1153	IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER("type")="v"			
		#OR#@LENGTH(@CELLPOINTER("contents"))>15}{BRANCH ret1153}			
28	!	/RNLR~{IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER			
		("type")="v"}{BRANCH ret1153}			
29	!	{LET nam153,@CELLPOINTER("contents")}~			
30	!	/RND{nam153}~{ESC 6}			
31	ret1153	{RETURN}			

To make sure that the current cell contains a valid label string that can be a range name, the [cancel1153] routine issues:

```
{IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER("type")="v"#OR#
    @LENGTH(@CELLPOINTER("contents"))>15}
```

using the `@CELLPOINTER("type")="b"` macro code to check that the cell is not blank, `@CELLPOINTER("type")="v"` to check that the cell doesn't contain a value or a formula, and `@LENGTH(@CELLPOINTER("contents"))>15` ensure that the string in the cell is not longer than 15 characters, the maximum length for a range name. If one of the conditions is TRUE, the macro ends the routine using `{BRANCH ret1153}` that activates `{RETURN}` in the routine named [ret1153].

	A	B	C	D	E
28	!	/RNLR~{IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER ("type")="v"}{BRANCH ret1153}			

If none of the previous conditions is met, the cell contains a label shorter or equal to 15 characters long, therefore the macro creates a range name using the label in the cell as the name by issuing the `/RNLR` macro keys. Here the macro tricks Lotus by creating the range name to make sure that the range name exists before it deletes it. If the range name already exists no harm is done, but if the there is no range name for the name in the cell, an error will occur and stop the macro. Therefore the macro issues the `/RNLR` macro keys to make sure that a range name exists before deleting it. Again the macro issues:

```
{IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER("type")="v"}
    {BRANCH ret1153}
```

which seems to be unnecessary because we already have used it. However, our experience has shown that because of some incompatibility between the many versions of Lotus 1-2-3, this repetition may be needed.

	A	B	C	D	E
29	!	{LET nam153,@CELLPOINTER("contents")}~			
30	!	/RND{nam153}~{ESC 6}			
31	ret1153	{RETURN}			

Now the macro copies the current cell content to cell [nam153] using:

```
{LET nam153,@CELLPOINTER("contents")}~
```

and the `/RND{nam153}~` indirect macro command to delete the range whose name is stored in cell [nam153]. The macro starts the `/RND` macro keys and then activates the [nam153] routine using the `{nam153}` routine command. Because the [nam153] routine contains only the name of the range, the macro types the name into the panel to complete the `/RND` process and issues the tilde "~" macro key, which acts like the ENTER key. The `{ESC 6}` makes sure that the spreadsheet returns to ready mode and `{RETURN}` ends the routine and returns the control to the `{FOR}` loop command.

Now we can continue with the [labels1153] routine:

	A	B	C	D	E
19	labels1153	{RIGHT}{LET here153,@CELLPOINTER("address")}~{LEFT} {FOR counter1a153,0,@ROWS(Names RANGE ?)-1,1,labels1a153}~			

```
{IF counter1153<@COLS(Names range ?)-1}{GOTO}{here153}~
{LET counter1a153,0}~
```

Next the macro issues `{IF counter1153<@COLS(Names range ?)-1}` to check the value in [counter1153] which serves as the counter for the number of columns that have been processed. If the value is less than the number of columns in the [Names range ?] range minus 1, the macro moves the cell pointer to the next column using the `{GOTO}{here153}~` indirect command. The cell/routine [here153] contains the address of the first cell in the next column stored at the beginning of this routine. Last the routine resets the value in [counter1a153], the counter for the number of rows already processed, to zero.

	A	B	C	D	E
14	!	{LET rel153,@INFO("release")}~{IF @LEFT(rel153,1)<>"@"} {GOTO}{hereabs153}~{LET counterb153,counterb153+1}~ {IF counterb153<@SHEETS(Names range ?)}{NS}{GOTO} {hereabs153}~{BRANCH cont153}			

Now the macro checks if you are using a 3-D or a 2-D Lotus 1-2-3 release . The macro stores the result of the `@INFO("release")` function in cell [rel153] using the `{LET rel153, @INFO("release")}~` and compares the first character of the cell content to the "@" character using `{IF @LEFT(REL153,1)<>"@"}`. If you are using a 3-D release, the condition is TRUE and the macro issues the `{GOTO}{hereabs153}~` indirect command to move the cell pointer to the address stored in cell [hereabs153]. The `{LET counterb153 ,counterb153+1}~` commands increase the content of [counterb153] by one.

Cell [counterb153] serves as the counter for the number of sheets in the [Names range ?] range. The macro uses `{IF counterb153<@SHEETS(Names range ?)}` to check if all the sheets in [Names range ?] have been processed and `@SHEETS(Names range ?)` function to find the number of sheets in [Names range ?]. If the value in [counterb153] is less than the number of sheets, the macro moves the cell pointer to the next sheet using `{NS}` and then moves the cell pointer to the address stored in [hereabs153] using `{GOTO}{hereabs153}~`. Then it branches back to process the new sheet in the [Names range ?] range using `{BRANCH cont153}`.

	A	B	C	D	E
15	!	{GOTO}Names range ?~/RNDNames range ?~			

Before the macro ends, it moves the cell pointer to the top left cell of the [Names range ?] range using `{GOTO}Names range ?~` and deletes the [Names range ?] range name using the `/RNDNames range ?~` macro code.

Important: This macro can delete range names only if the current cell contains a string identical to an existing range name. To delete hidden range names even if the range is empty see note **To Find If a Range Name Exists**.

[6] Delete All the Range Names in a Range

```
     A                B                C                D                E
 1 *---A macro to DELETE all range names in a specified 3-D or 2-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                      IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 UNNAME          {WINDOWSOFF}{PANELOFF}/RNCNames range ?~~/RND
                   Names range ?~/RNC{WINDOWSON}{PANELON}Names range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}
                   {LET counterb166,0}~{GOTO}Names range ?~
12 cont166         {LET hereabs166,@CELLPOINTER("address")}~
                   {LET counter1166,0}
13 !               {FOR counter1166,0,@COLS(Names range ?)-1,1,labels1166}
14 !               {LET rel166,@INFO("release")}~{IF @LEFT(rel166,1)<>"@"}
                   {GOTO}{hereabs166}~{LET counterb166,counterb166+1}~
                   {IF counterb166<@SHEETS(Names range ?)}{NS}{GOTO}
                   {hereabs166}~{BRANCH cont166}
15 !               {GOTO}Names range ?~/RNDNames range ?~
16 !
17 counter1166           1
18 counter1a166          0
19 labels1166      {RIGHT}{LET here166,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a166,0,@ROWS(names RANGE ?)-1,1,labels1a166}~
                   {IF counter1166<@COLS(Names range ?)-1}{GOTO}{here166}~
                   {LET counter1a166,0}~
20 !
21 here166         $F$1
22 !
23 labels1a166     {unname1166}
24 !               {DOWN}
25 !
26 unname1166      {WINDOWSOFF}{LET here1166,@CELLPOINTER("address")}~
27 lop166          {LET rangname166,@CELLPOINTER("contents")}~
28 !               {LET numcheck166,@@(rangname166)}~
29 !               {IF #NOT#@ISERR(numcheck166)}{BRANCH delete166}
30 !
31 delete166       /RND
32 rangname166     ppp
33 !               ~{BRANCH lop166}
34 !
35 here1166        $A$4
36 !
37 numcheck166         ERR
38 !
39 counterb166           2
40 hereabs166      $A$1
41 !
42 rel166
```

It is common knowledge that using range names in a worksheet and in macros has its benefits, but there is also a drawback. As the worksheet or application grows and gets more complicated, the number of range names increases and tends to slow the worksheet. When the application development is finished, there are always a large number of unused range names we would like to delete. It can be a tedious process to delete them manually. This macro removes and automatically delete all the range name in a specified range.

The macro first issues the `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities, and uses the "safe technique" to assign the [Names range ?] name to the range that

contains the range names to be deleted. Simultaneously the macro uses [Names range ?] as a prompt and then uses {LET counterb166,0}~ to set the value in the B39 cell named [counterb166] to zero, which is used as a counter.

	A	B	C	D	E
12	cont166	{LET hereabs166,@CELLPOINTER("address")}~			
		{LET counter1166,0}			
13	!	{FOR counter1166,0,@COLS(Names range ?)-1,1,labels1166}			
14	!	{LET rel166,@INFO("release")}~{IF @LEFT(rel166,1)<>"@"}			
		{GOTO}{hereabs166}~{LET counterb166,counterb166+1}~			
		{IF counterb166<@SHEETS(Names range ?)}{NS}{GOTO}			
		{hereabs166}~{BRANCH cont166}			
15	!	{GOTO}Names range ?~/RNDNames range ?~			

To be able to return to the point of origin when the macro is finished, the macro issues {LET hereabs166,@CELLPOINTER("address")}~ storing the current cell pointer address in cell [hereabs166], and then {LET counter1166,0} to set the content of [counter1166] to zero, which also serves as a counter.

The macro issues {FOR counter1166,0,@COLS(Names range ?)-1,1,labels1166} loop command activating the [labels1166] routine as many times as the number of columns in the [Names range ?] range. Before we continue with this code, let's look at the [labels1166] routine.

	A	B	C	D	E
19	labels1166	{RIGHT}{LET here166,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a166,0,@ROWS(names RANGE ?)-1,1,labels1a166}~			
		{IF counter1166<@COLS(Names range ?)-1}{GOTO}{here166}~			
		{LET counter1a166,0}~			

The routine issues {RIGHT}{LET here166,@CELLPOINTER("address")}~{LEFT} which record the address of the first cell of the next column in cell [here166]. When the macro finishes processing the current column, it will use the content of [here166] to move the cell pointer directly to the top of the next column. The {FOR counter1a166,0,@ROWS (Names range ?)-1,1,labels1a166} command activates the [labels1a166] routine as many times as the number of rows in the [Names range ?] range.

	A	B	C	D	E
23	labels1a166	{unname1166}			
24	!	{DOWN}			
25	!				
26	unname1166	{WINDOWSOFF}{LET here1166,@CELLPOINTER("address")}~			
27	lop166	{LET rangname166,@CELLPOINTER("contents")}~			
28	!	{LET numcheck166,@@(rangname166)}~			
29	!	{IF #NOT#@ISERR(numcheck166)}{BRANCH delete166}			
30	!				
31	delete166	/RND			
32	rangname166	budget			
33	!	~{BRANCH lop166}			

This routine issues {unname1166} to start [unname1166]. The [unname1166] routine issues {WINDOWSOFF} to freeze the screen display activity and uses {LET here1166, @CELLPOINTER("address")}~ to store the current cell address in cell [here1166]. It continues with {LET rangname166,@CELLPOINTER("contents")}~ to store the cell content in [rangname166].

The current cell may contain a label, a value or even can be empty. Therefore, the macro issues

{LET numcheck166,@@(rangname166)}~ that store the content of the cell in which its address is stored in cell [rangname166]. Recall that [rangname166] holds the cell pointer content, while the @@(rangname166) function expects to find an address or a range name. If [rangname166] contains a label that is not a range name or a value or anything else but a range name, the @@() function will return an error and cell [numcheck166] will contain the ERR value. The macro issues {IF #NOT#@ISERR(numcheck166)} to check if [numcheck166] contains the ERR value or not. If so, and [numcheck166] contains a label, it means that a range name with the same name exists, therefore the macro uses {BRANCH delete166} to initiate the [delete166] routine that deletes the range name. This is a special example of using the @@() to determine if a range name exist or not.

The [delete166] routine uses /RNDbudget~ to delete the [budget] range name. Notice that the [budget] range name was entered into the [delete166] routine when the macro issued {LET rangname166,@CELLPOINTER("contents")}~. This is another example of using dynamic code in a macro when the code is changed before Lotus processes it. Let's continue with the [labels1166] routine.

	A	B	C	D	E
19	labels1166	{RIGHT}{LET here166,@CELLPOINTER("address")}~{LEFT}			
		{FOR counter1a166,0,@ROWS(names RANGE ?)-1,1,labels1a166}~			
		{IF counter1166<@COLS(Names range ?)-1}{GOTO}{here166}~			
		{LET counter1a166,0}~			

When this {FOR} loop is finished with the first column, the macro uses {IF counter1166<@COLS(Names range ?)-1} to check if there are more columns to process, if so, the macro issues the {GOTO}{here166}~ indirect macro command to move to the first cell of the next column. The [here166] cell holds the address of the first cell in the next column. The last macro commands in this routine is {LET counter1a166,0}~ which reset the counter in [counter1a166] to zero. Now we can go back to the [cont166] routine.

	A	B	C	D	E
12	cont166	{LET hereabs166,@CELLPOINTER("address")}~			
		{LET counter1166,0}			
13	!	{FOR counter1166,0,@COLS(Names range ?)-1,1,labels1166}			
14	!	{LET rel166,@INFO("release")}~{IF @LEFT(rel166,1)<>"@"}			
		{GOTO}{hereabs166}~{LET counterb166,counterb166+1}~			
		{IF counterb166<@SHEETS(Names range ?)}{NS}{GOTO}			
		{hereabs166}~{BRANCH cont166}			
15	!	{GOTO}Names range ?~/RNDNames range ?~			

The macro continues with {LET rel166,@INFO("release")}~ that store the result of the @INFO("release") function in cell [rel166]. Then the macro issues {IF @LEFT(rel166,1)<>"@"} to check if you are using a 2-D or a 3-D Lotus release. If "@" is the first character of the content of [rel166], you are using a 2-D Lotus release, otherwise you are using a 3-D release. If you are using a 3-D release, the macro uses the {GOTO} {hereabs166}~ indirect command to move to the first cell in the current sheet range, and then {LET counterb166,counterb166 +1}~ to increase the counter in [counterb166] by one.

Next the macro uses {IF counterb166<@SHEETS(Names range ?)} to compare the counter value in [counterb166] to the number of sheets in the [Names range ?] range. If the counter is still less than the number of sheets in [Names range ?], there are more sheets to process. Therefore the macro issues {NS}{GOTO}{hereabs166}~ to move the cell pointer to the top left cell to process in the new sheet and {BRANCH cont166} to loop back to the beginning of the [cont166] routine.

When the counter in [counterb166] is equal to the number of sheets in the [Names range ?] range, the macro issues `{GOTO}Names range ?~` to place the cell pointer back on the point of origin just before starting, and then uses `/RNDNames range ?~` to delete the [Names range ?] range name to leave a clean worksheet.

Important: This macro can delete range names only if the current cell contains a string identical to an existing range name. To delete hidden range names even if the range is empty see **To Find If a Range Name Exists**.

[3] Delete Range Names Easily

```
     A          B          C          D          E
 1 *---A macro to DELETE range names
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Use Ctrl-Break to quit
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 NAMESDEL        {GETLABEL "One at a time or All together [O/A] ? [O] "
                   ,key1128}~
12 !               {WINDOWSOFF}{PANELOFF}
13 !               {IF @UPPER(key1128)="A"}/RNR{QUIT}
14 !               {BRANCH one1128}
15 !
16 one1128         /RN{PANELON}{WINDOWSON}D{NAME}{?}
17 !               {WINDOWSOFF}{PANELOFF}~{BRANCH one1128}
18 !
19 key1128
```

The literature recommended using range names instead of cell addresses, which is true because a range name always stays with the data, even if the address has been changed or moved. However, the more complicated the worksheet becomes, the more range names the worksheet contains, mostly due to the many changes made to the worksheet, so there are many range names that are no longer needed. Deleting them is recommended to save memory and make the worksheet faster. To save many keys and to make the process of deleting range names easier, this macro displays a full screen list of all the range names in the worksheet. Point to the range name and press ENTER to delete the range name. In return, the macro displays the full screen list again and the process continues until you press the Ctrl-Break keys.

The macro starts with

```
{GETLABEL "One at a time or All together [O/A] ? [O]  ",key1128}~
```

which display the "`One at a time or All together [O/A] ? [O]`" prompt message in the panel. When you press ENTER, Lotus stores your response in the B19 cell named [key1128]. Next the macro issues `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities while the macro issues `{IF @UPPER(key1128)="A"}` to check your response. If so, the macro issues `/RNR{QUIT}` which resets all the range names in the worksheet and quits. If the condition is false, the macro issues `{BRANCH one1128}` to route the macro control to the [one1128] routine.

```
     A          B          C          D          E
16 one1128         /RN{PANELON}{WINDOWSON}D{NAME}{?}
17 !               {WINDOWSOFF}{PANELOFF}~{BRANCH one1128}
```

The [one1128] routine starts with the `/RN` keys and then issues `{WINDOWSON}{PANELON}`, and reactivates the screen and panel display activities, just before the "`D`" key is issued to complete the `/RND` keys sequence, needed to delete a range name. Next the macro issues `{NAME}`, which displays a full screen list of all the range names in the worksheet. Then the macro issues `{?}` which halts the macro execution and waits until you point to the range name and press ENTER. When you press ENTER Lotus deletes the range name.

Next the macro issues `{WINDOWSOFF}{PANELOFF}` again to freeze the screen and panel display activities, and then issues `{BRANCH one1128}` to branch to the beginning of the [one1128] routine to start the process again. To end the macro you can press the Ctrl-Break keys.

[3] The Range Name Delete Macro

```
     A                B                C                D                E
 1 *---A macro to DELETE range names one by one
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Highlight the range name and press the [ENTER] key
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 RANGEDEL        {BREAKON}
12 loop030         /RND{NAME}{?}~
13 !               Delete more range names ? [Y/N] Y {GET key030}{ESC}~
14 !               {IF @UPPER(key030)="N"}{BRANCH ret030}
15 !               {BRANCH loop030}
16 !
17 key030          n
18 !
19 ret030
```

This macro displays a full screen list of all the range names in the worksheet, and using point and shoot, you delete the range names one by one. The macro starts with the `/RND` (/ **R**ange **N**ame **D**elete) macro keys and issues `{NAME}`, which displays a full screen view of all the range names in the worksheet instead of a row list. Then the macro issues `{?}` pausing the macro until you point to the name to delete and press ENTER. Then the macro issues the tilde "`~`" which is the same as ENTER and deletes the range name. Next the macro types "`Delete more range names ? (Y/N) Y `" into the panel and immediately follows it with `{GET key030}`, which halts macro execution and waits until you press a key. Lotus stores the key's code in cell [key030].

For example, if you press the RIGHT key, Lotus stores the `{RIGHT}` string in [key030], and if you press "`Y`", Lotus stores "`Y`" in [key030]. Next the macro issues `{ESC}` to clear the prompt message from the panel before Lotus writes it into the current cell. Now it issues `{IF @UPPER(key030)="N"}`, to check if you pressed "`N`" or "`n`". The `@UPPER` function assures that the macro is case insensitive. If so, the macro issues `{BRANCH ret030}`, which routes macro control to the empty [ret030] routine and quits. However, if the condition is false, the macro uses `{BRANCH loop030}` which routes macro control to the start allowing you to delete the next range name.

Print Macros

[6] Print Ranges

```
     A              B              C              D              E
 1 *---A macro to print the FIRST GROUP OF SPECIFIED number of RANGES,
 2     you can print all ranges in the worksheet in ascending order.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z            {BREAKON}
11 PRTRANGS      {WINDOWSOFF}{PANELOFF}{LET counter027,1}~{PANELON}/RNC!~~
                 /RND!~{GETNUMBER "How many ranges to print ? ",number027}
                 {PANELOFF}
12 !             {BRANCH routine027}
13 !
14 routine027    {IF counter027>number027}{BRANCH ret027}
15 !             {WINDOWSOFF}{PANELOFF}{err1027}{RECALC move1027}/PPR
                 {WINDOWSON}{PANELON}{NAME}
16 move1027      {RIGHT 2}~
17 !             {WINDOWSOFF}{PANELOFF}AGLLLQ{LET counter027,counter027+1}~
                 {BRANCH routine027}
18 !
19 number027           2
20 counter027          3
21 !
22 err1027       {ONERROR err2027}
23 !
24 err2027       {LET counter027,counter027+1}~{BRANCH routine027}
25 !
26 ret027        {BRANCH \y}
```

Before we start, notice that the code in the B16 cell [move1027] is the result of the following dynamic string formula:

```
16 move1027      +"{RIGHT "&@STRING(B20-1,0)&"}~"
```

If you plan to key this macro into Lotus 1-2-3, remember to key the formula as it appears here into the B16 cell, NOT the `{RIGHT 2}` as it appears in the main listing. This macro allows you to print many ranges in one session. For example, if you have four ranges in your worksheet that you want to print one after another, you need to assign a range name for every one of them. When the macro starts, it prompts you to enter the number of ranges to print. Then the macro prints the ranges in the same order that they are listed when you press the F5 (GOTO) key from the keyboard. If for example, you assign four range names like 111, 222, 333, and 444 to the four ranges that you want to print, then when you press the F5 key followed by the F3 key the following list of range names appears in the screen.

```
!              111            222            333            444
COUNTER027     ERR1027        ERR2027        MOVE1027       NUMBER027
PRTRANGS       RET027         ROUTINE027     \Z
```

The exclamation mark [!] range name is not completely a valid range name, which usually does not affect the macros; however, because it is the first range name, the macro will try to print it first causing an error. Therefore the macro issues `/RNC!~~/RND!~` to delete it when it starts (the [!] range name was created when you issued the **`/ Range Name Labels Right [END] [DOWN]~`** to assign the range names for the macro itself). Notice that the macro first re-creates the [!] range name to make sure that it exists before it deletes it, which is the safest way to delete range names without getting into errors. We have chosen the 111,..

range names because they will appear first in the list. The range to be printed must always have range names which are listed first. Normally, Lotus sorts the range names in such a way that numbers are sorted before characters. When the macro starts it first prints the 111 range, next the 222 range, then the 333 range and last the 444 range. The macro uses the cell pointer movement to point to the range names, which is why they must appear first in the range names list.

```
          A              B                C              D              E
11 PRTRANGS       {WINDOWSOFF}{PANELOFF}{LET counter027,1}~{PANELON}/RNC!~~
                  /RND!~{GETNUMBER "How many ranges to print ? ",number027}
                  {PANELOFF}
12 !              {BRANCH routine027}
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the screen and panel display activities and then the macro issues `{LET counter027,1}~` to set the value of cell [counter027] to "`1`". Next the macro issues `{PANELON}` which resumes the panel display activity and then issues `{GETNUMBER "How many ranges to print ? ",number027}` , which displays "`How many ranges to print ? `" in the panel and waits for you to insert the number and press ENTER. When you type a number and press ENTER, Lotus stores the number in [counter027]. Next the macro issues `{PANELOFF}` to freeze the panel display activity and then `{BRANCH routine027}`, which routes the macro control to the [routine027] routine.

```
          A              B                C              D              E
14 routine027     {IF counter027>number027}{BRANCH ret027}
15 !              {WINDOWSOFF}{PANELOFF}{err1027}{RECALC move1027}/PPR
                  {WINDOWSON}{PANELON}{NAME}
16 move1027       {RIGHT 2}~
17 !              {WINDOWSOFF}{PANELOFF}AGLLLQ{LET counter027,counter027+1}~
                  {BRANCH routine027}
```

The routine starts with `{IF counter027>number027}`, checking if the value in cell [counter027] is greater than the value in cell [number027], which holds the number of ranges to print. If so, the macro finished printing all the ranges; therefore the macro issues `{BRANCH ret027}`, which routes macro control to the [ret027] empty routine and quits. If the condition is false, the macro still have more ranges to print, therefore the macro issues `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities and then issues `{err1027}`, which initializes the [err1027] error handling routine which we will look into later. The macro continues with `{RECALC move1027}` to update the dynamic string formula in cell [move1027]. The B16 cell [move1027] holds the following formula:

```
16 move1027       +"{RIGHT "&@STRING(B20-1,0)&"}~"
```

You can see that the `@STRING(B20-1,0)` function returns the number that the macro uses to decide how many times to use the RIGHT arrow key to point to the next range name. The B20 cell [counter027] holds the count of the printed ranges. When the formula is updated, the macro issues the `/PPR` macro keys which start the printing process, and then the macro issues `{WINDOWSON}{PANELON}` to resume the screen and panel display activity allowing you to see the name of the printed range. Then the macro issues `{NAME}`, which displays a full screen list of all the range names in the worksheet, and issues `{RIGHT n}~` to print the n'th range name in the list.

Now the macro issues `{WINDOWSOFF}{PANELOFF}` to again freeze the screen and panel

activities, while the macro issues the AGLLLQ macro keys which align the paper and advance the paper three lines, and quits to the READY mode. Then the macro issues {LET counter027,counter027+1}~ to increase the value in [counter027] by one, and then issues {BRANCH routine027}, which routes macro control back to the beginning to print the next range in the list.

Last we need to look at the error handling routine:

```
          A          B          C          D          E
22 err1027       {ONERROR err2027}
23 !
24 err2027       {LET counter027,counter027+1}~{BRANCH routine027}
```

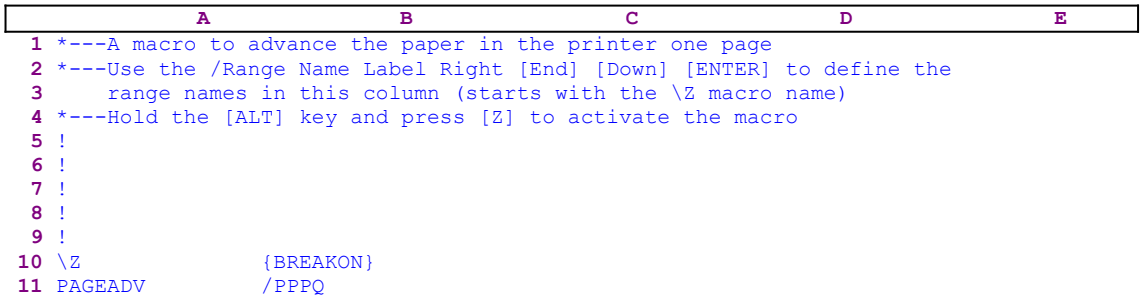
When an error occurs, the macro starts the [err2027] routine. In that case, the macro issues {LET counter027,counter027+1}~, which increases the value in [counter027] by one, and then {BRANCH routine027}, which routes the macro control back to the beginning to print the next range in the list. Therefore, in case an error occurs, the macro just skips this range and moves to the next range.

[0] Advance the Paper One Line

	A	B	C	D	E
1	*---A macro to the advance the paper in the printer one line.				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	LINEADV	/PPLQ			

This is a simple four key macro (/ **P**rint **P**rinter **L**ine **Q**uit), which allows you to fine-tune your page alignment. This macro advances the paper one line at a time and is handy when you want to move the paper seven lines, for example. You need to hold the ALT key and press the "Z" key seven times, thereby saving 21 key strokes.

[0] Advance the Paper one Page



```
     A              B              C              D              E
 1 *---A macro to advance the paper in the printer one page
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 PAGEADV         /PPPQ
```

This is a simple four key macro (/ **P**rint **P**rinter **P**age **Q**uit), which allows you to advances the paper in the printer one page at a time.

[0] Set Margin to None

	A	B	C	D	E
1	*---A macro to clear and reset print margin				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	!				
6	!				
7	THIS MACRO WORKS WITH LOTUS 2.2 AND UP				
8	!				
9	!				
10	\Z	{BREAKON}			
11	MARGINON	/PPOMNQQ			

This is a simple keyboard macro which saves you eight key strokes any time you need to set the margins to NONE. The macro executes / **P**rint **P**rinter **O**ption **M**argin **N**one **Q**uit **Q**uit.

[3] Set Margins to None (Improved)

```
     A                B                C                D                E
 1 *---A macro to set the margins to NONE. In release 2/2.01 the left,
 2     bottom and top margins are set to "0" and the right margin to
 3     "240"
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                   IT WILL WORK IN RELEASE 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 MARGNONE        {RECALC rel525}
12 rel525          /PPOMNQQ
```

This macro sets the margins to NONE. It may look simple but it is not. The code in the B12 cell named [rel525] is the result of the following dynamic string formula:

```
12 rel525          @IF(@ISERR(@CELLPOINTER("filename")),"/PPOML0~~R240~~
                   T0~~B0~~{ESC}QQ","/PPOMNQQ")
```

If you intend to key this macro into 1-2-3, you need to key this formula into the B12 cell of the macro. When Lotus 2.2 and 3.0 were introduced a new menu option was added, which allows you to set the margin to NONE. This macro option is viable with 2.2 and up, but it also works with the 2.0/2.01 older versions.

The macro starts with `{RECALC rel525}`, which updates the string formula in cell [rel525]. The formula starts with the `@ISERR(@CELLPOINTER("filename"))` function. This is the trick: the `@CELLPOINTER("filename")` function is valid only from Lotus 2.2 and up, therefore if you are using release 2.0 or 2.01, this function must return an error. If the result is an error the formula returns `/PPOML0~~R240~~T0~~B0~~{ESC}QQ` which sets the left margin to zero, the right margin to 240, the top border to zero and the bottom borders to zero, exactly the same as having no margins. If on the other hand, the result is not an error, Lotus understands that you are using Lotus 2.2 and up, therefore the result is `/PPOMNQQ`, which sets the margins to NONE.

[4] Insert a Ruler for Printing Width Measurement

```
     A                B                C                D                E
 1 *---A macro to insert a ruler for printing measurement. The ruler lines
 2     will be ignored when included in the print range. The ruler takes
 3     two rows, insert or clean two empty rows for the ruler.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Place the cell pointer where you want the ruler
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8 !
 9 !
10 \Z              {BREAKON}
11 RULER           [I]nsert two rows and the ruler or [O]verwrite with the
                   ruler [I]..{GET key530}{ESC}
12 !               {IF @UPPER(key530)="O"}{ruler530}{BRANCH ret530}
13 !               /WIR{DOWN}~{BRANCH ruler530}
14 !
15 ruler530        |0      10       20       30       40       50       60
                   70      80       90      100      110      120      130
                   140    150      160      170      180      190      200
                   210
16 !               |012345678901234567890123456789012345678901234567890123
                   456789012345678901234567890123456789012345678901234567
                   901234567890123456789012345678901234567891234567890123
                   567890123456789012345678901234567890123456789012345678
17 !               {DOWN}
18 !
19 ret530
20 !
21 key530          ~
```

Sometimes it is easier to use Lotus to print a letter or a short document. However, Lotus does not include a ruler which can show you the width of the text. This macro inserts a two row ruler into the worksheet making it easy to measure the width of the text. For example, let's look at the following worksheet with text:

```
     A                B                C                D                E
 1 *---A macro to insert a ruler for printing measurement. The ruler lines
 2     will be ignored when included in the print range. The ruler takes
 3     two rows, insert or clean two empty rows for the ruler.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Place the cell pointer where you want the ruler
```

Let's assume that the cell pointer is on the A1 cell. When you starts the macro it displays:

```
     A                B                C                D                E
 1 0         10        20        30        40        50        60        70
 2 01234567890123456789012345678901234567890123456789012345678901234567890
 3 *---A macro to insert a ruler for printing measurement. The ruler lines
 4     will be ignored when included in the print range. The ruler takes
 5     two rows, insert or clean two empty rows for the ruler.
 6 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 7     range names in this column (starts with the \Z macro name)
 8 *---Place the cell pointer where you want the ruler
```

You can see that the macro inserted two rows of numbers above the text which, the macro allows you to insert or you can overwrite the text with the two rows of numbers. The macro inserts the rows with the two "'|" characters preceding the list of the numbers, the first character is the apostrophe "'" which serves as the prefix and the second is the split bar "|" character. When Lotus sees "|" it ignores the rest of the text in the cell and does not print it. If

you intend to key this macro into Lotus 1-2-3, you have to type the two characters "'|" at the beginning of the numbers in the B15 and the B16 cells of the macro.

	A	B	C	D	E
11	RULER	[I]nsert two rows and the ruler or [O]verwrite with the ruler [I]..{GET key530}{ESC}			
12	!	{IF @UPPER(key530)="O"}{ruler530}{BRANCH ret530}			
13	!	/WIR{DOWN}~{BRANCH ruler530}			

The macro types

```
[I]nsert two rows and the ruler or [O]verwrite with the ruler [I]..
```

to the panel because Lotus does not find any valid command in this text. To hold the message in the panel so you can read it and respond, the macro issues {GET key530} halting macro execution and waiting for your response. When you press a key, the macro stores the key in cell [key530] and immediately issues {ESC} to clear the message from the panel before Lotus writes it into the current cell. Next the macro issues {IF @UPPER(KEY530)="O"} to check if you pressed the "O" or the "o" character. If so, the macro issues the {ruler530} routine command which types the two rows of numbers into the worksheet because the routine does not contain any valid macro command except the two rows of numbers preceded by the "'|" characters. Next the macro issues {BRANCH ret530} which transfers macro control to the empty routine named [ret530] and quits.

If you press the "I" or the "i" character or any other key, the macro first issue /WIR{DOWN}~ which insert two empty rows above the current row, and then issues {ruler530} which types the two rows of numbers of the [ruler530] routine into the two new empty rows.

[6] Global Setup String for Printing

```
     A              B          C          D          E      F  G    H    I
 1 *---A macro to create a new GLOBAL printing setup string, or insert
 2     a setup string into a cell, can be activated in the middle of Lotus
 3     printing menu too.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Place the cell pointer where you want the setup string to be
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                     IT WILL WORK IN LOTUS 2.0 AND UP
10 \Z              {BREAKON}
11 SETUPSTR        {LET rel162,@INFO("release")}~{RECALC loc162}
                   {RECALC form162}{MENUBRANCH menu1162}
12 !
13 menu1162        Global   Insert-s Row & st View   Up  Down  More Quit
14 !               Change thInsert thInsert roView cuMove thM  More Quit
15 !               {ESC 6}{f{ESC 6}{f{ESC 6}/w/PPOS{?{UP}{DOWN}{MENUCALL
16 string162       \027@\027{EDIT}{ho{EDIT}{ho{MENUBR{MEN{MENUB{MENUBRANC
17 !               ~{ESC 6}{GOTO}{here1162}~{MENUBRANCH menu1162}
18 !
19 sheets162       Right    Left     Next-shee PrevioQuit
20 !               Move the Move the Move to nMove toPrevious menu
21 !               {RIGHT}  {LEFT}   {IF @LEFT{IF @LEFT(rel162,1)<>"@"}{PS}
22 !               {MENUBRAN{MENUBRAN{MENUBRAN{MENUBRANCH sheets162}
23 !
24 here1162        $A:$E$67
25 !
26 rel162          3.00.00
27 !
28 loc162          coord
29 !
30 ret162
31 !
32 form162         {LET here1162,@CELLPOINTER("coord")}~
33 !
34 setuptable1162          ************************************************
35 !                       * Point to the desired setup and press [ENTER] *
36 !                       ************************************************
37 !
38 !                       Standard CondensedEnlarged Double-Spaced
39 !               Normal   \027@    \027@\027\027@\027\027@\027A\024
40 !               Double-St\027@\027\027@\027\027@\027\027@\027A\024\027\069
41 !               Emphasize\027@\027\027@\027\027@\027\027@\027A\024\027\071
42 !               Both last\027@\027\027@\027\027@\027\027@\027A\024\027\069
43 !               Italic   \027@\027\027@\027\027@\027\027@\027A\024\0274...
44 !               Underline\027@\027\027@\027\027@\027\027@\027A\024\027\...
```

This macro allows you to manipulate the printing setup string using point and shoot. When the macro starts, it displays a custom menu. If you choose to change the global setup string, the macro displays a table of setup strings from which to choose. Then you move the cell pointer and point to the desired setup string and press ENTER. The table of setup strings in this macro is built for the popular EPSON printer ESC codes. If you use a different printer, you must consult your printer manual for how to change the setup strings in the macro code.

Because the custom menus and the setup string table in this macro occupy a few columns their macro code cannot be clearly displayed on one page; therefore we show each menu option separately. To further assist you to key this macro into 1-2-3, you will find a full list of cell contents and the macro codes at the end of this section.

The custom menu is:

```
            Global
            Change the GLOBAL printing setup string (affects the whole
                worksheet)
            {ESC 6}{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}
Result of       {DOWN 5}{?}~/PPOS {ESC}{RECALC string162}
a formula-> \027@\027\015
            ~{ESC 6}{GOTO}{here1162}~{MENUBRANCH menu1162}

            Insert-setup-string
            Insert the setup string to a cell (affects the printing setup
                locally)
            {ESC 6}{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}
                {DOWN 5}{?}~/C~{here1162}~{GOTO}{here1162}~
            {EDIT}{HOME}{DEL}||~{MENUBRANCH menu1162}

            Row & string insert
            Insert row above and than insert the setup string in the new
                row
            {ESC 6}/WIR~{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}
                {DOWN 5}{?}~/C~{here1162}~{GOTO}{here1162}~
            {EDIT}{HOME}{DEL}||~{MENUBRANCH menu1162}

            View
            View current print setup string
            /PPOS{?}~{ESC 6}
            {MENUBRANCH menu1162}

            Up
            Move the cell pointer up
            {UP}
            {MENUBRANCH menu1162}

            Down
            Move the cell pointer down
            {DOWN}
            {MENUBRANCH menu1162}

            More
            More movement keys
            {MENUCALL sheets162}
            {MENUBRANCH menu1162}

            Quit
            Quit the macro
            {BRANCH ret162}
```

The second custom menu code is:

```
            Right
            Move the cell pointer to the right
            {RIGHT}
            {MENUBRANCH sheets162}

            Left
            Move the cell pointer to the left
            {LEFT}
            {MENUBRANCH sheets162}

            Next-sheet
            Move to next sheet
            {IF @LEFT(rel162,1)<>"@"}{NS}
            {MENUBRANCH sheets162}

            Previous-sheet
            Move to previous sheet
            {IF @LEFT(rel162,1)<>"@"}{PS}
            {MENUBRANCH sheets162}

            Quit
            Previous menu
```

The codes in B16 [string162], B28 [loc162], and B32 [form162] are the result of the following string formulas:

```
16 string162        @CELLPOINTER("contents")
28 loc162           @IF(@LEFT(rel162,1)<>"@","coord","address")
32 form162          +"{LET here1162,@CELLPOINTER("""&B28&""")}~"
```

This is the code you should type into the B16, the B28, and the B32 cells, NOT the code as it appears in the previous main listing.

```
          A         B         C         D         E      F   G   H   I
11 SETUPSTR         {LET rel162,@INFO("release")}~{RECALC loc162}
                    {RECALC form162}{MENUBRANCH menu1162}
```

The macro starts with `{LET rel162,@INFO("release")}~` which store the result of the `@INFO("release")` function in cell [rel162] for later use. The macro will use it to check if you are using a 2-D or a 3-D Lotus release. Before the macro continues, it uses `{RECALC loc162}{RECALC form162}` to update the dynamic string formulas in B28 [loc162] and B32 [form162]. To activate the [menu1162] custom menu, the macro issues `{MENUBRANCH menu1162}`.

```
          Global
          Change the GLOBAL printing setup string (affects the whole
              worksheet)
          {ESC 6}{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}
              {DOWN 5}{?}~/PPOS {ESC}{RECALC string162}
          \027@\027\015
          ~{ESC 6}{GOTO}{here1162}~{MENUBRANCH menu1162}
```

The first is the [**G**lobal] menu option, which allows you to easily select a printing setup string from a pre-prepared table. To make sure that the worksheet is in the READY mode, the macro issues `{ESC 6}`. Next it issues the `{form162}` routine command. Before continuing let's discuss the [form162] routine.

```
          A         B         C         D         E      F   G   H   I
32 form162          {LET here1162,@CELLPOINTER("coord")}~
```

As we have said earlier, the code in [form162] is the result of a string formula which is:

```
32 form162          +"{LET here1162,@CELLPOINTER("""&B28&""")}~"
```

The formula uses the content of B28 [loc162] which is also the result of a formula. Therefore, we need to look at the formula in [loc162] before continuing.

```
          A         B         C         D         E      F   G   H   I
28 loc162           coord
```

The code in [loc162] is too the result of a string formula which is:

```
28 loc162           @IF(@LEFT(rel162,1)<>"@","coord","address")
```

This formula uses the result of the `@INFO("release")` function, which was earlier stored by the macro in [rel162]. The formula checks the first character of the content of [rel162]. If "`@`" is the character, you are using a 2-D Lotus release, otherwise you are using a 3-D Lotus release.

This formula can have two possible results. If you are using a 2-D release, the result is the "address" string, but is you are using a 3-D release, the result is the "coord" string. Therefore, the formula in the [form162] cell can have two possible results:

The first result is the code:

```
32 form162          {LET here1162,@CELLPOINTER("address")}~
```

The second result is the code:

```
32 form162          {LET here1162,@CELLPOINTER("coord")}~
```

This way the macro adapts its code dynamically to the type of worksheet that you are using.

```
          A          B          C          D          E      F   G   H   I
32 form162          {LET here1162,@CELLPOINTER("coord")}~
```

The macro uses {LET here1162,@CELLPOINTER("coord")}~ to store the current cell pointer location in cell [here1162]. Later, the macro uses this address to return to this location. The {GOTO}IV1~{GOTO}setuptable1162~ macro code in the [**G**lobal] menu option moves the setup string table [setuptable1162] to the upper left corner of the screen, and {RIGHT} {DOWN 5} puts the cell pointer on the first setup string. The macro issues {?} and waits for you to highlight the new setup string and press ENTER. Then the macro issues /PPOS{ESC}, which start the setup string's change process and clear the panel for the new setup string.

Notice the SPACE character before the {ESC} macro command. The macro uses what we call the "safe technique" to insert data to the panel. Before it continues, it issues {RECALC string162} to update the formula in the next cell of code [string162]. The cell [string162] contains the @CELLPOINTER("contents") function that returns the current cell content, which is the highlighted setup string. Let's assume that you highlighted the "\027@\027\015" setup string in the table, the @CELLPOINTER("contents") formula returns the "\027@\027\015" string, which is typed into the panel because the macro treats "\027@\027\015" as a plain text, but when Lotus finds plain text which does not include macro commands, it types it directly to the panel. Now the macro issues the tilde "~" to end the process, and then uses {ESC 6} to return the worksheet to the READY mode. To return to its point of origin, the macro issues the {GOTO}{here1162}~ indirect macro command and then {MENUBRANCH menu1162} to re-activate the [menu1162] custom menu.

The second is the [**I**nsert-setup-string] menu option. This menu option allows you to embed a printing setup string in the current cell. The setup string prefix is the double split bar character "||". When Lotus see a legal printing setup code with this type of prefix, it sends the setup string to the printer. The setup string is ignored and is not printed with the worksheet.

```
Insert-setup-string
Insert the setup string to a cell (affects the printing setup
locally)
{ESC 6}{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}{DOWN 5}{?}~
    /C~{here1162}~{GOTO}{here1162}~
{EDIT}{HOME}{DEL}||~{MENUBRANCH menu1162}
```

The {ESC 6}{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}{DOWN 5} {?}~, is the same code as in the previous menu option. The macro continues with the /C~

{here1162}~ indirect macro command to copy the current cell's contents to the cell whose address is written in the cell named [here1162], which is a tricky copy command. The macro starts with the /C~ macro keys and then issues {here1162} to enter the target cell. Because the [here1162] routine contains the origin location address of the cell pointer before the macro started (stored at the beginning), the routine command injects the address into the panel and the tilde "~" finishes the copy process. Now, the macro uses the indirect {GOTO}{here1162}~ command to move the cell pointer to the point of origin. These are two fine examples of indirect macro commands that inject a cell content into panel.

To add the double split bar character "||" prefix to the setup string the macro issues {EDIT} and enters the EDIT mode, and then {HOME} moves the cursor to the beginning of the panel, and {DEL} deletes the apostrophe "'". Now the macro types the two split bar "||" characters and issues the tilde "~" to write the content of the panel to the current cell. Then the macro issues {MENUBRANCH menu1162} to re-activate the [menu1162] custom menu.

The third menu option is [**R**ow & string insert], which allows you to embed a printing setup string in the worksheet, but it inserts a new row and types the embedded string into the new row, while the previous menu option entered the setup string into the current cell position.

```
Row & string insert
Insert row above and than insert the setup string in the new row
{ESC 6}/WIR~{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}
    {DOWN 5}{?}~/C~{here1162}~{GOTO}{here1162}~
{EDIT}{HOME}{DEL}||~{MENUBRANCH menu1162}
```

The code of this menu option is exactly the same as the previous one except /WIR~ that inserts the new row. The fourth menu option allows you to view the setup string and simultaneously type a new global setup string or accept it.

```
View
View current print setup string
/PPOS{?}~{ESC 6}
{MENUBRANCH menu1162}
```

The macro issues /PPOS to start the printer global setup string change process, and then {?} to pause and wait for your response. Then you can either type a new global setup string and press ENTER, or accept the current global setup string and press ENTER. When you press ENTER, the macro issues {ESC 6} to insure that the worksheet is again in the READY mode. Next the macro uses {MENUBRANCH menu1162} to re-activate the [menu1162] custom menu. The fifth menu option is [**U**p].

```
Up
Move the cell pointer up
{UP}
{MENUBRANCH menu1162}
```

It uses {UP} to move the cell pointer in the UP direction, and then {MENUBRANCH menu1162} to re-activate the [menu1162] custom menu. The [**D**own] menu option is the same as [**U**p]. We will skip the seventh menu options for now and look at the eighth menu option, which is [**Q**uit]:

```
Quit
Quit the macro
{BRANCH ret162}
```

The [**Q**uit] menu option contains only one macro command, {BRANCH ret162}, which

branches to the [ret162] empty routine and quits. The sixth menu option is [**M**ore].

```
More
More movement keys
{MENUCALL sheets162}
{MENUBRANCH menu1162}
```

This menu option uses `{MENUCALL sheets162}` to start a second custom menu named [sheets162]. This custom menu adds more movement options to the [**U**p] menu option in the previous menu. The first is [**R**ight].

```
Right
Move the cell pointer to the right
{RIGHT}
{MENUBRANCH sheets162}
```

This is exactly the same as the [**U**p] menu option except that the command is `{RIGHT}` and it uses `{MENUBRANCH sheets162}` instead. The second is [**L**eft].

```
Left
Move the cell pointer to the right
{LEFT}
{MENUBRANCH sheets162}
```

This is exactly the same as the [**U**p] menu option except that the command is `{LEFT}` and it uses `{MENUBRANCH sheets162}` instead. The next two menu options are only for a 3-D worksheet. The first is the [**N**ext-sheet] menu option.

```
Next-sheet
Move to next sheet
{IF @LEFT(rel162,1)<>"@"}{NS}
{MENUBRANCH sheets162}
```

The macro uses `{IF @LEFT(rel162,1)<>"@"}` to check if you are using a 3-D Lotus release before it issues `{NS}` and then `{MENUBRANCH sheets162}` to re-activate the [sheets162] custom menu. The second is the [**P**revious-sheet] menu option.

```
Previous-sheet
Move to previous sheet
{IF @LEFT(rel162,1)<>"@"}{PS}
{MENUBRANCH sheets162}
```

The macro again uses `{IF @LEFT(rel162,1)<>"@"}` to check if you are using a 3-D Lotus release before it issues `{PS}` and `{MENUBRANCH sheets162}` to re-activate the [sheets162] custom menu. The last is the [**Q**uit] menu option.

```
Quit
Quit the macro
```

Here the macro simply quits because it reached an empty cell.

Because this macro is quite complicated and has two custom menus that cannot be displayed simultaneously on one page, the full list of all the cell contents is shown here.

```
A1: U [W13] '*---A macro to create a new GLOBAL printing setup string, or
            insert
A2: U [W13] '    a setup string into a cell, can be activated in the middle
                 of Lotus
```

```
A3: U [W13] '    printing menu too.
A4: [W13] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A5: [W13] '    range names in this column (starts with the \Z macro name)
A6: [W13] '*---Place the cell pointer where you want the setup string to be
A7: [W13] '*---Hold the [ALT] key and press [Z] to activate the macro
A8: U [W13] '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
A9: U [W13] '                    IT WILL WORK IN LOTUS 2.0 AND UP
A10: U [W13] '\Z
B10: [W14] '{BREAKON}
A11: U [W13] 'SETUPSTR
B11: [W14] '{LET rel162,@INFO("release")}~{RECALC loc162}{RECALC form162}
           {MENUBRANCH menu1162}
A12: [W13] '!
A13: [W13] 'menu1162
B13: [W14] 'Global
C13: [W15] 'Insert-setup-string
D13: [W15] 'Row & string insert
E13: [W12] 'View
F13: 'Up
G13: 'Down
H13: 'More
I13: 'Quit
A14: [W13] '!
B14: [W14] 'Change the GLOBAL printing setup string (affects the whole
           worksheet)
C14: [W15] 'Insert the setup string to a cell (affects the printing setup
           locally)
D14: [W15] 'Insert row above and than insert the setup string in the new row
E14: [W12] 'View current print setup string
F14: 'Move the cell pointer up
G14: 'Move the cell pointer down
H14: 'More movement keys
I14: 'Quit the macro
A15: [W13] '!
B15: [W14] '{ESC 6}{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}{DOWN 5}{?}~
           /PPOS {ESC}{RECALC string162}
C15: [W15] '{ESC 6}{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}{DOWN 5}{?}~
           /C~{here1162}~{GOTO}{here1162}~
D15: [W15] '{ESC 6}/wir~{form162}{GOTO}IV1~{GOTO}setuptable1162~{RIGHT}
           {DOWN 5}{?}~/C~{here1162}~{GOTO}{here1162}~
E15: [W12] '/PPOS{?}~{ESC 6}
F15: '{UP}
G15: '{DOWN}
H15: '{MENUCALL sheets162}
I15: '{BRANCH ret162}
A16: [W13] 'string162
B16: U [W14] @CELLPOINTER("contents")
C16: [W15] '{EDIT}{HOME}{DEL}||~{MENUBRANCH menu1162}
D16: [W15] '{EDIT}{HOME}{DEL}||~{MENUBRANCH menu1162}
E16: [W12] '{MENUBRANCH menu1162}
F16: '{MENUBRANCH menu1162}
G16: '{MENUBRANCH menu1162}
H16: '{MENUBRANCH menu1162}
A17: [W13] '!
B17: [W14] '~{ESC 6}{GOTO}{here1162}~{MENUBRANCH menu1162}
A18: [W13] '!
A19: [W13] 'sheets162
B19: [W14] 'Right
C19: [W15] 'Left
D19: [W15] 'Next-sheet
E19: [W12] 'Previous-sheet
F19: 'Quit
A20: [W13] '!
B20: [W14] 'Move the cell pointer to the right
C20: [W15] 'Move the cell pointer to the left
D20: [W15] 'Move to next sheet
E20: [W12] 'Move to previous sheet
F20: 'Previous menu
A21: [W13] '!
B21: [W14] '{RIGHT}
```

```
C21: [W15] '{LEFT}
D21: [W15] '{IF @LEFT(rel162,1)<>"@"}{NS}
E21: [W12] '{IF @LEFT(rel162,1)<>"@"}{PS}
A22: [W13] '!
B22: [W14] '{MENUBRANCH sheets162}
C22: [W15] '{MENUBRANCH sheets162}
D22: [W15] '{MENUBRANCH sheets162}
E22: [W12] '{MENUBRANCH sheets162}
A23: [W13] '!
A24: [W13] 'here1162
B24: [W14] '$A:$E$67
A25: [W13] '!
A26: [W13] 'rel162
B26: [W14] '3.00.00
A27: [W13] '!
A28: [W13] 'loc162
B28: U [W14] @IF(@LEFT(B26,1)<>"@","coord","address")
A29: [W13] '!
A30: [W13] 'ret162
A31: [W13] '!
A32: [W13] 'form162
B32: U [W14] +"{LET here1162,@cellpointer(""&B28&"")}~"
A33: [W13] '!
A34: [W13] 'setuptable1162
C34: U [W15] '************************************************
A35: [W13] '!
C35: U [W15] '* Point to the desired setup and press [ENTER] *
A36: [W13] '!
C36: U [W15] '************************************************
A37: [W13] '!
A38: [W13] '!
C38: U [W15] ^Standard
D38: U [W15] ^Condensed
E38: U [W12] ^Enlarged
F38: U 'Double-Spaced
A39: [W13] '!
B39: U [W14] 'Normal
C39: [W15] '\027@
D39: [W15] '\027@\027\015
E39: [W12] '\027@\027\014
F39: '\027@\027A\024
A40: [W13] '!
B40: U [W14] 'Double-Strike
C40: [W15] '\027@\027\069
D40: [W15] '\027@\027\015\027\069
E40: [W12] '\027@\027\014\027\069
F40: '\027@\027A\024\027\069
A41: [W13] '!
B41: U [W14] 'Emphasized
C41: [W15] '\027@\027\071
D41: [W15] '\027@\027\015\027\071
E41: [W12] '\027@\027\014\027\071
F41: '\027@\027A\024\027\071
A42: [W13] '!
B42: U [W14] 'Both last two
C42: [W15] '\027@\027\069\027\071
D42: [W15] '\027@\027\015\027\069\027\071
E42: [W12] '\027@\027\014\027\069\027\071
F42: '\027@\027A\024\027\069\027\071
A43: [W13] '!
B43: U [W14] 'Italic
C43: [W15] '\027@\0274
D43: [W15] '\027@\027\015\0274
E43: [W12] '\027@\027\014\0274
F43: '\027@\027A\024\0274
A44: [W13] '!
B44: U [W14] 'Underline
C44: [W15] '\027@\027\045\001
D44: [W15] '\027@\027\015\027\045\001
E44: [W12] '\027@\027\014\027\045\001
F44: '\027@\027A\024\027\045\001
```

[6] Print a Range with Row and Column Headings

```
     A              B                C                D              E
 1 *---A macro to print any range including the column and row headings
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                    IT WILL WORK IN RELEASE 2.0 AND UP
 9 !
10 \Z               {BREAKON}
11 FRAMERNG         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                    Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                    {LET counterb523,0}~
12 cont523          {LET hereabs523,@CELLPOINTER("address")}~
13 !                {LET rel523,@INFO("release")}~{IF @LEFT(rel523,1)="@"}
                    {labels1a523}
14 !                {IF @LEFT(rel523,1)<>"@"}/PPRWhich range ?~OBFQGOBNQQ
15 !                {GOTO}Which range ?~/RNDWhich range ?~
16 !
17 labels1a523      {GOTO}Which range ?~/WIC~/WIR~
18 !                {LET rows523,@CELLPOINTER("row")}~
19 !                /DF{BS}{DOWN}.{DOWN @ROWS(Which range ?)-1}~rows523~1~
                    8191~
20 !                @REPEAT(" ",@CELL("width",{DOWN}{UP})/2)&@MID(@CELL
                    ("address",{DOWN}{UP}),1,@FIND("$",@CELL("address",
                    {DOWN}{UP}),1)-1)~
21 !                /C~.{RIGHT @COLS(Which range ?)-1}~/RV.{END}{RIGHT}~~
                    /M.{END}{RIGHT}~{RIGHT}~{LEFT}
22 !                {GOTO}Which range ?~{UP}{LEFT}/RE~
23 !                /PPRWhich range ?~R..{UP}{LEFT}~GPQ/WDC~/WDR~
24 !
25 rows523                      1
26 cols523                      2
27 hereabs523       $A$1
28 counterb523                  3
29 rel523           3.10.00
```

This macro allows you to print a range from a worksheet including the row and columns headings.

```
     A              B                C                D              E
11 FRAMERNG         {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                    Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                    {LET counterb523,0}~
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands to freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to print and simultaneously assigns the [Which range ?] name to the same range which also acts as a prompt. Next, the macro issues `{LET counterb523,0}~` to set the content of the B28 cell named [counterb523] to zero, which serves as a counter.

```
     A              B                C                D              E
12 cont523          {LET hereabs523,@CELLPOINTER("address")}~
13 !                {LET rel523,@INFO("release")}~{IF @LEFT(rel523,1)="@"}
                    {labels1a523}
14 !                {IF @LEFT(rel523,1)<>"@"}/PPRWhich range ?~OBFQGOBNQQ
15 !                {GOTO}Which range ?~/RNDWhich range ?~
```

To be able to return to the point of origin when the macro is finished, the macro issues `{LET hereabs523,@CELLPOINTER("address")}~`, which store the current cell pointer address in cell [hereabs523], and then issues `{LET rel523,@INFO("release")}~` to store the result of the `@INFO("release")` 3-D function in cell [rel523]. The macro uses this result to determine if you are using a 2-D or a 3-D Lotus release. Now the macro issues `{IF @LEFT (rel523,1)="@"}`. If so, you are using a 2-D Lotus release and the macro issues the `{labels1a523}` routine command.

	A	B	C	D	E
17	labels1a523	{GOTO}Which range ?~/WIC~/WIR~			
18	!	{LET rows523,@CELLPOINTER("row")}~			
19	!	/DF{BS}{DOWN}.{DOWN @ROWS(Which range ?)-1}~rows523~1~8191~			
20	!	@REPEAT(" ",@CELL("width",{DOWN}{UP})/2)&@MID(@CELL("address",{DOWN}{UP}),1,@FIND("$",@CELL("address",{DOWN}{UP}),1)-1)~			
21	!	/C~.{RIGHT @COLS(Which range ?)-1}~/RV.{END}{RIGHT}~~/M.{END}{RIGHT}~{RIGHT}~{LEFT}			
22	!	{GOTO}Which range ?~{UP}{LEFT}/RE~			
23	!	/PPRWhich range ?~R..{UP}{LEFT}~GPQ/WDC~/WDR~			

The routine starts with `{GOTO}Which range ?~`, which moves the cell pointer to the upper left cell of the range to print. Then the macro issues `/WIC~/WIR~` which insert a new row and a new column. Now the macro issues `{LET rows523,@CELLPOINTER("row")}~` which store the row number of the current cell pointer position in cell [rows523]. Next the macro issues `/DF{BS}` which starts the / **D**ata **F**ill process. The `{BS}` macro command is necessary to clear any previous **D**ata **F**ill range. Now the macro issues `{DOWN}` to point to the first cell in the range to fill, and then issues the period "`.`" key to anchor the cell pointer. Then the macro issues `{DOWN @ROWS(Which range ?)-1}~`, which paints down as many cells as the number of rows in the [Which range ?] range. Next the macro issues `rows523~1~8191~` to fill the range of numbers in the painted range, starting with the row number of the current cell (which is stored in the cell named [rows523]), using step of "1".

To fill the row with column letters/headers the process is more involved. The macro writes a formula into the panel which centers the letters. To create the formula, the macro also uses the pointing method. To better understand the macro code. let's follow the code with an example. Let's assume that the range to print is the A1..C4 range and we know that the cell pointer is on the A1 cell:

	A	B	C	D	E
1	1	5	7		
2	5	3	8		
3	4	7	3		
4	2	2	1		

When the macro inserts the new row, the range is changed to look like:

	A	B	C	D	E
1					
2	1	1	5	7	
3	2	5	3	8	
4	3	4	7	3	
5	4	2	2	1	

Now the macro needs to insert the column headings in the B1..D1 range. The macro writes a formula which starts with the `@REPEAT(" ",@CELL("width",` text, therefore the panel

displays:

```
@REPEAT(" ",@CELL("width",
```

Next the macro issues `{DOWN}{UP}`, which add the current cell address to the formula, therefore the panel displays:

```
@REPEAT(" ",@CELL("width",A1
```

Next the macro writes `)/2)&@MID(@CELL("address",` into the panel which now displays:

```
@REPEAT(" ",@CELL("width",A1)/2)&@MID(@CELL("address",
```

Again the macro issues `{DOWN}{UP}`, which add the A1 cell address to the panel, which displays:

```
@REPEAT(" ",@CELL("width",A1)/2)&@MID(@CELL("address",A1
```

The macro continues to write `),1,@FIND("$",@CELL("address",` so panel displays:

```
@REPEAT(" ",@CELL("width",A1)/2)&@MID(@CELL("address",A1),1,
    @FIND("$",@CELL("address",
```

Again the macro issues `{DOWN}{UP}`, which add the A1 cell address to the panel, which displays:

```
@REPEAT(" ",@CELL("width",A1)/2)&@MID(@CELL("address",A1),1,
    @FIND("$",@CELL("address",A1
```

To finish the formula, the macro writes `),1)-1)` into the panel which finally displays:

```
@REPEAT(" ",@CELL("width",A1)/2)&@MID(@CELL("address",A1),1,
    @FIND("$",@CELL("address",A1),1)-1)
```

The formula uses the `@MID,@FIND` and `@CELL("address",A1)` functions to find the column letter and then `@REPEAT(" ",@CELL("width",A1)/2)` formula to add spaces to center the heading letter in the cell. Now the macro issues the tilde "`~`" to write the formula into the current cell. Next the macro issues `/C~.{RIGHT @COLS(Which range ?)-1}~` which copy the formula to the B1, C1 and the D1 cells. To change the formulas in the A1..D1 range into labels, the macro issues `/RV.{END}{RIGHT}~~`. The range now looks like:

	A	B	C	D	E
1	A	B	C	D	
2	1	1	5	7	
3	2	5	3	8	
4	3	4	7	3	
5	4	2	2	1	

However, all the headings are shifted one cell to the left because of the new column the macro inserted earlier. For example, this new column changed the "C" column to the "D" column. Therefore, the macro issues `/M.{END}{RIGHT}~{RIGHT}~` to move the letters in the A1..D1 range to the B1..E1 range, which now looks like:

	A	B	C	D	E
1		A	B	C	D

2	1	1	5	7
3	2	5	3	8
4	3	4	7	3
5	4	2	2	1

Now all that is left is to print the A1..D5 range. The macro issues `{GOTO}Which range ?~{UP}{LEFT}/RE~` which move the cell pointer to the A1 cell and erase its contents. Then the macro issues `/PPRWhich range ?~R..{UP}{LEFT}~GPQ` to print the range, and `/WDC~/WDR~` to erase the inserted column and the inserted row. Let's continue with the [cont523] routine.

	A	B	C	D	E
12	cont523	{LET hereabs523,@CELLPOINTER("address")}~			
13	!	{LET rel523,@INFO("release")}~{IF @LEFT(rel523,1)="@"}{labels1a523}			
14	!	{IF @LEFT(rel523,1)<>"@"}/PPRWhich range ?~OBFQGOBNQQ			
15	!	{GOTO}Which range ?~/RNDWhich range ?~			

The macro issues `{IF @LEFT(rel523,1)<>"@"}` to check if you are using a 3-D release. If so, the macro issues `/PPRWhich range ?~OBFQGOBNQQ` code which prints the range with the **F**rame **H**eadings menu options. In the 3-D releases, Lotus added the option to print a range with the headings, therefore the code is much shorter and simpler. When the macro finishes the printing process, it issues `{GOTO}Which range ?~` to place the cell pointer on the top left cell of the printed range and then `/RNDWhich range ?~` to delete the temporary [Which range ?] range name to leave a clean worksheet.

[7] Print a List of Ranges Picked Using Point and Shoot

```
     A          B          C          D          E          F          G
 1 *---A macro to print a LIST of ranges picked from the screen or
 2     entered manually
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                     IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z               {BREAKON}
11 RANGSPRT         {LET rel155,@INFO("release")}~{RECALC add1155}
                    {RECALC add2155}{RECALC add3155}
12 add1155          {LET here1155,@CELLPOINTER("coord")}~
13 !                {GOTO}names155~{DOWN 3}{MENUBRANCH menu1155}
14 !
15 menu1155         View/cha Add/Edit New_list Print_al Quit
16 !                View or change theEnter newPrint outQuit the macro
17 again155         Select th{GOTO}nam{GOTO}nam{GOTO}nam{BRANCH end155}
18 !                {IF key11{MENUCALL{IF @cell{LET print2155,@CELLPOINTER
19 !                {ESC}{IF {MENUBRAN{MENUCALL{LET print3155,@CELLPOINTER
20 !                {key1155}{?}~{LET {MENUBRAN{BRANCH print1155}
21 end1155          {MENUBRANCH menu1155}      {MENUBRANCH menu1155}
22 !
23 key1155          {UP}
24 !
25 change155        /RNC
26 change1155       MENU2155
27 !                ~....{?}~{DOWN}
28 !
29 rel155           3
30 !
31 loc1155          $B$33
32 !
33 new1155          MENU2155
34 !
35 menu2155         Select   Create   Up       Down     Print_all Quit
36 !                Select raCreate neMove the Move the Print out Previous...
37 add2155          {LET here{GETLABEL{UP}     {DOWN}   {GOTO}nam {BRANCH ...
38 !                {GOTO}{NA{windowso{MENUBRAN{MENUBRAN{LET print2155,@CE...
39 !                {WINDOWSO{MENUBRANCH menu2155}      {LET print3155,@CE...
40 add3155          {LET loc1155,@CELLPOINTER("coord")}~{BRANCH print1155}...
41 !                {LET new1155,+"@SUM("&@CELL("CONTENT{MENUBRANCH menu21...
42 !                {WINDOWSOFF}{PANELOFF}{GOTO}names155~{DOWN}{LEFT}/C~.....
43 !                {MENUBRANCH menu2155}
44 !
45 print1155        /PPCAR
46 print2155        Budget
47 !                ~OH {ESC}@|
48 print3155        Budget
49 !                |Page #
50 !                ~qagllq{DOWN}
51 !                {IF @CELLPOINTER("type")="b"}{MENUBRANCH menu1155}
52 !                {LET print2155,@CELLPOINTER("contents")}~
53 !                {LET print3155,@CELLPOINTER("contents")}~
                    {BRANCH print1155}
54 !
55 end155          {ESC 6}{RETURN}
56 !
57 here1155        $B$69
58 !
59 names155        =========
60 !                RANGES TO PRINT
61 !                =========
62 !                Budget
63 !                Receiving
64 !                Accounts
```

This macro allows you to pick the ranges to print from a full screen list of range names using point and shoot. Then the macro builds a list of the selected range names and prints them in order when you choose the [**P**rint_all] menu option from the custom menu. This is quite a complicated macro with two custom menus which cannot fit on one page without overlapping. Therefore we show every menu option's code separately and, at the end of this section we bring a full list of all the cell addresses and contents. Here is the list of the dynamic string formulas which this macro uses in the B12, B37 and the B40 cells.

```
12 add1155          @IF(@LEFT(B29,1)<>"@","{LET here1155,@CELLPOINTER(""coord"")
                    }~","{LET here1155,@CELLPOINTER(""address"")}~")
37 add2155          @IF(@LEFT(B29,1)<>"@","{LET here1155,@CELLPOINTER(""coord"")
                    }~","{LET here1155,@CELLPOINTER(""address"")}~")
40 add3155          @IF(@LEFT(B29,1)<>"@","{LET loc1155,@CELLPOINTER(""coord"")
                    }~","{LET loc1155,@CELLPOINTER(""address"")}~")
```

If you intend to key this macro code into 1-2-3, you need to key in these formulas, NOT the code as it appears in the main listing, which is the result of these formulas.

	A	B	C	D	E	F	G
11	RANGSPRT	{LET rel155,@INFO("release")}~{RECALC add1155} {RECALC add2155}{RECALC add3155}					

The macro begins with the `{LET rel155,@INFO("release")}~` macro commands, which store the result of the `@INFO("release")` function in the B29 cell named [rel155] for later use. later, the macro uses the stored result to determine if you are using a 3-D or 2-D Lotus release . Then it issues `{RECALC add1155}{RECALC add2155}{RECALC add3155}` to update the formulas in the B12, B37 and B40 cells named [add1155], [add2155] and [add3155], respectively.

	A	B	C	D	E	F	G
12	add1155	{LET here1155,@CELLPOINTER("coord")}~					
13	!	{GOTO}names155~{DOWN 3}{MENUBRANCH menu1155}					

Now the macro processes the code in the B12 cell [add1155]. Before we continue, we will look at the string formulas behind the code in this cell. The formula is:

```
12 add1155          @IF(@LEFT(B29,1)<>"@","{LET here1155,@CELLPOINTER(""coord"")
                    }~","{LET here1155,@CELLPOINTER(""address"")}~")
```

The formula uses `@IF(@LEFT(rel155,1))<>@` to check if the first character of the contents of cell [rel155] is not equal to the "`@`" character. If so, you are using a 2-D Lotus release and the formula in the B12 cell returns:

```
{LET here1155,@CELLPOINTER(""address"")}~
```

Otherwise you are using a 3-D Lotus release and the formula in the B12 cell returns:

```
{LET here1155,@CELLPOINTER(""coord"")}~
```

The macro uses the result of the formula as the code to process. In our example, the code is

	A	B	C	D	E	F	G
12	add1155	{LET here1155,@CELLPOINTER("coord")}~					

which means that a 3-D release was used when this macro list was recorded. The code in the B12 cell [add1155] stores the result of the `@CELLPOINTER("coord")` in [here1155] for later use.

	A	B	C	D	E	F	G
13	!	{GOTO}names155~{DOWN 3}{MENUBRANCH menu1155}					

Next the macro issues `{GOTO}names155~{DOWN 3}` to move the cell pointer to the beginning of the range names list in B62 and `{MENUBRANCH menu1155}` to start the first custom menu.

```
View/change
View or change the range area to print
Select the range to view and press [ENTER] {GET key1155}{ESC}
{IF key1155="{ESC}"}{DOWN}{UP}{BRANCH end155}
{IF key1155="~"}{DOWN}{UP}{LET change1155,@CELLPOINTER("contents")}~
    {change155}{BRANCH end1155}
{key1155}{?}~{LET change1155,@CELLPOINTER("contents")}~
    {change155}{UP}{DOWN}
{MENUBRANCH menu1155}
```

The first menu option in [menu1155] is [**V**iew/change], which allows you to view and change the size of the range to print. The code of this menu option occupies the B15..B21 range. This menu option should be used after you have used the [Add/Edit] menu option to add range names to the list of the ranges to print. The code in the B17 cell of this menu option is a result of a formula. Therefore we have assigned the [again155] range name to the B17 cell by writing the name of the range in the A17 cell, when you define the range names as instructed, the [again155] range name is assigned to the B17 cell. This is the same trick we have used with the B21 cell, which has the [end1155] range name. We could use this technique because [View/change] is the first menu option and resides in the second column of the macro. The code in the [View/change] menu option starts with the prompt

```
Select the range to view and press [ENTER]
```

The text in this line does not include any Lotus command, so Lotus types it directly to the panel as it appears in the code and issues `{GET key1155}` that halts the macro with the prompt in the panel and waits until you press a key. The key you press is stored in cell [key1155]. Next the macro issues `{ESC}` to clear the message from the panel before Lotus writes it into the current cell.

```
{IF key1155="{ESC}"}{DOWN}{UP}{BRANCH end155}
```

Now the macro uses `{IF key1155="{ESC}"}` to check if you pressed ESC. If so, the macro issues `{DOWN}{UP}{BRANCH end155}` to branch to the [end155] routine, which contains `{MENUBRANCH menu1155}` returning the macro back to the [menu1155] menu. Again, it seems that the `{DOWN}{UP}` macro commands are not needed; however as we have mentioned in similar cases, in some of the Lotus releases, the macro does not perform correctly without these commands due to incompatibilities between the many versions of Lotus 1-2-3. The second menu option is [**A**dd/Edit], which allows you to add more names to the list by activating a second custom menu, the [menu2155] menu, which lets you select names using point and shoot or to type a name and paint the range.

```
Add/Edit
Add range names to the list or edit a range size
{GOTO}names155~{END}{DOWN 2}
{MENUCALL menu2155}
{MENUBRANCH menu1155}
```

It uses {GOTO}names155~{END}{DOWN 2} to move the cell pointer to the end of the list of names, which at the beginning were assigned the [names155] range name. Then the macro issues {MENUCALL menu2155} to activates the second custom menu, the [menu2155] menu. Before we look into the second custom menu [menu2155], let's continue and see the rest of the menu options in this menu. The third menu option is [**N**ew_list]:

```
New_list
Enter new list of ranges
{GOTO}names155~{DOWN 3}
{IF @CELLPOINTER("type")<>"b"}/RE.{END}{DOWN}~
{MENUCALL menu2155}
{MENUBRANCH menu1155}
```

It uses {GOTO}names155~{DOWN 3} to move the cell pointer to the beginning of the list of the range names and then issues {IF @CELLPOINTER("type")<>"b"} to checks if the current cell is blank. If the cell is not blank, the macro erases the list using /RE.{END}{DOWN}~ and activates the second custom menu using {MENUCALL menu2155}. The fourth menu option is [**P**rint_all]:

```
Print_all
Print out ranges shown
{GOTO}names155~{DOWN 3}
{LET print2155,@CELLPOINTER("contents")}~
{LET print3155,@CELLPOINTER("contents")}~
{BRANCH print1155}
{MENUBRANCH menu1155}
```

The menu uses {GOTO}names155~{DOWN 3} to move the cell pointer to the first name in the list and issues {LET print2155,@CELLPOINTER("contents")}~ to copy the cell content to cell [print2155], and issues {LET print3155,@CELLPOINTER("contents")}~ to copy the cell pointer content to cell [print3155]. Next the macro issues {BRANCH print1155} to branch to the [print1155] routine. The [print1155] routine is in charge of the printing process of the ranges.

	A	B	C	D	E	F	G
45	print1155	/PPCAR					
46	print2155	Budget					
47	!	~OH {ESC}@|					
48	print3155	Budget					
49	!	|Page #					
50	!	~qagllq{DOWN}					
51	!	{IF @CELLPOINTER("type")="b"}{MENUBRANCH menu1155}					
52	!	{LET print2155,@CELLPOINTER("contents")}~					
53	!	{LET print3155,@CELLPOINTER("contents")}~					
		{BRANCH print1155}					

It starts with the /PPCAR macro keys which clear the printing setup and then issues the range name In this example the name is [Budget] because it is the first name in the list. The content in B46 [print2155] and B48 [print3155], have been stored there earlier by the [Print_all] menu option routine. This is an example of using dynamic code in macro programming. The macro, itself, inserts the name of the range to print into its code before it processes that part of the code. The full code of the printing command is:

```
/PPCARBudjet~OH {ESC}@|Budget|Page #~QAGLLQ{DOWN}
```

The range to print is the [Budget] range. The macro uses the OH {ESC}@|Budget|Page #~

code to define the page header. Pay attention to the space character between the `OH` and the `{ESC}`. The macro uses here the "safe technique" to enter data into the panel. The "@" character instructs Lotus to type the current date in the left part of the header, the `|Budget|` instructs Lotus to type the "Budget" string in the middle of the header, and the `Page #` instructs Lotus to type the "Page " string at the right side of the header, and the "#" character instructs Lotus to type the page number following the "Page " string. Now the macro issues `QAGLLQ{DOWN}` which sends the printed range to the printer, aligns the new page, inserts two empty lines, and moves the cell pointer to the next item in the list.

```
     A          B          C          D          E          F          G
51 !          {IF @CELLPOINTER("type")="b"}{MENUBRANCH menu1155}
```

The macro uses `{IF @CELLPOINTER("type")="b"}` to check if the current cell is blank, if so, the macro has reached the end of the list and it issues `{MENUBRANCH menu1155}` to return control to the main custom menu.

```
     A          B          C          D          E          F          G
52 !          {LET print2155,@CELLPOINTER("contents")}~
53 !          {LET print3155,@CELLPOINTER("contents")}~{BRANCH print1155}
```

If the cell is not blank, the macro stores the new range name from the list into B46 [print2155], and B48 [print3155] and issues `{BRANCH print1155}` to start the printing process again. The last menu option is [**Q**uit]:

```
Quit
Quit the macro
{BRANCH end155}
```

The macro uses `{BRANCH end155}` to transfer control to the [end155] routine

```
     A          B          C          D          E          F          G
55 end155     {ESC 6}{RETURN}
```

it issues `{ESC 6}` to return Lotus 1-2-3 to READY mode and then uses `{RETURN}` to quit. There is no routine to return to, because the macro used `{MENUBRANCH}` to activate the first custom menu and not `{MENUCALL}`. It seems unnecessary to use this type of quitting, we could just leave the [end155] routine blank but there is always more than one way to do a specific job. Now we can start with the second custom menu, the first menu option is, which displays a full screen list of all the range names in the worksheet and allows you to select a range name to print by pointing and pressing ENTER. Then the macro adds this name to the list.

```
Select
Select ranges to print from existing range names
{LET here1155,@CELLPOINTER("address")}~
{GOTO}{NAME}{NAME}{?}~
{WINDOWSOFF}{PANELOFF}
{LET loc1155,@CELLPOINTER("address")}~
{LET new1155,+"@SUM("&@CELL("contents",loc1155)&")"}~{GOTO}new1155~
    {EDIT}{HOME}{DEL}{END}{LEFT 2}{ABS}{HOME}{DEL 5}{END}{BS}~
    {GOTO}{here1155}~/Cnew1155~~{LET change1155,@CELLPOINTER("
    contents")}~{WINDOWSON}{PANELON}{change155}
{WINDOWSOFF}{PANELOFF}{GOTO}names155~{DOWN}{LEFT}/C~.{RIGHT}{END}
    {DOWN}{LEFT}~{GOTO}IV1~{GOTO}names155~{END}{DOWN 2}{WINDOWSON}
    {PANELON}
{MENUBRANCH menu2155}
```

The [**S**elect] menu option code starts with

```
{LET here1155,@CELLPOINTER("address")}~
```

which is the result of the following formula:

```
37 add2155          @IF(@LEFT(B29,1)<>"@","{LET here1155,@CELLPOINTER(""coord"")
                    }~","{LET here1155,@CELLPOINTER(""address"")}~")
```

The `@IF(@LEFT(B29,1)<>"@"` condition checks if the first character of the content of the cell named [rel155] is the "`@`" character, if so, then you are using a 2-D Lotus release, otherwise you are using a 3-D Lotus release. If you use a 3-D release the result of the formula is the following code:

```
{LET here1155,@CELLPOINTER("coord")}~
```

If you are using a 2-D release, the result of the formula is the code:

```
{LET here1155,@CELLPOINTER("address")}~
```

Either the address or coord of the current cell is stored in cell [here1155] for later use. Next the macro issues `{GOTO}{NAME}{NAME}`. This set of commands display a full screen list of all the range names in the worksheet, and `{?}` halts the macro allowing you point to the name to go to. When you press ENTER, the macro resumes control and uses `{WINDOWSOFF} {PANELOFF}` to freeze the screen and panel display activities. Now the macro issues

```
{LET loc1155,@CELLPOINTER("address")}~
```

to store the current cell position, the chosen range position, in cell [loc1155]. This line of code is again the result of the

```
40 add3155          @IF(@LEFT(B29,1)<>"@","{LET loc1155,@CELLPOINTER(""coord"")
                    }~","{LET loc1155,@CELLPOINTER(""address"")}~")
```

string formula. The content of [loc1155] which is the address of the upper left cell of the chosen range, is now used by the macro to extract the actual name of the range in an interesting way.

```
{LET new1155,+"@SUM("&@CELL("contents",loc1155)&")"}~{GOTO}new1155~
    {EDIT}{HOME}{DEL}{END}{LEFT 2}{ABS}{HOME}{DEL 5}{END}{BS}~
    {GOTO}{here1155}~/Cnew1155~~{LET change1155,@CELLPOINTER("
    contents")}~{WINDOWSON}{PANELON}{change155}
```

The macro starts with

```
{LET new1155,+"@SUM("&@CELL("contents",loc1155)&")"}~
```

If the content of [loc1155] is the address `$B$33`, this specific command builds the following string:

```
'@SUM($B$33)
```

in cell [new1155]. Next the macro issues `{GOTO}new1155~`, which puts the cell pointer on [new1155] and uses `{EDIT}` to enter EDIT mode and the panel shows:

```
'@SUM($B$33)_
```

The cursor position is marked by the underscore after the closing bracket. Now the macro issues {HOME}, which sends the cursor to the very beginning of the panel, shows:

```
'@SUM($B$33)
^
```

The cursor position is marked by the "^" under the apostrophe "'" character. Next the macro issues {DEL}, which deletes the apostrophe "'" thereby converting the text in [new1155] into the

```
@SUM($B$33)
```

active formula. Now the macro moves the cursor back to the end of the panel using {END}, which again puts the cursor at the end of the text; therefore the panel displays:

```
@SUM($B$33)_
```

The cursor position is again marked by the underscore after the closing bracket. Now the macro issues {LEFT 2}, which moves the cursor two characters to the left. At this point, the panel displays:

```
@SUM($B$33)
         ^
```

The cursor position is marked by the "^" under the "3" character. Now the macro issues {ABS}, which changes the text in the panel to:

```
@SUM(Budget)
```

The name of the B33 cell is the [Budget] range name. Now the macro issues {HOME}, which moves the cursor to the beginning of the panel and then issues {DEL 5} which deletes the first five characters and the panel displays:

```
Budget)
^
```

Now the macro issues {END}, which moves the cursor to the end of the text in the panel. Therefore the panel displays:

```
Budget)_
```

For the final touch, the macro issues {BS} (the macro equivalent of the BACKSPACE key), which deletes the closing bracket. Next the macro issues the tilde "~" macro command, (the equivalent of the ENTER key). The result is that the content of [new1155] will become the string "Budget" which is the name of the range that you chose from the screen. Isn't it beautiful? this what drives us the group of macro developers to love the Lotus macro language and dig deeper into it. If you repeat the process manually you will see how it evolves. **Pick a Range Name from the Screen** will show you an even easier way without the @SUM function.

```
Create
Create new range names or change existing range names
{GETLABEL "Enter name to create: ",new1155}~/Cnew1155~~
    /RNC{new1155}~{?}~
```

```
{WINDOWSOFF}{PANELOFF}{GOTO}names155~{DOWN}{LEFT}/C~.{RIGHT}{END}
    {DOWN}{LEFT}~{GOTO}IV1~{GOTO}names155~{END}{DOWN 2}{WINDOWSON}
    {PANELON}
{MENUBRANCH menu2155}
```

The [**C**reate] menu option issues `{GETLABEL "Enter name to create: ",new1155}~` which displays `"Enter name to create: "` in the panel and stores the name that you type in [new1155]. Then the macro uses `/Cnew1155~~` to copy the name from the [new1155] cell to the current cell position. Using `/RNC{new1155}~{?}~` the macro prompts you to paint the range whose name was typed earlier. Here the macro uses a trick; it first asks for the range name and stores the name in [new1155], then it starts a **R**ange **N**ame **C**reate process and uses the `{new1155}` routine command to inject the name into the panel. This combination of the `{GETLABEL}` and the `/RNC` a has more professional look and greater clarity because of the addition of the `"Enter name to create: "` prompt message compared to the common `/RNC{?}~{?}~` command code.

```
{WINDOWSOFF}{PANELOFF}{GOTO}names155~{DOWN}{LEFT}/C~.{RIGHT}{END}
    {DOWN}{LEFT}~{GOTO}IV1~{GOTO}names155~{END}{DOWN 2}{WINDOWSON}
    {PANELON}
```

This code has meaning only when the macro is used with a macro manager. The macro uses `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities and then issues `{GOTO}names155~{DOWN}{LEFT}` to place the cell pointer on the A60 cell of the macro, which contains the exclamation mark "!". The macro now adjusts and adds exclamation marks to make the A column the same length as the B column. The macro copies the exclamation mark "!" all the way down to fit to the changing length because more range names have been added to the list at the end of the "B" column. The macro issues `/C~.{RIGHT}{END} {DOWN}{LEFT}~` that use the adjacent "B" column's length to adjust the length of the "A" column by adding exclamation marks.

To bring the list of range names to print to the top left corner of the screen, the macro uses `{GOTO}IV1~{GOTO}names155~`. When the cell pointer is in place, it issues `{WINDOWSON}{PANELON}` which resume the screen and panel display activities. To end the routine and return to the custom menu the macro issues `{MENUBRANCH menu2155}`.

```
Up
Move the cell pointer up
{UP}
{MENUBRANCH menu2155}

Down
Move the cell pointer down
{DOWN}
{MENUBRANCH menu2155}
```

These two menu options allow you to move the cell pointer up and down in the list of the range names in case you want to edit or change a range name.

```
Print_all
Print out ranges shown
{GOTO}names155~{DOWN 3}
{LET print2155,@CELLPOINTER("contents")}~
{LET print3155,@CELLPOINTER("contents")}~
{BRANCH print1155}
{MENUBRANCH menu2155}
```

This menu option has the same code as the [**P**rint_all] menu option in the first custom menu

except that {MENUBRANCH menu2155} resumes control to the second custom menu.

```
Quit
Previous menu
{BRANCH end155}
```

This [**Q**uit] menu option is exactly the same as the [**Q**uit] menu option in the first custom menu. To make it easy and absolutely clear for you to type in this macro code, we show a full list of all the cell contents in the macro.

```
A1: U [W14] '*---A macro to print a LIST of ranges picked from the screen or
A2: U [W14] '    entered manually
A3: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A4: [W14] '    range names in this column (starts with the \Z macro name)
A5: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A6: [W14] '!
A7: U [W14] '        THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
A8: U [W14] '                    IT WILL WORK IN LOTUS 2.0 AND UP
A9: [W14] '!
A10: U [W14] '\Z
B10: [W14] '{BREAKON}
A11: U [W14] 'RANGSPRT
B11: [W14] '{LET rel155,@INFO("release")}~{RECALC add1155}{RECALC add2155}
           {RECALC add3155}
A12: [W14] 'add1155
B12: U [W14] @IF(@LEFT(B29,1)<>"@","{LET here1155,@CELLPOINTER(""coord"")}~",
             "{LET here1155,@CELLPOINTER(""address"")}~")
A13: [W14] '!
B13: [W14] '{GOTO}names155~{DOWN 3}{MENUBRANCH menu1155}
A14: [W14] '!
A15: [W14] 'menu1155
B15: [W14] 'View/change
C15: [W15] 'Add/Edit
D15: [W15] 'New_list
E15: 'Print_all
F15: 'Quit
A16: [W14] '!
B16: [W14] 'View or change the range area to print
D16: [W15] 'Enter new list of ranges
E16: 'Print out ranges shown
F16: 'Quit the macro
A17: [W14] 'again155
B17: [W14] 'Select the range to view and press [ENTER] {GET key1155}
C17: [W15] '{GOTO}names155~{END}{DOWN 2}
D17: [W15] '{GOTO}names155~{DOWN 3}
E17: '{GOTO}names155~{DOWN 3}
F17: '{BRANCH end155}
A18: [W14] '!
B18: [W14] '{IF key1155="{ESC}"}{DOWN}{UP}{BRANCH end155}
C18: [W15] '{MENUCALL menu2155}
D18: [W15] '{IF @CELLPOINTER("type")<>"b"}/RE.{END}{DOWN}~
E18: '{LET print2155,@CELLPOINTER("contents")}~
A19: [W14] '!
B19: [W14] '{ESC}{IF key1155="~"}{DOWN}{UP}{LET change1155,@CELLPOINTER("
           contents")}~{change155}{BRANCH end1155}
C19: [W15] '{MENUBRANCH menu1155}
D19: [W15] '{MENUCALL menu2155}
E19: '{LET print3155,@CELLPOINTER("contents")}~
A20: [W14] '!
B20: [W14] '{key1155}{?}~{LET change1155,@CELLPOINTER("contents")}~
            {change155}{UP}{DOWN}
D20: [W15] '{MENUBRANCH menu1155}
E20: '{BRANCH print1155}
A21: [W14] 'end1155
B21: [W14] '{MENUBRANCH menu1155}
E21: '{MENUBRANCH menu1155}
A22: [W14] '!
```

```
A23: [W14] 'key1155
B23: [W14] '{UP}
A24: [W14] '!
A25: [W14] 'change155
B25: [W14] '/RNC
A26: [W14] 'change1155
B26: [W14] 'MENU2155
A27: [W14] '!
B27: [W14] '~....{?}~{DOWN}
A28: [W14] '!
A29: [W14] 'rel155
B29: [W14] '3
A30: [W14] '!
A31: [W14] 'loc1155
B31: [W14] '$B$33
A32: [W14] '!
A33: [W14] 'new1155
B33: [W14] 'MENU2155
A34: [W14] '!
A35: [W14] 'menu2155
B35: [W14] 'Select
C35: [W15] 'Create
D35: [W15] 'Up
E35: 'Down
F35: 'Print_all
G35: 'Quit
A36: [W14] '!
B36: [W14] 'Select ranges to print from existing range names
C36: [W15] 'Create new range names or change existing range names
D36: [W15] 'Move the cell pointer up
E36: 'Move the cell pointer down
F36: 'Print out ranges shown
G36: 'Previous menu
A37: [W14] 'add2155
B37: U [W14] @IF(@LEFT(B29,1)<>"@","{LET here1155,@CELLPOINTER(""coord"")}~",
             "{LET here1155,@CELLPOINTER(""address"")}~")
C37: [W15] '{GETLABEL "Enter name to create: ",new1155}~/cnew1155~~
           /RNC{new1155}~{?}~
D37: [W15] '{UP}
E37: '{DOWN}
F37: '{GOTO}names155~{DOWN 3}
G37: '{BRANCH end155}
A38: [W14] '!
B38: [W14] '{GOTO}{NAME}{NAME}{?}~
C38: [W15] '{WINDOWSOFF}{PANELOFF}{GOTO}names155~{DOWN}{LEFT}/c~.{RIGHT}
           {END}{DOWN}{LEFT}~{GOTO}IV1~{GOTO}names155~{END}{DOWN 2}{WINDOWSON}
           {PANELON}
D38: [W15] '{MENUBRANCH menu2155}
E38: '{MENUBRANCH menu2155}
F38: '{LET print2155,@CELLPOINTER("contents")}~
A39: [W14] '!
B39: [W14] '{WINDOWSOFF}{PANELOFF}
C39: [W15] '{MENUBRANCH menu2155}
F39: '{LET print3155,@CELLPOINTER("contents")}~
A40: [W14] 'add3155
B40: U [W14] @IF(@LEFT(B29,1)<>"@","{LET loc1155,@CELLPOINTER(""coord"")}~",
             "{LET loc1155,@CELLPOINTER(""address"")}~")
F40: '{BRANCH print1155}
A41: [W14] '!
B41: [W14] '{LET new1155,+"@SUM("&@CELL("contents",loc1155)&")"}~{GOTO}
           new1155~{EDIT}{HOME}{DEL}{END}{LEFT 2}{ABS}{HOME}{DEL 5}{END}{BS}~
           {GOTO}{here1155}~/Cnew1155~~{LET change1155,@CELLPOINTER
           ("contents")}~{WINDOWSON}{PANELON}{change155}
F41: '{MENUBRANCH menu2155}
A42: [W14] '!
B42: [W14] '{WINDOWSOFF{PANELOFF}{GOTO}names155~{DOWN}{LEFT}/C~.
           {RIGHT}{END}{DOWN}{LEFT}~{GOTO}IV1~{GOTO}names155~{END}{DOWN 2}
           {WINDOWSON}{PANELON}
A43: [W14] '!
B43: [W14] '{MENUBRANCH menu2155}
A44: [W14] '!
```

```
A45: [W14] 'print1155
B45: [W14] '/PPCAR
A46: [W14] 'print2155
B46: [W14] 'BudgetA47: [W14] '!
B47: [W14] '~OH {ESC}@|
A48: [W14] 'print3155
B48: [W14] 'Budget
A49: [W14] '!
B49: [W14] '|Page #
A50: [W14] '!
B50: [W14] '~QAGLLQ{DOWN}
A51: [W14] '!
B51: [W14] '{IF @CELLPOINTER("type")="b"}{MENUBRANCH menu1155}
A52: [W14] '!
B52: [W14] '{LET print2155,@CELLPOINTER("contents")}~
A53: [W14] '!
B53: [W14] '{LET print3155,@CELLPOINTER("contents")}~{BRANCH print1155}
A54: [W14] '!
A55: [W14] 'end155
B55: [W14] '{ESC 6}{RETURN}
A56: [W14] '!
A57: [W14] 'here1155
B57: [W14] '$B$69
A58: [W14] '!
A59: [W14] 'names155
B59: U [W14] \=
A60: [W14] '!
B60: U [W14] 'RANGES TO PRINT
A61: [W14] '!
B61: U [W14] \=
A62: [W14] '!
B62: [W14] 'again155
A63: [W14] '!
B63: [W14] 'change155
A64: [W14] '!
B64: [W14] 'change1155
```

[7] Print Every Other Column in a Range

```
     A                B                C                D                E
 1 *---A macro to print only every other specified number of columns. The
 2     macro prompts the user for the number of columns to skip. Example:
 3     To print only every 4th column, answer 3 to the "How many columns to
 4     skip ? " prompt.
 5 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 6     range names in this column (starts with the \Z macro name).
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                      IT WILL WORK IN LOTUS 2.0 AND UP
10 \Z              {BREAKON}
11 PRINTALT        {LET rel206,@INFO("release")}~{RECALC loc206}
                   {RECALC form206}
12 form206         {LET here1206,@CELLPOINTER("coord")}~
13 !               {WINDOWSOFF}{PANELOFF}/RNCRange to print?~~/RND
                   Range to print?~/RNC{PANELON}{WINDOWSON}Range to print?~
                   {BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}
14 !               {PANELON}{GETLABEL "How many columns to skip ? ",skip206}~
15 !               {LET temp1206,@STRING(@MOD(@COLS(Range to print?),(@VALUE(
                   skip206)+1)),0)}~
16 !               {GETNUMBER "No. of copies ? ",copies206}~{PANELOFF}
                   {RECALC colshide206}
17 !               {FOR counter1206,1,(@COLS(Range to print?)-@MOD(@COLS
                   (Range to print?),(@VALUE(skip206)+1)))/(@VALUE(skip206)
                   +1)+1,1,colshide206}{ESC 10}
18 !               {FOR counter2206,1,copies206,1,prints206}{ESC 10}
19 !               /WCDRange to print?~{GOTO}{here1206}~/RNDRange to print?~
                   {ESC 10}{CALC}
20 !
21 colshide206     {RIGHT}/WCH.{RIGHT 2}~{ESC 10}{RETURN}
22 !
23 prints206       {ONERROR err206}{IF @MOD(@COLS(Range to print?),@VALUE
                   (skip206)+1)>0}/PPCRRRange to print?~R{LEFT}{RIGHT}~
                   AGPQ{ESC 10}{RETURN}
24 !               {IF @MOD(@COLS(Range to print?),@VALUE(skip206)+1)=0}
                   /PPCRRRange to print?~R{LEFT}{RIGHT}~AGPQ{ESC 10}{RETURN}
25 !
26 temp1206        0
27 skip206         3
28 counter1206            6
29 counter2206            2
30 copies206              1
31 here1206        $B:$A$1
32 key206          ~
33 err206          {IF @LEFT(rel206,1)<>"@"}{BEEP}Turn on your computer
                   and press any key to continue...{GET key206}{ESC}/PRQ
                   /WCDA1..IV1~
34 !               {IF @LEFT(rel206,1)="@"}{BEEP}Turn on your computer
                   and press any key to continue...{GET key206}{ESC}
                   /WCDA1..IV1~
35 !
36 rel206          3.00.00
37 !
38 loc206          coord
```

Notice that the codes in the B12..B12, B21..B21 and the B38 cells are the result of the following dynamic string formulas. If you intend to key this macro into 1-2-3, you need to key in the following formulas and NOT the code appearing in the main macro text.

```
12 form206         +"{LET here1206,@CELLPOINTER("""&B38&""")}~"
21 colshide206     +"{RIGHT}/WCH.{RIGHT "&@STRING(@VALUE(B27)-1,0)&"}~
                   {ESC 10}{RETURN}"
38 loc206          @IF(@LEFT(B36,1)<>"@","coord","address")
```

Using this macro, you can print table columns apart from each other by a specified number of columns. For example, let's look at the following table of numbers:

```
0       11      22      33      44      55      66      77      88      99      110
1       12      23      34      45      56      67      78      89      100     111
2       13      24      35      46      57      68      79      90      101     112
3       14      25      36      47      58      69      80      91      102     113
4       15      26      37      48      59      70      81      92      103     114
5       16      27      38      49      60      71      82      93      104     115
6       17      28      39      50      61      72      83      94      105     116
7       18      29      40      51      62      73      84      95      106     117
8       19      30      41      52      63      74      85      96      107     118
9       20      31      42      53      64      75      86      97      108     119
10      21      32      43      54      65      76      87      98      109     120
```

If you specify two columns apart, the macro will print the following table:

```
0       33      66      99
1       34      67      100
2       35      68      101
3       36      69      102
4       37      70      103
5       38      71      104
6       39      72      105
7       40      73      106
8       41      74      107
9       42      75      108
10      43      76      109
```

The macro prints the first, fourth, seventh and tenth columns, which are two columns apart from each other.

	A	B	C	D	E
11	PRINTALT	{LET rel206,@INFO("release")}~{RECALC loc206} {RECALC form206}			

The macro issues the `{LET rel206,@INFO("release")}~` commands, which store the result of the `@INFO("release")` function in the B36 cell named [rel206]. Later, the macro uses it to decide if you are using a 3-D or 2-D Lotus 1-2-3 release. Next the macro issues `{RECALC loc206}{RECALC form206}` to update the two formulas in the B38 cell, [loc206], and the B12 cell, [form206]. The formulas in the B38 and the B12 cells are part of the macro code which changes dynamically when the formulas are calculated. We will look into the formula in the B38 cell when we reach the explanation of the macro code.

	A	B	C	D	E
12	form206	{LET here1206,@CELLPOINTER("coord")}~			

This code is the result of the following formula:

```
12 form206          +"{LET here1206,@CELLPOINTER("""&B38&""")}~"
```

the "dynamic" part of this formula is the content of the B38 cell, [loc206], shown here:

	A	B	C	D	E
38	loc206	@IF(@LEFT(rel206,1)<>"@","coord","address")			

The formula in [loc206] depends on the content of the B36 cell, [rel206], (the content of this cell is the result of the `@INFO("release")` function). If the first character of the string in [rel206] is the "`@`" character, then you are using a 2-D version of Lotus 1-2-3, and the result of

the formula in [loc206] is:

```
38 loc206          address
```

otherwise, you are using a 3-D release of Lotus 1-2-3, and the result of the formula in [loc206] is:

```
38 loc206          coord
```

This way the macro "knows" what type of Lotus 1-2-3 you are using and adapts the code of the macro accordingly.

	A	B	C	D	E
13	!	{WINDOWSOFF}{PANELOFF}/RNCRange to print?~~/RND Range to print?~/RNC{PANELON}{WINDOWSON}Range to print?~ {BS}{BS}{?}~{WINDOWSOFF}{PANELOFF}			

The macro continues with `{WINDOWSOFF}{PANELOFF}` which freeze the screen and panel display activities. Then the macro uses the "safe technique" to create the [Range to print?] range name for the range that you select to print, which it uses as a guiding prompt.

	A	B	C	D	E
14	!	{PANELON}{GETLABEL "How many columns to skip ? ",skip206}~			

Now the macro issues `{GETLABEL "How many columns to skip ? ",skip206}~` which displays `"How many columns to skip ? "` in the panel. When you insert the number of column to skip and press ENTER, the macro stores your response in the B27 cell, [skip206].

	A	B	C	D	E
15	!	{LET temp1206,@STRING(@MOD(@COLS(Range to print?),(@VALUE(skip206)+1)),0)}~			

Next the macro issues this advanced `{LET}` command using the `@MOD`, `@COLS`, `@VALUE` and `@STRING` functions to checks if the number of columns in the [Range to print?] range is exactly divided in the number of columns to skip plus one, without a reminder. If there is a reminder, the macro stores it as a string in cell [temp1206].

	A	B	C	D	E
16	!	{GETNUMBER "No. of copies ? ",copies206}~{PANELOFF} {RECALC colshide206}			

Next the macro issues `{GETNUMBER "No. of copies ? ",copies206}~`, which displays the `"No. of copies ? "` prompt message in the panel. If you insert the number of copies to print and press ENTER, the macro stores your response in the B30 cell, [copies206]. Now the macro issues `{RECALC colshide206}`, which updates the formula in the B21 cell, [colshide206]:

```
21 colshide206     +"{RIGHT}/WCH.{RIGHT "&@STRING(@VALUE(B27)-1,0)&"}~
                   {ESC 10}{RETURN}"
```

to give the result of:

```
21 colshide206     {RIGHT}/WCH.{RIGHT 2}~{ESC 10}{RETURN}
```

The macro uses the value in [skip206] to determine how many columns to hide.

	A	B	C	D	E
17	!	{FOR counter1206,1,(@COLS(Range to print?)-@MOD(@COLS (Range to print?),(@VALUE(skip206)+1)))/(@VALUE(skip206) +1)+1,1,colshide206}{ESC 10}			

Now the macro uses a {FOR} command to execute the [colshide206] routine as many times as needed to hide the unwanted columns in the [Range to print?] range.

	A	B	C	D	E
18	!	{FOR counter2206,1,copies206,1,prints206}{ESC 10}			

Only the columns to print are displayed in the range, and the rest are hidden and the macro issues {FOR counter2206,1,copies206,1,prints206} to execute the [prints206] routine as many times as the value stored in the B30 cell, [copies206] .

	A	B	C	D	E
19	!	/WCDRange to print?~{GOTO}{here1206}~/RNDRange to print?~ {ESC 10}{CALC}			

Next the macro uses /WCDRange to print?~ to reveal the hidden columns and then issues the indirect {GOTO}{here1206}~ macro command, which sends the cell pointer to the cell address where it was located before the macro started.

What's left to analyze is the [prints206] routine which is also, a result of a string formula:

	A	B	C	D	E
23	prints206	{ONERROR err206}{IF @MOD(@COLS(Range to print?),@VALUE (skip206)+1)>0}/PPCRRRange to print?~R{LEFT}{RIGHT}~ AGPQ{ESC 10}{RETURN}			
24	!	{IF @MOD(@COLS(Range to print?),@VALUE(skip206)+1)=0} /PPCRRRange to print?~R{LEFT}{RIGHT}~AGPQ{ESC 10}{RETURN}			

This routine executes the printing job, however the printer may be off, or a printing error may occur; therefore the macro issues {ONERROR err206}, to activate the [err206] error handing routine when the next error will occur. This code, as it appears, really looks unusual. It seems that the {IF} commands is not needed because all situations are covered. However, this macro has shown erratic behavior in different versions of Lotus 1-2-3; but we could not pinpoint to the exact reason yet, for example this code creates an error and the error handling routine is activated in Lotus 1-2-3W version 1.1 for Windows, while it works smoothly in Lotus 1-2-3 version 2.4 and other Lotus versions. This is the best way we could find. The curious reader is urged to try this macro and to come up with a solution that will better fit all versions of Lotus 1-2-3.

	A	B	C	D	E
33	err206	{IF @LEFT(rel206,1)<>"@"}{BEEP}Turn on your computer and press any key to continue...{GET key206}{ESC}/PRQ /WCDA1..IV1~			
34	!	{IF @LEFT(rel206,1)="@"}{BEEP}Turn on your computer and press any key to continue...{GET key206}{ESC} /WCDA1..IV1~			

The [err206] error handing routine again uses duplicate code and the {IF} command seems to be unnecessary, but, again we show here the actual macro code that works the best in all Lotus releases. When the error routine is activated:

```
Turn on your printer and/or press any key to continue...
```

appears in the panel. You need to press any key to finish the macro. Unless the printer is not ready, the dummy error does not disable the printing job.

[6] Print a Worksheet

```
     A                B                C                D          E          F
 1 *---A macro to PRINT a worksheet
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer where you want the setup string to be
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 PRINTWKS        {LET here1140,@CELLPOINTER("address")}~
                   {MENUBRANCH menu1140}
12 !
13 menu1140        Save_&_PrNow_PrintPrint_SetMargin   View     Quit
14 !               Save sprePrint witChange thSet the rView currQuit the mac
15 !               {CALC}/FS{CALC}/PP{ESC 6}{G/PPOMR{?}/PPOS{?}~{ESC 6}
16 !               /PPCRAR{?OH*{ESC}@~{ESC 6}{{MENUBRAN{MENUBRANCH menu1140}
17 !               OH*{ESC}@{MENUBRANCH menu1140}
18 !               {MENUBRANCH menu1140}
19 !
20 string140       \027@\027\071
21 !
22 !
23 here1140        $F$13
24 !
25 setuptable1140          ************************************************
26 !                       * Point to the desired setup and press [ENTER] *
27 !                       ************************************************
28 !
29 !                       Standard CondensedEnlarged Double-Spaced
30 !               Normal   \027@    \027@\027\027@\027\027@\027A\024
31 !               Double-St\027@\027\027@\027\027@\027\027@\027A\024\027\069
32 !               Emphasize\027@\027\027@\027\027@\027\027@\027A\024\027\071
33 !               Both last\027@\027\027@\027\027@\027\027@\027A\024\027\069
34 !               Italic   \027@\027\027@\027\027@\027\027@\027A\024\0274
35 !               Underline\027@\027\027@\027\027@\027\027@\027A\024\027\045
```

This macro uses a custom menu of five menu items, and each of them in different columns, therefore the menu item's code in the B13..G18 range cannot fit on the page. If you intend to key this macro into 1-2-3, you need to key the code as it appears here for every menu item. This macro allows you to print a range using different setup strings that you can choose from a table, using point and shoot. This macro is especially built for Epson compatible printers. If you use a different type of printer you must change the setup strings in the C30..F35 range.

```
     A                B                C                D          E
11 PRINTWKS        {LET here1140,@CELLPOINTER("address")}~
                   {MENUBRANCH menu1140}
```

The macro starts with the `{LET here1140,@CELLPOINTER("address")}~` macro commands which store the current cell pointer address in the B23 cell named [here1140]. Then the macro uses `{MENUBRANCH menu1140}` to activate the print custom menu. The first menu option is [Save_&_Print]:

```
Save_&_Print
Save spreadsheet and then print (Align paper in printer before
    printing)
{CALC}/FS{?}~R{ESC}
/PPCRAR{?}~{WINDOWSOFF}{PANELOFF}
OH*{ESC}@|Page #~QGQ{PANELON}{WINDOWSON}
{MENUBRANCH menu1140}
```

The macro first issues {CALC} to re-calculate the worksheet before it saves and prints it, and then the macro issues /FS{?}~, which starts the **F**ile **S**ave process, prompts you to insert a file name or accept the default name, and press ENTER. When you press ENTER, the macro issues the "R" key and {ESC} to account for the following two optional cases: 1) where the file name is new and 2) the file name already exists. Now the macro issues /PPCRAR to start the printing procedure, clear the previous range and align the paper. Then it issues {?} to allow you to paint the range to print and press ENTER.

After the tilde "~", the macro issues {WINDOWSOFF}{PANELOFF}, freeze the screen and panel activities. Next the macro issues OH*{ESC} clearing the header in case a previous header exists. The OH code displays the header prompt in the panel, and the macro types the astrix "*" into the panel replacing the previous header if it exists, or just types the "*" into the panel. Now that the panel includes only the astrix "*" character, {ESC} is enough to clear it from the panel, resetting the header. Later, the macro issues the "@" character, which tells Lotus to print the current date into the header then the split bar "|" character informs Lotus to print the following "Page " string in the center of the line of the header and the "#" character tells Lotus to print the page number in the header.

When this process is finished, the macro uses {PANELON}{WINDOWSON} to resume the screen and panel display activities, and {MENUBRANCH menu1140} to branch back to the menu. The second menu option is [**N**ow_Print]:

```
Now_Print
Print without saving (Align paper in printer before printing)
{CALC}/PPCRAR{?}~{WINDOWSOFF}{PANELOFF}
OH*{ESC}@|Page #~QGQ{PANELON}{WINDOWSON}
{MENUBRANCH menu1140}
```

This menu option has exactly the same code as the previous menu option except for the /FS{?}~R{ESC} macro code because this menu option does not save the file before printing. The third menu option is [Print_Setup]:

```
Print_Setup
Change the GLOBAL printing setup string
{ESC 6}{GOTO}IV1~{GOTO}setuptable1140~{RIGHT}{DOWN 5}{?}~
    /PPOS {ESC}
{LET string140,@CELLPOINTER("contents")}{string140}
~{ESC 6}{GOTO}IV1~{GOTO}{here1140}~{MENUBRANCH menu1140}
```

When you select this menu option, Lotus allows you to select a setup string from a prepared table using point and shoot. The macro first issues {ESC 6} to make sure that Lotus is in the READY mode, then {GOTO}IV1~ to move the cell pointer to the IV1 cell address, and finally {GOTO}setuptable1140~ to bring the setup strings table to the upper left cornet of the screen. Then the macro issues {RIGHT}{DOWN 5}' which move the cell pointer to the first cell of the setup strings table. Next the macro issues {?} which halts the macro and waits for you to point to the setup string and press ENTER.

Then the macro issues /PPOS {ESC} to start the setup string process. Notice the space character between the /PPOS key commands and {ESC}. This space character does the same job as the astrix "*" character. It clears the previous setup string if it exists and types a space instead or just types a space if there was no previous setup string. The {ESC} command clears the panel and a new setup string can be typed to the panel.

To insert the content of the current cell (the setup string from the table) into the panel, the macro issues {LET string140,@CELLPOINTER("contents")}, which stores the content of the current cell in [string140], then {string140} routine command which injects the content of [string140] into the panel as a response to the prompt, and finally ENTER using the tilde "~". After {ESC 6} it returns the worksheet to the READY mode. Next the macro issues {GOTO}IV1~ the indirect {GOTO}{here1140}~ macro command to bring the cell pointer to the origin address, and the origin cell to the upper left corner of the screen.

Notice that {LET string140,@CELLPOINTER("contents")} and {string140} were issued while the macro was in the middle of the process to define a new setup string. This is a fine example of an advanced use of the {LET} command. To end this menu option and return to the custom menu, the macro issues {MENUBRANCH menu1140}. The fourth menu option is [**M**argin]:

```
Margin
Set the right margin
/PPOMR{?}~QQ
{MENUBRANCH menu1140}
```

This menu option allows you to set the right margin as needed. The macro issues /PPOMR and then {?}, which pauses the macro and waits until you type the new margin, and press ENTER or accept the current margin and press ENTER. When you press ENTER, the macro issues the Lotus **Q**uit menu option twice to return to READY mode and {MENUBRANCH menu1140} to branch back to the menu. The fifth menu option is [**V**iew]:

```
View
View current global printing setup string
/PPOS{?}~{ESC 6}
{MENUBRANCH menu1140}
```

This menu option allows to view and/or edit the setup string manually. The macro starts with /PPOS, issues the {?} macro command, and waits for you to view the setup string, edit it or enter a new setup string, and press ENTER. When you press ENTER, the macro issues {ESC 6} and returns the worksheet to the READY mode. Then it issues {MENUBRANCH menu1140} and branches back to the custom menu .

```
Quit
Quit the macro
```

When you select this menu option, the macro reaches an empty cell and quits.

This macro contains a table of setup strings that you can choose from using point and shoot. Because this table and the custom menu cannot fit on one page without overlapping we show a full list of all the cell contents of the macro:

```
A1: U [W15] '*---A macro to PRINT a worksheet
A2: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W15] '    range names in this column (starts with the \Z macro name)
A4: [W15] '*---Place the cell pointer where you want the setup string to be
A5: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A6: [W15] '!
A7: [W15] '!
A8: [W15] '!
A9: [W15] '!
```

```
A10: U [W15] '\Z
B10: [W14] '{BREAKON}
A11: U [W15] 'PRINTWKS
B11: [W14] '{LET here1140,@CELLPOINTER("address")}~{MENUBRANCH menu1140}
A12: [W15] '!
A13: [W15] 'menu1140
B13: [W14] 'Save_&_Print
C13: [W10] 'Now_Print
D13: [W11] 'Print_Setup
E13: [W10] 'Margin
F13: 'View
G13: 'Quit
A14: [W15] '!
B14: [W14] 'Save spreadsheet and then print (Align paper in printer before
           printing)
C14: [W10] 'Print without saving (Align paper in printer before printing)
D14: [W11] 'Change the GLOBAL printing setup string
E14: [W10] 'Set the right margin
F14: 'View current global printing setup string
G14: 'Quit the macro
A15: [W15] '!
B15: [W14] '{CALC}/FS{?}~R{ESC}
C15: [W10] '{CALC}/PPCRAR{?}~{WINDOWSOFF{PANELOFF}
D15: [W11] '{ESC 6}{GOTO}IV1~{GOTO}setuptable1140~{RIGHT}{DOWN 5}{?}~
           /PPOS {ESC}{LET string140,@CELLPOINTER("contents")}{string140}
E15: [W10] '/PPOMR{?}~QQ
F15: '/PPOS{?}~{ESC 6}
A16: [W15] '!
B16: [W14] '/PPCRAR{?}~{WINDOWSOFF{PANELOFF}
C16: [W10] 'OH*{ESC}@|Page #~QGQ{PANELON}{WINDOWSON}
D16: [W11] '~{ESC 6}{GOTO}IV1~{GOTO}{here1140}~{MENUBRANCH menu1140}
E16: [W10] '{MENUBRANCH menu1140}
F16: '{MENUBRANCH menu1140}
A17: [W15] '!
B17: [W14] 'OH*{ESC}@|Page #~QGQ{PANELON}{WINDOWSON}
C17: [W10] '{MENUBRANCH menu1140}
A18: [W15] '!
B18: [W14] '{MENUBRANCH menu1140}
A19: [W15] '!
A20: [W15] 'string140
B20: [W14] '\027@\027\071
A21: [W15] '!
A22: [W15] '!
A23: [W15] 'here1140
B23: [W14] '$F$13
A24: [W15] '!
A25: [W15] 'setuptable1140
C25: U [W10] '************************************************
A26: [W15] '!
C26: U [W10] '* Point to the desired setup and press [ENTER] *
A27: [W15] '!
C27: U [W10] '************************************************
A28: [W15] '!
A29: [W15] '!
C29: U [W10] ^Standard
D29: U [W11] ^Condensed
E29: U [W10] ^Enlarged
F29: U 'Double-Spaced
A30: [W15] '!
B30: U [W14] 'Normal
C30: [W10] '\027@
D30: [W11] '\027@\027\015
E30: [W10] '\027@\027\014
F30: '\027@\027A\024
A31: [W15] '!
B31: U [W14] 'Double-Strike
C31: [W10] '\027@\027\069
D31: [W11] '\027@\027\015\027\069
E31: [W10] '\027@\027\014\027\069
F31: '\027@\027A\024\027\069
A32: [W15] '!
```

```
B32: U [W14] 'Emphasized
C32: [W10] '\027@\027\071
    D32: [W11] '\027@\027\015\027\071
E32: [W10] '\027@\027\014\027\071
F32: '\027@\027A\024\027\071
A33: [W15] '!
B33: U [W14] 'Both last two
C33: [W10] '\027@\027\069\027\071
D33: [W11] '\027@\027\015\027\069\027\071
E33: [W10] '\027@\027\014\027\069\027\071
F33: '\027@\027A\024\027\069\027\071
A34: [W15] '!
B34: U [W14] 'Italic
C34: [W10] '\027@\0274
D34: [W11] '\027@\027\015\0274
E34: [W10] '\027@\027\014\0274
F34: '\027@\027A\024\0274
A35: [W15] '!
B35: U [W14] 'Underline
C35: [W10] '\027@\027\045\001
D35: [W11] '\027@\027\015\027\045\001
E35: [W10] '\027@\027\014\027\045\001
F35: '\027@\027A\024\027\045\001
```

[6] Print a Worksheet and Headings

```
     A                B                C                D                E
 1 *---A macro to print the worksheet including the columns and rows
 2     headings.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                    IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z              {BREAKON}
11 PRNTHEAD        {LET rel141,@INFO("release")}~{RECALC loc141}
                   {RECALC form141}
12 form141         {WINDOWSOFF}{PANELOFF}{LET here141,@CELLPOINTER("coord")}~
13 !               {IF @LEFT(rel141,1)<>"@"}/WGGE/WTC{HOME}/WIC~/WIR~
14 !               {IF @LEFT(rel141,1)="@"}/WTC{HOME}/WIC~/WIR~{ESC 10}
15 !               /RNCPaint down~~/RNDPaint down~/RNC{PANELON}{WINDOWSON}
                   Paint down~.{PGDN 2}{?}~{WINDOWSOFF}{PANELOFF}
16 !               /DFPaint down~0~1~~{RIGHT}
17 !               /RNCPaint right~~/RNDPaint right~/RNC{PANELON}{WINDOWSON}
                   Paint right~.{BIGRIGHT 2}{?}~{WINDOWSOFF}{PANELOFF}
18 !               {LET A1,'@REPEAT(" ",@CELL("width",A2..A2)/2)&@MID(@CELL
                   ("address",A2..A2),1,@FIND("$",@CELL("address",A2..A2),1)
                   -1)}~{GOTO}A1~{EDIT}{HOME}{DEL 2}~
19 !               {IF @LEFT(rel141,1)<>"@"}/C~.{RIGHT @COLS(Paint right)}~
                   /M.{END}{RIGHT}~{RIGHT}~/RV{END}{RIGHT}~~
20 !               {IF @LEFT(rel141,1)="@"}/C~.{RIGHT @COLS(Paint right)}~
                   {ESC 10}/M.{END}{RIGHT}~{RIGHT}~/RV.{END}{RIGHT}~~
21 !               {IF @LEFT(rel141,1)<>"@"}/RNCPaint down~{LEFT}~
                   /CPaint right~.{LC}{HOME}~/CPaint down~.{LC}{HOME}~/RND
                   Paint down~/RNDPaint right~/RE.{LC}{HOME}~
22 !               {WINDOWSON}{PANELON}{ESC 10}/PPCAAR{BS}.{?}~
23 !               {IF @LEFT(rel141,1)<>"@"}{WINDOWSOFF}{PANELOFF}GPQ{HOME}
                    /WDC~/WDR~{GOTO}{here141}~/WGGD
24 !               {IF @LEFT(rel141,1)="@"}{WINDOWSOFF}{PANELOFF}GPQ{HOME}
                   /WDC~/WDR~{GOTO}{here141}~
25 !
26 here141         $A:$A$1
27 !
28 rel141          3.00.00
29 !
30 loc141          coord
```

Notice that the code in the B12 and the B30 cells is the result of the following dynamic string formulas:

```
12 form141         +"{WINDOWSOFF}{PANELOFF}{LET here141,@CELLPOINTER
                   (""""&B30&""")}~"
30 loc141          @IF(@LEFT(B28,1)<>"@","coord","address")
```

Using this macro, you can also print a range with rows and columns headings.

```
     A                B                C                D                E
11 PRNTHEAD        {LET rel141,@INFO("release")}~{RECALC loc141}
                   {RECALC form141}
```

The macro starts with the `{LET rel141,@INFO("release")}~` macro commands to store the result of the `@INFO("release")` function in the B28 cell named [rel141]. Later, the macro uses it to detect if you are using a 3-D or a 2-D Lotus release. Next the macro issues `{RECALC loc141}{RECALC form141}` to update the string formulas in the B30 cell, [loc141] and the

B12 cell, [form141].

```
12 form141          +"{WINDOWSOFF}{PANELOFF}{LET here141,@CELLPOINTER
                    (""""&B30&""")}~"
```

The formula in the B12 cell uses the content of [loc141] to create the correct code for the macro while the formula in the B30 cell

```
30 loc141           @IF(@LEFT(B28,1)<>"@","coord","address")
```

uses the content of cell [rel141], which holds the results of the `@INFO("release")` function. The B30 cell can have one of two results: if you are using a 3-D Lotus release, the result of the formula is the "`coord`" string, otherwise the result is the "`address`" string. Therefore the code in the B12 cell can look:

	A	B	C	D	E
12	form141	{WINDOWSOFF}{PANELOFF}{LET here141,@CELLPOINTER("coord")}~			

or:

	A	B	C	D	E
12	form141	{WINDOWSOFF}{PANELOFF}{LET here141,@CELLPOINTER("address")}~			

The code in the B13 cell is:

	A	B	C	D	E
13	!	{IF @LEFT(rel141,1)<>"@"}/WGGE/WTC{HOME}/WIC~/WIR~			

This effective if you are using a 3-D Lotus release like Lotus 1-2-3 version 3.1. The macro enables the **G**lobal **G**roup option to give all the sheets the same format as the first sheet using the `/WGGE` macro keys. You may change this command may if you think it's dangerous to use. The macro clears any titles using `/WTC` and sends the cell pointer to the A1 cell using `{HOME}`, where the macro inserts a row and a column to make a place for the row and column headings using the `/WIC~/WIR~` macro keys.

	A	B	C	D	E
14	!	{IF @LEFT(rel141,1)="@"}/WTC{HOME}/WIC~/WIR~{ESC 10}			

The code in B14 is effective if you are using a 2-D Lotus release like Lotus 1-2-3 version 2.4. Therefore the macro clears the titles using `/WTC` and sends the cell pointer to the A1 cell using `{HOME}`. There the macro issues `/WIC~/WIR~` to insert a row and a column and to make a place for the row and the column headings.

	A	B	C	D	E
15	!	/RNCPaint down~~/RNDPaint down~/RNC{PANELON}{WINDOWSON} Paint down~.{PGDN 2}{?}~{WINDOWSOFF}{PANELOFF}			

Here the macro uses the "safe technique" to assign the [Paint down] name to the column of row headings and uses the range name as a prompt message. Simultaneously the macro issues `{PGDN 2}` to paint a column two pages long to fill with the row heading numbers, and `{?}`, which halts the macro and until you accept or change the suggested painted column and press ENTER. Then the macro issues the tilde "~" and finishes assigning the column range for the row headings.

	A	B	C	D	E
16	!	/DFPaint down~0~1~~{RIGHT}			

Next the macro issues `/DFPaint down~0~1~~` to fill the column range named [Paint down] with numbers, starting with "0", in step increments of "1", and then moves the cell pointer to B1 using `{RIGHT}`.

	A	B	C	D	E
17	!	/RNCPaint right~~/RNDPaint right~/RNC{PANELON}{WINDOWSON} Paint right~.{BIGRIGHT 2}{?}~{WINDOWSOFF}{PANELOFF}			

Again the macro uses the safe technique to assign the [Paint right] range name to the row of headings, which acts as a prompt. Then it issues `{BIGRIGHT 2}` to paint a row two pages wide to fill with the column heading letters, and `{?}`, which halts the macro until you accept or change the suggested painted row and press ENTER. Then the macro issues the tilde "~" and finishes assigning the row range for the row headings.

	A	B	C	D	E
18	!	{LET A1,'@REPEAT(" ",@CELL("width",A2..A2)/2)&@MID(@CELL ("address",A2..A2),1,@FIND("$",@CELL("address",A2..A2),1) -1)}~{GOTO}A1~{EDIT}{HOME}{DEL 2}~			

The `{LET}` command inserts a `@REPEAT` formula in the A1 cell but as a text with the apostrophe "'" character as the first character. If we omit the apostrophe "'", the result of the formula as a text will appear in the A1 cell instead. Because we need an active formula in the A1 cell, a formula that we can copy across the row to create the letter headings for the columns of the range, we had to add this apostrophe to the formula's text. After the `{LET}` command the A1 contains the following text.

```
''@REPEAT(" ",@CELL("width",A2..A2)/2)&@MID(@CELL("address",A2..A2)
   ,1,@FIND("$",@CELL("address",A2..A2),1)-1)
```

Note the two apostrophes "''". The second apostrophe was entered by the `{LET}` command. Next the macro issues `{GOTO}A1~`, which moves the cell pointer to A1, and `{EDIT}` to enter the EDIT mode. The `{HOME}` command moves the cell pointer to the first apostrophe in the panel. Now the macro issues `{DEL 2}~`, which deletes the two apostrophes and writes the formula back to A1. The formula, itself, is worth studying, we recommend that you type the formula into the A1 and copy it across the row at least to the "AA" column to see the results. The formula returns the column's characters heading in a centered position.

	A	B	C	D	E
19	!	{IF @LEFT(rel141,1)<>"@"}/C~.{RIGHT @COLS(Paint right)}~ /M.{END}{RIGHT}~{RIGHT}~/RV{END}{RIGHT}~~			

Now the macro issues `{IF @LEFT(rel141,1)<>"@"}`, which checks the first character of the content of cell [rel141]. If the condition is TRUE, you use a 3-D Lotus release and the macro issues `/C~.{RIGHT @COLS(Paint right)}~` to copy the formula in the A1 cell all the way to the right to all the cells of range [Paint right]. Notice `{RIGHT @COLS(Paint right)}` that uses the `@COLS(Paint right)` formula to determine how many cells to expand the range. To place the headers on the right columns, the macro issues `/M.{END} {RIGHT}~{RIGHT}~` which moves the [Paint right] range one cell to the right, and finally `/RV.{END}{RIGHT}~~` to turn all the formulas into values.

	A	B	C	D	E

```
20 !              {IF @LEFT(rel141,1)="@"}/C~.{RIGHT @COLS(Paint right)}~
                  {ESC 10}/M.{END}{RIGHT}~{RIGHT}~/RV.{END}{RIGHT}~~
```

Here the macro issues `{IF @LEFT(rel141,1)="@"}` to check if you are using a 2-D Lotus release. The rest of the code is exactly the same as for the 3-D release. However, the added `{ESC 10}` helps some 2-D releases of Lotus 1-2-3 work correctly. This is another example of incompatibility between different releases of Lotus 1-2-3.

```
     A              B                C              D              E
21 !              {IF @LEFT(rel141,1)<>"@"}/RNCPaint down~{LEFT}~
                  /CPaint right~.{LC}{HOME}~/CPaint down~.{LC}{HOME}~/RND
                  Paint down~/RNDPaint right~/RE.{LC}{HOME}~
```

The macro continues with `{IF @LEFT(rel141,1)<>"@"}`, which checks if you are using a 3-D release of 1-2-3. If so, the macro issues:

```
/RNCPaint down~{LEFT}~/CPaint right~.{LC}{HOME}~/CPaint down~.{LC}
    {HOME}~
```

which expand the [Paint down] and [Paint right] ranges to all the sheets in the worksheet.

Note: The `{LC}` command or the `{LASTCELL}` sends the cell pointer to the last cell in the worksheet. This is like a combination of the `{LASTSHEET}` and `{END}{HOME}` macro commands.

Next the macro issues `/RNDPaint down~/RNDPaint right~` to delete the [Paint down] and [Paint right] temporary range names, and `/RE.{LC}{HOME}~` to erase the content of the A1 cell in all the sheets of the range.

```
     A              B                C              D              E
22 !              {WINDOWSON}{PANELON}{ESC 10}/PPCAAR{BS}.{?}~
```

Now the macro issues `{WINDOWSON}{PANELON}` to freeze the screen and panel display activities and `{ESC 10}` to return the worksheet to the READY mode. Next the macro issues `/PPCAAR{BS}.{?}~` and starts the printing process. The `{?}` halts the macro and waits until you paint or type the range to print. When you paint the range starting from the A1 cell to include the column and row headings and press ENTER, the macro reaches the following code:

```
     A              B                C              D              E
23 !              {IF @LEFT(rel141,1)<>"@"}{WINDOWSOFF}{PANELOFF}GPQ{HOME}
                   /WDC~/WDR~{GOTO}{here141}~/WGGD
```

In this cell of code the macro issues `{IF @LEFT(rel141,1)<>"@"}` to check if you are using a 3-D release, if so, the macro uses `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities and continues to execute the print job using the "`GPQ`" macro keys. Next the macro uses `{HOME}` to move the cell pointer to A1 and `/WDC~/WDR~` to delete the row and column of headings. Last, the macro issues the `{GOTO}{here141}~` indirect macro command, which returns the cell pointer to its point of origin. Last the macro issues `/WGGD` to disable the **G**lobal **G**roup mode and leave a clean worksheet.

```
     A              B                C              D              E
24 !              {IF @LEFT(rel141,1)="@"}{WINDOWSOFF}{PANELOFF}GPQ{HOME}
                  /WDC~/WDR~{GOTO}{here141}~
```

In this cell of code the macro uses `{IF @LEFT(rel141,1)="@"}` to check if you are using a 2-D Lotus release. If so, the macro uses the same code as in the case of the 3-D Lotus release, except for the `/WGGD` macro keys that are irrelevant in this case. This macro is useful when you want to print a worksheet beginning with the A1 cell. For an all purpose macro which can print any range with column and row headings, see the FRAMERNG.WK1 macro.

[6] Measure Range Width before Printing

```
     A              B                C                D              E
 1 *---A macro to measure range width ( before printing for example )
 2 *---Use the /Range Name Label Down [ENTER] to define the \Z macro name
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column ( starts with RANGWIDE )
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 RANGWIDE       {WINDOWSOFF}{PANELOFF}/RNCMeasure range ?~~/RND
                  Measure range ?~/RNC{PANELON}Measure range ?~{WINDOWSON}
                  {?}~
12 !              {WINDOWSOFF}{PANELOFF}{GOTO}Measure range ?~
13 !              {LET total031,0}~
14 !              {FOR counter031,1,@COLS(Measure range ?),1,sum031}
15 !              {LET message031,+"The range width is "&@STRING(total031,0)
                  &" characters long, Press [ESC] to Quit. "}~{PANELON}
                  {message031}{GET key031}{ESC}/RNDMeasure range ?~
16 !
17 sum031         {LET total031,total031+@CELLPOINTER("width")}~
18 !              {RIGHT}
19 !
20 counter031                  8
21 !
22 message031     The range width is 68 characters long, Press [ESC] to Quit.
23 !
24 total031                   68
25 !
26 key031         {ESC}
```

This macro solves the problem of measuring the width of a range before printing. If you have ever tried to print a range into a file or to the printer from Lotus 1-2-3, you had to cope with the question of: how wide is the painted range or will it fit in the printer's page width? This macro allows you to calculate the range width before printing it.

```
     A              B                C                D              E
11 RANGWIDE       {WINDOWSOFF}{PANELOFF}/RNCMeasure range ?~~/RND
                  Measure range ?~/RNC{PANELON}Measure range ?~{WINDOWSON}
                  {?}~
12 !              {WINDOWSOFF}{PANELOFF}{GOTO}Measure range ?~
13 !              {LET total031,0}~
14 !              {FOR counter031,1,@COLS(Measure range ?),1,sum031}
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the screen and panel display activities. Then the macro uses the "safe technique" to prompt you to paint the range to process, and assigns the [Measure range ?] name to the same range, which acts as a prompt. Next the macro issues `{GOTO}Measure range ?~`, which moves the cell pointer to the upper left cell of the [Measure range ?] range. The macro continues with `{LET total031,0}~` to reset the value in the B24 cell named [total031] to zero, which holds the total width of the range to print. To calculate the width of the [Measure range ?] range the macro issues

```
{FOR counter031,1,@COLS(Measure range ?),1,sum031}
```

This command executes the [sum031] routine as many times as the number of columns of the [Measure range ?] range.

	A	B	C	D	E
17	sum031	{LET total031,total031+@CELLPOINTER("width")}~			
18	!	{RIGHT}			

The [sum031] routine adds the column width of each column in the [Measure range ?] range to cell [total031]. The macro starts when the cell pointer is on the upper left cell of the [Measure range ?] range. Then it issues `{LET total031,total031+@CELLPOINTER ("width")}`, which adds the width of the first column of the [Measure range ?] range to [total031]. Then the macro issues `{RIGHT}`, which moves the cell pointer to the upper cell of the second column and returns to the `{FOR}` loop, which activates the [sum031] routine again. This time the macro adds the width of the second column to [total031]. When the `{FOR}` loop command is finished, [total031] holds the sum of all the columns widths in the [Measure range ?] range, which is the width of the range to print.

	A	B	C	D	E
15	!	{LET message031,+"The range width is "&@STRING(total031,0) &" characters long, Press [ESC] to Quit. "}~{PANELON} {message031}{GET key031}{ESC}/RNDMeasure range ?~			

This is the interesting part of the macro. The cell named [total031] already contains the range width, however the macro still has to display it to the user. Therefore the macro issues:

```
{LET message031,+"The range width is "&@STRING(total031,0)&
    " characters long, Press [ESC] to Quit. "}~
```

which adds the "`The range width is `" string to the `@STRING(total031,0)` function changing the value in [total031] into a string format and then adds "` characters long, Press [ESC] to Quit. `" string to create the complete message to the user. To better understand the code, let's assume that the printed range has eight columns with a total width of 68 characters. Therefore the `@STRING(total031,0)` returns the "`68`" string. Therefore the `{LET}` macro command creates the

```
The range width is 68 characters long, Press [ESC] to Quit.
```

string and stores it in cell [message031]. Here comes the trick! The macro issues `{PANELON}` to resume the panel display activity and then `{message031}`, which injects the content of the [message031] routine into the panel because the routine contains only text without any Lotus command. However the [message031] routine holds:

```
The range width is 68 characters long, Press [ESC] to Quit.
```

Therefore `{message031}` writes the prompt message into the panel. To allow you to read the message, the macro issues `{GET key031}`, which halts the macro execution and waits until you press a key. Normally, when you press a key, Lotus stores the key's macro code in cell [key031], but this time we do not need to use this code. The macro uses `{GET key031}` only to allow you to read the message with the range width. When you press a key, the macro issues `{ESC}`, which clears the message from the panel before Lotus writes it into the current cell. This is one of the finest examples of how we can manipulate the text and formulas in the panel to create an effective and professional applications in Lotus 1-2-3. Combine it with the advanced use of `{LET}`, and you get the ultimate level of Lotus macro language programming technique. Last, the macro issues `/RNDMeasure range ?~` to delete the temporary [Measure range ?] range name and leave a clean worksheet.

[1] Insert a Print Footer

```
     A                B                C                D                E
 1 *---A macro to prompt the user for a PRINT FOOTER
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 FOOTER         /{PANELOFF}PPO{PANELON}F{?}~QQ
```

This macro simplifies the process of assigning a printing footer, and saves a few key strokes every time you need to re-assign a new footer for printing. The macro starts with the slash "/" macro key and then issues the {PANELOFF} command to suppress the panel display activity. Then it issues the PPO macro keys to start the process, and {PANELON}F to resume the panel display activity and allow you to insert the new footer. The macro issues {?}, which halts the macro, and waits until you insert the new footer, and press ENTER. After pressing ENTER, the macro issues the ~QQ macro keys to save the footer and exit to the READY mode. Notice that the macro issues {PANELON} just before it issues the "F" key. This way the macro resumes panel display activity allowing Lotus to display the "Enter footer:" prompt.

[1] Insert a Print Header

```
     A                B                C                D                E
 1 *---A macro to prompt the user for a PRINT HEADER
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 HEADER         /{PANELOFF}PPO{PANELON}H{?}~QQ
```

This macro simplifies the process of assigning a printing header, and saves a few key strokes every time you need to re-assign a new header for printing. The macro starts with the slash "`/`" macro key and then issues `{PANELOFF`, to suppress the panel display activity. Then it issues the `PPO` macro keys to start the process, and `{PANELON}H` to resume the panel display activity and allow you to insert the new header. The macro issues `{?}`, which halts the macro, and waits until you insert the new header, and press ENTER. After pressing ENTER, the macro issues the `~QQ` macro keys to save the header and exit to the READY mode. Notice that the macro issues `{PANELON}` just before it issues the "H" key. This way the macro resumes panel display activity allowing Lotus to display the "`Enter Header:`" prompt.

[1] Print Cells Content as Displayed

	A	B	C	D	E
1	*---A macro to print cells as displayed to the printer				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer at the upper leftmost cell of the print range				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	PRTDISPL	{WINDOWSOFF}{PANELOFF}/PP{PANELON}{WINDOWSON}R{?}~			
		{PANELOFF}OOAQGQ			

This macro allows you to print the contents of a range of cells as displayed. The macro starts with the {WINDOWSOFF}{PANELOFF} commands to suspend screen and panel activities. Next the macro issues the /PP macro keys and then issues {PANELON}{WINDOWSON} to resume screen and panel activities. Now the macro issues the R{?} macro commands to allow you to paint the range to print. When you finish and press ENTER, the macro issues the tilde "~" and again issues {PANELOFF} to freeze the panel activity while the macro continues to execute the OOAQGQ macro keys to print the range and exit to the READY mode.

[1] Print Cells Content as Formulas

```
     A                B                C                D                E
 1 *---A macro to print cell formulas to the printer
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer at the upper leftmost cell of the print range
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 PRTFORMU       {WINDOWSOFF}{PANELOFF}/PP{PANELON}{WINDOWSON}R{?}~
                  {PANELOFF}OOCQGQ
```

This macro allows you to print the contents of a range of cells as formulas. The macro starts with the `{WINDOWSOFF}{PANELOFF}` commands to suspend screen and panel activities. Next the macro issues the `/PP` macro keys and `{PANELON}{WINDOWSON}` to resume screen and panel activities. Now the macro issues `R{?}` to allow you to paint the range to print. When you finish to paint the range and press ENTER, the macro issues the tilde "`~`" and again `{PANELOFF}` to freeze the panel display activity while the macro continues to execute the `OOCQGQ` macro keys to print the range and exit to the READY mode.

[1] Print a Range

	A	B	C	D	E
1	*---A macro to print SELECTED RANGE				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer at the upper leftmost cell of the range				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	PRTRANGE	/PPR{NAME}{NAME}{?}~AGQ			

This macro is used to print a range to the printer. The macro displays a full screen list of all the range names in the worksheet. The macro starts with the standard `/PPR` printing commands and then issues the `{NAME}{NAME}{?}` commands which display a full screen list of all range names in the worksheet and waits for your input. If the range to print has a range name, use the direction keys to highlight the range name and press ENTER. The macro then issues the tilde "`~`" and finishes the printing process with the `AGQ` macro keys.

[0] Page Align

	A	B	C	D	E
1	*---A macro to ALIGN the paper before printing				
2	*---Use the /Range Name Label Down [ENTER] to define the \Z macro name				
3	*---Hold the [ALT] key and press [Z] to activate the macro				
4	!				
5	!				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	ALIGNPG	/PPAQ			

This is a simple five key macro to align a page before printing, it saves three key strokes every time you use it. This macro executes the **/ P**rint **P**rinter **A**lign **Q**uit Lotus menu commands.

[7] Print Form Letters from an Address Database

```
     A                B                C                D                E
 1 *---A macro to print FORM LETTERS from an address database. The macro
 2     compensates for M.I. and missing COMPANY NAME.
 3 *---Type your form letter right to the LETTER range name, the macro
 4     will take care of the date, address, etc.
 5 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 6     range names in this column (starts with the \Z macro name)
 7 *---Place the highlight on the upper left cell of the labels list
 8 *---Hold the [ALT] key and press [Z] to activate the macro
 9 !
10 \Z              {BREAKON}
11 FORMLETR        {LET rel110,@INFO("release")}~{RECALC add1110}
                   {RECALC add2110}
12 add1110         {LET here1110,@CELLPOINTER("coord")}~{BEEP}
13 !               {GETLABEL "Are you using WYSIWYG (Y/N)? N  ",wys110}~
14 !               Highlight the labels list USING THE ARROW KEYS and press
                   [ENTER]: {GET key1110}{ESC 6}~{PANELOFF}{WINDOWSOFF}
                   /RNClabels~~/RNDlabels~/RNC{PANELON}{WINDOWSON}labels~
                   {key1110}{?}~
15 !               {GOTO}now1110~/WCS72~{BEEP}Highlight the letter range
                   USING THE ARROW KEYS and press [ENTER]:
16 !               {GET key1110}{ESC 6}~{WINDOWSOFF}{PANELOFF}/RNCletter110~~
                   /RNDletter110~/RNC{PANELON}{WINDOWSON}letter110~{key1110}
                   {?}~{WINDOWSOFF}{PANELOFF}{GOTO}line9110~{DOWN 4}{LEFT}
                   /C~.{DOWN @ROWS(letter110)-14}~{GOTO}now1110~
17 !               /WCS72~{GOTO}labels~
18 !               {FOR counter110,1,@ROWS(labels),1,loop1110}
19 !               {GOTO}now1110~/WCS9~{GOTO}labels~/RNDlabels~
20 !
21 loop1110        {WINDOWSOFF}{PANELOFF}{LET row1110,@CELLPOINTER("row")
                   -@CELL("row",labels)}~
22 !               {LET line1110,@INDEX(labels,0,row1110)&" "&@INDEX(labels,
                   1,row1110)&@IF(@ISSTRING(@INDEX(labels,2,row1110)),""&
                   @INDEX(labels,2,row1110),"")&" "&@INDEX(labels,3,row1110)}
23 !               {LET line2110,@IF(@ISSTRING(@INDEX(labels,4,row1110))=0,
                   "",@INDEX(labels,4,row1110))}~
24 !               {LET line3110,@IF(@ISSTRING(@INDEX(labels,5,row1110))=0,
                   "",@INDEX(labels,5,row1110))}~
25 !               {LET line4110,@IF(@ISSTRING(@INDEX(labels,6,row1110))=0,
                   "",@INDEX(labels,6,row1110))&","&@IF(@ISSTRING(@INDEX
                   (labels,7,row1110))=0,"","  "&@INDEX(labels,7,row1110))&"
                   "&@IF(@ISSTRING(@INDEX(labels,8,row1110))=0,"",@INDEX
                   (labels,8,row1110))}
26 !               {LET line5110,@IF(@ISSTRING(@INDEX(labels,9,row1110))=0,
                   "",@INDEX(labels,9,row1110))}~
27 !               {LET line6110," "}~{LET line7110," "}~
28 !               {LET line9110,+"Dear "&@INDEX(labels,0,row1110)&" "&@IF
                   (@INDEX(labels,3,row1110)="","",@INDEX(labels,3,row1110))
                   &":"}~{CALC}
29 add2110         {LET here1110,@CELLPOINTER("coord")}~
30 !               {IF @ISSTRING(@INDEX(labels,4,row1110))=0}{GOTO}line3110~
                   /M.{DOWN @ROWS(letter110)-4}~{UP}~{GOTO}{here1110}~
31 !               {RECALC ver110}
32 ver110          :PRSletter110~G
33 !               {IF @ISSTRING(@INDEX(labels,4,row1110))=0}{GOTO}line1110~
                   /M.{DOWN @ROWS(letter110)-4}~{DOWN}~
34 !               {GOTO}now1110~{LEFT}{DOWN}/RNLR{DOWN 10}~{GOTO}{here1110}
                   ~{DOWN}{WINDOWSON}{WINDOWSOFF}
35 !
36 counter110             4
37 wys110          y
38 rel110          3.00.00
39 !
40 row1110                2
41 !
42 here1110        $B:$A$4
43 !
```

```
44 key1110        {PGDN}
45 !
46 form110
47 now1110        February 1, 1993
48 !
49 !
50 line1110       Mrs. Ann Bradley
51 line2110       HARRIS CO.
52 line3110       234 Main St.
53 line4110       Newton, MA 02005
54 line5110       U.S.A.
55 line6110
56 line7110
57 line9110       Dear Mrs. Bradley:
58 !
59 !
60 letter110      Start your form letter here. The macro leaves two empty
                  lines between
61 !              the date and the address, between the address and
                  salution line and
62 !              between the salution line and the letter's body.
63 !
64 !
65 !
66 !
67 !
68 !
69 !
70 !
71 !
72 !
73 !
74 !              Sincerely,
75 !
76 !
77 !              Macro user
78 !
```

This macro is a complementary macro to the MAILLABL.WK1 macro. It prints a standard form letter with the address, the date and the salutation line using the address records in a database. You need to type the form letter in the reserved place which starts at the B60 cell, [letter110], in this macro. You type only the main body of the letter and the macro adds the address, the date and the salutation line. The macro expects the database to be in the following format:

	A	B	C	D	E	F	G	H	I	J
1	TITLE	FNAME	M.I.	LNAME	COMPANY	ADDRESS	CITY	STATE	ZIP	COUNTRY
2	Mr.	Robert	C	Lane	Parker co	111 Grele	Portla	OR	97777	U.S.A.
3	Mrs.	Ann	S	Brad	HARRIS CO	234 Main	Newton	MA	02005	U.S.A.

The fields should contain labels only. You can use the ADDDATA.WK1 macro to easily fill such a database. The M.I., COMPANY and COUNTRY fields are optional and can be left empty. The macro uses the records in the database to print the form letters.

Notice that the codes in the B12, B29, B32 and B47 cells are the results of dynamic string formulas. If you intend to key this macro into Lotus 1-2-3, you have to key the code of the following formulas in these cells, NOT the code as they appear in the main macro listing above.

```
12 add1110        @IF(@LEFT(B38,1)<>"@","{LET here1110,@CELLPOINTER
                  (""coord"")}~{BEEP}","{LET here1110,@CELLPOINTER
                  (""address"")}~{BEEP}")
29 add2110        @IF(@LEFT(B38,1)<>"@","{LET here1110,@CELLPOINTER
                  (""coord"")}~","{LET here1110,@CELLPOINTER
                  (""address"")}~")
32 ver110         @IF(@LEFT(B38,1)="1","{ALT}FPletter110~",@IF(@UPPER
                  (B37)="Y",":PRSletter110~G","{WINDOWSOFF}{PANELOFF}
```

```
                  /PPCAARletter110~OOFQGPQ{ESC 6}"))
47 now1110        @CHOOSE(@MONTH(@NOW)-1,"January","February","March",
                  "April","May","June","July","August","September",
                  "October","November","December")&" "&@STRING(@DAY(@NOW),
                  0)&","&@STRING(@YEAR(@NOW)+1900,0)
```

Because this macro has a few complicated formulas and codes, we include a list of all the cell contents at the end of this section to make it easier for you to key this macro into 1-2-3.

```
     A            B                C          D          E
11 FORMLETR       {LET rel110,@INFO("release")}~{RECALC add1110}
                  {RECALC add2110}
12 add1110        {LET here1110,@CELLPOINTER("coord")}~{BEEP}
13 !              {GETLABEL "Are you using WYSIWYG (Y/N)? N  ",wys110}~
```

The macro starts with the `{LET rel110,@INFO("release")}~` macro commands which store the result of the `@INFO("release")` function in the B38 cell, [rel110]. Later, the macro uses this information to check if you are using a 2-D or a 3-D Lotus release. Next the macro issues `{RECALC add1110}{RECALC add2110}`, which recalculate and update the formulas in the B12 cell, [add1110], and the B29 cell, [add2110]. Before we continue let's look at the first formula:

```
12 add1110        @IF(@LEFT(B38,1)<>"@","{LET here1110,@CELLPOINTER
                  (""coord"")}~{BEEP}","{LET here1110,@CELLPOINTER
                  (""address"")}~{BEEP}")
```

This formula returns `{LET here1110,@CELLPOINTER("coord")}~{BEEP}` code if you are using a 2-D Lotus release or `{LET here1110,@CELLPOINTER("address")}~{BEEP}` code if you are using a 3-D Lotus release. This a good example of how to use dynamic code in Lotus macro language. The macro checks the first character of the content of the B38 cell, [rel110]. If the left character is not the "`@`" character, you are using a 3-D Lotus release, otherwise you are using a 2-D release. No matter what the result of the formula, `{LET here1110,@CELLPOINTER("location")}~` store the current cell pointer location in the B44 cell, [here1110]. Next the macro issues `{GETLABEL "Are you using WYSIWYG (Y/N)? N ",wys110}~`, which display "`Are you using WYSIWYG (Y/N)? N  `" in the panel.

The macro is built so that, if you use the TEXT mode, WYSIWYG is not active or is not attached, or the Lotus release does not support WYSIWYG, then the macro uses the standard Lotus printing scheme in the unformatted mode. However, if you are using Lotus for Windows or if Lotus is in the WYSIWYG mode, the macro will print the form letter in the format and fonts that you previously set before the macro was activated. You may ask why we do not use one of the `@ISAPP` functions to automatically check if the WYSIWYG is active and skip the prompt message? The reason is simple: first, even if WYSIWYG is attached, it does not mean that it is invoked; second, even if it is invoked, you may still want to print only a simple form letter without any fancy fonts; therefore this message prompt allows you to make a decision.

When you answer the prompt, Lotus stores your answer in the B35 cell, [wys110] for later use.

```
     A            B                C          D          E
14 !              Highlight the labels list USING THE ARROW KEYS and press
                  [ENTER]: {GET key1110}{ESC 6}~{PANELOFF}{WINDOWSOFF}
                  /RNClabels~~/RNDlabels~/RNC{PANELON}{WINDOWSON}labels~
                  {key1110}{?}~
```

Now the macro displays the following prompt in the panel

```
Highlight the labels list USING THE ARROW KEYS and press [ENTER]:
```

Because the macro processor does not find any Lotus command in this text, it just writes it into the panel and immediately issues {GET key1100}, which pauses the macro and waits until you press a key. That key's macro code is stored in the B42 cell, [key1100]. The macro issues {ESC 6} to clear the panel and return to the READY mode before Lotus writes the message into the current cell and alters the worksheet. Next the macro issues {PANELOFF} {WINDOWSOFF} to freeze the screen and panel display activities, while it uses the "safe technique" to start the process of assigning the [labels] range name to the database range of records that you paint, which acts as a prompt.

Now the macro uses the key you pressed earlier, which is stored in [key1100] by issuing {key1110}. However, this only contains the key that you pressed, therefore the routine command executes this key. For example: if you pressed the DOWN arrow key to start to paint the database range, the macro stored {DOWN} in [key1100]. Therefore when the macro issues the {key1100} routine command, it executes {DOWN} which moves the cell pointer one cell down exactly, which is what you wanted when you pressed the DOWN key. It seems that Lotus just carried what you pressed from the keyboard, but the truth is that the macro already started the process of assigning the [labels] range name to the database range before it issued {DOWN}.

The macro continues with {?} to allow you to paint the database range, press ENTER so the macro issues the tilde "~", and finish assigning the [labels] range name to the database range. In this piece of code, the macro assigned the [labels] range name to the database while you highlight it. The process looks like regular range painting, but the macro gives the range a name that will later be used by it. This process is equivalent to the trick to find the address of a range painted by the user from within a macro.

	A	B	C	D	E
15	!	{GOTO}now1110~/WCS72~{BEEP}Highlight the letter range USING THE ARROW KEYS and press [ENTER]:			
16	!	{GET key1110}{ESC 6}~{WINDOWSOFF}{PANELOFF}/RNCletter110~~ /RNDletter110~/RNC{PANELON}{WINDOWSON}letter110~{key1110} {?}~{WINDOWSOFF}{PANELOFF}{GOTO}line9110~{DOWN 4}{LEFT} /C~.{DOWN @ROWS(letter110)-14}~{GOTO}now1110~			

Now the macro uses {GOTO}now1110~, which moves the cell pointer to the B47 cell, [now1110], which contains the current date, and /WCS72~ which set the column width to 72. Next the macro writes the following prompt

```
Highlight the letter range USING THE ARROW KEYS and press [ENTER]:
```

into the panel and issues {GET key1110}, pausing the macro to wait until you press a key. The key's macro code is stored in [key1100]. The macro issues {ESC 6} to clear the panel and to return to the READY mode before Lotus writes the message into the current cell and alters the worksheet. Next the macro issues {PANELOFF}{WINDOWSOFF} to freeze screen and panel display activities while it uses the "safe technique" to start the process to assign the [letter110] range name to the range containing the body of the form letter. Then the macro issues {key1110}, which executes the key you pressed and {?} to allow you to finish painting the range containing the form letter (for example: the B47..B77 range in the current macro list). When you finish painting the range and press ENTER, the macro issues the tilde "~" and finishes the process.

Next the macro issues {WINDOWSOFF}{PANELOFF} to freeze the screen and panel display activities, {GOTO}line9110~, which moves the cell pointer to the B57 cell, [line9110], which contains the salutation line, and then {DOWN 4}{LEFT} which move the cell pointer to the A61 cell containing the exclamation mark "!". Now the macro uses /C~.{DOWN @ROWS (letter110)-14}~ which copy the exclamation mark "!" in the A61 cell down as many lines/rows in the form letter minus fourteen, because the date, address, salutation line, two empty lines between the date and the address, two empty lines between the address and the salutation line and the line in the B60 cell are a total of fourteen lines/rows. This way, the macro makes sure that the "A" column of the macro is contiguous up to the same length of the form letter.

Next the macro issues {GOTO}now1110~, which moves the cell pointer back to the top cell of the form letter [now1110]. The process of copying the exclamation mark "!" is not necessary for the macro, if it is being used manually, but it is needed when the macro is used from the macro manager because the manager uses the contiguous first column of the macro to automatically erase the macro when it is finished.

	A	B	C	D	E
17	!	/WCS72~{GOTO}labels~			
18	!	{FOR counter110,1,@ROWS(labels),1,loop1110}			
19	!	{GOTO}now1110~/WCS9~{GOTO}labels~/RNDlabels~			

The macro again issues /WCS72~ to make sure that the column width is 72 characters wide and then {GOTO}labels~ to move the cell pointer to the address database. Now the macro issues {FOR counter110,1,@ROWS(labels),1,loop1110}, which executes the [loop1110] routine as many times as the number of rows/records in the [labels] database.

	A	B	C	D	E
21	loop1110	{WINDOWSOFF}{PANELOFF}{LET row1110,@CELLPOINTER("row") -@CELL("row",labels)}~			
22	!	{LET line1110,@INDEX(labels,0,row1110)&" "&@INDEX(labels, 1,row1110)&@IF(@ISSTRING(@INDEX(labels,2,row1110)),""& @INDEX(labels,2,row1110),"")&" "&@INDEX(labels,3,row1110)}			
23	!	{LET line2110,@IF(@ISSTRING(@INDEX(labels,4,row1110))=0, "",@INDEX(labels,4,row1110))}~			
24	!	{LET line3110,@IF(@ISSTRING(@INDEX(labels,5,row1110))=0, "",@INDEX(labels,5,row1110))}~			
25	!	{LET line4110,@IF(@ISSTRING(@INDEX(labels,6,row1110))=0, "",@INDEX(labels,6,row1110))&","&@IF(@ISSTRING(@INDEX (labels,7,row1110))=0,"","&@INDEX(labels,7,row1110))&" "&@IF(@ISSTRING(@INDEX(labels,8,row1110))=0,"",@INDEX (labels,8,row1110))}			
26	!	{LET line5110,@IF(@ISSTRING(@INDEX(labels,9,row1110))=0, "",@INDEX(labels,9,row1110))}~			
27	!	{LET line6110," "}~{LET line7110," "}~			
28	!	{LET line9110,+"Dear "&@INDEX(labels,0,row1110)&" "&@IF (@INDEX(labels,3,row1110)="","",@INDEX(labels,3,row1110)) &":"}~{CALC}			
29	add2110	{LET here1110,@CELLPOINTER("coord")}~			
30	!	{IF @ISSTRING(@INDEX(labels,4,row1110))=0}{GOTO}line3110~ /M.{DOWN @ROWS(letter110)-4}~{UP}~{GOTO}{here1110}~			
31	!	{RECALC ver110}			
32	ver110	:PRSletter110~G			
33	!	{IF @ISSTRING(@INDEX(labels,4,row1110))=0}{GOTO}line1110~ /M.{DOWN @ROWS(letter110)-4}~{DOWN}~			
34	!	{GOTO}now1110~{LEFT}{DOWN}/RNLR{DOWN 10}~{GOTO}{here1110} ~{DOWN}{WINDOWSON}{WINDOWSOFF}			

The [loop1100] routine is a long routine, therefore we will study it part by part. The routine

starts with {WINDOWSOFF}{PANELOFF}, which freeze screen and panel display activities. Then the macro issues {LET row1100,@CELLPOINTER("row")-@CELL("row" ,labels)}~, which calculate how many rows are between the current cell pointer position and the top of the address database you painted, and stores the result in [row1100]. The value in [row1100] serves as an INDEX for the macro and is used to extract the parts of the address from the database to create the address to print with the form letter.

	A	B	C	D	E
22	!	{LET line1110,@INDEX(labels,0,row1110)&" "&@INDEX(labels, 1,row1110)&@IF(@ISSTRING(@INDEX(labels,2,row1110)),""& @INDEX(labels,2,row1110),"")&" "&@INDEX(labels,3,row1110)}			
23	!	{LET line2110,@IF(@ISSTRING(@INDEX(labels,4,row1110))=0, "",@INDEX(labels,4,row1110))}~			
24	!	{LET line3110,@IF(@ISSTRING(@INDEX(labels,5,row1110))=0, "",@INDEX(labels,5,row1110))}~			
25	!	{LET line4110,@IF(@ISSTRING(@INDEX(labels,6,row1110))=0, "",@INDEX(labels,6,row1110))&","&@IF(@ISSTRING(@INDEX (labels,7,row1110))=0,"","&@INDEX(labels,7,row1110))&" "&@IF(@ISSTRING(@INDEX(labels,8,row1110))=0,"",@INDEX (labels,8,row1110))}			
26	!	{LET line5110,@IF(@ISSTRING(@INDEX(labels,9,row1110))=0, "",@INDEX(labels,9,row1110))}~			

The code in the B22 cell uses the @INDEX function to create the first line of the address in the form letter. The formula adds the Title field, the First name field, the Middle initials and the Last name field to create the first line of the address in the form letter. Before it adds the Middle initials, it uses the @IF and the @ISSTRING Lotus functions to check if the middle initial of the name is missing or not. The next four cells in the B23..B26 range have similar formulas and code to extract the rest of the address parts, one formula for each line of the 6 lines of the address in the B50..B54 range.

	A	B	C	D	E
27	!	{LET line6110," "}~{LET line7110," "}~			
28	!	{LET line9110,+"Dear "&@INDEX(labels,0,row1110)&" "&@IF (@INDEX(labels,3,row1110)="","",@INDEX(labels,3,row1110)) &":"}~{CALC}			

Next the macro issues {LET line6110," "}~{LET line7110," "}~ to insert a space character in the two cells between the address and the salutation line. The macro continues with {LET ...}~ in the B28 cell, which uses the @INDEX function to extract the Title field and the Last name field and combine it with the "Dear" label and the colon ":" label to create the salutation line for the addressee, such as "Dear Mrs. Bradley:" as you can see in the B57 cell of the macro. Next the macro follows with {CALC}, which updates all the formulas in the macro.

	A	B	C	D	E
29	add2110	{LET here1110,@CELLPOINTER("coord")}~			
30	!	{IF @ISSTRING(@INDEX(labels,4,row1110))=0}{GOTO}line3110~ /M.{DOWN @ROWS(letter110)-4}~{UP}~{GOTO}{here1110}~			
31	!	{RECALC ver110}			

Because the macro needs to temporarily move the cell pointer, it issues {LET here1110, @CELLPOINTER("coord")}~ which store the current cell pointer location so the macro can use it later to return to the same address. Next the macro issues {IF @ISSTRING(@INDEX (labels,4,row1110))=0} to check if the Company name field in the record is empty. If so, the B51 cell is an empty cell between the addressee name at the B50 cell and the street address at the B52 cell. Therefore, the macro issues /M.{DOWN @ROWS(letter110)-4}~{UP}~, which

move the letter range beginning with the street address at the B52 one cell up to close the gap in the B51 cell. Notice how the macro uses the `@ROWS(letter110)-4` formula inside the `{DOWN}` command to paint the letter range down to the empty B51 cell. Next, the macro uses the `{GOTO}{here1110}~` indirect macro command to return the cell pointer to the previous location before the temporary move.

Next the macro issues `{RECALC ver110}`, which updates the string formula in the B32 cell, [ver110]. Let's look at this formula.

```
32 ver110        @IF(@LEFT(B38,1)="1","{ALT}FPletter110~",@IF(@UPPER
                 (B37)="Y",":PRSletter110~G","{WINDOWSOFF}{PANELOFF}
                 /PPCAARletter110~OOFQGPQ{ESC 6}"))
```

The formula starts with the `@IF(@LEFT(B38,1)="1"` condition which checks if you are using Lotus for windows 123W. If so, the formula returns the following code:

```
{ALT}FPletter110~
```

which is the code to print the [letter110] range. If you are not using 123W, the formula issues `@IF(@UPPER(B37)="Y"` condition, which checks if you answered "Y" to the prompt. If you are using the WYSIWYG add-in, this formula returns the following code

```
:PRSletter110~G
```

which is the code to print the [letter110] range. If you are not using the WYSIWYG add-in and you are not using Lotus 123 for Windows then the result of this formula is:

```
{WINDOWSOFF}{PANELOFF}/PPCAARletter110~OOFQGPQ{ESC 6}
```

This starts with `{WINDOWSOFF}{PANELOFF}` to freeze screen and panel display activities and then uses the standard keys to print the [letter110] range as a formatted range to the printer. Then the macro issues `{ESC 6}` to make sure that the worksheet returns to the READY mode. Here is an excellent example of how we can design a macro with a string formula which changes the code dynamically to work with the different versions and options of Lotus 1-2-3.

```
          A              B              C              D              E
33 !             {IF @ISSTRING(@INDEX(labels,4,row1110))=0}{GOTO}line1110~
                 /M.{DOWN @ROWS(letter110)-4}~{DOWN}~
34 !             {GOTO}now1110~{LEFT}{DOWN}/RNLR{DOWN 10}~{GOTO}{here1110}
                 ~{DOWN}{WINDOWSON}{WINDOWSOFF}
```

Now the macro issues `{IF @ISSTRING(@INDEX(labels,4,row1110))=0}` to check if the Company name field was empty. If so, the macro moves the range beginning with the street address and the cell below one cell down to restore the letter's length because it previously closed the gap between the addressee name and the street address. First the macro issues `{GOTO}line1110~`, which moves the cell pointer to the addressee line, [line1110], and `{DOWN @ROWS(letter110)-4}~{DOWN}~` to move the range back down one cell. To make sure that all the range names are correct, the macro issues `{GOTO}now1110~{LEFT} {DOWN}` to move the cell pointer to the B47 cell and then `/RNLR{DOWN 10}~` to re-assigns the nine [line*110] range names.

Next the macro issues `{GOTO}{here1110}~`, which moves the cell pointer back to the current record in the address database and `{DOWN}` to move to the next record. Last, the macro issues

{WINDOWSON} to display the database and the new position of the cell pointer and immediately issues {WINDOWSOFF} to again freeze the screen activity. Now, the macro returns control to the {FOR} loop to process the new record. When the {FOR} loop ends and the macro finishes printing the form letter for all the addresses in the [labels] range, the macro issues {GOTO} now1110~/WCS9~, which resets the "B" column width back to nine characters width and then {GOTO}labels~, to return the cell pointer to the upper left cell of the address database. Finally it issues /RNDlabels~ to delete the [labels] temporary range name and leave a clean a worksheet.

Here is a complete list of all the cell contents in the macro to assist you in keying this macro into Lotus 1-2-3.

```
A1: U [W15] '*---A macro to print FORM LETTERS from an address database.
          The macro
A2: U [W15] '    compensates for M.I. and missing COMPANY NAME.
A3: [W15] '*---Type your form letter right to the LETTER range name, the
         macro
A4: [W15] '    will take care of the date, address, etc.
A5: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
         define the
A6: [W15] '    range names in this column (starts with the \Z macro name)
A7: [W15] '*---Place the highlight on the upper left cell of the labels list
A8: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A9: U [W15] '!
A10: U [W15] '\Z
B10: [W9] '{BREAKON}
A11: U [W15] 'FORMLETR
B11: [W9] '{LET rel110,@INFO("release")}~{RECALC add1110}{RECALC add2110}
A12: [W15] 'ADD1110
B12: U [W9] @IF(@LEFT(B38,1)<>"@","{LET here1110,@CELLPOINTER(""coord"")}~
          {BEEP}","{LET here1110,@CELLPOINTER(""address"")}~{BEEP}")
A13: [W15] '!
B13: [W9] '{GETLABEL "Are you using WYSIWYG (Y/N)? N  ",wys110}~
A14: [W15] '!
B14: [W9] 'Highlight the labels list USING THE ARROW KEYS and press [ENTER]:
         {GET key1110}{ESC 6}~{PANELOFF}{WINDOWSOFF}/RNClabels~~/RNDlabels~
         /RNC{PANELON}{WINDOWSON}labels~{key1110}{?}~
A15: [W15] '!
B15: [W9] '{GOTO}now1110~/WCS72~{BEEP}Highlight the letter range USING THE
         ARROW KEYS
         and press [ENTER]:
A16: [W15] '!
B16: [W9] '{GET key1110}{ESC 6}~{WINDOWSOFF}{PANELOFF}/RNCletter110~~
         /RNDletter110~/RNC{PANELON}{WINDOWSON}letter110~{key1110}{?}~
         {WINDOWSOFF}PANELOFF}{GOTO}line9110~{DOWN 4}{LEFT}/C~.
         {DOWN @ROWS(letter110)-14}~{GOTO}now1110~
A17: [W15] '!
B17: [W9] '/WCS72~{GOTO}labels~
A18: [W15] '!
B18: [W9] '{FOR counter110,1,@ROWS(labels),1,loop1110}
A19: [W15] '!
B19: [W9] '{GOTO}now1110~/WCS9~{GOTO}labels~/RNDlabels~
A20: [W15] '!
A21: [W15] 'loop1110
B21: [W9] '{WINDOWSOFF}{PANELOFF}{LET row1110,@CELLPOINTER("row")-@CELL
         ("row",labels)}~
A22: [W15] '!
B22: [W9] '{LET line1110,@INDEX(labels,0,row1110)&" "&@INDEX(labels,1,
         row1110)&@IF(@ISSTRING(@INDEX(labels,2,row1110))," "&@INDEX(labels
         ,2,row1110),"")&" "&@INDEX(labels,3,row1110)}
A23: [W15] '!
B23: [W9] '{LET line2110,@IF(@ISSTRING(@INDEX(labels,4,row1110))=0,"",
         @INDEX(labels,4,row1110))}~
A24: [W15] '!
B24: [W9] '{LET line3110,@IF(@ISSTRING(@INDEX(labels,5,row1110))=0,"",
         @INDEX(labels,5,row1110))}~
A25: [W15] '!
```

```
B25: [W9] '{LET line4110,@IF(@ISSTRING(@INDEX(labels,6,row1110))=0,"",
          @INDEX(labels,6,row1110))&","&@IF(@ISSTRING(@INDEX(labels,7
          ,row1110))=0,"","  "&@INDEX(labels,7,row1110))&" "&@IF(@ISSTRING
          (@INDEX(labels,8,row1110))=0,"",@INDEX(labels,8,row1110))}
A26: [W15] '!
B26: [W9] '{LET line5110,@IF(@ISSTRING(@INDEX(labels,9,row1110))=0,"",
          @INDEX(labels,9,row1110))}~
A27: [W15] '!
B27: [W9] '{LET line6110," "}~{LET line7110," "}~
A28: [W15] '!
B28: [W9] '{LET line9110,+"Dear "&@INDEX(labels,0,row1110)&""&@IF(@INDEX
          (labels,3,row1110)="","",@INDEX(labels,3,row1110))&":"}~{CALC}
A29: [W15] 'add2110
B29: U [W9] @IF(@LEFT(B38,1)<>"@","{LET here1110,@CELLPOINTER(""coord"")}~",
             "{LET here1110,@CELLPOINTER(""address"")}~")
A30: [W15] '!
B30: [W9] '{IF @ISSTRING(@INDEX(labels,4,row1110))=0}{GOTO}line3110~/M.
          {DOWN @ROWS(letter110)-4}~{UP}~{GOTO}{here1110}~
A31: [W15] '!
B31: [W9] '{RECALC ver110}
A32: [W15] 'ver110
B32: [W9] @IF(@LEFT(B38,1)="1","{ALT}FPletter110~",@IF(@UPPER(B37)="Y","
          :PRSletter110~g","{WINDOWSOFF}{PANELOFF}/PPCAARLETTER110~OOFQGPQ
          {ESC 6}"))
A33: [W15] '!
B33: [W9] '{IF @ISSTRING(@INDEX(labels,4,row1110))=0}{GOTO}line1110~/M.
          {DOWN @ROWS(letter110)-4}~{DOWN}~{GOTO}{here1110}~
A34: [W15] '!
B34: [W9] '{GOTO}now1110~{LEFT}{DOWN}/RNLR{DOWN 10}~{GOTO}{here1110}~{DOWN}
          {WINDOWSON}{WINDOWSOFF}
A35: [W15] '!
A36: [W15] 'counter110
B36: [W9] 4
A37: [W15] 'wys110
B37: [W9] 'y
A38: [W15] 'rel110
B38: [W9] '3.00.00
A39: [W15] '!
A40: [W15] 'row1110
B40: [W9] 2
A41: [W15] '!
A42: [W15] 'here1110
B42: [W9] '$B:$A$4
A43: [W15] '!
A44: [W15] 'key1110
B44: [W9] '{PGDN}
A45: [W15] '!
A46: U [W15] 'form110
A47: U [W15] 'now1110
B47: U [W9] @CHOOSE(@MONTH(@NOW)-1,"January","February","March","April",
             "May","June","July","August","September","October","November",
             "December")&" "&@STRING(@DAY(@NOW),0)&","&@STRING(@YEAR(@NOW)
             +1900,0)
A48: [W15] '!
A49: [W15] '!
A50: U [W15] 'line1110
B50: [W9] 'Mrs. Ann Bradley
A51: U [W15] 'line2110
B51: [W9] 'HARRIS CO.
A52: U [W15] 'line3110
B52: [W9] '234 Main St.
A53: U [W15] 'line4110
B53: [W9] 'Newton, MA 02005
A54: U [W15] 'line5110
B54: [W9] 'U.S.A.
A55: U [W15] 'line6110
B55: [W9] '
A56: U [W15] 'line7110
B56: [W9] '
A57: U [W15] 'line9110
B57: [W9] 'Dear Mrs. Bradley:
```

```
A58: [W15] '!
A59: [W15] '!
A60: U [W15] 'letter110
B60: [W9] 'Start your form letter here. The macro leaves two empty lines
          between
A61: [W15] '!
B61: [W9] 'the date and the address, between the address and salution line
          and
A62: [W15] '!
B62: [W9] 'between the salution line and the letter's body.
A63: [W15] '!
A64: [W15] '!
A65: [W15] '!
A66: [W15] '!
A67: [W15] '!
A68: [W15] '!
A69: [W15] '!
A70: [W15] '!
A71: [W15] '!
A72: [W15] '!
A73: [W15] '!
A74: [W15] '!
B74: [W9] 'Sincerely,
A75: [W15] '!
A76: [W15] '!
A77: [W15] '!
B77: [W9] 'Macro user
A78: [W15] '!
```

[7] Create One Across Mailing Labels from an Address Database

```
     A                B                C                D                E
 1 *---A macro to print MAILING labels from an address database. The macro
 2     compensates for M.I. and missing COMPANY NAME.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Place the highlight on the upper left cell of the labels list
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8
 9 !
10 \Z              {BREAKON}
11 MAILLABL        {LET here1127,@CELLPOINTER("address")}~
12 !               {BEEP}Highlight the labels list USING THE ARROW KEYS and
                   press [ENTER]:{GET key1127}{ESC 6}~{WINDOWSOFF}{PANELOFF}
                   /RNClabels~~/RNDlabels~/RNC{PANELON}{WINDOWSON}labels~
                   {key1127}{?}~{WINDOWSOFF}{PANELOFF}
13 !               {GOTO}line1127~/WCS30~/RNCaddressa~~/RNDaddressa~/RNC
                   addressa~.{DOWN 5}~{GOTO}labels~{FOR counter127,1,@ROWS
                   (labels),1,loop1127}{GOTO}line1127~/WCS9~{GOTO}labels~
                   /RNDlabels~/ENDaddressa~
14 !
15 loop1127        {WINDOWSOFF}{PANELOFF}{LET row1127,@CELLPOINTER("row")-
                   @CELL("row",labels)}~
16 !               {LET line1127,@INDEX(labels,0,row1127)&" "&@INDEX(labels,
                   1,row1127)&@IF(@ISSTRING(@INDEX(labels,2,row1127)),""&
                   @INDEX(labels,2,row1127),"")&" "&@INDEX(labels,3,row1127)}
17 !               {LET line2127,@IF(@ISSTRING(@INDEX(labels,4,row1127))=0,""
                   ,@INDEX(labels,4,row1127))}~
18 !               {LET line3127,@IF(@ISSTRING(@INDEX(labels,5,row1127))=0,""
                   ,@INDEX(labels,5,row1127))}~
19 !               {LET line4127,@IF(@ISSTRING(@INDEX(labels,6,row1127))=0,""
                   ,@INDEX(labels,6,row1127))&","&@IF(@ISSTRING(@INDEX
                   (labels,7,row1127))=0,""," "&@INDEX(labels,7,row1127))
                   &""&@IF(@ISSTRING(@INDEX(labels,8,row1127))=0,"",@INDEX
                   (labels,8,row1127))}
20 !               {LET line5127,@IF(@ISSTRING(@INDEX(labels,9,row1127))=0,""
                   ,@INDEX(labels,9,row1127))}~
21 !               {LET line6127," "}~{LET here1127,@CELLPOINTER("address")}~
22 !               {IF @ISSTRING(@INDEX(labels,4,row1127))=0}{GOTO}line3127~
                   /M.{DOWN 3}~{UP}~{GOTO}line1127~/RNCaddressa~{DOWN}~
                   {GOTO}{here1127}~
23 !               {WINDOWSOFF}{PANELOFF}/PPCARaddressa~OML0~OUQGQ
24 !               {GOTO}row1127~{LEFT}{DOWN}/RNLR{DOWN 6}~{GOTO}{here1127}~
                   {DOWN}{WINDOWSON}{WINDOWSOFF}
25 !
26 counter127             4
27 !
28 row1127                2
29 !
30 line1127        Mrs. Ann Bradley
31 line2127        HARRIS CO.
32 line3127        234 Main St.
33 line4127        Newton, MA 02005
34 line5127        U.S.A.
35 line6127
36 !
37 here1127        $A$4
38 !
39 key1127         {DOWN}
```

This macro is a complementary macro to the FORMLETR.WK1 macro. It prints one across mailing labels using the address records in a database. The macro expects the database to be in the following format:

	A	B	C	D	E	F	G	H	I	J
1	TITLE	FNAME	M.I.	LNAME	COMPANY	ADDRESS	CITY	STATE	ZIP	COUNTRY
2	Mr.	Robert	C	Lane	Parker co	111 Grele	Portla	OR	97777	U.S.A.
3	Mrs.	Ann	S	Brad	HARRIS CO	234 Main	Newton	MA	02005	U.S.A.

The fields should contain labels only. You can use the ADDDATA.WK1 macro to easily fill such a database. The M.I., COMPANY and COUNTRY fields are optional and can be left empty. The macro uses the records in the database to print the mailing labels.

Because this macro has a few complicated formulas and codes, we include a list of all the cell contents at the end of this section to make it easier for you to key this macro into 1-2-3.

	A	B	C	D	E
11	MAILLABL	{LET here1127,@CELLPOINTER("address")}~			
12	!	{BEEP}Highlight the labels list USING THE ARROW KEYS and press [ENTER]:{GET key1127}{ESC 6}~{WINDOWSOFF}{PANELOFF}/RNClabels~~/RNDlabels~/RNC{PANELON}{WINDOWSON}labels~{key1127}{?}~{WINDOWSOFF}{PANELOFF}			
13	!	{GOTO}line1127~/WCS30~/RNCaddressa~~/RNDaddressa~/RNCaddressa~.{DOWN 5}~{GOTO}labels~{FOR counter127,1,@ROWS(labels),1,loop1127}{GOTO}line1127~/WCS9~{GOTO}labels~/RNDlabels~/ENDaddressa~			

The macro starts with the {LET here1127,@CELLPOINTER("address")}~ macro commands which record the cell pointer position and stores it in the B37 cell, [here1127]. Later, the macro uses this address to return to its point of origin. Next the macro issues {BEEP} to sound the BEEP and then writes:

```
Highlight the labels list USING THE ARROW KEYS and press [ENTER]:
```

directly into the panel. Then it issues {GET key1127}, which pauses macro execution and waits until you press a key. Then Lotus stores the key's code in the B39 cell, [key1127], and issues {ESC 6} to clear the panel and return to READY mode. Next the macro issues {WINDOWSOFF}{PANELOFF} to freeze the screen and panel display activities. The macro continues with the "safe technique" and assigns the [labels] range name to the address database range.

However, before the macro issues {?} it issues the {key1127} routine command to execute the key that you just pressed. The B39 cell, [key1127] holds the code of the key that you just pressed. To you it seems that Lotus executes the key immediately when you pressed it, but the macro executes the keys only after it issues few other macro commands. The {?} allows you to paint the address database and press ENTER. Then the macro issues the tilde "~" and finishes assigning the [labels] range name. Later the macro uses the [labels] range name to process the address labels.

The macro continues with {GOTO}line1127~, which moves the cell pointer to the B30 cell, [line1127], and /WCS30~ to change the column width to 30. Then it issues /RNCaddressa~~ /RNDaddressa~/RNCaddressa~.{DOWN 5}~ (the "safe technique") to assign the [addressa] range name to the B30..B34 range that holds the address to print. The macro will use the [addressa] range name when it will refer to the address range. Next the macro issues {GOTO}labels~, which returns the cell pointer to the upper left cell of address database, and {FOR counter127,1,@ROWS(labels),1,loop1127} to execute the [loop1127] routine as many times as the number of rows (records) in the address database.

	A	B	C	D	E
15	loop1127	{WINDOWSOFF}{PANELOFF}{LET row1127,@CELLPOINTER("row")-@CELL("row",labels)}~			
16	!	{LET line1127,@INDEX(labels,0,row1127)&" "&@INDEX(labels,1,row1127)&@IF(@ISSTRING(@INDEX(labels,2,row1127))," "&@INDEX(labels,2,row1127),"")&" "&@INDEX(labels,3,row1127)}			
17	!	{LET line2127,@IF(@ISSTRING(@INDEX(labels,4,row1127))=0,"",@INDEX(labels,4,row1127))}~			
18	!	{LET line3127,@IF(@ISSTRING(@INDEX(labels,5,row1127))=0,"",@INDEX(labels,5,row1127))}~			
19	!	{LET line4127,@IF(@ISSTRING(@INDEX(labels,6,row1127))=0,"",@INDEX(labels,6,row1127))&","&@IF(@ISSTRING(@INDEX(labels,7,row1127))=0,""," "&@INDEX(labels,7,row1127))&""&@IF(@ISSTRING(@INDEX(labels,8,row1127))=0,"",@INDEX(labels,8,row1127))}			
20	!	{LET line5127,@IF(@ISSTRING(@INDEX(labels,9,row1127))=0,"",@INDEX(labels,9,row1127))}~			
21	!	{LET line6127," "}~{LET here1127,@CELLPOINTER("address")}~			
22	!	{IF @ISSTRING(@INDEX(labels,4,row1127))=0}{GOTO}line3127~/M.{DOWN 3}~{UP}~{GOTO}line1127~/RNCaddressa~{DOWN}~{GOTO}{here1127}~			
23	!	{WINDOWSOFF}{PANELOFF}/PPCARaddressa~OML0~OUQGQ			
24	!	{GOTO}row1127~{LEFT}{DOWN}/RNLR{DOWN 6}~{GOTO}{here1127}~{DOWN}{WINDOWSON}{WINDOWSOFF}			

The [loop1127] routine issues `{WINDOWSOFF}{PANELOFF}` to freeze screen and panel activities and `{LET row1127,@CELLPOINTER("row")-@CELL("row",labels)}~` to calculate the number of rows between the current cell pointer position and the upper left cell of the address database, and then it stores the result in the B28 cell, [row1127].

	A	B	C	D	E
16	!	{LET line1127,@INDEX(labels,0,row1127)&" "&@INDEX(labels,1,row1127)&@IF(@ISSTRING(@INDEX(labels,2,row1127))," "&@INDEX(labels,2,row1127),"")&" "&@INDEX(labels,3,row1127)}			

The macro uses the `@INDEX` function to create the first line of the address, and then stores the result in [line1127]. The first `@INDEX(labels,0,row1127)` function returns the title (Mr., Mrs. etc.). The `@INDEX(labels,1,row1127)` function returns the first name, the `@IF(@ISSTRING(@INDEX(labels,2,row1127))," "&@INDEX(labels,2,row1127),"")` returns middle initial if it exists or the null string "" if it is missing. The `@INDEX(labels,3,row1127)` returns the last name. For example: when the cell pointer is on the second record of the following database

	A	B	C	D	E	F	G	H	I	J
1	TITLE	FNAME	M.I.	LNAME	COMPANY	ADDRESS	CITY	STATE	ZIP	COUNTRY
2	Mr.	Robert	C	Lane	Parker co	111 Grele	Portla	OR	97777	U.S.A.
3	Mrs.	Ann	S	Brad	HARRIS CO	234 Main	Newton	MA	02005	U.S.A.

The command in B16 is equivalent to `{LET line1127,Mrs. Ann S Brad}` which stores the "`Mrs. Ann S Brad`" text in [line1127]. The `@ISSTRING(@INDEX(labels,2,row1127)` checks if the middle initial field is empty or not. If so, the result is the null string.

	A	B	C	D	E
17	!	{LET line2127,@IF(@ISSTRING(@INDEX(labels,4,row1127))=0,"",@INDEX(labels,4,row1127))}~			

Here the macro uses the `@INDEX` function to create the second line of the address and then stores the result in [line1127]. The equivalent of this command is the `{LET line2127, HARRIS CO.}`. Again the `@ISSTRING(@INDEX(labels,4,row1127))` function checks if the

COMPANY field is empty. If so, the result is the null string, otherwise the result is the company name.

```
     A          B          C          D          E
18 !          {LET line3127,@IF(@ISSTRING(@INDEX(labels,5,row1127))=0,""
              ,@INDEX(labels,5,row1127))}~
```

This command returns the street address. The rest of the {LET} commands in the B18..B21 range are quite the same, they extract the data from the database and combine it to create the address. The result is the address as it appears in the B30..B35 range.

```
     A          B          C          D          E
30 line1127   Mrs. Ann Bradley
31 line2127   HARRIS CO.
32 line3127   234 Main St.
33 line4127   Newton, MA 02005
34 line5127   U.S.A.
35 line6127
```

In case that the company name is omitted, the code in B22 pushes the content of the B32..B35 range one row up to close the gap. For example: if the company name is missing in the database the result is:

```
     A          B          C          D          E
30 line1127   Mrs. Ann Bradley
31 line2127   234 Main St.
32 line3127   Newton, MA 02005
33 line4127   U.S.A.
34 line5127
35 line6127
```

The code that closes the gap is:

```
     A          B          C          D          E
22 !          {IF @ISSTRING(@INDEX(labels,4,row1127))=0}{GOTO}line3127~
              /M.{DOWN 3}~{UP}~{GOTO}line1127~/RNCaddressa~{DOWN}~
              {GOTO}{here1127}~
```

If the COMPANY field in the record is empty the @ISSTRING(@INDEX(labels,4, row1127))=0} function returns a zero "0" value, and the macro issues {GOTO}line3127~, which moves the cell pointer to the B32 cell, [line3127], and /M.{DOWN 3}~{UP}~ which move the B32..B35 range one row up and to close the gap in B31. Because the macro moved part of the B30..B35 range which also carry the [addressa] range name, the macro has to re-assign the [addressa] range name to the B30..B35 range. Therefore the macro issues /RNCaddressa~{DOWN}~ to expand the [addressa] range name to include the B30..B35 range again. Next the macro issues the {GOTO}{here1127}~ indirect command to move the cell pointer back to the first field in the current record.

```
     A          B          C          D          E
23 !          {WINDOWSOFF}{PANELOFF}/PPCARaddressa~OML0~OUQGQ
24 !          {GOTO}row1127~{LEFT}{DOWN}/RNLR{DOWN 6}~{GOTO}{here1127}~
              {DOWN}{WINDOWSON}{WINDOWSOFF}
```

The macro continues with {WINDOWSOFF}{PANELOFF}, which freeze screen and panel display activities and issues /PPCARaddressa~OML0~OUQGQ to print the B30..B35 range, [addressa]. This is the simplest form of printing and it best fits the Epson FX 80 compatible printers. If you are using a different printer you may need to manually issue the Form Feed to eject the last

page, and you may need to manually reset the printer to truly align it back. Any way, you need to experiment with your printer and probably change this code to fit your needs. For example, in this code the left margin are set to zero "0", you can change it to fit your needs, the `/PPCA` code clears any previous printing setup, if you prefer, you may want to change it to `/PP` and omit the "**C**lear All".

The macro continues with `{GOTO}row1127~`, which moves the cell pointer to the B28 cell, [row1127], and issues `{LEFT}{DOWN}` to move the cell pointer to the A29 cell and then issues `/RNLR{DOWN 6}~` to re-assign the [line1127], [line2127], [line3127], [line4127], [line5127] and [line6127] range names to the B30, B31, B32, B33, B34 and B35 cells respectively. The reason is simple: when the macro moved the last three lines of the address to close the gap, all the range names were lost, therefore before the macro can process a new record it needs to re-assign those range name.

The macro continues and issues the `{GOTO}{here1127}~` indirect command which moves the cell pointer back to the first field of the current record, and then `{DOWN}` to move to the next record. That's concludes the [loop1127] routine. When the `{FOR}` loop finishes processing all the records in the database, the macro issues `{GOTO}line1127~/WCS9~ {GOTO}labels~/RNDlabels~/ENDaddressa~` which move the cell pointer to the beginning of the address area and resets the column width back to 9. Then it moves the cell pointer back to the upper left cell of the address database and deletes the [labels] and [addressa] temporary range names and leaves a clean worksheet.
Here is a complete list of all the cell contents in the macro to assist you in keying this macro into Lotus 1-2-3.

```
A1: U [W15] '*---A macro to print MAILING labels from an address database.
            The macro
A2: U [W15] '    compensates for M.I. and missing COMPANY NAME.
A3: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A4: [W15] '    range names in this column (starts with the \Z macro name)
A5: [W15] '*---Place the highlight on the upper left cell of the labels list
A6: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A7: [W15] '!
A8: U [W15] '
A9: [W15] '!
A10: U [W15] '\Z
B10: [W9] '{BREAKON}
A11: U [W15] 'MAILLABL
B11: [W9] '{LET here1127,@CELLPOINTER("address")}~
A12: [W15] '!
B12: [W9] '{BEEP}Highlight the labels list USING THE ARROW KEYS and press
          [ENTER]: {GET key1127}{ESC 6}~{GOTO}{here1127}~{WINDOWSOFF}
          {PANELOFF}/RNClabels~~/RNDlabels~/RNC{PANELON}{WINDOWSON}labels~
          {key1127}{?}~{WINDOWSOFF}{PANELOFF}
A13: [W15] '!
B13: [W9] '{GOTO}line1127~{WINDOWSOFF}{PANELOFF}/WCS30~/RNCaddressa~~
          /RNDaddressa~/RNCaddressa~.{DOWN 5}~{GOTO}labels~{FOR counter127,
          1,@ROWS(labels),1,loop1127}{GOTO}line1127~/WCS9~{GOTO}labels~
          /RNDlabels~/ENDaddressa~
A14: [W15] '!
A15: [W15] 'loop1127
B15: [W9] '{WINDOWSOFF}{PANELOFF}{LET row1127,@CELLPOINTER("row")-
          @CELL("row",labels)}~
A16: [W15] '!
B16: [W9] '{LET line1127,@INDEX(labels,0,row1127)&" "&@INDEX(labels,1,
          row1127)&@IF(@ISSTRING(@INDEX(labels,2,row1127))," "&@INDEX
          (labels,2,row1127),"")&" "&@INDEX(labels,3,row1127)}
A17: [W15] '!
B17: [W9] '{LET line2127,@IF(@ISSTRING(@INDEX(labels,4,row1127))=0,""
```

```
          ,@INDEX(labels,4,row1127))}~
A18: [W15] '!
B18: [W9] '{LET line3127,@IF(@ISSTRING(@INDEX(labels,5,row1127))=0,""
          ,@INDEX(labels,5,row1127))}~
A19: [W15] '!
B19: [W9] '{LET line4127,@IF(@ISSTRING(@INDEX(labels,6,row1127))=0,"",@INDEX
          (labels,6,row1127))&","&@IF(@ISSTRING(@INDEX(labels,7,row1127))=
          0,""          ," "&@INDEX(labels,7,row1127))&" "&@IF(@ISSTRING
          (@INDEX(labels,8,row1127))=0,"",@INDEX(labels,8,row1127))}
A20: [W15] '!
B20: [W9] '{LET line5127,@IF(@ISSTRING(@INDEX(labels,9,row1127))=0,""
          ,@INDEX(labels,9,row1127))}~
A21: [W15] '!
B21: [W9] '{LET line6127," "}~{LET here1127,@CELLPOINTER("address")}~
A22: [W15] '!
B22: [W9] '{IF @ISSTRING(@INDEX(labels,4,row1127))=0}{GOTO}line3127~/M.
          {DOWN 3}~{UP}~{GOTO}line1127~/RNCaddressa~{DOWN}~{GOTO}{here1127}~
A23: [W15] '!
B23: [W9] '{WINDOWSOFF}{PANELOFF}/PPCARaddressa~OML0~OUQGQ
A24: [W15] '!
B24: [W9] '{GOTO}row1127~{LEFT}{DOWN}/RNLR{DOWN 6}~{GOTO}{here1127}~{DOWN}
          {WINDOWSON}{WINDOWSOFF}
A25: [W15] '!
A26: [W15] 'counter127
B26: [W9] 4
A27: [W15] '!
A28: [W15] 'row1127
B28: [W9] 2
A29: [W15] '!
A30: [W15] 'line1127
B30: [W9] 'Mrs. Ann Bradley
A31: [W15] 'line2127
B31: [W9] 'HARRIS CO.
A32: [W15] 'line3127
B32: [W9] '234 Main St.
A33: [W15] 'line4127
B33: [W9] 'Newton, MA 02005
A34: [W15] 'line5127
B34: [W9] 'U.S.A.
A35: [W15] 'line6127
B35: [W9] '
A36: [W15] '!
A37: [W15] 'here1127
B37: [W9] '$A$4
A38: [W15] '!
A39: [W15] 'key1127
B39: [W9] '{DOWN}
```

Miscellaneous Macros

[5] View the Range Inside the @AVG Formula

```
     A          B          C          D          E
 1 *---A macro to VIEW the range to be @AVGed, the range is highlighted
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer on the cell containing the @AVG formula
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 VIEW@AVG        {WINDOWSOFF}{PANELOFF}{EDIT}{HOME}'~
12 !               {LET formula178,@MID(@CELLPOINTER("contents"),5,
                   @LENGTH(@CELLPOINTER("contents"))-6)}~
13 !               /RNCAVGrange is ?~~/RNDAVGrange is ?~/RNCAVGrange is ?~
14 formula178      A10..G20
15 !               ~/RNC{PANELON}AVGrange is ?{WINDOWSON}~{?}
16 !               {WINDOWSOFF}{PANELOFF}~@AVG(AVGrange is ?)~
17 !               /RNDAVGrange is ?~
```

The @AVG, @COUNT, @MAX, @MIN, @STD, @SUM and @VAR are functions which operate on a range of values. If the worksheet contains such a function this macro displays a visual view of the range in the function and then allows you to manipulate the range size. For example, if the worksheet contains the @AVG(A10..G20) function and you start the macro it displays the actual A10..G20 range highlighted. At that point you can use the direction keys and the period [.] key to adjust the range size. When the macro is finished the function reflects the changes you made. For example, if you expand the A10..G20 range to the A5..G30 range and press ENTER, the function shows @AVG(A5..G30).

The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands which freeze the screen and panel activities. Then it issues {EDIT}{HOME}'~ to precede the @AVG formula in the cell with an apostrophe "'" to change it into a label. Now the macro issues the

```
{LET formula178,@MID(@CELLPOINTER("contents"),5,@LENGTH(@CELLPOINTER
    ("contents"))-6)}~
```

commands to insert the result of

```
@MID(@CELLPOINTER("contents"),5,@LENGTH(@CELLPOINTER("contents"))-6)
```

to cell [formula178]. For example, if the formula in the current cell is @AVG(A10..G20), this formula extracts the A10..G20 range address and stores it in cell [formula178]. Next the macro uses the "safe technique"

```
/RNCAVGrange is ?~~/RNDAVGrange is ?~/RNCAVGrange is ?~A10..G20~
```

to assign the [AVGrange is ?] name to the A10..G20 range, which also acts as a prompt. Now that the A10..G20 range also has a range name, the macro issues /RNC{PANELON} AVGrange is ? {WINDOWSON}~{?} to allow you to re-assign the [AVGrange is ?] range name. The {?} halts macro execution and displays the A10..G20 range highlighted. Use the direction keys to expand or shrink the highlighted range. When you press ENTER, the macro issues {WINDOWSOFF}{PANELOFF}~ to suppress all screen and panel display activities. Then it types the @AVG(AVGrange is ?) formula into the panel, and issues the tilde "~" to write the formula into the current cell. Because [AVGrange is ?] is the new updated range, the formula

reflects the changes you made. For example, if you expanded the A10..G20 to the A5..G30 range. Then [AVGrange is ?] is now the A5..G30 range, and the formula will show `@AVG(A5..G30)`.

[5] View the Range Inside the @COUNT Formula

```
     A                B                C                D                E
 1 *---A macro to VIEW the range to be @COUNTed, the range is highlighted
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer on the cell containing the @COUNT formula
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z               {BREAKON}
11 VIEW@CNT         {WINDOWSOFF}{PANELOFF}{EDIT}{HOME}'~
12 !                {LET formula177,@MID(@CELLPOINTER("contents"),7,
                    @LENGTH(@CELLPOINTER("contents"))-8)}~
13 !                /RNCCNTrange is ?~~/RNDCNTrange is ?~/RNCCNTrange is ?~
14 formula177       C18..E25
15 !                ~/RNC{PANELON}CNTrange is ?{WINDOWSON}~{?}
16 !                {WINDOWSOFF}{PANELOFF}~@COUNT(CNTrange is ?)~
17 !                /RNDCNTrange is ?~
```

This macro is similar to the VIEW@AVG.WK1 macro.

[5] View the Range Inside the @MAX Formula

	A	B	C	D	E
1	*---A macro to VIEW the range to be @MAXed, the range is highlighted				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer on the cell containing the @MAX formula				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	VIEW@MAX	{WINDOWSOFF}{PANELOFF}{EDIT}{HOME}'~			
12	!	{LET formula176,@MID(@CELLPOINTER("contents"),5,			
		@LENGTH(@CELLPOINTER("contents"))-6)}~			
13	!	/RNCMAXrange is ?~~/RNDMAXrange is ?~/RNCMAXrange is ?~			
14	formula176	B14..B17			
15	!	~/RNC{PANELON}MAXrange is ?{WINDOWSON}~{?}			
16	!	{WINDOWSOFF}{PANELOFF}~@MAX(MAXrange is ?)~			
17	!	/RNDMAXrange is ?~			

This macro is similar to the VIEW@AVG.WK1 macro.

[5] View the Range Inside the @MIN Formula

```
     A              B                C              D              E
 1 *---A macro to VIEW the range to be @MINed, the range is highlighted
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer on the cell containing the @MIN formula
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z               {BREAKON}
11 VIEW@MIN         {WINDOWSOFF}{PANELOFF}{EDIT}{HOME}'~
12 !                {LET formula175,@MID(@CELLPOINTER("contents"),5,
                    @LENGTH(@CELLPOINTER("contents"))-6)}~
13 !                /RNCMINrange is ?~~/RNDMINrange is ?~/RNCMINrange is ?~
14 formula175       B14..B17
15 !                ~/RNC{PANELON}MINrange is ?{WINDOWSON}~{?}
16 !                {WINDOWSOFF}{PANELOFF}~@MIN(MINrange is ?)~
17 !                /RNDMINrange is ?~
```

This macro is similar to the VIEW@AVG.WK1 macro.

[5] View the Range Inside the @STD Formula

	A	B	C	D	E
1	*---A macro to VIEW the range to be @STDrd, the range is highlighted				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer on the cell containing the @STD formula				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	VIEW@STD	{WINDOWSOFF}{PANELOFF}{EDIT}{HOME}'~			
12	!	{LET formula172,@MID(@CELLPOINTER("contents"),5,			
		@LENGTH(@CELLPOINTER("contents"))-6)}~			
13	!	/RNCSTDrange is ?~~/RNDSTDrange is ?~/RNCSTDrange is ?~			
14	formula172	B14..B17			
15	!	~/RNC{PANELON}STDrange is ?{WINDOWSON}~{?}			
16	!	{WINDOWSOFF}{PANELOFF}~@STD(STDrange is ?)~			
17	!	/RNDSTDrange is ?~			

This macro is similar to the VIEW@AVG.WK1 macro.

[5] View the Range Inside the @SUM Formula

```
     A                B                C                D                E
 1 *---A macro to VIEW the range to be @SUMmed, the range is highlighted
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Place the cell pointer on the cell containing the @SUM formula
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7 !
 8 !
 9 !
10 \Z               {BREAKON}
11 VIEW@SUM         {WINDOWSOFF}{PANELOFF}{EDIT}{HOME}'~
12 !                {LET formula171,@MID(@CELLPOINTER("contents"),5,
                    @LENGTH(@CELLPOINTER("contents"))-6)}~
13 !                /RNCSUMrange is ?~~/RNDSUMrange is ?~/RNCSUMrange is ?~
14 formula171       B14..B17
15 !                ~/RNC{PANELON}SUMrange is ?{WINDOWSON}~{?}
16 !                {WINDOWSOFF}{PANELOFF}~@SUM(SUMrange is ?)~
17 !                /RNDSUMrange is ?~
```

This macro is similar to the VIEW@AVG.WK1 macro.

[5] View the Range Inside the @VAR Formula

	A	B	C	D	E
1	*---A macro to VIEW the range to be @VARianced, the range is highlighted				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Place the cell pointer on the cell containing the @VAR formula				
5	*---Hold the [ALT] key and press [Z] to activate the macro				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	VIEW@VAR	{WINDOWSOFF}{PANELOFF}{EDIT}{HOME}'~			
12	!	{LET formula170,@MID(@CELLPOINTER("contents"),5,			
		@LENGTH(@CELLPOINTER("contents"))-6)}~			
13	!	/RNCVARrange is ?~~/RNDVARrange is ?~/RNCVARrange is ?~			
14	formula170	B14..B17			
15	!	~/RNC{PANELON}VARrange is ?{WINDOWSON}~{?}			
16	!	{WINDOWSOFF}{PANELOFF}~@VAR(VARrange is ?)~			
17	!	/RNDVARrange is ?~			

This macro is similar to the VIEW@AVG.WK1 macro.

[5] Grade Student's Scores

```
     A                B                C                D                E
 1 *---A macro to GRADE students scores (F = 0-55, D = 56-74, C = 75-84
 2     B = 85-94 and A = 95-100). To change it change the numbers in the
 3     gradesa111 range.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Point to the cell right to the upper cell of the grades list
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                      IT WILL WORK IN LOTUS 2.0 AND UP
10 \Z             {BREAKON}
11 GRADES         {LET rel111,@INFO("release")}~{RECALC loc111}
                  {RECALC form111}
12 form111        {WINDOWSOFF}{PANELOFF}{LET here1111,@CELLPOINTER("loc111")}~
                  {GOTO}gradesa111~/RNCgradesa111~~/RNDgradesa111~
                  /RNCgradesa111~.{END}{DOWN}{RIGHT}~
13 !              {GOTO}{here1111}~
14 !              {PANELON}@VLOOKUP({LEFT},$gradesa111,1)~{PANELOFF}
15 !              /C~.{LEFT}{END}{DOWN}{RIGHT}~{LEFT}/RV.{END}{DOWN}{RIGHT}~~
16 !
17 !
18 !
19 here1111       $B:$D$1
20 !
21 rel111         3.00.00
22 !
23 loc111         coord
24 !
25 gradesa111                0 F
26 !                        56 D
27 !                        75 C
28 !                        85 B
29 !                        95 A
```

This macro uses dynamic string formulas to create variable codes. Notice that the codes in the B12 and B23 cells is the result of the following dynamic string formulas:

```
12 form111        +"{WINDOWSOFF}{PANELOFF}{LET here1111,@CELLPOINTER
                  (""""&B23&""")}~{GOTO}gradesa111~/RNCgradesa111~~
                  /RNDgradesa111~/RNCgradesa111~.{END}{DOWN}{RIGHT}~"
23 loc111         @IF(@LEFT(B21,1)<>"@","coord","address")
```

If you intend to key the code of this macro into Lotus 1-2-3, you have to key the code of the formulas as they appear here, NOT the code as it appears in the main listing of the macro. For teachers who use a letter system (A,B,C,D,E,F) to grade students scores, this macro will automatically create a column of the correct grade letters for the student scores given in numbers.

```
     A          B          C          D          E
 1              50
 2              59
 3              77
 4              86
 5              97
 6
```

For example, if the student's scores are in the A1..A5 range, the macro will create the characters to the right of the numbers. To create the letter grades in the B1..B5 range, place the cell pointer on the B1 cell and activate the macro. The results are:

```
     A          B          C          D          E
```

```
1                   50             F
2                   59             D
3                   77             C
4                   86             B
5                   97             A
6
```

The macro starts with the `{LET rel111,@INFO("release")}~` macro commands which store the result of the `@INFO("release")` function in the B21 cell named [rel111]. Later, the macro uses this result to decide if you are using a 2-D or a 3-D Lotus release. Next, the macro issues `{RECALC loc111}{RECALC form111}` which update the formulas in cells [loc111] and [form111] before the macro processes their code. [loc111] contains the following formula:

```
@IF(@LEFT(B21,1)<>"@","coord","address")
```

This formula returns the "address" string if you are using a 2-D Lotus release, or the "coord" string if you are using a 3-D Lotus release. Cell [loc111] holds the result of the `@INFO ("release")` function. The formula in [form111] is:

```
+"{WINDOWSOFF}{PANELOFF}{LET here1111,@CELLPOINTER("""&B23&""")}~
              {GOTO}gradesa111~/RNCgradesa111~~/RNDgradesa111~
              /RNCgradesa111~.{END}{DOWN}{RIGHT}~"
```

This formula uses the content of the B23 cell, [loc111], which can have two optional results:

1. If you are using a 2-D Lotus release the formula returns:

```
{WINDOWSOFF}{PANELOFF}{LET here1111,@CELLPOINTER("address")}~
    {GOTO}gradesa111~/RNCgradesa111~~/RNDgradesa111~
    /RNCgradesa111~.{END}{DOWN}{RIGHT}~"
```

2. If you are using a 3-D Lotus release the formula returns:

```
{WINDOWSOFF}{PANELOFF}{LET here1111,@CELLPOINTER("coord")}~
    {GOTO}gradesa111~/RNCgradesa111~~/RNDgradesa111~
    /RNCgradesa111~.{END}{DOWN}{RIGHT}~"
```

```
                A                B                 C                  D                E
12 form111         {WINDOWSOFF}{PANELOFF}{LET here1111,@CELLPOINTER("coord")}~
                   {GOTO}gradesa111~/RNCgradesa111~~/RNDgradesa111~
                   /RNCgradesa111~.{END}{DOWN}{RIGHT}~
```

The macro processes the code in [form111] starting with `{WINDOWSOFF}{PANELOFF}` to freeze the screen and panel display activities and then the `{LET here1111,@CELLPOINTER ("coord")}~`, which stores the cell pointer position in the B19 cell, [here1111]. Next the macro issues `{GOTO}gradesa111~`, which moves the cell pointer to the B25 cell, [gradesa111]. Now the macro uses the "safe technique" to re-assign the [gradesa111] range name to the B25..C29 (Lookup table). The macro first issues `/RNCgradesa111~~` to assign (or re-assign) the [gradesa111] range name, then issues `/RNDgradesa111~` to delete the [gradesa111] range name, and finally `/RNCgradesa111~.{END}{DOWN}{RIGHT}~`, which re-assign the [gradesa111] range name to the B25..C29 range. Now that the B25..C29 range has a range name, the macro can use it as a lookup table.

```
                A                B                 C                  D                E
13 !               {GOTO}{here1111}~
14 !               {PANELON}@VLOOKUP({LEFT},$gradesa111,1)~{PANELOFF}
15 !               /C~.{LEFT}{END}{DOWN}{RIGHT}~{LEFT}/RV.{END}{DOWN}{RIGHT}~~
```

The macro issues the indirect {GOTO}{here1111}~ command, which moves the cell pointer back to the point of origin, recorded in [here1111] (the B1 cell in the sample worksheet). Next the macro issues {PANELON} to allow the panel display activity, and then directly writes a formula into the panel. To understand the technique let's look at the worksheet sample:

	A	B	C	D	E
1	50				
2	59				
3	77				
4	86				
5	97				

The cell pointer is now located on the B1 cell (right to the first number in the column of numbers). When the macro writes @VLOOKUP(into the panel, it display:

```
@VLOOKUP(
```

Next the macro issues {LEFT}, which moves the cell pointer to the A1 cell and the panel displays:

```
@VLOOKUP(A1
```

Then the macro writes ,$gradesa111,1) into the panel, and panel displays the:

```
@VLOOKUP(A1,$gradesa111,1)
```

formula. Last, the macro issues the tilde "~", which is the same as ENTER, and writes the formula into the current cell, B1. Therefore B1 should display the "F" character which is the result of the lookup formula. We have used the pointing method technique to force the formula in B1 to refer to the cell to its left (the A1 cell). Next the macro issues {PANELOFF} to freeze the panel display activity and issues /C~.{LEFT}{END}{DOWN}{RIGHT}~, which use the length of the adjacent column to the left, to copy the formula in the B1 cell down to the same column length. The result is a column of formulas in the B1..B5 range resulting in the following worksheet:

	A	B	C	D	E
1	50	F			
2	59	D			
3	77	C			
4	86	B			
5	97	A			

To change the formulas in the B1..B5 range into values, the macro issues {LEFT}/RV. {END}{DOWN}{RIGHT}~~.

[8] Link Cells to External Worksheets for Release 2.0/2.01

	A	B	C	D
1	*---LINK cells in worksheet to cells/ranges in other worksheets			
2	*---Combine this macro to any worksheet that needs to be linked to			
3	other worksheets. Combine it to an empty area of your			
4	worksheet and make sure that there will be enough empty rows to			
5	accommodate for the linking information table.			
6	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the			
7	range names in this column (starts with the \Z range name)			
8	*---Use the [ALT] key and press [Y] to ESTABLISH linking between the			
9	current cells to a cells in other worksheets.			
10	*---Use the [ALT] key and press [Z] to UPDATE all the linked cells/ranges			
11	*---After the worksheet containing this macro is saved the next			
12	time it will be retrieved the AUTO macro \0 will UPDATE all			
13	linked cell automatically.			
14	!			
15				
16	!			
17	\Z		{RESTART}	
18	\0		{GOTO}IV1~{GOTO}linkk282~{WINDOWSOFF}{PANELOFF}{DOWN 3}	
19	!		{executea282}{HOME}{WINDOWSON}{PANELON}	
20	!			
21	\Y		{RESTART}	
22	more282		{WINDOWSON}{PANELON}Point to the CELL to be linked and press [RETURN] or press [ESC] to Quit	
23	!		{GET key1@l282}{ESC}{IF key1@l282="{ESC}"}{ESC 6}{QUIT}	
24	!		{IF key1@l282="~"}{BRANCH creat@l282}	
25	!		{IF key1@l282="{LEFT}"#OR#key1@l282="{RIGHT}"#OR#key1@l282="{DOWN}"#OR#key1@l282="{UP}"#OR#key1@l282="{PGDN}"#OR#key1@l282="{PGUP}"}{key1@l282}~{BRANCH more282}	
26	!		{BRANCH \Y}	
27	!			
28	here1@l282		LINK1	
29	!			
30	file@l282		TEST1	
31	!			
32	manee@l282		D1	
33	!			
34	namecreate282		/RNC	
35	name2@l282		LINK6	
36	!		~{here1@l282}~	
37	!			
38	key1@l282		{DOWN}	
39	!			
40	creat@l282		{LET here1@l282,@CELLPOINTER("address")}~{GOTO}linkk282~{END}{DOWN 2}	
41	creat1@l282		Move up to edit and press [RETURN] or just press [RETURN]	
42	!		{GET key1@l282}{ESC}{IF key1@l282="{ESC}"}{ESC 6}{QUIT}	
43	!		{IF key1@l282="~"}{BRANCH creat2@l282}	
			{IF key1@l282="{DOWN}"#OR#key1@l282="{UP}"#OR#key1@l282="{PGDN}"#OR#key1@l282="{PGUP}"}{key1@l282}~{BRANCH creat1@l282}	
44	!		{BRANCH creat1@l282}	
45	!			
46	creat2@l282		{GETLABEL "Which file to link: ",file@l282}~/Cfile@l282~~{RIGHT}	
47	!		{GETLABEL "Insert a range name or range address to link to this cell: ",manee@l282}~/Cmanee@l282~~{RIGHT}	
48	!		{GETLABEL "Assign a range name to this cell: ",name2@l282}~/Cname2@l282~~{RIGHT}	
49	!		/Chere1@l282~~{LEFT 3}{namecreate282}{executea282}{GOTO}{here1@l282}~{BRANCH more282}	
50	!			
51	executea282		{err1@l282}{WINDOWSOFF}{PANELOFF}{LET here2@l282,@CELLPOINTER("address")}~/C~file1@l282~{RIGHT}/C~namee1@l282~{RIGHT}/C~here1@l282~{GOTO}{here1@l282}~	
52	!		/FCCN	

```
53 namee1@l282     B2
54 !               ~*{ESC}
55 file1@l282      TEST1
56 !               ~
57 !               {GOTO}{here2@l282}~{DOWN}{IF @CELLPOINTER("type")="b"}
                   {RETURN}
58 !               {BRANCH executea282}
59 !
60 here2@l282      $B$71
61 !
62 err1@l282       {ONERROR err2@l282,message@l282}
63 !
64 err2@l282       {BEEP}{message@l282} press ANY KEY to continue ...
                   {GET key1@l282}{ESC}{BRANCH \Y}
65 !
66 message@l282    File does not exist
67 !
68 linkk282        ------------------------------------------------------------
69                 EXTERNAL-FILE  EXTERNAL-RANGE  LINKED-CELL-NAME LINKED-CELL
70                 ------------------------------------------------------------
71                 TEST1          B2              LINK1             $A$71
72                 TEST1          A1              LINK2             $A$72
73                 TEST1          C1              LINK3             $A$73
74                 TEST1          C2              LINK4             $A$74
75                 TEST1          D1              LINK5             $A$95
```

One of the new features in the new releases of Lotus 1-2-3 2.2 and up is the linking feature, which allows you to build multiple worksheet applications by linking cells in one worksheet to cells in other worksheets. The **/ Range Combine** macro command was used by many Lotus users as a manual and primitive linking feature. This macro makes the process easier and automatic. When this macro is included in a worksheet, you can create a link table that holds the relations of the cells in the current worksheet with cells in the external worksheets in the disk. Next time you retrieve the worksheet, the macro is activated and "uses" the table to automatically combine all the cells from the external worksheets to the current worksheet, thereby simulating the linking process of the newer Lotus releases.

Because this macro needs to "remember" the table you created, it must become part of the worksheet to be able to link the worksheet with all the cells in the external worksheets. The macro contains three macros: the "\Y" macro creates the link table, the "\Z" macro activates the linking manually every time you need to refresh the link; and the "\0" macro is an automatic macro that is invoked every time the worksheet is retrieved and updates the links. Let's begin with the \Y macro:

```
          A              B                                    C                    D
21 \Y              {RESTART}
22 more282         {WINDOWSON}{PANELON}Point to the CELL to be linked and
                   press [RETURN] or press [ESC] to Quit
23 !               {GET key1@l282}{ESC}{IF key1@l282="{ESC}"}{ESC 6}{QUIT}
24 !               {IF key1@l282="~"}{BRANCH creat@l282}
25 !               {IF key1@l282="{LEFT}"#OR#key1@l282="{RIGHT}"#OR#key1@l282
                   ="{DOWN}"#OR#key1@l282="{UP}"#OR#key1@l282="{PGDN}"#OR#
                   key1@l282="{PGUP}"}{key1@l282}~{BRANCH more282}
26 !               {BRANCH \Y}
```

The {RESTART} command is used here as a do nothing routine, when you use this macro manually; but when you call this macro from another macro, the {RESTART} command clears the subroutine stack and ends the macro when the current subroutine ends. When 1-2-3 encounters {RESTART} command, it executes the remaining instructions in the current subroutine. Instead of returning control to the original macro location after it completes the current subroutine, the macro ends, unless it continues with an explicit branching command like

{BRANCH routine}. To return to the calling macro, use {BRANCH}. The macro continues with {WINDOWSON}{PANELON} to freeze the screen and panel display activities and then the macro writes:

```
Point to the CELL to be linked and press [RETURN] or press [ESC] to Quit
```

to the panel. The macro prompts you to point to the cell you want to link to an external cell in another worksheet and will accept the value from the external worksheet file. To hold the prompt message in the panel and allow you to respond, the macro issues {GET key1@l282}, which halts macro execution and waits until you press a key. Lotus stores the key in cell [key1@l282], and immediately follows with {ESC}, which clears the message from the panel before Lotus writes the content of the panel into the current cell.

Note: In the new releases of Lotus 1-2-3, a special command {CE} or the {CLEARENTRY} can be used to clear the panel. However, to make the macro compatible with all releases of 1-2-3, we use {ESC} instead.

The rest of the code in the \Y macro is devoted to monitor and control the keys you can use. First the macro issues {IF key1@l282="{ESC}"} to check if you pressed {ESC}. If so, the macro follows with {ESC 6} which returns the worksheet to the READY mode and issues {QUIT} and quits. Next the macro issues {IF key1@l282="~"}, which checks if you pressed ENTER. If so, the macro issues {BRANCH creat@l282}, which routes macro control to the [creat@l282] routine creating the link. Before we look at the [creat@l282] routine, let's continue with the \Y macro. Next the macro issues:

```
{IF key1@l282="{LEFT}"#OR#key1@l282="{RIGHT}"#OR#key1@l282
    ="{DOWN}"#OR#key1@l282="{UP}"#OR#key1@l282="{PGDN}"#OR#
    key1@l282="{PGUP}"}
```

a compound macro command, which checks if you pressed either the LEFT, RIGHT, DOWN, UP, PGDN or PGUP direction keys. If so, the macro issues {key1@l282}, which executes the macro command in the [key1@l282] routine. Cell [key1@l282] holds the key you just pressed, therefore the macro executes that key. You may think that Lotus just executes the key immediately when you press the keyboard, in fact, the macro issued the key after the macro checked which key you pressed. Now the macro continues with the tilde "~" to force Lotus to carry the [key1@l282] routine immediately, and then issues {BRANCH more282} to route the macro control back to the start and display the prompt message again. The last three {IF} commands take complete control of the keys that you can to use. Using such macro techniques, we can control the keys you can use, and create safer applications or macros. If none of the conditions was true, the macro issues {BRANCH \Y} and starts all over again. Now we can continue with the [creat@l282] routine:

	A	B	C	D
40	creat@l282	{LET here1@l282,@CELLPOINTER("address")}~{GOTO}linkk282~		
		{END}{DOWN 2}		
41	creat1@l282	Move up to edit and press [RETURN] or just press [RETURN]		
		{GET key1@l282}{ESC}{IF key1@l282="{ESC}"}{ESC 6}{QUIT}		
42	!	{IF key1@l282="~"}{BRANCH creat2@l282}		
43	!	{IF key1@l282="{DOWN}"#OR#key1@l282="{UP}"#OR#key1@l282=		
		"{PGDN}"#OR#key1@l282="{PGUP}"}{key1@l282}~		
		{BRANCH creat1@l282}		
44	!	{BRANCH creat1@l282}		

The [creat@l282] routine starts with {LET here1@l282,@CELLPOINTER("address")}~,

which record the current cell pointer address in [here1@l282], and then issues `{GOTO} linkk282~{END}{DOWN 2}`, which move the cell pointer to the cell under the link table, [linkk282].

	A	B	C	D
68	linkk282	----------	----------	----------
69		EXTERNAL-FILE	EXTERNAL-RANGE	LINKED-CELL-NAME	LINKED-CELL
70		----------	----------	----------
71		TEST1	B2	LINK1	$A$71
72		TEST1	A1	LINK2	$A$72
73		TEST1	C1	LINK3	$A$73
74		TEST1	C2	LINK4	$A$74
75		TEST1	D1	LINK5	$A$95

As we can see from the table which already comes with the macro as an example, the cells in the A71..A75 range (named [LINK1] to [LINK5]) are linked to the B2, A1, C1, C2 and D1 cells of the TEST1.WK1 file which contains the following data:

	A	B	C	D	E
1	1	2	3		
2	7	4	5		
3	8	9	5		

In the future, when you retrieve the LINK.WK1 file, the \0 macro uses the link table to combine the data in the TEST1.WK1 file into the A71..A75 range. Later, we will explain how the table is created using this macro. Now that the cell pointer is on the B76 cell in this example, the macro writes:

```
Move up to edit and press [RETURN] or just press [RETURN]
```

into the panel as we have seen for the pervious prompt, and then issues `{GET key1@l282}` to halt macro execution and wait until you press a key. Then the macro issues `{ESC}` to clear the message from the panel. This routine is the same as the [more282] routine, therefore it is unnecessary to repeat the explanation of the code. When you press ENTER, the macro issues `{BRANCH creat2@l282}`, which routes the macro control to the [creat2@l282] routine.

	A	B	C	D
46	creat2@l282	{GETLABEL "Which file to link: ",file@l282}~/Cfile@l282~~ {RIGHT}		
47	!	{GETLABEL "Insert a range name or range address to link to this cell: ",manee@l282}~/Cmanee@l282~~{RIGHT}		
48	!	{GETLABEL "Assign a range name to this cell: ",name2@l282}~ /Cname2@l282~~{RIGHT}		
49	!	/Chere1@l282~~{LEFT 3}{namecreate282}{executea282} {GOTO}{here1@l282}~{BRANCH more282}		

To clarify the code in this routine, let's look at the sample worksheet:

	A	B	C	D
68	linkk282	----------	----------	----------
69		EXTERNAL-FILE	EXTERNAL-RANGE	LINKED-CELL-NAME	LINKED-CELL
70		----------	----------	----------
71	4	TEST1	B2	LINK1	$A$71
72	1	TEST1	A1	LINK2	$A$72
73	3	TEST1	C1	LINK3	$A$73
74	5	TEST1	C2	LINK4	$A$74
75	1	TEST1	D1	LINK5	$A$75

Let's assume that you want to link the A76 cell to the B1 cell in the TEST1.WK1 file. Recall

that {LET here1@l282,@CELLPOINTER("address")}~ in the beginning of the [creat@l282] routine stored the A76 cell address in [here1@l282]. We also assume that you placed the cell pointer on the B76 cell. This routine starts with {GETLABEL "Which file to link:",file@l282}, which displays the "Which file to link: " prompt message in the panel and waits until you enter the external worksheet's file name (TEST1 in our example) and press ENTER. When you type the TEST1 file name and press ENTER, Lotus stores the name of the file in cell [file@l282]. Then the macro issues the tilde "~" forcing Lotus to immediately update the content of [file@l282] to reflect the last change. The macro continues with /Cfile@l282~~ which copies the content of [file@l282] to the current cell, and then issues {RIGHT} to move the cell pointer to the C76 cell. The link table should now display:

	A	B	C	D
68	linkk282	----------	----------	----------
69		EXTERNAL-FILE	EXTERNAL-RANGE	LINKED-CELL-NAME	LINKED-CELL
70		----------	----------	----------
71	4	TEST1	B2	LINK1	$A$71
72	1	TEST1	A1	LINK2	$A$72
73	3	TEST1	C1	LINK3	$A$73
74	5	TEST1	C2	LINK4	$A$74
75	1	TEST1	D1	LINK5	$A$75
76		TEST1			

The routine continues with

```
{GETLABEL "Insert a range name or range address to link to this
cell:",manee@l282}~
```

which displays

```
Insert a range name or range address to link to this cell:
```

in the panel and waits until you insert the cell address or name of the cell of the external worksheet (the B1 cell in the TEST1 file). When you type the B1 cell address and press ENTER, Lotus stores the B1 text in cell [manee@l282]. Then the macro issues /Cmanee@l282~~, which copies the content of [manee@l282] (the B1 text) into the current cell, C76. Next, the macro issues {RIGHT}, which moves the cell pointer to the D76 cell. Therefore the table displays the following:

	A	B	C	D
68	linkk282	----------	----------	----------
69		EXTERNAL-FILE	EXTERNAL-RANGE	LINKED-CELL-NAME	LINKED-CELL
70		----------	----------	----------
71	4	TEST1	B2	LINK1	$A$71
72	1	TEST1	A1	LINK2	$A$72
73	3	TEST1	C1	LINK3	$A$73
74	5	TEST1	C2	LINK4	$A$74
75	1	TEST1	D1	LINK5	$A$75
76		TEST1	B1		

Next the macro needs to assign a range name to the A76 cell (the linked cell). Therefore, the macro issues {GETLABEL "Assign a range name to this cell: ",name2@l282}~, which display "Assign a range name to this cell: " in the panel, and waits until you type a range name for the A76 cell and press ENTER. Let's assume that you typed the [LINK6] range name and pressed ENTER. Lotus stores the "LINK6" label in cell [name2@l282] and the macro issues /Cname2@l282~~{RIGHT} which copy the "LINK6" label, the content of [name2@l282] into the current cell. Therefore the table displays the following:

	A	B	C	D
68	linkk282	----------	----------	----------
69		EXTERNAL-FILE	EXTERNAL-RANGE	LINKED-CELL-NAME	LINKED-CELL
70		----------	----------	----------
71	4	TEST1	B2	LINK1	$A$71
72	1	TEST1	A1	LINK2	$A$72
73	3	TEST1	C1	LINK3	$A$73
74	5	TEST1	C2	LINK4	$A$74
75	1	TEST1	D1	LINK5	$A$75
76		TEST1	B1	LINK6	

Now the macro needs to enter the address of cell [LINK6] into the E76 cell to complete the table. Therefore the macro issues `/Chere1@l282~~`, which copies the content of [here1@l282] into the current cell. Cell [here1@l282] holds the address $A$76. Next the macro issues `{LEFT 3}`, which moves the cell pointer to the B76 cell. Therefore the table displays the following:

	A	B	C	D
68	linkk282	----------	----------	----------
69		EXTERNAL-FILE	EXTERNAL-RANGE	LINKED-CELL-NAME	LINKED-CELL
70		----------	----------	----------
71	4	TEST1	B2	LINK1	$A$71
72	1	TEST1	A1	LINK2	$A$72
73	3	TEST1	C1	LINK3	$A$73
74	5	TEST1	C2	LINK4	$A$74
75	1	TEST1	D1	LINK5	$A$75
76		TEST1	B1	LINK6	$A$76

The macro continues with `{namecreate282}{executea282}` routine commands which execute the [namecreate282] routine and then the [executea282] routine. Let's start with [namecreate282]:

	A	B	C	D
34	namecreate282	/RNC		
35	name2@l282	LINK6		
36	!	~{here1@l282}~		

This routine assigns the [LINK6] range name to the A76 cell. The "LINK6" label in the B35 cell was stored by `{GETLABEL "Assign a range name to this cell:", name2@l282}~`. This is one form of dynamic code programming, where the macro changes the code before processing it. To understand this routine, let's see how it works:

The macro issues `/RNC` and the panel shows:

```
Enter name:
```

Then the macro types the "`LINK6`" text into the panel, which now displays:

```
Enter name: LINK6
```

The macro continues with the tilde "`~`", which is the same as ENTER and the panel displays:

```
Enter name: LINK6                    Enter range: B76..B76
```

Now the macro issues `{here1@l282}` which injects the content of [here1@l282] into the panel. Cell [here1@l282] holds the $A$76 cell address, therefore the panel displays:

```
Enter name: LINK6                    Enter range: $A$76
```

Last the macro issues the tilde "~" and finishes assigning the [LINK6] range name to the A76 cell.

Note: When you key this macro into Lotus 1-2-3, you do not have to key the links information listed in the table, they are given here as an example of how the macro works. When you want to create a link from your application to other worksheets, combine the LINK.WK1 macro, then assign the range names as explained in the A1..A13 range of the macro, and activate the \Y macro.

When the [namecreate282] routine is finished, the macro starts the [executea282] routine:

	A	B	C	D
51	executea282	{err1@l282}{WINDOWSOFF}{PANELOFF}{LET here2@l282, @CELLPOINTER("address")}~/C~file1@l282~{RIGHT}/C~ namee1@l282~{RIGHT}/C~here1@l282~ {GOTO}{here1@l282}~		
52	!	/FCCN		
53	namee1@l282	B2		
54	!	~*{ESC}		
55	file1@l282	TEST1		
56	!	~		
57	!	{GOTO}{here2@l282}~{DOWN}{IF @CELLPOINTER("type")="b"} {RETURN}		
58	!	{BRANCH executea282}		
59	!			
60	here2@l282	$B$71		
61	!			
62	err1@l282	{ONERROR err2@l282,message@l282}		
63	!			
64	err2@l282	{BEEP}{message@l282} press ANY KEY to continue ... {GET key1@l282}{ESC}{BRANCH \Y}		
65	!			
66	message@l282	File does not exist		

The [executea282] routine starts with {err1@l282}, which starts the [err1@l282] error handling routine. We will study [err1@l282] later. The macro issues {WINDOWSOFF} {PANELOFF} which freeze screen and panel display activities, continuing with {LET here2@l282,@CELLPOINTER("address")}~, which store the current cell's address in [here2@l282]. The cell pointer is on the B76 cell which contains the "TEST1" label. When the macro next issues /C~file1@l282~, the content of the B76 cell is copied to B55, [file1@l282]. Next the macro issues {RIGHT}, which moves the cell pointer to the C76 cell containing the B1 cell address label.

Therefore, /C~namee1@l282~ copies the content of the current cell to B53, [namee1@l282]. Again the macro issues {RIGHT}, which moves the cell pointer to the D76 cell, containing the "LINK6" label. The /C~here1@l282~ copies the content of the current cell to [here1@l282]. All the copied text becomes part to the macro code which the macro immediately uses. The {GOTO} {here1@l282}~ indirect command uses the content of [here1@l282] to move the cell pointer to the A76 cell, [here1@l282], containing the "LINK6" text; therefore {GOTO} {here1@l282}~ is actually {GOTO}LINK6~.

	A	B	C	D
52	!	/FCCN		
53	namee1@l282	B1		
54	!	~*{ESC}		
55	file1@l282	TEST1		
56	!	~		

The code in the B52..B56 range is summed up as /FCCNB1~*{ESC}TEST1~. The macro starts the combining process with /FCCNB1~ which means the macro combines the content of the B1 cell in the TEST1.WK1 external file. Recall that the "B1" text in the B53 cell was copied from C76 by /C~file1@l282~. To enter the TEST1 file name, the macro needs to clear the panel from any previous text, therefore the macro types the astrix "*" character (any other character will do), now a single {ESC} is enough to clear the panel, no matter how long the text is. If a previous file combine command has been issued or the default directory is a nested directory, the panel may look like this:

```
C:\LOTUS\WORK\SAMPLE\FILENAME
```

Every ESC will take the file name one step down. For example, after the first ESC, the panel will show:

```
C:\LOTUS\WORK\SAMPLE
```

and the next ESC will change the panel to display:

```
C:\LOTUS\WORK
```

and so on. Therefore, because we cannot know how many directories and sub directories are listed in the panel, we cannot know how many times to press ESC. Therefore we write something to the panel (the astrix "*": character in our macro), and issue one {ESC}, which is enough to clear the whole text from the panel and make it ready for the TEST1 file name. Next the macro types the "TEST1" text into the panel and issues the tilde "~" to complete the combine process. Now A76 contains the data from the B1 cell of the TEST1.WK1 file.

	A	B	C	D
57	!	{GOTO}{here2@l282}~{DOWN}{IF @CELLPOINTER("type")="b"} {RETURN}		
58	!	{BRANCH executea282}		

Next the macro issues {GOTO}{here2@l282}~, which moves the cell pointer back to B76. Cell [here2@l282] holds the $B$76 cell address previously stored there by {LET here2@l282 ,@CELLPOINTER("address")}~. Next the macro issues {DOWN}, which moves the cell pointer to B77. Now the macro issues {IF @CELLPOINTER("type")="b"}, which checks if the cell is empty. If so, the macro issues {RETURN} which routes macro control back to the [creat2@l282] routine. If the cell is not empty, the macro issues {BRANCH executea282} which starts the [executea282] routine and combines the next cell according to the link record information in the table. Before we return to the [creat2@l282] routine, let's examine the error handing routine.

	A	B	C	D
62	err1@l282	{ONERROR err2@l282,message@l282}		
63	!			
64	err2@l282	{BEEP}{message@l282} press ANY KEY to continue ... {GET key1@l282}{ESC}{BRANCH \Y}		
65	!			
66	message@l282	File does not exist		

When an error occurs, {ONERROR err2@l282,message@l282} writes the error message in B66, [message@l282], and activates the [err2@l282] routine. The [err2@l282] routine starts with a {BEEP}, to warn you that an error occurred. Then the macro issues {message@l282}

which injects5 the content of [message@l282] into the panel because the {message@l282} routine contains only the error message text and then the macro continue to write2 " press ANY KEY to continue ..." into the panel to create a complete and informative message.

Let's see how this works. First, when an error occurs, the macro writes the Lotus standard error message into B66, [message@l282]. In this case, the error message is "File does not exist". Such an error occurs if you insert a wrong file name to link. After the BEEP, the macro issues {message@l282} which injects5 the message into the panel which displays:

```
File does not exist
```

Next the macro writes " press ANY KEY to continue ..." into the panel which now displays:

```
File does not exist press ANY KEY to continue ...
```

We have combined the standard Lotus error message with our instructions to the user. This is another example of how we can manipulate the panel to achieve a desired effect. To keep the message in the panel so you can read it, the macro issues {GET key1@l282}. When you press a key, Lotus stores the key in cell [key1@l282] and the macro issues {ESC}, which clears the combined message from the panel before Lotus writes it into the current cell. Then the macro issues {BRANCH \Y} which routes macro control to the beginning to start all over again. Now we can continue with the [creat2@l282] routine:

	A	B	C	D
46	creat2@l282	{GETLABEL "Which file to link: ",file@l282}~/Cfile@l282~~ {RIGHT}		
47	!	{GETLABEL "Insert a range name or range address to link to this cell: ",manee@l282}~/Cmanee@l282~~{RIGHT}		
48	!	{GETLABEL "Assign a range name to this cell: ",name2@l282}~ /Cname2@l282~~{RIGHT}		
49	!	/Chere1@l282~~{LEFT 3}{namecreate282}{executea282} {GOTO}{here1@l282}~{BRANCH more282}		

When the macro returns to the [creat2@l282] mother routine the macro issues the {GOTO}{here1@l282}~ indirect command to move the cell pointer back to the last linked cell, A76 cell in our example, and then issues {BRANCH more282}, which branches back to the beginning of the macro to allow you to establish another link. Up till now, we have looked into the \Y macro, which is that allows you to create or update the link table records. However, when the table already exists, the macro \Z is used when you want to update or refresh all the linked cells.

	A	B	C	D
17	\Z	{RESTART}		
18	\0	{GOTO}IV1~{GOTO}linkk282~{WINDOWSOFF}{PANELOFF}{DOWN 3}		
19	!	{executea282}{HOME}{WINDOWSON}{PANELON}		

This routine starts with {RESTART}, which clears the subroutine stack, if you called this macro from within another macro, and ends the macro when the current subroutine ends. When 1-2-3 encounters {RESTART}, it executes the remaining instructions in the current subroutine, but instead of returning control to the original macro location after it completes the current subroutine, the macro ends, unless it is told to continue with an explicit branching command such as {BRANCH routine}. To return to the calling macro, use {BRANCH}.

The macro continues with {GOTO}IV1~ and then issues {GOTO}linkk282~ to move the cell pointer to the link table. Why move to the last column and back to the link table while the macro could move the cell pointer directly to the link table? The answer is simple: this way the cell pointer will be on the top left corner of the screen, while moving directly to the link table may put the cell pointer anywhere on the screen if the current cell position is less than seven (7) columns or/and nineteen (19) rows away from the link table.

Next the macro issues {WINDOWSOFF}{PANELOFF} to freeze the screen and panel activities while the macro combines the cell from the external worksheets according to the link table. The macro issues {DOWN 3}, placing the cell pointer on the first record of the link table, and then {executea282}, which starts the [executea282] routine. We studied the [executea282] routine when we studied the \Y macro and we show it again for clarity:

	A	B	C	D
51	executea282	{err1@l282}{WINDOWSOFF}{PANELOFF}{LET here2@l282,		
		@CELLPOINTER("address")}~/C~file1@l282~{RIGHT}/C~		
		namee1@l282~{RIGHT}/C~here1@l282~		
		{GOTO}{here1@l282}~		
52	!	/FCCN		
53	namee1@l282	B2		
54	!	~*{ESC}		
55	file1@l282	TEST1		
56	!	~		
57	!	{GOTO}{here2@l282}~{DOWN}{IF @CELLPOINTER("type")="b"}		
		{RETURN}		
58	!	{BRANCH executea282}		

To see how it will affect the link table now, let's look at the sample table:

	A	B	C	D
68	linkk282	----------	----------	----------
69		EXTERNAL-FILE	EXTERNAL-RANGE	LINKED-CELL-NAME	LINKED-CELL
70		----------	----------	----------
71	4	TEST1	B2	LINK1	$A$71
72	1	TEST1	A1	LINK2	$A$72
73	3	TEST1	C1	LINK3	$A$73
74	5	TEST1	C2	LINK4	$A$74
75	1	TEST1	D1	LINK5	$A$75
76		TEST1	B1	LINK6	$A$76

When the \Z macro calls the [executea282] routine, the cell pointer is on the B41 cell. Therefore the [executea282] routine processes the data in the first record and combines the content of B2 in the TEST1.WK1 file to A71. Next the [executea282] routine moves the cell pointer to the B72 cell and checks if it's empty. Because the B72 cell is not empty, the [executea282] routine issues {BRANCH executea282} and branches to the beginning to process this record, and combines the content of A1 in the TEST1.WK1 file to A72l. This process continues until {IF @CELLPOINTER("type")="b"} condition is true, which is when the cell pointer reaches the B77 cell. This way the \Z macro uses the [executea282] routine to update all the cells based on the link records in the table.

When the [executea282] routine is finished, the macro issues {HOME}{WINDOWSON} {PANELON} to move the cell pointer to the A1 cell of the worksheet and to resume screen and panel display activities, and quits because it reaches an empty cell. This was quite a complicated macro to analyze and study. To help you key this macro into Lotus 1-2-3, we show the full list of the cell contents.

```
A1: U [W15] '*---LINK cells in worksheet to cells/ranges in other worksheets
```

```
A2: [W15] '*---Combine this macro to any worksheet that needs to be linked to
A3: [W15] '    other worksheets. Combine it to an empty area of your
A4: [W15] '    worksheet and make sure that there will be enough empty rows to
A5: [W15] '    accommodate for the linking information table.
A6: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define
          the
A7: [W15] '    range names in this column (starts with the \Z range name)
A8: [W15] '*---Use the [ALT] key and press [Y] to ESTABLISH linking between
          the
A9: [W15] '    current cell to cells/ranges in other worksheets.
A10: [W15] '*---Use the [ALT] key and press [Z] to UPDATE all linked cells/
           ranges
A11: [W15] '*---After the worksheet containing this macro is saved the next
A12: [W15] '    time it will be retrieved the AUTO macro \0 will UPDATE all
A13: [W15] '    linked cell automatically.
A14: [W15] '!
A15: U [W15] '
A16: [W15] '!
A17: U [W15] '\Z
B17: [W15] '{RESTART}
A18: [W15] '\0
B18: [W15] '{GOTO}IV1~{GOTO}linkk282~{WINDOWSOFF}{PANELOFF}{DOWN 3}
A19: [W15] '!
B19: [W15] '{executea282}{HOME}{WINDOWSON}{PANELON}
A20: [W15] '!
A21: U [W15] '\Y
B21: [W15] '{RESTART}
A22: [W15] 'more282
B22: [W15] '{WINDOWSON}{PANELON}Point to the CELL to be linked and press
           [RETURN] or
            press [ESC] to Quit
A23: [W15] '!
B23: [W15] '{GET key1@l282}{ESC}{IF key1@l282="{ESC}"}{ESC 6}{QUIT}
A24: [W15] '!
B24: [W15] '{IF key1@l282="~"}{BRANCH creat@l282}
A25: [W15] '!
B25: [W15] '{IF key1@l282="{LEFT}"#OR#key1@l282="{RIGHT}"#OR#key1@l282=
           "{DOWN}"#OR#key1@l282="{UP}"#OR#key1@l282="{PGDN}"#OR#key1@l282=
           "{PGUP}"}{key1@l282}~{BRANCH more282}
A26: [W15] '!
B26: [W15] '{BRANCH \Y}
A27: [W15] '!
A28: [W15] 'here1@l282
B28: [W15] 'LINK1
A29: [W15] '!
A30: [W15] 'file@l282
B30: [W15] 'TEST1
A31: [W15] '!
A32: [W15] 'manee@l282
B32: [W15] 'D1
A33: [W15] '!
A34: [W15] 'namecreate282
B34: [W15] '/RNC
A35: [W15] 'name2@l282
B35: [W15] 'LINK5
A36: [W15] '!
B36: [W15] '~{here1@l282}~
A37: [W15] '!
A38: [W15] 'key1@l282
B38: [W15] '{DOWN}
A39: [W15] '!
A40: [W15] 'creat@l282
B40: [W15] '{LET here1@l282,@CELLPOINTER("address")}~{GOTO}linkk282~{END}
           {DOWN 2}
A41: [W15] 'creat1@l282
B41: [W15] 'Move up to edit and press [RETURN] or just press [RETURN]
           {GET key1@l282}{ESC}{IF key1@l282="{ESC}"}{ESC 6}{QUIT}
A42: [W15] '!
B42: [W15] '{IF key1@l282="~"}{BRANCH creat2@l282}
A43: [W15] '!
B43: [W15] '{IF key1@l282="{DOWN}"#OR#key1@l282="{UP}"#OR#key1@l282="{PGDN}"
```

```
           #OR#key1@l282="{PGUP}"}{key1@l282}{?}{BRANCH creat1@l282}
A44: [W15] '!
B44: [W15] '{BRANCH creat1@l282}
A45: [W15] '!
A46: [W15] 'creat2@l282
B46: [W15] '{GETLABEL "Which file to link: ",file@l282}~/Cfile@l282~~{RIGHT}
A47: [W15] '!
B47: [W15] '{GETLABEL "Insert a range name or range address to link to this
           cell: ",manee@l282}~/Cmanee@l282~~{RIGHT}
A48: [W15] '!
B48: [W15] '{GETLABEL "Assign a range name to this cell: ",name2@l282}~
           /Cname2@l282~~{RIGHT}
A49: [W15] '!
B49: [W15] '/Chere1@l282~~{LEFT 3}{namecreate282}{executea282}
           {GOTO}{here1@l282}~{BRANCH more282}
A50: [W15] '!
A51: [W15] 'executea282
B51: [W15] '{err1@l282}{WINDOWSOFF}{PANELOFF}{LET here2@l282,@CELLPOINTER
           ("address")}~/C~file1@l282~{RIGHT}/C~namee1@l282~{RIGHT}/C~
           here1@l282~{GOTO}{here1@l282}~
A52: [W15] '!
B52: [W15] '/FCCN
A53: [W15] 'namee1@l282
B53: [W15] 'B2
A54: [W15] '!
B54: [W15] '~*{ESC}
A55: [W15] 'file1@l282
B55: [W15] 'TEST1
A56: [W15] '!
B56: [W15] '~
A57: [W15] '!
B57: [W15] '{GOTO}{here2@l282}~{DOWN}{IF @CELLPOINTER("type")="b"}{RETURN}
A58: [W15] '!
B58: [W15] '{BRANCH executea282}
A59: [W15] '!
A60: [W15] 'here2@l282
B60: [W15] '$B$71
A61: [W15] '!
A62: [W15] 'err1@l282
B62: [W15] '{ONERROR err2@l282,message@l282}
A63: [W15] '!
A64: [W15] 'err2@l282
B64: [W15] '{BEEP}{message@l282} press ANY KEY to continue ...{GET key1@l282}
           {ESC}{BRANCH \Y}
A65: [W15] '!
A66: [W15] 'message@l282
B66: [W15] 'File does not exist
A67: [W15] '!
A68: [W15] 'linkk282
B68: U [W15]
'------------------------------------------------------------------------------
B69: U [W15] ^EXTERNAL-FILE
C69: U [W16] ^EXTERNAL-RANGE
D69: U [W18] ^LINKED-CELL-NAME
E69: U [W10] ^LINKED-CELL-ADDRESS
B70: U [W15]
'------------------------------------------------------------------------------
A71: [W15] 4
B71: [W15] 'TEST1
C71: [W16] 'B2
D71: [W18] 'LINK1
E71: [W10] @CELL("address",$LINK1)
A72: [W15] 1
B72: [W15] 'TEST1
C72: [W16] 'A1
D72: [W18] 'LINK2
E72: [W10] @CELL("address",$LINK2)
A73: [W15] 3
B73: [W15] 'TEST1
C73: [W16] 'C1
D73: [W18] 'LINK3
```

```
E73: [W10] @CELL("address",$LINK3)
A74: [W15] 5
B74: [W15] 'TEST1
C74: [W16] 'C2
D74: [W18] 'LINK4
E74: [W10] @CELL("address",$LINK4)
B75: [W15] 'TEST1
C75: [W16] 'D1
D75: [W18] 'LINK5
E75: [W10] @CELL("address",$A$95..$A$95)
```

[4] Link a Range (2.2 and Up)

```
     A              B              C              D              E
 1 *---A macro to LINK a range. Lotus 1-2-3 allows only cells linking, this
 2     macro links a whole range, it creates a range of cell linking
 3     formulas.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Place the cell pointer where you want the linked range
 7 *---Hold the [ALT] key and press [Z] to activate the macro
 8 *---Ranges should be entered as address ONLY!  ( For example: A1..D5 )
 9              * * *     A LOTUS 2.2 AND UP MACRO    * * *
10 \Z              {BREAKON}
11 LINKRANG        {GETLABEL "External file name: ",ext215}~
                   {GETLABEL "External range: ",rng215}~
12 !               {WINDOWSOFF}{PANELOFF}{RECALC columns215}{RECALC rows215}
                   {RECALC copy215}
13 !               {LET firstcell215,@LEFT(rng215,@FIND(".",rng215,0))}~
14 !               '<<{ext215}>>{firstcell215}{EDIT}{HOME}{DEL}+~
15 copy215         /C~.{DOWN 4}{RIGHT 3}~
16 !
17 ext215          A.WK1
18 !
19 rng215          A1..D5
20 !
21 columns215      3
22 !
23 rows215         4
24 !
25 firstcell215    A1
```

The following are the dynamic string formulas that create the code in the B15, B21 and B23 cells.

```
15 copy215         +"/C~.{DOWN "&B23&"}{RIGHT "&B21&"}~"
21 columns215      @STRING(@COLS(@@(B19))-1,0)
23 rows215         @STRING(@ROWS(@@(B19))-1,0)
```

when you key this code in Lotus 1-2-3, you must use these formulas, NOT the code as it appears in the main macro listing.

Lotus 1-2-3 release 2.2 and up allows only cells linking, while this macro links a whole range and creates a range of cell linking formulas. The macro start with the `{GETLABEL "External file name: ",ext215}` command that displays the "`External file name :`" prompt in the panel. When you enter the external file name to link to and press ENTER, Lotus stores the file name in cell [ext215]. Next, the macro issues `{GETLABEL "External range: ",rng215}`, displays the "`External range: `" prompt in the panel. When you enter the external range address to link to and press ENTER, Lotus stores the range address in cell [rng215].

```
     A              B              C              D              E
12 !               {WINDOWSOFF}{PANELOFF}{RECALC columns215}{RECALC rows215}
                   {RECALC copy215}
13 !               {LET firstcell215,@LEFT(rng215,@FIND(".",rng215,0))}~
14 !               '<<{ext215}>>{firstcell215}{EDIT}{HOME}{DEL}+~
15 copy215         /C~.{DOWN 4}{RIGHT 3}~
```

Now the macro uses `{WINDOWSOFF}{PANELOFF}` to freeze screen and panel display activities while it uses `{RECALC columns215}{RECALC rows215}{RECALC copy215}` to update the formulas in cells [columns215] [rows215] and [copy215]. Next the macro stores the first external cell's address in the cell [firstcell215]. To extract the cell address the macro issues:

```
{LET firstcell215,@LEFT(rng215,@FIND(".",rng215,0))}~
```

The macro uses an advanced form of the {LET} combined with the @LEFT and the @FIND Lotus functions to extract the address of the upper left cell address of the external range. For example, if the external range is A1..D5, its address is stored in cell [rng215], then the @FIND(".",rng215,0)) formula which looks for the first occurrence of the period "." in the address, returns the number "2" because Lotus starts counting from zero. Therefore, the @LEFT(rng215,@FIND(".",rng215,0)) formula, which is equivalent to the @LEFT (rng215,2) function, returns the "A1" string and stores it in cell [firstcell215]. Next the macro types:

```
'<<
```

to the panel, because Lotus cannot find a valid Lotus command in this text, and {ext215} to inject the content of cell [ext215] to the panel. Recall that [ext215] holds the external file name. Therefore, if the external file name is "A.WK1", the panel will display:

```
'<<A.WK1
```

Again the macro types:

```
>>
```

to the panel that now displays:

```
'<<A.WK1>>
```

Then the macro issues {firstcell215} that injects the content of [firstcell215] into the panel, which now displays:

```
'<<A.WK1>>A1
```

To turn this text into an active linking formula, the macro issues {EDIT}{HOME}{DEL}+ that delete the apostrophe and add the "+" character instead; therefore the panel displays:

```
+<<A.WK1>>A1
```

before the macro issues the tilde "~" that writes the formula into the current cell. The next macro command copies this formula to create a range that has the same size as the external range. In our example, the macro uses /C~.{DOWN 4}{RIGHT 3}~ to copy the formula and create a 5X4 range of linking cell formulas. This code is the result of the dynamic string formula:

```
15 copy215          +"/C~.{DOWN "&B23&"}{RIGHT "&B21&"}~"
```

in the B15 cell, [copy215], which uses the content of [rows215] and [columns215]. Cell [rows215] and cell [columns215] contain these formulas:

```
21 columns215       @STRING(@COLS(@@(B19))-1,0)
23 rows215          @STRING(@ROWS(@@(B16))-1,0)
```

These two formulas use the indirect @@(rng215) and the indirect @@(rng215) functions to return the range address that is stored in [rng215]. Therefore, if [rng215] contains the A1..D5 range address, the @COLS(@@(rng215)) formula returns the number "4" and the @ROWS(@@

`(rng215)` formula returns the number "`5`". To use these values, Lotus needs to turn them into labels. Therefore the `@STRING(@COLS(@@(rng215))-1,0)` formula returns the label "`3`" and the `@STRING(@ROWS(@@(rng215))-1,0)` formula returns the "`4`" label. The `+"/C~.{DOWN "&rows215&"}{RIGHT "&columns215&"}~"` string formula in [copy215] uses both labels to create the `/C~.{DOWN 4}{RIGHT 3}~` code in [copy215].

[7] Help Macro

	A	B	C	D	E
1	*---A macro to HELP WRITING MACROS and FORMULAS, allows you to				
2	CONTINUOUSLY write every Lotus function or command by POINTING to				
3	the syntax on display and SHOOTING (press RETURN). There is no more				
4	typing errors or a need to remember the exact syntax.				
5	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
6	range names in this column (starts with the \Z macro name)				
7	*---Hold the [ALT] key and press [Z] to activate the macro				
8	THIS MACRO WORKS IN LOTUS 2.0 AND UP				
9	!	SEE MORE DETAILS IN THE README.PRN FILE			
10	\Z	{BREAKON}			
11	MACROHLP	{PANELOFF}{LET rel126,@INFO("release")}~{RECALC loc126}			
		{RECALC form126}{RECALC hhh1126}/RNC!~~/RND!~{PANELON}			
		{IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER("type")="v"}			
		{EDIT}{HOME}'~			
12	form126	{err1126}{LET back1126,@CELLPOINTER("coord")}~{PANELOFF}			
		{GOTO}{NAME}{PANELON}{NAME}{GET key126}{IF key126="{ESC}"}			
		{ESC 8}{QUIT}			
13	!	{IF key126="~"}{BRANCH hhh1126}			
14	key126	{DOWN}			
15	hhh126	{?}~			
16	hhh1126	~{LET routin126,@CELLPOINTER("coord")}~{back126}			
		{DISPATCH routin126}~			
17	!				
18	key1126	@			
19	!				
20	err1126	{ONERROR err2126}			
21	!				
22	err2126	{BEEP}INVALID MACRO NAME! PRESS ANY KEY TO CONTINUE ...			
		{GET key126}{ESC}{BRANCH \Z}}			
23	!				
24	routin126	$A:$B$85			
25	!				
26	back126	{GOTO}			
27	back1126	$A:$A$85			
28	!	~			
29	!				
30	rel126	3.00.00			
31	!				
32	loc126	coord			
33	!				
34	rep126	{LEFT}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}			
35	rep1126	{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}			
36	rep2126	{LEFT 2}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}			
37	rep3126	{LEFT 12}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}			
38	rep4126	{LEFT 23}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}			
39	rep5126	{LEFT 6}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}			
40	!				
41	!				
42	"ABORT	{IF @LEFT(@CELLPOINTER("contents"),1)="@"#OR#@LEFT			
		(@CELLPOINTER("contents"),1)="-"#OR#@LEFT(@CELLPOINTER			
		("contents"),1)="+"}{EDIT}{HOME}{DEL}~{BRANCH ret126}			
43	!				
44	ret126				
45	!				
46	"ADDRESS"	{EDIT}ADDRESS",{BRANCH rep1126}			
47	"COL"	{EDIT}COL",{BRANCH rep1126}			
48	"COLOR"	{EDIT}color",{BRANCH rep1126}			
49	"CONTENTS"	{EDIT}CONTENTS",{BRANCH rep1126}			
50	"COORD"	{EDIT}coord",{BRANCH rep1126}			
51	"FILENAME"	{EDIT}filename",{BRANCH rep1126}			
52	"PORMAT"	{EDIT}format",{BRANCH rep1126}			
53	"PARENTHESES"	{EDIT}parentheses",{BRANCH rep1126}			
54	"PREFIX"	{EDIT}PREFIX",{BRANCH rep1126}			
55	"PROTECT"	{EDIT}protect",{BRANCH rep1126}			
56	"ROW"	{EDIT}ROW",{BRANCH rep1126}			
57	"SHEET"	{EDIT}sheet",{BRANCH rep1126}			

```
 58 "type"           {EDIT}TYPE",{BRANCH rep1126}
 59 "WIDTH"          {EDIT}WIDTH",{BRANCH rep1126}
 60 @ABS             {EDIT}@ABS(X){BRANCH rep2126}
 61 @ACOS            {EDIT}@ACOS(X){BRANCH rep2126}
 62 @ASIN            {EDIT}@ASIN(X){BRANCH rep2126}
 63 @ATAN            {EDIT}@ATAN(X){BRANCH rep2126}
 64 @ATAN2           {EDIT}@ATAN2(X,Y){BRANCH rep2126}
 65 @CELL            {EDIT}@CELL("{BRANCH rep1126}
 66 @CELLPOINTER     {EDIT}@CELLPOINTER("{BRANCH rep1126}
 67 @CHAR            {EDIT}@CHAR(CODE){BRANCH rep1126}
 68 @CHOOSE          {EDIT}@CHOOSE(X,"","","","){BRANCH rep1126}
 69 @CODE            {EDIT}@CODE(character){LEFT 10}{BRANCH rep1126}
 70 @COLS            {EDIT}@COLS(range){BRANCH rep5126}
 71 @COORD           {EDIT}@COORD(worksheet,column,row,absolute){LEFT 30}
                     {BRANCH rep1126}
 72 @COS             {EDIT}@COS(X){BRANCH rep2126}
 73 @COUNT           {EDIT}@COUNT(range){BRANCH rep5126}
 74 @CTERM           {EDIT}@CTERM(interest,future-value,present-value)
                     {LEFT 36}{BRANCH rep1126}
 75 @DATE            {EDIT}@DATE(YR,MO,DAY){LEFT 10}{BRANCH rep1126}
 76 @DAVG            {EDIT}@DAVG(input,offset,criterion){BRANCH rep4126}
 77 @DAY             {EDIT}@DAY(date_number){BRANCH rep3126}
 78 @DCOUNT          {EDIT}@DCOUNT(input,offset,criterion){BRANCH rep4126}
 79 @DDB             {EDIT}@DDB(cost,salvage,life,period){LEFT 25}
                     {BRANCH rep1126}
 80 @DGET            {EDIT}@DGET(input1,input2,...,inputn,field,criteria)
                     {LEFT 40}{BRANCH rep1126}
 81 @DMAX            {EDIT}@DMAX(input,offset,criterion){BRANCH rep4126}
 82 @DMIN            {EDIT}@DMIN(input,offset,criterion){BRANCH rep4126}
 83 @DQUERY          {EDIT}@DQUERY(function,ext-arguments){LEFT 23}
                     {BRANCH rep1126}
 84 @DSTD            {EDIT}@DSTD(input,offset,criterion){BRANCH rep4126}
 85 @DSTDS           {EDIT}@DSTDS(input1,input2,...,inputn,field,criteria)
                     {LEFT 40}
                     {BRANCH rep1126}
 86 @DSUM            {EDIT}@DSUM(input,offset,criterion){BRANCH rep4126}
 87 @DTVALUE         {EDIT}@DATEVALUE(date_string){BRANCH rep3126}
 88 @DVAR            {EDIT}@DVAR(input,offset,criterion){BRANCH rep4126}
 89 @DVARS           {EDIT}@DVARS(input1,input2,...,inputn,field,criteria)
                     {LEFT 40}{BRANCH rep1126}
 90 @ERR             {EDIT}@ERR{BRANCH rep1126}
 91 @EXACT           {EDIT}@EXACT(string1,string2){LEFT 16}{BRANCH rep1126}
 92 @EXP             {EDIT}@EXP(X){BRANCH rep2126}
 93 @FALSE           {EDIT}@FALSE{BRANCH rep1126}
 94 @FIND            {EDIT}@FIND(searched_string,string,offset){LEFT 30}
                     {BRANCH rep1126}
 95 @FV              {EDIT}@FV(pmt,int,term){LEFT 13}{BRANCH rep1126}
 96 @HLOOKUP         {EDIT}@HLOOKUP(X,range,offset_row_number){LEFT 26}
                     {BRANCH rep1126}
 97 @HOUR            {EDIT}@HOUR(time_number){BRANCH rep3126}
 98 @IF              {EDIT}@IF(condition,x,y){LEFT 14}{BRANCH rep1126}
 99 @INDEX           {EDIT}@INDEX(range,col,row){LEFT 14}{BRANCH rep1126}
100 @INT             {EDIT}@INT(X){BRANCH rep2126}
101 @IRR             {EDIT}@IRR(guess,range){BRANCH rep3126}
102 @ISERR           {EDIT}@ISERR(X){BRANCH rep1126}
103 @ISNA            {EDIT}@ISNA(X){BRANCH rep1126}
104 @ISNUMBER        {EDIT}@ISNUMBER(X){BRANCH rep1126}
105 @ISSTRING        {EDIT}@ISstring(X){BRANCH rep1126}
106 @LEFT            {EDIT}@LEFT(string,n){LEFT 9}{BRANCH rep1126}
107 @LENGTH          {EDIT}@LENGTH(string){LEFT 7}{BRANCH rep1126}
108 @LN              {EDIT}@LN(X){BRANCH rep2126}
109 @LOG             {EDIT}@LOG(X){BRANCH rep2126}
110 @LOWER           {EDIT}@LOWER(string){LEFT 7}{BRANCH rep1126}
111 @MAX             {EDIT}@MAX(list){LEFT 5}{BRANCH rep1126}
112 @MID             {EDIT}@MID(string,START NUMBER,N){LEFT 22}
                     {BRANCH rep1126}
113 @MIN             {EDIT}@MIN(list){LEFT 5}{BRANCH rep1126}
114 @MINUTE          {EDIT}@MINUTE(time_number){BRANCH rep3126}
115 @MOD             {EDIT}@MOD(X,Y){LEFT 4}{BRANCH rep1126}
116 @MONTH           {EDIT}@MONTH(date_number){BRANCH rep3126}
117 @N               {EDIT}@N(range){BRANCH rep1126}
```

```
@NA             {EDIT}@NA{BRANCH rep1126}
@NOW            {EDIT}@NOW{BRANCH rep1126}
@NPV            {EDIT}@NPV(int range){LEFT 10}{BRANCH rep1126}
@PI             {EDIT}@PI{BRANCH rep1126}
@PMT            {EDIT}@PMT(principal,interest,terms){LEFT 25}
                {BRANCH rep1126}
@PROPER         {EDIT}@PROPER(string){LEFT 7}{BRANCH rep1126}
@PV             {EDIT}@PV(pmt,int,term){LEFT 13}{BRANCH rep1126}
@RAND           {EDIT}@RAND{BRANCH rep1126}
@RATE           {EDIT}@RATE(fv,pv,term){LEFT 11}{BRANCH rep1126}
@REPEAT         {EDIT}@REPEAT(string,N){LEFT 9}{BRANCH rep1126}
@REPLACE        {EDIT}@REPLACE(old_string,start_loc,no_of_characters
                ,new_string){LEFT 9}{BRANCH rep1126}
@RIGHT          {EDIT}@RIGHT(string,n){LEFT 9}{BRANCH rep1126}
@ROUND          {EDIT}@ROUND(X,N){BRANCH rep1126}
@ROWS           {EDIT}@ROWS(range){BRANCH rep5126}
@S              {EDIT}@S(range){BRANCH rep5126}
@SECOND         {EDIT}@SECOND(time_number){BRANCH rep3126}
@SIN            {EDIT}@SIN(X){BRANCH rep2126}
@SLN            {EDIT}@SLN(cost,salvage,life){LEFT 18}{BRANCH rep1126}
@SQRT           {EDIT}@SQRT(X){BRANCH rep2126}
@STD            {EDIT}@STD(list){LEFT 5}{BRANCH rep1126}
@STRING         {EDIT}@STRING(X,N){LEFT 4}{BRANCH rep1126}
@SUM            {EDIT}@SUM(list){LEFT 5}{BRANCH rep1126}
@SYD            {EDIT}@SYD(cost,salvage,life,period){LEFT 25}
                {BRANCH rep1126}
@TAN            {EDIT}@TAN(X){BRANCH rep2126}
@TERM           {EDIT}@TERM(pmt,int,fv){LEFT 11}{BRANCH rep1126}
@TIME           {EDIT}@TIME(HR,MIN,SEC){LEFT 11}{BRANCH rep1126}
@TIMEVALUE      {EDIT}@TIMEVALUE(time_string){BRANCH rep3126}
@TRIM           {EDIT}@TRIM(string){LEFT 7}{BRANCH rep1126}
@TRUE           {EDIT}@TRUE{BRANCH rep1126}
@UPPER          {EDIT}@UPPER(string){LEFT 7}{BRANCH rep1126}
@VALUE          {EDIT}@VALUE(string){LEFT 7}{BRANCH rep1126}
@VAR            {EDIT}@VAR(list){LEFT 5}{BRANCH rep1126}
@VLOOKUP        {EDIT}@VLOOKUP(X,range,col_number){LEFT 19}
                {BRANCH rep1126}
@YEAR           {EDIT}@YEAR(DATE number){BRANCH rep3126}
{BEEP}          {EDIT}{{}BEEP [number]}{LEFT 9}{BRANCH rep1126}
{BLANK}         {EDIT}{{}BLANK location}{LEFT 9}{BRANCH rep1126}
{BRANCH}        {EDIT}{{}BRANCH location}{LEFT 9}{BRANCH rep1126}
{BREAKOFF}      {EDIT}{{}BREAKOFF}{BRANCH rep1126}
{BREAKON}       {EDIT}{{}BREAKON}{BRANCH rep1126}
{CALC}          {EDIT}{{}CALC}{BRANCH rep1126}
{CONTENTS}      {EDIT}{{}CONTENTS destination_location,source_location
                ,width,format_number}{LEFT 57}{BRANCH rep1126}
{DEFINE}        {EDIT}{{}DEFINE location:TYPE1,LOCATION:TYPE2,....}
                {LEFT 35}{BRANCH rep1126}
{DISPATCH}      {EDIT}{{}DISPATCH location}{LEFT 9}{BRANCH rep1126}
{DOWN}          {EDIT}{{}DOWN number_of_cells}{LEFT 16}{BRANCH rep1126}
{EDIT}          {EDIT}{{}EDIT}{BRANCH rep1126}
{END}           {EDIT}{{}END}{BRANCH rep1126}
{ESC}           {EDIT}{{}ESC}{BRANCH rep1126}
{FILESIZE}      {EDIT}{{}FILESIZE location}{LEFT 9}{BRANCH rep1126}
{FORBREAK}      {EDIT}{{}FORBREAK}{BRANCH rep126}
{FOR}           {EDIT}{{}FOR COUNTER,START,STOP,SKIP,SUBROUTINE}
                {BRANCH REP126}
{GETLABEL}      {EDIT}{{}GETLABEL "string prompt",location}{LEFT 24}
                {BRANCH rep1126}
{GETNUMBER}     {EDIT}{{}GETNUMBER "string prompt",location}{LEFT 24}
                {BRANCH rep1126}
{GETPOS}        {EDIT}{{}GETPOS }{BRANCH rep1126}
{GET}           {EDIT}{{}GET location}{LEFT 9}{BRANCH rep1126}
{GOTO}          {EDIT}{{}GOTO}location{LEFT 8}{BRANCH rep1126}
{GRAPH}         {EDIT}{{}GRAPH}{BRANCH REP126}
{HOME}          {EDIT}{{}HOME}{BRANCH rep1126}
{IF}            {EDIT}{{}IF condition}{LEFT 10}{BRANCH rep1126}
{INDICATE}      {EDIT}{{}INDICATE [string]}{LEFT 9}{BRANCH rep1126}
{LEFT}          {EDIT}{{}LEFT number_of_cells}{LEFT 16}{BRANCH rep1126}
{LET}           {EDIT}{{}LET location,number/string/formula}{LEFT 31}
                {BRANCH rep1126}
```

```
179 {LOOK}          {EDIT}{{}LOOK location}{LEFT 9}{BRANCH rep1126}
180 {MENUBRANCH}   {EDIT}{{}MENUBRANCH menu}{LEFT 5}{BRANCH rep1126}
181 {MENUCALL}     {EDIT}{{}MENUCALL menu}{LEFT 5}{BRANCH rep1126}
182 {NAME}         {EDIT}{{}NAME}{BRANCH rep1126}
183 {ONERROR}      {EDIT}{{}ONERROR SUBROUTINE}{LEFT 11}{BRANCH rep1126}
184 {OPEN}         {EDIT}{{}OPEN FILE}{BRANCH rep1126}
185 {PANELOFF}     {EDIT}{{}PANELOFF}{BRANCH rep1126}
186 {PANELON}      {EDIT}{{}PANELON}{BRANCH rep1126}
187 {PGDN}         {EDIT}{{}PGDN}{BRANCH rep1126}
188 {PGUP}         {EDIT}{{}PGUP}{BRANCH rep1126}
189 {PUT}          {EDIT}{{}PUT location,col,row,number/string}{LEFT 31}
                   {BRANCH rep1126}
190 {QUERY}        {EDIT}{{}QUERY}{BRANCH rep1126}
191 {QUIT}         {EDIT}{{}QUIT}{BRANCH rep1126}
192 {READLN}       {EDIT}{{}READLN location}{LEFT 9}{BRANCH rep1126}
193 {READ}         {EDIT}{{}READ byte_count,location}{LEFT 20}
                   {BRANCH rep1126}
194 {RECALCOL}     {EDIT}{{}RECALCOL cell/range}{LEFT 11}{BRANCH rep1126}
195 {RECALC}       {EDIT}{{}RECALC cell/range}{LEFT 11}{BRANCH rep1126}
196 {RESTART}      {EDIT}{{}RESTART}{BRANCH REP126}
197 {RIGHT}        {EDIT}{{}RIGHT number_of_cells}{LEFT 16}{BRANCH rep1126}
198 {UP}           {EDIT}{{}UP number_of_cells}{LEFT 16}{BRANCH rep1126}
199 {WAIT}         {EDIT}{{}WAIT(@NOW+.0)}{BRANCH rep1126}
200 {WINDOWSOFF}   {EDIT}{{}WINDOWSOFF}{BRANCH rep1126}
201 {WINDOWSON}    {EDIT}{{}WINDOWSON}{BRANCH rep1126}
202 {WINDOW}       {EDIT}{{}WINDOW}{BRANCH rep1126}
203 {WRITELN}      {EDIT}{{}WRITELN "string"}{LEFT 8}{BRANCH rep1126}
204 {WRITE}        {EDIT}{{}WRITE "string"}{LEFT 8}{BRANCH rep1126}
205 "ASCII         {GOTO}tableb126~Point to the character to type and
                  press [RETURN]...{GET key1126}{ESC}
206 !             {IF key1126="~"}{ESC}/RV~key1126~{back126}{EDIT}{key1126}
                  {BRANCH rep1126}
207 !             {IF key1126="{ESC}"}{ESC}{back126}{BRANCH macrohlp}
208 !             {key1126}{?}~/RV~key1126~{back126}{EDIT}{key1126}
                  {BRANCH rep1126}
209 !
210 !
211 tableb126              @       `       ?       f       ?       à
212 !             !        A       a       ?       ¡       ?       á
213 !             "        B       b       ?       ¢       ?       â
214 !             #        C       c       ~       £       ?       ?
215 !             $        D       d       ?       ?       Ä       ä
216 !             %        E       e       ?       ¥       Å       å
217 !             &        F       f       ?       p       Æ       æ
218 !             '        G       g       ?       §       Ç       ç
219 !             (        H       h       ?       ¤       ?       è
220 !             )        I       i       ?       ?       É       é
221 !             *        J       j       ?       ª       ?       ê
222 !             +        K       k       ?       «       ?       ë
223 !             ,        L       l       ?       ?       ?       ì
224 !             -        M       m       ?       ¶       ?       í
225 !             .        N       n       ?       _       ?       î
226 !             /        O       o               ¯       ?       ï
227 !             0        P       p       ?       °       ?       ?
228 !             1        Q       q       ?       ±       Ñ       ñ
229 !             2        R       r       ?       ²       ?       ò
230 !             3        S       s       ?       ?       ?       ó
231 !             4        T       t       ~       ?       ?       ô
232 !             5        U       u       ?       µ       ?       ?
233 !             6        V       v       ?       ¶       Ö       ö
234 !             7        W       w       ?       ·       ?       ?
235 !             8        X       x       ?       ?       ?
236 !             9        Y       y       ?       ?       ?       ù
237 !             :        Z       z               º       ?       ú
238 !             ;        [       {       ?       »       ?       û
239 !             <        \       |       ?       ¼       Ü       ü
240 !             =        ]       }       ?       ½       ?       ÿ
241 !             >        ^       ~       ?       _       ?       ?
242 !             ?        _       ?       ?       ¿       ß       ?
```

The codes in the cells B12, B16 and B27 are the result of the following dynamic string formulas. If you intend to key this macro into 1-2-3, you need to key the code as it appears here, NOT the code as it appears in the main listing.

```
12 form126          +"{err1126}{LET back1126,@CELLPOINTER("""&B32&""")}~
                    {PANELOFF}{GOTO}{NAME}{PANELON}{NAME}{GET key126}
                    {IF key126=""{ESC}""}{ESC 8}{QUIT}"
16 hhh1126          +"~{LET routin126,@CELLPOINTER("""&B32&""")}~{back126}
                    {DISPATCH routin126}~"
32 loc126           @IF(@LEFT(B30,1)<>"@","coord","address")
```

The ASCII characters in the B211..H242 range are the result of the `@CHAR(code #)` function that returns the ASCII character for every ASCII code number. You can use this macro to continuously write macros and formulas, the macro displays all the macro commands, attributes and Lotus function's syntax on the screen. Point to the command and press ENTER, then the macro adds the command to the text at the current cell. Using this macro, you can write macros and formulas almost without typing. It saves typing errors and you do not have to remember the commands or the function's syntax.

	A	B	C	D	E
11	MACROHLP	{PANELOFF}{LET rel126,@INFO("release")}~{RECALC loc126}			
		{RECALC form126}{RECALC hhh1126}/RNC!~~/RND!~{PANELON}			
		{IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER("type")="v"}			
		{EDIT}{HOME}'~			

The macro starts with `{PANELOFF}` to freeze the panel activity, then `{LET rel126,@INFO("release")}~` to store the result of the `@INFO("release")` function in cell [rel126]. Later, the macro uses it to determine if you are using a 2-D or a 3-D Lotus release. Next the macro issues `{RECALC loc126}{RECALC form126}{RECALC hhh1126}` to update the dynamic string formulas in cells [loc12], [form126] and [hhh1126]. When you use the instructions in the A5..A7 range to assign the range names for this macro, it also assigns the "!" range name. Therefore the macro uses `/RNC!~~/RND!~` to safely delete the "!" range name. Now the macro issues `{PANELON}` to resume panel activity and

```
{IF @CELLPOINTER("type")="b"#OR#@CELLPOINTER("type")="v"}
```

to check if the current cell is empty or contains a value. If so, the macro issues `{EDIT}{HOME}'~` to change the cell's content into a label.

	A	B	C	D	E
12	form126	{err1126}{LET back1126,@CELLPOINTER("coord")}~{PANELOFF}			
		{GOTO}{NAME}{PANELON}{NAME}{GET key126}{IF key126="{ESC}"}			
		{ESC 8}{QUIT}			
13	!	{IF key126="~"}{BRANCH hhh1126}			
14	key126	{DOWN}			
15	hhh126	{?}~			
16	hhh1126	~{LET routin126,@CELLPOINTER("coord")}~{back126}			
		{DISPATCH routin126}~			

The code in cell [form126] is a result of:

```
12 form126          +"{err1126}{LET back1126,@CELLPOINTER("""&B32&""")}~
                    {PANELOFF}{GOTO}{NAME}{PANELON}{NAME}{GET key126}
                    {IF key126=""{ESC}""}{ESC 8}{QUIT}"
```

We can see that the dynamic part of the formula is the value of the B32 cell, [loc126], which contains:

```
 32 loc126          @IF(@LEFT(B30,1)<>"@","coord","address")
```

This formula also depends on the content of the B30 cell, [rel126], which holds the result of the `@INFO("release")` function. If "`@`" is the first character of the label in [rel126] you are using a 2-D Lotus release, otherwise you are using a 3-D Lotus release. If you are using a 2-D release, the result of the formula in [loc126] is the "`address`" string, otherwise it is the "`coord`" string. Therefore the result of the formula in cell [form126] will be:

```
     A              B                C              D              E
12 form126         {err1126}{LET back1126,@CELLPOINTER("coord")}~{PANELOFF}
                   {GOTO}{NAME}{PANELON}{NAME}{GET key126}{IF key126="{ESC}"}
                   {ESC 8}{QUIT}
```

if you are using a 3-D Lotus release, or

```
     A              B                C              D              E
12 form126         {err1126}{LET back1126,@CELLPOINTER("address")}~{PANELOFF}
                   {GOTO}{NAME}{PANELON}{NAME}{GET key126}{IF key126="{ESC}"}
                   {ESC 8}{QUIT}
```

if you are using a 2-D Lotus release. The macro issues `{err1126}` to activates `{ONERROR err2126}` which traps and handles errors occurring while a macro is running. Normally, when an error occurs while a macro is running, 1-2-3 displays an error message and, except for background errors, changes the mode indicator to ERROR and ends the macro. However, if `{ONERROR}` is in effect when the error occurs, 1-2-3 returns to READY mode and branches to the [err2126] routine for further instructions. Before the macro continues, it issues `{LET back1126,@CELLPOINTER("address")}~` which store the current cell pointer position in cell [back1126]. Later, the macro uses this address to bring the cell pointer to the same location.

The macro starts a set of macro commands that display a full screen list of all the range names in the macro. If you look at the A46..A205 range, you can see a list of all the macro names and functions that Lotus supports. When you assign the range names as instructed, for the A5..A7 range and use `/RNLR [END] [DOWN] [ENTER]` the macro assigns these labels as range names to the B46..B205 cells. To clarify, let's look at

```
     A              B                C              D              E
60 @ABS            {EDIT}@ABS(X){BRANCH rep2126}
61 @ACOS           {EDIT}@ACOS(X){BRANCH rep2126}
```

A60..B61 range and at

```
     A              B                C              D              E
172 {GOTO}         {EDIT}{{}GOTO}location{LEFT 8}{BRANCH rep1126}
173 {GRAPH}        {EDIT}{{}GRAPH}{BRANCH REP126}
```

A172..B173 range that represent the rest of the A46..B205 range. When you use `/RNLR [END] [DOWN] [ENTER]` the macro assigns the `@ABS` range name to the B60 cell and the `{GRAPH}` range name to the B173 cell with all the other range names. The result is a list of 150 range names that resemble the macro commands, attributes and functions that Lotus supports. Next the macro issues `{PANELOFF}{GOTO}{NAME}{PANELON}{NAME}`, which display a full screen list of the all the range names in the macro, and issues `{GET key126}` that halts the `{GOTO}` process and waits until you press a key.

You may think that what you see on-screen are Lotus commands, but they are the range names that look like Lotus commands and functions. Now the macro uses a series of {IF} commands to check which key you pressed. The macro starts with {IF key126={ESC}"} {ESC 8} {QUIT} that check if you pressed ESC. If so, the macro issues {ESC 8} to return to READY mode and then {QUIT} and quits.

	A	B	C	D	E
13	!	{IF key126="~"}{BRANCH hhh1126}			
14	key126	{DOWN}			
15	hhh126	{?}~			
16	hhh1126	~{LET routin126,@CELLPOINTER("coord")}~{back126} {DISPATCH routin126}~			

If you pressed ENTER, the macro issues {BRANCH hhh1126} that starts the [hhh1126] routine. The code in the [hhh1126] routine is also a result of a dynamic string formula similar to the previous formulas we have seen. Before we get into the [hhh1126] routine, let's see what happens if you press keys other then ESC or ENTER. When you press a key, {GET key126} stores this key's macro code in cell [key126], which happens to be the next cell that the macro processes. For example, if you press the DOWN key, [key126] will contain {DOWN} that becomes a dynamic part of the macro code. Therefore the macro processes the key that you pressed as you intended it to. Next the macro issues {?} that allows you to type a range name or an address to go to, or to point to one of the range names on the screen and press ENTER. If you make an error in a range name or an address, the [err2126] routine is activated and routes the macro to correct the error.

Let's assume that you point to the @ABS range name and press ENTER. The [hhh1126] routine starts with the tilde "~" that finishes the {GOTO} process and puts the cell pointer on B60, [@ABS]. Then the macro issues {LET routin126,@CELLPOINTER("coord")}~ to store the current cell position (the B60 address) in cell [routin126]. Now the macro issues the {back126} routine command that activates the [back126] routine to send the cell pointer to the point of origin before the {GOTO} process started.

	A	B	C	D	E
26	back126	{GOTO}			
27	back1126	$A:$A$85			
28	!	~			

Cell [back1126] holds the address of the cell pointer before the macro started. The [hhh1126] routine continues with {DISPATCH routin126}. This command activates the routine that its address or range name is stored in the cell named [routin126]. Because this cell holds the B60 address of cell [@ABS] in our example, the [@ABS] routine is activated.

	A	B	C	D	E
60	@ABS	{EDIT}@ABS(X){BRANCH rep2126}			

The [@ABS] routine starts with {EDIT} and then types the "@ABS(X)" string to the panel. If the current cell is empty the panel shows:

```
@ABS(X)
```

Next the routine issues {BRANCH rep2126} to start the [rep2126] routine.

	A	B	C	D	E
36	rep2126	{LEFT 2}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}			

The [rep2126] routine starts with the {LEFT 2} macro command that moves the cursor to the "x" (marked by "^") and the panel shows:

```
@ABS(X)
^
```

The macro continues and issues {?} which allows you to change the "x" in the parenthesis into a value or formula. When you finish and press ENTER, the macro issues {DOWN}{UP} to insure that the content of the panel is written to the current cell. Now the macro issues {macrohlp} to allow you to continue typing to the panel by picking the macro or function syntax from the screen. This was an example of picking a function from the screen. The next example shows how to pick a macro command syntax from the screen. For example, if you choose the [{GOTO}] range name from the screen, the [{GOTO}] routine is activated.

	A	B	C	D	E
172	{GOTO}	{EDIT}{{}GOTO}location{LEFT 8}{BRANCH rep1126}			

The [{GOTO}] routine starts with {EDIT} and then issues {{} that writes the "{" character to the panel and then types the "GOTO}location" string to the panel. Remember that {GOTO} is a Lotus reserved macro command, therefore we could not type it to the panel directly because Lotus would execute it as a command. Therefore we had to type first the opening curled parenthesis and then the "GOTO}" string. If the panel was empty before, it will now show:

```
{GOTO}location
      ^
```

The {LEFT 8} macro command places the cursor under the "l" character (marked by "^") and waits until you replace the "location" string with your own. This macro contains 150 routines and functions such as {GOTO} and @ABS. Using these routines and functions, you can write macros and formulas almost without typing.

	A	B	C	D	E
205	"ASCII	{GOTO}tableb126~Point to the character to type and press [RETURN]...{GET key1126}{ESC}			
206	!	{IF key1126="~"}{ESC}/RV~key1126~{back126}{EDIT}{key1126} {BRANCH rep1126}			
207	!	{IF key1126="{ESC}"}{ESC}{back126}{BRANCH macrohlp}			
208	!	{key1126}{?}~/RV~key1126~{back126}{EDIT}{key1126} {BRANCH rep1126}			

When you pick the ["ASCII] range name from the screen, the macro issues {GOTO} tableb126~ that sends the cell pointer to the ASCII table in the B211..H242 range and the ASCII table appears on the screen. Now the macro types

```
Point to the character to type and press [RETURN]...
```

to the panel because it cannot find any valid macro command in this message. Next it issues {GET key1126} that halts the macro and waits until you press a key. The macro stores the key in cell [key1126] and then issues {ESC}, which clears the panel from the message before Lotus writes it to the current cell. If you press ENTER, Lotus stores the tilde "~" in cell [key1126]. The {IF key1126="~"} macro condition is true; therefore the macro issues /RV~key1126~ that copies the ASCII character as a label to [key1126]. The next macro command is {back126} that sends the cell pointer to the point of origin. To add the ASCII character to the

panel, the macro issues {EDIT}. Now let's assume that the ASCII character is a "+" and earlier you picked the @ABS name and changed the "x" to "-8". Therefore the panel displays:

```
@ABS(-8)_
```

where the underscore represents the cursor position. Now the macro issues the {key1126} routine command, but this routine contains only the "+" character. Therefore, when this routine is activated, Lotus types a "+" to the panel which now displays:

```
@ABS(-8)+
```

Now you can continue and pick another function like the @COS function and change the "x" to "0.5". Therefore the panel will show:

```
@ABS(-8)+@COS(0.5)
```

This way, you can write macros or functions by picking elements from the screen. The elements can be macro commands, functions, attributes and ASCII characters. If you do not press ENTER, the macro issues {IF key1126="{ESC}"}. If you did press ESC, the macro issues {ESC} that clear the message from the panel and then use {back126}{BRANCH macrohlp} to send the cell pointer back to its point of origin and reactivate the macro so you can continue picking text from the screen. If you press any other key, such as one of the direction keys, the macro issues {key1126} that executes the key that you pressed and issues {?} that allows you to move the cell pointer to any text on the screen and press ENTER to pick this text. The rest of the code is the same as before.

The last range name is the ["ABORT] range name. When you choose this text from the screen and press ENTER, the ["ABORT] routine is activated.

	A	B	C	D	E
42	"ABORT	{IF @LEFT(@CELLPOINTER("contents"),1)="@"#OR#@LEFT (@CELLPOINTER("contents"),1)="-"#OR#@LEFT(@CELLPOINTER ("contents"),1)="+"}{EDIT}{HOME}{DEL}~{BRANCH ret126}			

Before you can abort the macro, it is be wise to check if you want to write a formula or a macro. If you want to write a formula, the macro has to delete the apostrophe "'" in front of the text in the cell to turn it into an active formula. Therefore the macro issues:

```
{IF @LEFT(@CELLPOINTER("contents"),1)="@"#OR#@LEFT(@CELLPOINTER(
    "contents"),1)="-"#OR#@LEFT(@CELLPOINTER("contents"),1)="+"}
```

to check if "@" or "+" or "-" is the first character in the cell. If so, it is logical to assume that you wanted to write a formula, therefore the macro issues {EDIT}{HOME}{DEL}~ to delete the apostrophe "'" in front of the text in the cell and turn it into an active formula. Last, the macro uses {BRANCH ret126} to branch to an empty cell to quit.

Because this macro is a complicated macro we bring here the complete list of cell formulas and contents in this macro.

```
A1: U '*---A macro to HELP WRITING MACROS and FORMULAS, allows you to
A2: U '    CONTINUOUSLY write every Lotus function or command by POINTING
      to
A3: U '    the syntax on display and SHOOTING (press RETURN). There is no
      more
```

```
A4: U '    typing errors or a need to remember the exact syntax.
A5: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A6: '    range names in this column (starts with the \Z macro name)
A7: '*---Hold the [ALT] key and press [Z] to activate the macro
A8: U '                    THIS MACRO WORKS IN LOTUS 2.0 AND UP
A9: U '!                SEE MORE DETAILS IN THE README.PRN FILE
A10: U '\Z
B10: '{BREAKON}
E10: '
F10: '
A11: U 'MACROHLP
B11: '{PANELOFF}{LET rel126,@INFO("release")}~{RECALC loc126}{RECALC form126}
     {RECALC hhh1126}/RNC!~~/RND!~{PANELON}{IF @CELLPOINTER("type")="b"#OR#
     @CELLPOINTER("type")="v"}{EDIT}{HOME}'~
A12: 'form126
B12: U +"{err1126}{LET back1126,@CELLPOINTER("""&B32&""")}~{PANELOFF}
       {GOTO}{NAME}{PANELON}{NAME}{GET key126}{IF key126=""{ESC}""}{ESC 8}
       {QUIT}"
A13: '!
B13: '{IF key126="~"}{BRANCH hhh1126}
A14: 'key126
B14: '{DOWN}
E14: '
A15: 'hhh126
B15: '{?}~
A16: 'hhh1126
B16: U +"~{LET routin126,@CELLPOINTER("""&B32&""")}~{back126}
       {DISPATCH routin126}~"
A17: '!
A18: 'key1126
B18: U '@
A19: '!
A20: 'err1126
B20: '{ONERROR err2126}
A21: '!
A22: 'err2126
B22: '{BEEP}INVALID MACRO NAME! PRESS ANY KEY TO CONTINUE ...{GET key126}
     {ESC}{BRANCH \Z}}
A23: '!
A24: 'routin126
B24: '$A:$B$85
D24: '
A25: '!
A26: 'back126
B26: '{GOTO}
A27: 'back1126
B27: '$A:$A$85
A28: '!
B28: '~
A29: '!
A30: 'rel126
B30: '3.00.00
A31: '!
A32: 'loc126
B32: @IF(@LEFT(B30,1)<>"@","coord","address")
A33: '!
A34: 'rep126
B34: '{LEFT}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}
A35: 'rep1126
B35: '{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}
A36: 'rep2126
B36: '{LEFT 2}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}
A37: 'rep3126
B37: '{LEFT 12}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}
A38: 'rep4126
B38: '{LEFT 23}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}
A39: 'rep5126
B39: '{LEFT 6}{?}~{DOWN}{UP}{macrohlp}{BRANCH ret126}
A40: '!
A41: '!
A42: '"ABORT
```

```
B42: '{IF @LEFT(@CELLPOINTER("contents"),1)="@"#OR#@LEFT(@CELLPOINTER
     ("contents"),1)="-"#OR#@LEFT(@CELLPOINTER("contents"),1)="+"}{EDIT}
     {HOME}{DEL}~{BRANCH ret126}
A43: '!
A44: 'ret126
A45: '!
A46: '"ADDRESS"
B46: '{EDIT}ADDRESS",{BRANCH rep1126}
A47: '"COL"
B47: '{EDIT}COL",{BRANCH rep1126}
A48: '"COLOR"
B48: '{EDIT}color",{BRANCH rep1126}
A49: '"CONTENTS"
B49: '{EDIT}CONTENTS",{BRANCH rep1126}
A50: '"COORD"
B50: '{EDIT}coord",{BRANCH rep1126}
A51: '"FILENAME"
B51: '{EDIT}filename",{BRANCH rep1126}
A52: '"PORMAT"
B52: '{EDIT}format",{BRANCH rep1126}
A53: '"PARENTHESES"
B53: '{EDIT}parentheses",{BRANCH rep1126}
A54: '"PREFIX"
B54: '{EDIT}PREFIX",{BRANCH rep1126}
A55: '"PROTECT"
B55: '{EDIT}protect",{BRANCH rep1126}
A56: '"ROW"
B56: '{EDIT}ROW",{BRANCH rep1126}
A57: '"SHEET"
B57: '{EDIT}sheet",{BRANCH rep1126}
A58: '"type"
B58: '{EDIT}TYPE",{BRANCH rep1126}
A59: '"WIDTH"
B59: '{EDIT}WIDTH",{BRANCH rep1126}
A60: '@ABS
B60: '{EDIT}@ABS(X){BRANCH rep2126}
A61: '@ACOS
B61: '{EDIT}@ACOS(X){BRANCH rep2126}
A62: '@ASIN
B62: '{EDIT}@ASIN(X){BRANCH rep2126}
A63: '@ATAN
B63: '{EDIT}@ATAN(X){BRANCH rep2126}
A64: '@ATAN2
B64: '{EDIT}@ATAN2(X,Y){BRANCH rep2126}
A65: '@CELL
B65: '{EDIT}@CELL("{BRANCH rep1126}
A66: '@CELLPOINTER
B66: '{EDIT}@CELLPOINTER("{BRANCH rep1126}
A67: '@CHAR
B67: '{EDIT}@CHAR(CODE){BRANCH rep1126}
A68: '@CHOOSE
B68: '{EDIT}@CHOOSE(X,"","","",""){BRANCH rep1126}
A69: '@CODE
B69: '{EDIT}@CODE(character){LEFT 10}{BRANCH rep1126}
A70: '@COLS
B70: '{EDIT}@COLS(range){BRANCH rep5126}
A71: '@COORD
B71: '{EDIT}@COORD(worksheet,column,row,absolute){LEFT 30}{BRANCH rep1126}
A72: '@COS
B72: '{EDIT}@COS(X){BRANCH rep2126}
A73: '@COUNT
B73: '{EDIT}@COUNT(range){BRANCH rep5126}
A74: '@CTERM
B74: '{EDIT}@CTERM(interest,future-value,present-value){LEFT 36}
     {BRANCH rep1126}
A75: '@DATE
B75: '{EDIT}@DATE(YR,MO,DAY){LEFT 10}{BRANCH rep1126}
A76: '@DAVG
B76: '{EDIT}@DAVG(input,offset,criterion){BRANCH rep4126}
A77: '@DAY
B77: '{EDIT}@DAY(date_number){BRANCH rep3126}
```

```
A78: '@DCOUNT
B78: '{EDIT}@DCOUNT(input,offset,criterion){BRANCH rep4126}
I78: '
A79: '@DDB
B79: '{EDIT}@DDB(cost,salvage,life,period){LEFT 25}{BRANCH rep1126}
A80: '@DGET
B80: '{EDIT}@DGET(input1,input2,...,inputn,field,criteria){LEFT 40}
     {BRANCH rep1126}
A81: '@DMAX
B81: '{EDIT}@DMAX(input,offset,criterion){BRANCH rep4126}
A82: '@DMIN
B82: '{EDIT}@DMIN(input,offset,criterion){BRANCH rep4126}
A83: '@DQUERY
B83: '{EDIT}@DQUERY(function,ext-arguments){LEFT 23}{BRANCH rep1126}
A84: '@DSTD
B84: '{EDIT}@DSTD(input,offset,criterion){BRANCH rep4126}
A85: '@DSTDS
B85: '{EDIT}@DSTDS(input1,input2,...,inputn,field,criteria){LEFT 40}
     {BRANCH rep1126}
A86: '@DSUM
B86: '{EDIT}@DSUM(input,offset,criterion){BRANCH rep4126}
A87: '@DTVALUE
B87: '{EDIT}@DATEVALUE(date_string){BRANCH rep3126}
A88: '@DVAR
B88: '{EDIT}@DVAR(input,offset,criterion){BRANCH rep4126}
A89: '@DVARS
B89: '{EDIT}@DVARS(input1,input2,...,inputn,field,criteria){LEFT 40}
     {BRANCH rep1126}
A90: '@ERR
B90: '{EDIT}@ERR{BRANCH rep1126}
A91: '@EXACT
B91: '{EDIT}@EXACT(string1,string2){LEFT 16}{BRANCH rep1126}
A92: '@EXP
B92: '{EDIT}@EXP(X){BRANCH rep2126}
A93: '@FALSE
B93: '{EDIT}@FALSE{BRANCH rep1126}
A94: '@FIND
B94: '{EDIT}@FIND(searched_string,string,offset){LEFT 30}{BRANCH rep1126}
A95: '@FV
B95: '{EDIT}@FV(pmt,int,term){LEFT 13}{BRANCH rep1126}
A96: '@HLOOKUP
B96: '{EDIT}@HLOOKUP(X,range,offset_row_number){LEFT 26}{BRANCH rep1126}
A97: '@HOUR
B97: '{EDIT}@HOUR(time_number){BRANCH rep3126}
A98: '@IF
B98: '{EDIT}@IF(condition,x,y){LEFT 14}{BRANCH rep1126}
A99: '@INDEX
B99: '{EDIT}@INDEX(range,col,row){LEFT 14}{BRANCH rep1126}
A100: '@INT
B100: '{EDIT}@INT(X){BRANCH rep2126}
A101: '@IRR
B101: '{EDIT}@IRR(guess,range){BRANCH rep3126}
A102: '@ISERR
B102: '{EDIT}@ISERR(X){BRANCH rep1126}
A103: '@ISNA
B103: '{EDIT}@ISNA(X){BRANCH rep1126}
A104: '@ISNUMBER
B104: '{EDIT}@ISNUMBER(X){BRANCH rep1126}
A105: '@ISSTRING
B105: '{EDIT}@ISstring(X){BRANCH rep1126}
A106: '@LEFT
B106: '{EDIT}@LEFT(string,n){LEFT 9}{BRANCH rep1126}
A107: '@LENGTH
B107: '{EDIT}@LENGTH(string){LEFT 7}{BRANCH rep1126}
A108: '@LN
B108: '{EDIT}@LN(X){BRANCH rep2126}
A109: '@LOG
B109: '{EDIT}@LOG(X){BRANCH rep2126}
A110: '@LOWER
B110: '{EDIT}@LOWER(string){LEFT 7}{BRANCH rep1126}
A111: '@MAX
```

```
B111: '{EDIT}@MAX(list){LEFT 5}{BRANCH rep1126}
A112: '@MID
B112: '{EDIT}@MID(string,START NUMBER,N){LEFT 22}{BRANCH rep1126}
A113: '@MIN
B113: '{EDIT}@MIN(list){LEFT 5}{BRANCH rep1126}
A114: '@MINUTE
B114: '{EDIT}@MINUTE(time_number){BRANCH rep3126}
A115: '@MOD
B115: '{EDIT}@MOD(X,Y){LEFT 4}{BRANCH rep1126}
A116: '@MONTH
B116: '{EDIT}@MONTH(date_number){BRANCH rep3126}
A117: '@N
B117: '{EDIT}@N(range){BRANCH rep1126}
A118: '@NA
B118: '{EDIT}@NA{BRANCH rep1126}
A119: '@NOW
B119: '{EDIT}@NOW{BRANCH rep1126}
A120: '@NPV
B120: '{EDIT}@NPV(int range){LEFT 10}{BRANCH rep1126}
A121: '@PI
B121: '{EDIT}@PI{BRANCH rep1126}
A122: '@PMT
B122: '{EDIT}@PMT(principal,interest,terms){LEFT 25}{BRANCH rep1126}
A123: '@PROPER
B123: '{EDIT}@PROPER(string){LEFT 7}{BRANCH rep1126}
A124: '@PV
B124: '{EDIT}@PV(pmt,int,term){LEFT 13}{BRANCH rep1126}
A125: '@RAND
B125: '{EDIT}@RAND{BRANCH rep1126}
A126: '@RATE
B126: '{EDIT}@RATE(fv,pv,term){LEFT 11}{BRANCH rep1126}
A127: '@REPEAT
B127: '{EDIT}@REPEAT(string,N){LEFT 9}{BRANCH rep1126}
A128: '@REPLACE
B128: '{EDIT}@REPLACE(old_string,start_loc,no_of_characters,new_string)
      {LEFT 49}{BRANCH rep1126}
A129: '@RIGHT
B129: '{EDIT}@RIGHT(string,n){LEFT 9}{BRANCH rep1126}
A130: '@ROUND
B130: '{EDIT}@ROUND(X,N){BRANCH rep1126}
A131: '@ROWS
B131: '{EDIT}@ROWS(range){BRANCH rep5126}
A132: '@S
B132: '{EDIT}@S(range){BRANCH rep5126}
A133: '@SECOND
B133: '{EDIT}@SECOND(time_number){BRANCH rep3126}
A134: '@SIN
B134: '{EDIT}@SIN(X){BRANCH rep2126}
A135: '@SLN
B135: '{EDIT}@SLN(cost,salvage,life){LEFT 18}{BRANCH rep1126}
A136: '@SQRT
B136: '{EDIT}@SQRT(X){BRANCH rep2126}
A137: '@STD
B137: '{EDIT}@STD(list){LEFT 5}{BRANCH rep1126}
A138: '@STRING
B138: '{EDIT}@STRING(X,N){LEFT 4}{BRANCH rep1126}
A139: '@SUM
B139: '{EDIT}@SUM(list){LEFT 5}{BRANCH rep1126}
A140: '@SYD
B140: '{EDIT}@SYD(cost,salvage,life,period){LEFT 25}{BRANCH rep1126}
A141: '@TAN
B141: '{EDIT}@TAN(X){BRANCH rep2126}
A142: '@TERM
B142: '{EDIT}@TERM(pmt,int,fv){LEFT 11}{BRANCH rep1126}
A143: '@TIME
B143: '{EDIT}@TIME(HR,MIN,SEC){LEFT 11}{BRANCH rep1126}
A144: '@TIMEVALUE
B144: '{EDIT}@TIMEVALUE(time_string){BRANCH rep3126}
A145: '@TRIM
B145: '{EDIT}@TRIM(string){LEFT 7}{BRANCH rep1126}
A146: '@TRUE
```

```
B146: '{EDIT}@TRUE{BRANCH rep1126}
A147: '@UPPER
B147: '{EDIT}@UPPER(string){LEFT 7}{BRANCH rep1126}
A148: '@VALUE
B148: '{EDIT}@VALUE(string){LEFT 7}{BRANCH rep1126}
A149: '@VAR
B149: '{EDIT}@VAR(list){LEFT 5}{BRANCH rep1126}
A150: '@VLOOKUP
B150: '{EDIT}@VLOOKUP(X,range,col_number){LEFT 19}{BRANCH rep1126}
A151: '@YEAR
B151: '{EDIT}@YEAR(DATE number){BRANCH rep3126}
A152: '{BEEP}
B152: '{EDIT}{{}BEEP [number]}{LEFT 9}{BRANCH rep1126}
A153: '{BLANK}
B153: '{EDIT}{{}BLANK location}{LEFT 9}{BRANCH rep1126}
A154: '{BRANCH}
B154: '{EDIT}{{}BRANCH location}{LEFT 9}{BRANCH rep1126}
A155: '{BREAKOFF}
B155: '{EDIT}{{}BREAKOFF}{BRANCH rep1126}
A156: '{BREAKON}
B156: '{EDIT}{{}BREAKON}{BRANCH rep1126}
A157: '{CALC}
B157: '{EDIT}{{}CALC}{BRANCH rep1126}
A158: '{CONTENTS}
B158: '{EDIT}{{}CONTENTS destination_location,source_location,width,
      format_number}{LEFT 57}{BRANCH rep1126}
A159: '{DEFINE}
B159: '{EDIT}{{}DEFINE location:TYPE1,LOCATION:TYPE2,....}{LEFT 35}
      {BRANCH rep1126}
A160: '{DISPATCH}
B160: '{EDIT}{{}DISPATCH location}{LEFT 9}{BRANCH rep1126}
A161: '{DOWN}
B161: '{EDIT}{{}DOWN number_of_cells}{LEFT 16}{BRANCH rep1126}
A162: '{EDIT}
B162: '{EDIT}{{}EDIT}{BRANCH rep1126}
A163: '{END}
B163: '{EDIT}{{}END}{BRANCH rep1126}
A164: '{ESC}
B164: '{EDIT}{{}ESC}{BRANCH rep1126}
A165: '{FILESIZE}
B165: '{EDIT}{{}FILESIZE location}{LEFT 9}{BRANCH rep1126}
A166: '{FORBREAK}
B166: '{EDIT}{{}FORBREAK}{BRANCH rep126}
A167: '{FOR}
B167: '{EDIT}{{}FOR COUNTER,START,STOP,SKIP,SUBROUTINE}{BRANCH REP126}
A168: '{GETLABEL}
B168: '{EDIT}{{}GETLABEL "string prompt",location}{LEFT 24}{BRANCH rep1126}
A169: '{GETNUMBER}
B169: '{EDIT}{{}GETNUMBER "string prompt",location}{LEFT 24}{BRANCH rep1126}
A170: '{GETPOS}
B170: '{EDIT}{{}GETPOS }{BRANCH rep1126}
A171: '{GET}
B171: '{EDIT}{{}GET location}{LEFT 9}{BRANCH rep1126}
A172: '{GOTO}
B172: '{EDIT}{{}GOTO}location{LEFT 8}{BRANCH rep1126}
A173: '{GRAPH}
B173: '{EDIT}{{}GRAPH}{BRANCH REP126}
A174: '{HOME}
B174: '{EDIT}{{}HOME}{BRANCH rep1126}
A175: '{IF}
B175: '{EDIT}{{}IF condition}{LEFT 10}{BRANCH rep1126}
A176: '{INDICATE}
B176: '{EDIT}{{}INDICATE [string]}{LEFT 9}{BRANCH rep1126}
A177: '{LEFT}
B177: '{EDIT}{{}LEFT number_of_cells}{LEFT 16}{BRANCH rep1126}
A178: '{LET}
B178: '{EDIT}{{}LET location,number/string/formula}{LEFT 31}{BRANCH rep1126}
A179: '{LOOK}
B179: '{EDIT}{{}LOOK location}{LEFT 9}{BRANCH rep1126}
A180: '{MENUBRANCH}
B180: '{EDIT}{{}MENUBRANCH menu}{LEFT 5}{BRANCH rep1126}
```

```
A181: '{MENUCALL}
B181: '{EDIT}{{}MENUCALL menu}{LEFT 5}{BRANCH rep1126}
A182: '{NAME}
B182: '{EDIT}{{}NAME}{BRANCH rep1126}
A183: '{ONERROR}
B183: '{EDIT}{{}ONERROR SUBROUTINE}{LEFT 11}{BRANCH rep1126}
A184: '{OPEN}
B184: '{EDIT}{{}OPEN FILE}{BRANCH rep1126}
A185: '{PANELOFF}
B185: '{EDIT}{{}PANELOFF}{BRANCH rep1126}
A186: '{PANELON}
B186: '{EDIT}{{}PANELON}{BRANCH rep1126}
A187: '{PGDN}
B187: '{EDIT}{{}PGDN}{BRANCH rep1126}
A188: '{PGUP}
B188: '{EDIT}{{}PGUP}{BRANCH rep1126}
A189: '{PUT}
B189: '{EDIT}{{}PUT location,col,row,number/string}{LEFT 31}{BRANCH rep1126}
A190: '{QUERY}
B190: '{EDIT}{{}QUERY}{BRANCH rep1126}
A191: '{QUIT}
B191: '{EDIT}{{}QUIT}{BRANCH rep1126}
A192: '{READLN}
B192: '{EDIT}{{}READLN location}{LEFT 9}{BRANCH rep1126}
A193: '{READ}
B193: '{EDIT}{{}READ byte_count,location}{LEFT 20}{BRANCH rep1126}
A194: '{RECALCOL}
B194: '{EDIT}{{}RECALCOL cell/range}{LEFT 11}{BRANCH rep1126}
A195: '{RECALC}
B195: '{EDIT}{{}RECALC cell/range}{LEFT 11}{BRANCH rep1126}
A196: '{RESTART}
B196: '{EDIT}{{}RESTART}{BRANCH REP126}
A197: '{RIGHT}
B197: '{EDIT}{{}RIGHT number_of_cells}{LEFT 16}{BRANCH rep1126}
A198: '{UP}
B198: '{EDIT}{{}UP number_of_cells}{LEFT 16}{BRANCH rep1126}
A199: '{WAIT}
B199: '{EDIT}{{}WAIT(@NOW+.0)}{BRANCH rep1126}
A200: '{WINDOWSOFF}
B200: '{EDIT}{{}WINDOWSOFF}{BRANCH rep1126}
A201: '{WINDOWSON}
B201: '{EDIT}{{}WINDOWSON}{BRANCH rep1126}
A202: '{WINDOW}
B202: '{EDIT}{{}WINDOW}{BRANCH rep1126}
A203: '{WRITELN}
B203: '{EDIT}{{}WRITELN "string"}{LEFT 8}{BRANCH rep1126}
A204: '{WRITE}
B204: '{EDIT}{{}WRITE "string"}{LEFT 8}{BRANCH rep1126}
A205: '"ASCII
B205: '{GOTO}tableb126~Point to the character to type and press [RETURN]...
      {GET key1126}{ESC}
A206: '!
B206: '{IF key1126="~"}{ESC}/RV~key1126~{back126}{EDIT}{key1126}
      {BRANCH rep1126}
A207: '!
B207: '{IF key1126="{ESC}"}{ESC}{back126}{BRANCH macrohlp}
A208: '!
B208: '{key1126}{?}~/RV~key1126~{back126}{EDIT}{key1126}{BRANCH rep1126}
A209: '!
A210: '!
A211: 'tableb126
B211: U @CHAR(32)
C211: U @CHAR(64)
D211: U @CHAR(96)
E211: U @CHAR(129)
F211: U @CHAR(160)
G211: U @CHAR(192)
H211: U @CHAR(224)
A212: '!
B212: U @CHAR(33)
C212: U @CHAR(65)
```

```
D212: U @CHAR(97)
E212: U @CHAR(130)
F212: U @CHAR(161)
G212: U @CHAR(193)
H212: U @CHAR(225)
A213: '!
B213: U @CHAR(34)
C213: U @CHAR(66)
D213: U @CHAR(98)
E213: U @CHAR(131)
F213: U @CHAR(162)
G213: U @CHAR(194)
H213: U @CHAR(226)
A214: '!
B214: U @CHAR(35)
C214: U @CHAR(67)
D214: U @CHAR(99)
E214: U @CHAR(132)
F214: U @CHAR(163)
G214: U @CHAR(195)
H214: U @CHAR(227)
A215: '!
B215: U @CHAR(36)
C215: U @CHAR(68)
D215: U @CHAR(100)
E215: U @CHAR(133)
F215: U @CHAR(164)
G215: U @CHAR(196)
H215: U @CHAR(228)
A216: '!
B216: U @CHAR(37)
C216: U @CHAR(69)
D216: U @CHAR(101)
E216: U @CHAR(134)
F216: U @CHAR(165)
G216: U @CHAR(197)
H216: U @CHAR(229)
A217: '!
B217: U @CHAR(38)
C217: U @CHAR(70)
D217: U @CHAR(102)
E217: U @CHAR(135)
F217: U @CHAR(166)
G217: U @CHAR(198)
H217: U @CHAR(230)
A218: '!
B218: U @CHAR(39)
C218: U @CHAR(71)
D218: U @CHAR(103)
E218: U @CHAR(136)
F218: U @CHAR(167)
G218: U @CHAR(199)
H218: U @CHAR(231)
A219: '!
B219: U @CHAR(40)
C219: U @CHAR(72)
D219: U @CHAR(104)
E219: U @CHAR(137)
F219: U @CHAR(168)
G219: U @CHAR(200)
H219: U @CHAR(232)
A220: '!
B220: U @CHAR(41)
C220: U @CHAR(73)
D220: U @CHAR(105)
E220: U @CHAR(138)
F220: U @CHAR(169)
G220: U @CHAR(201)
H220: U @CHAR(233)
A221: '!
B221: U @CHAR(42)
```

```
C221: U @CHAR(74)
D221: U @CHAR(106)
E221: U @CHAR(139)
F221: U @CHAR(170)
G221: U @CHAR(202)
H221: U @CHAR(234)
A222: '!
B222: U @CHAR(43)
C222: U @CHAR(75)
D222: U @CHAR(107)
E222: U @CHAR(140)
F222: U @CHAR(171)
G222: U @CHAR(203)
H222: U @CHAR(235)
A223: '!
B223: U @CHAR(44)
C223: U @CHAR(76)
D223: U @CHAR(108)
E223: U @CHAR(141)
F223: U @CHAR(172)
G223: U @CHAR(204)
H223: U @CHAR(236)
A224: '!
B224: U @CHAR(45)
C224: U @CHAR(77)
D224: U @CHAR(109)
E224: U @CHAR(142)
F224: U @CHAR(173)
G224: U @CHAR(205)
H224: U @CHAR(237)
A225: '!
B225: U @CHAR(46)
C225: U @CHAR(78)
D225: U @CHAR(110)
E225: U @CHAR(143)
F225: U @CHAR(174)
G225: U @CHAR(206)
H225: U @CHAR(238)
A226: '!
B226: U @CHAR(47)
C226: U @CHAR(79)
D226: U @CHAR(111)
F226: U @CHAR(175)
G226: U @CHAR(207)
H226: U @CHAR(239)
A227: '!
B227: U @CHAR(48)
C227: U @CHAR(80)
D227: U @CHAR(112)
E227: U @CHAR(144)
F227: U @CHAR(176)
G227: U @CHAR(208)
H227: U @CHAR(240)
A228: '!
B228: U @CHAR(49)
C228: U @CHAR(81)
D228: U @CHAR(113)
E228: U @CHAR(145)
F228: U @CHAR(177)
G228: U @CHAR(209)
H228: U @CHAR(241)
A229: '!
B229: U @CHAR(50)
C229: U @CHAR(82)
D229: U @CHAR(114)
E229: U @CHAR(146)
F229: U @CHAR(178)
G229: U @CHAR(210)
H229: U @CHAR(242)
A230: '!
B230: U @CHAR(51)
```

```
C230: U @CHAR(83)
D230: U @CHAR(115)
E230: U @CHAR(147)
F230: U @CHAR(179)
G230: U @CHAR(211)
H230: U @CHAR(243)
A231: '!
B231: U @CHAR(52)
C231: U @CHAR(84)
D231: U @CHAR(116)
E231: U @CHAR(148)
F231: U @CHAR(180)
G231: U @CHAR(212)
H231: U @CHAR(244)
A232: '!
B232: U @CHAR(53)
C232: U @CHAR(85)
D232: U @CHAR(117)
E232: U @CHAR(149)
F232: U @CHAR(181)
G232: U @CHAR(213)
H232: U @CHAR(245)
A233: '!
B233: U @CHAR(54)
C233: U @CHAR(86)
D233: U @CHAR(118)
E233: U @CHAR(150)
F233: U @CHAR(182)
G233: U @CHAR(214)
H233: U @CHAR(246)
A234: '!
B234: U @CHAR(55)
C234: U @CHAR(87)
D234: U @CHAR(119)
E234: U @CHAR(151)
F234: U @CHAR(183)
G234: U @CHAR(215)
H234: U @CHAR(247)
A235: '!
B235: U @CHAR(56)
C235: U @CHAR(88)
D235: U @CHAR(120)
E235: U @CHAR(152)
F235: U @CHAR(184)
G235: U @CHAR(216)
H235: U @CHAR(248)
A236: '!
B236: U @CHAR(57)
C236: U @CHAR(89)
D236: U @CHAR(121)
E236: U @CHAR(153)
F236: U @CHAR(185)
G236: U @CHAR(217)
H236: U @CHAR(249)
A237: '!
B237: U @CHAR(58)
C237: U @CHAR(90)
D237: U @CHAR(122)
E237: U @CHAR(154)
F237: U @CHAR(186)
G237: U @CHAR(218)
H237: U @CHAR(250)
A238: '!
B238: U @CHAR(59)
C238: U @CHAR(91)
D238: U @CHAR(123)
E238: U @CHAR(155)
F238: U @CHAR(187)
G238: U @CHAR(219)
H238: U @CHAR(251)
A239: '!
```

```
B239: U @CHAR(60)
C239: U @CHAR(92)
D239: U @CHAR(124)
E239: U @CHAR(156)
F239: U @CHAR(188)
G239: U @CHAR(220)
H239: U @CHAR(252)
A240: '!
B240: U @CHAR(61)
C240: U @CHAR(93)
D240: U @CHAR(125)
E240: U @CHAR(157)
F240: U @CHAR(189)
G240: U @CHAR(221)
H240: U @CHAR(253)
A241: '!
B241: U @CHAR(62)
C241: U @CHAR(94)
D241: U @CHAR(126)
E241: U @CHAR(158)
F241: U @CHAR(190)
G241: U @CHAR(222)
H241: U @CHAR(254)
A242: '!
B242: U @CHAR(63)
C242: U @CHAR(95)
D242: U @CHAR(128)
E242: U @CHAR(159)
F242: U @CHAR(191)
G242: U @CHAR(223)
H242: U @CHAR(255)
```

[9] Create a Menu Range

	A	B	C	D	E
1	*---A macro to TURN any 2-D or 3-D range of cells into an active				
2	MENU-RANGE				
3	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
4	range names in this column (starts with the \Y macro name)				
5	*---Hold the [ALT] key and press [Y] to define the MENU RANGE. The MENU				
6	RANGE needs to be defined ONLY once.				
7	*---Hold the [ALT] key and press [Z] to USE the MENU-RANGE, or use				
8	{BRANCH MENURANG} inside or from a macro.				
9	!				
10	\Y	{WINDOWSOFF}{PANELOFF}{LET rel283,@INFO("release")}~			
		{RECALC form283}/RNCDefine range!~~/RNDDefine range !~			
		/RNC{PANELON}{WINDOWSON}Define range !~{BS}{BS}{?}~			
		{GOTO}Define range !~			
11	MENURANG	{IF @CELLPOINTER("col")<@CELL("col",Define range !)}			
		{WINDOWSOFF}{PANELOFF}{RIGHT @COLS(Define range !)+			
		@CELL("col",Define range !)-@CELLPOINTER("col")-1}			
12	!	{IF @CELLPOINTER("col")>@CELL("col",Define range !)+			
		@COLS(Define range !)-1}{WINDOWSOFF}{PANELOFF}			
		{LEFT @CELLPOINTER("col")-@CELL("col",Define range !)}			
13	!	{IF @CELLPOINTER("row")<@CELL("row",Define range !)}			
		{WINDOWSOFF}{PANELOFF}{DOWN @ROWS(Define range !)+			
		@CELL("row",Define range !)-@CELLPOINTER("row")-1}~			
14	!	{IF @CELLPOINTER("row")>@CELL("row",Define range !)+			
		@ROWS(Define range !)-1}{WINDOWSOFF}{PANELOFF}			
		{UP @CELLPOINTER("row")-@CELL("row",Define range !)}			
15	!	{IF @LEFT(rel283,1)<>"@"}{IF @CELLPOINTER("sheet")>			
		@CELL("sheet",Define range !)+@SHEETS(Define range !)-1}			
		{WINDOWSOFF}{PANELOFF}{PS @CELLPOINTER("sheet")-			
		@CELL("sheet",Define range !)}			
16	!	{IF @LEFT(rel283,1)<>"@"}{IF @CELLPOINTER("sheet")<			
		@CELL("sheet",Define range !)}{WINDOWSOFF}{PANELOFF}			
		{NS @SHEETS(Define range !)+@CELL("sheet",Define range !)			
		-@CELLPOINTER("sheet")-1}~			
17	form283	{LET here1283,@CELLPOINTER("coord")}~{IF @CELLPOINTER			
		("type")<>"b"}{LET place283,@CELLPOINTER("contents")}~			
18	!	{WINDOWSOFF}{PANELOFF}{err1283}{GOTO}{place283}~{GOTO}			
		Define range !~{GOTO}{here1283}~{WINDOWSON}{PANELON}			
		{DISPATCH place283}			
19	!				
20	!	{IF @CELLPOINTER("type")="b"}{err2283}			
21	!				
22	\Z	{BREAKON}{BRANCH MENURANG}			
23	!				
24	key283	~			
25	!				
26	place283	rout1			
27	!				
28	here1283	$C:$B$4			
29	!				
30	err1283	{ONERROR err2283}			
31	!				
32	err2283	{ESC}{RESTART}{BEEP}{GOTO}Define range !~{RESTART}{PANELON}			
		Invalid routine name! PRESS any key to continue ...			
		{GET key283}{ESC}{BRANCH MENURANG}			
33	!				
34	rel283	3.00.00			
35	!				
36	loc283	coord			
37	!				
38	exec283	{GET key283}{ESC}			
39	!	{IF key283="{ESC}"}{ESC 6}{QUIT}			
40	!	{IF key283="{DOWN}"#OR#key283="{UP}"#OR#key283="{RIGHT}"			
		#OR#key283="{LEFT}"#OR#key283="{PGDN}"#OR#key283="{PGUP}"			
		#OR#key283="{HOME}"#OR#key283="{BIGRIGHT}"#OR#key283=			
		"{BIGLEFT}"}{key283}{RESTART}{BRANCH MENURANG}			
41	!	{IF key283="{NS}"#OR#key283="{PS}"}{key283}{RESTART}			

```
                    {BRANCH MENURANG}
```

Notice that the codes in the B17 and B36 cells is the result of dynamic string formulas. If you intend to key this code into 1-2-3, you need to key in the following formulas, NOT the code which appears in the main macro text.

```
17 form283          +"{LET here1283,@CELLPOINTER(""""&B36&""""")}~
                    {IF @CELLPOINTER(""type"")<>""b""}{LET place283,
                    @CELLPOINTER(""contents"")}~"
36 loc283           @IF(@LEFT(B34,1)<>"@","coord","address")
```

The Lotus macro language supplies the means to create custom menus, thereby allowing application developers to build menu driven applications. However, a custom menu can hold up to eight menu items maximum; therefore if more than eight menu options are needed, the menus must be nested. If you have ever tried to nest menus, you have probably noticed that it is not an easy task to create a foolproof menu system immunized against common user mistreatment. This macro solves the problem by creating the option to define any rectangular range of cells in the worksheet as what we call a "MENU RANGE", an expansion of the Lotus "MENU BAR". A normal 80x25 screen can hold 160 menu options, if the column width is the standard nine characters wide, and many more if the column width is less.

```
          A              B              C              D              E
10 \Y               {WINDOWSOFF}{PANELOFF}{LET rel283,@INFO("release")}~
                    {RECALC form283}/RNCDefine range!~~/RNDDefine range !~
                    /RNC{PANELON}{WINDOWSON}Define range !~{BS}{BS}{?}~
                    {GOTO}Define range !~
```

The macro first issues the `{WINDOWSOFF}{PANELOFF}` commands to freeze screen and panel display activities, and then stores the result of the `@INFO("release")` function in the B34 cell, [rel283], for later use. The macro uses this data to decide which Lotus version you are using. Next, the macro calculates the formula in B17, [form283], using `{RECALC form283}`. Before we continue, let's look into the formula in B17:

```
17 form283          +"{LET here1283,@CELLPOINTER(""""&B36&""""")}~
                    {IF @CELLPOINTER(""type"")<>""b""}{LET place283,
                    @CELLPOINTER(""contents"")}~"
```

The result of this formula is a string that serves as part of the macro code, the formula depends on the result of another formula in B36, [loc283], therefore we will also look at that formula:

```
36 loc283           @IF(@LEFT(B34,1)<>"@","coord","address")
```

The formula in B36 can only have two results. If "`@`" is the first character of the content of cell [rel283], than the result is the string "`address`" because you are using a 2-D Lotus release, otherwise the string is "`coord`" and you are using a 3-D release. Cell [rel283] contains the result of the `@INFO("release")` Lotus function issued earlier by the macro. If the result of the formula is the "`coord`" string, the formula in the B17 cell named [form283] will show:

```
          A              B              C              D              E
17 form283          {LET here1283,@CELLPOINTER("coord")}~{IF @CELLPOINTER
                    ("type")<>"b"}{LET place283,@CELLPOINTER("contents")}~
```

If the result of the formula is the "`address`" string, the formula in [form283] will show:

```
          A              B              C              D              E
17 form283          {LET here1283,@CELLPOINTER("address")}~{IF @CELLPOINTER
```

```
                ("type")<>"b"}{LET place283,@CELLPOINTER("contents")}~
```

Let's continue with the macro.

	A	B	C	D	E
10	\Y	{WINDOWSOFF}{PANELOFF}{LET rel283,@INFO("release")}~			
		{RECALC form283}/RNCDefine range!~~/RNDDefine range !~			
		/RNC{PANELON}{WINDOWSON}Define range !~{BS}{BS}{?}~			
		{GOTO}Define range !~			

The macro starts a / **R**ange **N**ame **C**reate process to assign the [Define range!] name to the range that you paint. The macro uses the "safe technique" to assign [Define range!], which it uses as a guiding prompt. When you finish painting the range and press ENTER, the macro moves the cell pointer to the lop left cell of [Define range!], using `{GOTO}Define range !~`. This concludes the definition of the menu range area. Now the macro knows where the MENU RANGE is. But before you can use the menu range, the menu range must contain the menu items and they must be associated with the appropriate routines. Each menu item has to be associated with a routine with the same exact name that is executed when the menu option is activated. For example, let's assume the following menu range is the A2..D4 range:

	A	B	C	D	E
1					
2	rout1	rout2	rout3	rout4	
3	rout5	rout6	rout7	rout8	
4	rout9	rout10	rout11	rout12	

For each menu item in this range, a routine using the same name must be written. For example, let's choose the [rout1] menu option. The routine must be built the following way:

```
rout1       This is the help line to the menu item {EXEC283}
            here starts the routine code
            .
            .
            .
```

For our example, let's assume that the [rout1] routine sends the cell pointer to the A1 cell. The routine will look like this:

```
rout1       Sends the cell pointer to HOME (A1) cell {EXEC283}
            {HOME}
```

The routine starts with the prompt in the panel when the cell pointer will be placed on the menu item, analogous to using the Lotus menu. Then the `{EXEC283}` routine command must follow the prompt text. The [EXEC283] routine, an integral part of the MENURANG .WK1 macro, clears the prompt from the panel when you move the cell pointer to a new menu option and activates and manages the menu range operation. The next cell and downward should contain the routine that the menu option activates when you press ENTER. In our example, the only command in the routine is `{HOME}`, which sends the cell pointer to the HOME cell (normally the A1 cell when titles are clear).

	A	B	C	D	E
11	MENURANG	{IF @CELLPOINTER("col")<@CELL("col",Define range !)}			
		{WINDOWSOFF}{PANELOFF}{RIGHT @COLS(Define range !)+			
		@CELL("col",Define range !)-@CELLPOINTER("col")-1}			

Here the macro checks where the cell pointer is. If the cell pointer is outside the menu range area, the macro moves it back to the menu range area, meaning that the moment the menu range is

operative, there is no way to get out of it. The cell pointer movement is limited to cells in the menu range area, as it should be. For example, if the cell pointer is on the "rout8" menu option and you press the RIGHT arrow key, the macro moves the cell pointer in a cyclic movement to the [rout5] menu option. If the cell pointer is on the [rout5] menu option and you press the LEFT arrow key, the macro moves the cell pointer to the "rout8" menu option. To check if the cell pointer is moving out of the menu range area, the macro uses `@CELLPOINTER("col")` and `@CELL("col",Define range !)` Lotus functions to create a formula for how many cells to move the cell pointer and in what direction to bring it back to the menu range area:

	A	B	C	D	E
12	!	{IF @CELLPOINTER("col")>@CELL("col",Define range !)+			
		@COLS(Define range !)-1}{WINDOWSOFF}{PANELOFF}			
		{LEFT @CELLPOINTER("col")-@CELL("col",Define range !)}			
13	!	{IF @CELLPOINTER("row")<@CELL("row",Define range !)}			
		{WINDOWSOFF}{PANELOFF}{DOWN @ROWS(Define range !)+			
		@CELL("row",Define range !)-@CELLPOINTER("row")-1}~			
14	!	{IF @CELLPOINTER("row")>@CELL("row",Define range !)+			
		@ROWS(Define range !)-1}{WINDOWSOFF}{PANELOFF}			
		{UP @CELLPOINTER("row")-@CELL("row",Define range !)}			

The codes in B11, B12, B13 and B14 cells control the movement to the four direction in the worksheet.

	A	B	C	D	E
15	!	{IF @LEFT(rel283,1)<>"@"}{IF @CELLPOINTER("sheet")>			
		@CELL("sheet",Define range !)+@SHEETS(Define range !)-1}			
		{WINDOWSOFF}{PANELOFF}{PS @CELLPOINTER("sheet")-			
		@CELL("sheet",Define range !)}			
16	!	{IF @LEFT(rel283,1)<>"@"}{IF @CELLPOINTER("sheet")<			
		@CELL("sheet",Define range !)}{WINDOWSOFF}{PANELOFF}			
		{NS @SHEETS(Define range !)+@CELL("sheet",Define range !)			
		-@CELLPOINTER("sheet")-1}~			

The codes in the B15 and B16 cells control the movement between the sheets in a 3-D Lotus release. Yes, a menu range can be a 3-D range. In B15, the macro first checks to see if the first character of the cell content in the cell named [rel283] is equal to the "@" character or not. If it is not equal to the "@" character, you are using a 3-D Lotus release and the macro will handle a 3-D menu range in the same manner as a 2-D menu range. For example, if the menu range has five sheets (B, C, D, E, F) and the cell pointer is on B3 and you try to move to the 6th sheet (the G sheet), the macro will move the cell pointer to the first sheet (to the B sheet) to the same address as the B3 cell was in the B sheet. This time it uses the `@CELLPOINTER ("sheet")` and the `@CELL("sheet",Define range !)` to create a formula for how many sheets to move the cell pointer and in what direction to bring it back to the menu range area. The macro commands to move from one sheet to another one are `{NEXTSHEET}` or `{NS}` and `{PREVSHEET}` or `{PS}`.

	A	B	C	D	E
17	form283	{LET here1283,@CELLPOINTER("coord")}~{IF @CELLPOINTER			
		("type")<>"b"}{LET place283,@CELLPOINTER("contents")}~			
18	!	{WINDOWSOFF}{PANELOFF}{err1283}{GOTO}{place283}~{GOTO}			
		Define range !~{GOTO}{here1283}~{WINDOWSON}{PANELON}			
		{DISPATCH place283}			

Now that the macro makes sure that the cell pointer is always inside the menu range area, the macro checks that the current menu option has a matched routine. Therefore the macro issues `{LET here1283,@CELLPOINTER("coord")}`, which stores the current cell pointer position in cell [here1283], and then issues `{IF @CELLPOINTER("type")<>"b"}` to check if the

current cell is not empty. If the cell is occupied, the macro issues {LET place283, @CELLPOINTER("contents")} store the current cell contents in the B26, [place283.

Next the macro issues {WINDOWSOFF}{PANELOFF} to freeze screen and panel activities, and then issues an error routine command {err1283}. This routine command will activate the error handling routine, [err1283]. If an error occurs later during macro execution, it is probably because you didn't create a routine with the same name as the menu option name. The macro immediately issues the indirect {GOTO}{place283}~ macro command which tries to send the cell pointer to the range whose name is written in [place283], but remember that the macro stored the name of the menu option in [place283]. Therefore {GOTO}{place283}~ will try to send the cell pointer to the routine with the same name as the menu option name. If an error occurs, the [err1283] routine will be activated and continue the macro as needed.

To better understand the code, let's assume that the macro stored the [rout1] menu option name in [place283]; therefore {GOTO}{place283}~ will actually become {GOTO}rout1~. However, if there is no range name with the [rout1] range name, an error will occur and the [err1283] routine will be activated. The macro uses here a special technique to determine whether a range name exists.

In the 3-D releases of Lotus 1-2-3, the @ISRANGE(range) function tests for a defined range name or valid range address; but to keep compatibility with older versions, we used the previous technique that works in all the Lotus releases from 2.0 and up. If the range name exists, the macro sends the cell pointer back to its place and issues {DISPATCH place283}, which activates the routine whose name is in the [place283] routine.

	A	B	C	D	E
30	err1283	{ONERROR err2283}			
31	!				
32	err2283	{ESC}{RESTART}{BEEP}{GOTO}Define range !~{RESTART}{PANELON}			
		Invalid routine name! PRESS any key to continue ...			
		{GET key283}{ESC}{BRANCH MENURANG}			

The [err2283] error routine is worth studying. For example, {ESC} was needed only in Lotus for windows to clear the panel when an error occurs, while the same routine works fine without {ESC} in all other releases of Lotus 1-2-3, on the other hand {PANELON} was needed in Lotus 3.1+. Without it

```
Invalid routine name! PRESS any key to continue ...
```

does not appear in the panel when you point to a menu option that is not associated with a routine with the same name. Sometimes it may seem that there are unnecessary commands or repetitions of the same command in the codes of the macros in ***Super Power***, but our research indicates they are needed to cover the many versions of Lotus 1-2-3. In every Lotus release, there are some quirks and incompatibilities with the previous versions. Different releases behave differently in the same situation. There are many examples of this and sometimes when we look into our macros, we forget why a specific command is there, but we are very careful not to change it before we can extensively test it in all the versions of Lotus.

	A	B	C	D	E
38	exec283	{GET key283}{ESC}			
39	!	{IF key283="{ESC}"}{ESC 6}{QUIT}			
40	!	{IF key283="{DOWN}"#OR#key283="{UP}"#OR#key283="{RIGHT}"			
		#OR#key283="{LEFT}"#OR#key283="{PGDN}"#OR#key283="{PGUP}"			

```
                #OR#key283="{HOME}"#OR#key283="{BIGRIGHT}"#OR#key283=
                "{BIGLEFT}"}{key283}{RESTART}{BRANCH MENURANG}
41 !            {IF key283="{NS}"#OR#key283="{PS}"}{key283}{RESTART}
                {BRANCH MENURANG}
```

As we have seen, [exec283] is the main routine that controls the movement from one menu item to the others and the execution of the menu option when you press ENTER. The [exec283] routine begins with `{GET key283}` which follows the help line in the routine associated with the menu option. The `{GET key283}` command halts macro execution and waits until you press a key. When a key is pressed, `{ESC}` clears the help message from the panel.

Next the macro screens the key using a series of `{IF}` macro commands. The first `{IF}` checks if ESC was pressed, using `{IF key283="{ESC}"}{ESC 6}{QUIT}`, which quits the macro if ESC was pressed. If one of the arrow keys was pressed, the macro executes the `{key283}` routine, which contains the code of the key. Therefore the macro executes the key that you intended. This way, [exec283] controls and allows only the keys programed in the routine, including DOWN, UP, RIGHT, LEFT, HOME, CTRL-RIGHT, CTRL-LEFT, NEXT-SHEET, PREVIOUS-SHEET, PGDN, and PGUP.

If any other key is pressed, [exec283] ends and returns control to the calling routine. In our example [rout1], which continues executing the rest of the code in the routine. When the routine is finished, the macro quits. In applications based on this macro, `{BRANCH menurang}` at the end of a routine which is associated with a menu option (such as [route1]), returns the control to the menu range.

[5] Notepad Macro

```
     A                B                C                D                E
 1 *---A NOTEPAD MACRO that updates itself at end of work. Best utilized
 2     with the MACRO MANAGER macro (the MACROMGR.WK1 or the SMALLMGR.WK1
 3     macro library managers)
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 *---When you will choose the SAVE option the macro file will be updated
 8 !
 9 !
10 \Z              {BREAKON}
11 NOTEPAD         {WINDOWSOFF}{PANELOFF}{LET here1130,@CELLPOINTER
                   ("address")}~
12 !               {GOTO}IV1~{GOTO}notepada130~{END}{DOWN}{DOWN}{PANELON}
                   {WINDOWSON}Add notes ? (Y/N) Y {GET key1130}{IF key1130
                   <>"~"#AND#@UPPER(key1130)<>"Y"}{ESC 8}{RETURN}
13 !               {ESC 6}Type your notes and use the direction keys,
                   Press [ENTER] to finish{GET key1130}{ESC 6}~{key1130}
                   {EDIT}{?}~
14 !               {GOTO}{here1130}~Save changes ? (Y/N) Y {GET key1130}
                   {IF key1130<>"~"#AND#@UPPER(key1130)<>"Y"}{ESC 8}{RETURN}
15 !               {ESC 6}{GOTO}{here1130}~{WINDOWSOFF}{PANELOFF}{GOTO}
                   NOTEPAD~{LEFT}{UP 10}/FXFNOTEPAD~{END}{DOWN}{RIGHT}~R
                   {ESC}{GOTO}{here1130}~
16 !
17 !
18 key1130         y
19 !
20 here1130        $A$11
21 !
22 notepada130     ************************************************************
23 !               *                          MACRO  NOTEPAD                  *
24 !               ************************************************************
25 !               Meet with sales managers 14:30
26 !               Lunch with Jacob Murphy 12:30
27 !
28 !
29 !
30 !
31 !
32 !
33 !
34 !
35 !
36 !
37 !
```

This is a notepad macro that allows you to edit, view and save notes while using Lotus 1-2-3.

```
     A                B                C                D                E
11 NOTEPAD         {WINDOWSOFF}{PANELOFF}{LET here1130,@CELLPOINTER
                   ("address")}~
```

The macro starts with the `{WINDOWSOFF}{PANELOFF}` macro commands which freeze the screen and panel activities until they are needed, and then issues `{LET here1130 ,@CELLPOINTER("address")}~`, which stores the current cell address in the B20 cell, [here1130].

```
     A                B                C                D                E
12 !               {GOTO}IV1~{GOTO}notepada130~{END}{DOWN}{DOWN}{PANELON}
                   {WINDOWSON}Add notes ? (Y/N) Y {GET key1130}{IF key1130
                   <>"~"#AND#@UPPER(key1130)<>"Y"}{ESC 8}{RETURN}
```

The macro issues {GOTO}IV1~{GOTO}notepada130~ to move the cell pointer to the notepad area and place the upper left cell of the notepad area on the upper left corner of the screen. Then issues {END}{DOWN}{DOWN} to move the cell pointer to the cell below the last note in the notepad. Next the macro issues {PANELON}{WINDOWSON} to resume screen and panel display activities so you can scroll, move and edit. Next the macro types:

```
Add notes ? (Y/N) Y
```

into the panel and issues {GET key1130}. Now the macro pauses and waits for your response. When you press any key other then ENTER, "Y" or "y", the macro issues {ESC 8}{RETURN} to clear the panel, return to READY mode and quit.

```
     A          B                C          D          E
13 !          {ESC 6}Type your notes and use the direction keys,
              Press [ENTER] to finish{GET key1130}{ESC 6}~{key1130}
              {EDIT}{?}~
```

If you press ENTER, "Y" or "y", the macro issues {ESC 6} to clear the panel and types:

```
Type your notes and use the direction keys, Press [ENTER] to finish
```

into the panel followed by {GET key1130} and waits until you press a key. Then it issues {ESC 6} to clear the panel and executes the {key1130} routine command. Because the [key1130] routine holds the code for the key that you just pressed, the {key1130} command executes this key. Next the macro issues {EDIT} and {?} so you can write notes or edit the previous notes as long as you do not press ENTER.

```
     A          B                C          D          E
14 !          {GOTO}{here1130}~Save changes ? (Y/N) Y {GET key1130}
              {IF key1130<>"~"#AND#@UPPER(key1130)<>"Y"}{ESC 8}{RETURN}
```

When you press ENTER, the macro exits the EDIT mode and issues the {GOTO} {here1130}~ indirect macro command, sending the cell pointer back to the cell position before the macro started. Then the macro types

```
Save changes ? (Y/N) Y
```

into the panel followed by {GET key1130} and pauses. When you respond, the macro issues:

```
{IF key1130<>"~"#AND#@UPPER(key1130)<>"Y"}{ESC 8}{RETURN}
```

which is the same code as in the first message. If you press any key then but ENTER, "Y" or "y", the macro issues {ESC 8}{RETURN} and quits.

```
     A          B                C          D          E
15 !          {ESC 6}{GOTO}{here1130}~{WINDOWSOFF}{PANELOFF}{GOTO}
              NOTEPAD~{LEFT}{UP 10}/FXFNOTEPAD~{END}{DOWN}{RIGHT}~R
              {ESC}{GOTO}{here1130}~
```

If you press ENTER, "Y" or "y" to save the notepad, the macro issues {ESC 6} to clear the message from the panel, and then the {GOTO}{here1130}~ indirect command, which moves the cell pointer to the origin address (before the macro was activated) and uses {PANELOFF} {WINDOWSOFF} to freeze screen and panel display activities. Now the macro issues {GOTO} NOTEPAD~ to move the cell pointer to the notepad macro area and then:

```
{LEFT}{UP 10}/FXFNOTEPAD~{END}{DOWN}{RIGHT}~R{ESC}
```

macro code which extracts the notepad to the NOTEPAD.WK1 file. The `{LEFT}{UP 10}` macro commands bring the cell pointer to the first cell of the notepad macro and then the `/FXFNOTEPAD~{END}{DOWN}{RIGHT}~` paints all the macro area and extracts it to the NOTEPAD.WK1 file. The "`R{ESC}`" code needs some explanation. The "`R`" stands for **R**eplace if the NOTEPAD.WK1 file already exists in the default directory but if the file is new, `{ESC}` clears it from the panel.

[4] Make Formulas Error Proof

```
     A          B          C          D          E
 1 *---A macro to make a formula ERROR PROOF. The macro alters the formula
 2     in the cell to return ZERO instead of ERR in case of error condition
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Place the cell pointer on the formula to be altered
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8 !
 9 !
10 \Z            {BREAKON}
11 ERRPROOF      {WINDOWSOFF}{PANELOFF}{EDIT}{HOME}'~{LET temp016,@CELLPOINTER
                 ("contents")}~{EDIT}{HOME}{RIGHT}@IF(@ISERR({END}),0,
                 {temp016}){HOME}{DEL}~
12 !
13 temp016       @SIN(AA1)/@TAN(AB1)
```

This macro turns any formula to an immune error proof formula, in such a way, that the formula can never produce an `@ERR`, it will produce a "`0`" zero value instead. This macro uses the enveloping method to change the formula so it will return the correct answer as long as the formula is legal, where as when the original formula returned `ERR`, the upgraded formula returns zero. For example: if the current cell contains the `@SIN(AA1)/@TAN(AB1)` formula, which should return an `ERR` if the AB1 cell contains a zero value, the macro changes it to the `@IF(@ISERR(@SIN(AA1)/@TAN(AB1)),0,@SIN(AA1)/@TAN(AB1)` formula, which returns a zero instead.

The macro starts with the `{WINDOWSOFF}{PANELOFF}` commands which freeze the screen and panel display activity. Then the macro issues `{LET temp016,@CELLPOINTER ("contents")}~` to copy the content of the current cell as a label to the B13 cell named [temp016]. Next it issues `{EDIT}` to enter to the EDIT mode and `{HOME}` to move the cursor to the beginning of the formula in the panel, which now displays:

```
@SIN(AA1)/@TAN(AB1)
^
```

The "`^`" represents the cursor position. Then the macro writes the apostrophe "`'`" to turn the formula into a label, issues the tilde "`~`" command, which is the same as ENTER, and writes it back to the current cell, which now contains a label instead of the formula. Next, the macro copies the current cell's content (the label) to the B13 cell named [temp016]. Now the macro again issues `{EDIT}`, `{HOME}` and `{RIGHT}` to enter the EDIT mode, and move the cursor to the second character of the label in the panel, which now displays:

```
'@SIN(AA1)/@TAN(AB1)
^
```

Next the macro writes the "`@IF(@ISERR(`" text into the panel ,which shows:

```
'@IF(@ISERR(@SIN(AA1)/@TAN(AB1)
```

Next the macro issues `{END}` moving the cursor to the end of the panel and then writes the "`),0,`" text to the panel, which displays:

```
'@IF(@ISERR(@SIN(AA1)/@TAN(AB1)),0,
```

Now the macro needs to insert the original formula after the comma, but how does the macro know what the original formula was? The macro copied the original formula to cell [temp016], therefore the macro issues the {temp016} routine command which injects the content of [temp016] into the panel right where the cursor is. Therefore the panel displays:

```
'@IF(@ISERR(@SIN(AA1)/@TAN(AB1)),0,@SIN(AA1)/@TAN(AB1)
```

All that left to complete the formula is to write the closing parenthesis "`)`", delete the apostrophe and press ENTER. Therefore the macro issues {HOME}{DEL}, which delete the apostrophe "`'`" and turn the label back into a formula, which is now immune. Even if the original `@SIN(A1)/@TAN(B1)` formula had returned an ERR, the new formula returns zero "`0`" instead. Note how the macro wrote the original formula twice to get a correct formula. We recorded the formula in [temp016] as a label, and then injected the formula into the panel using the {temp016} routine command. This macro is one of the finest examples of panel manipulating techniques that stretch Lotus's macro language to the limit.

[4] Display a Flashing Message in the Panel

```
     A                B                C                D                E
 1 *---A macro to present a flashing message in the panel area
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Type your message right to the cell MESSAGE
 6 !
 7 !
 8!
 9 !
10 \Z              {BREAKON}
11 MESSAGE         {BREAKON}
12 loop023         {WINDOWSOFF}{PANELOFF}{message023}{PANELON}
13 !               {EDIT}{WAIT @NOW+@TIME(0,0,1)}{ESC}~{LOOK key023}~
14 !               {IF key023=""}{WAIT @NOW+@TIME(0,0,1)}{BRANCH loop023}
15 !               {GET key023}~
16 !
17 !
18 message023      This is our message .... Press any key to Quit
19 !
20 key023
```

This macro displays a flashing message in the panel. It uses {PANELOFF} and {PANELON} to freeze and release the panel display activities in approximately one second intervals to display and erase the flashing message in the panel. The macro starts with {WINDOWSOFF} {PANELOFF} which freeze screen and panel display activities, and then it issues the {message023} routine command to inject the message in cell [message023] into the panel. Then the macro issues {PANELON}{EDIT}, which resume the panel activity and allow you to see the message. Next, the macro issues {WAIT @NOW+@TIME(0,0,1)}, which waits one second before issuing {ESC} to exit the EDIT mode and erase the message from the panel, before Lotus writes it into the current cell.

Next the macro issues {LOOK key023} to trap the key you pressed to quit the message. Lotus stores the key in [key023]. If [key023] contains anything but the null string, it means that a key was pressed. The macro continues with {IF key023=""} to check if a key was pressed. If so, the macro issues {WAIT @NOW+@TIME(0,0,1)}, which waits one second and then issues {BRANCH loop023} routing macro control back to the beginning and re-displays the message. If you press a key, the macro issues {GET key023} which clears the key buffer from the key that you pressed and quits. Notice the tilde "~" after {LOOK} which forces the update of the worksheet immediately.

Note: Lotus does not remove the keystroke it records with {LOOK} from the type ahead buffer, so a subsequent {LOOK} command will record the same keystroke. To remove the first keystroke from the type ahead buffer, you must follow the {LOOK} command with a {GET} command. Lotus does not automatically recalculate formulas after executing a {LOOK} command when worksheet recalculation is set to Automatic. To force recalculation after a {LOOK} command, you must follow the command with a tilde "~" or {CALC}.

[4] Find Adjacent Duplicate Entries in a Column

```
     A                B                C                D                E
 1 *---A macro to find DUPLICATE entries in a column. The macro handles
 2     numbers and labels.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Point to the first entry cell in the column
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7 !
 8 !
 9 !
10 \Z              {BREAKON}
11 DUPENTRY        {BREAKON}
12 loop105         {WINDOWSOFF}{PANELOFF}{LET upcell105,@CELLPOINTER
                   ("contents")}{DOWN}{IF @CELLPOINTER("contents")=
                   upcell105}{RESTART}{BRANCH end105}
14 !               {BRANCH loop105}
15 !
16 upcell105
17 !
18 end105          {ESC}
```

This macro will find couples of identical cell contents inside a column of data. The macro starts with the {WINDOWSOFF}{PANELOFF} macro commands which freeze screen and panel display activities. Then it issues {LET upcell105,@CELLPOINTER("contents")}, which stores the current cell content in B16, [upcell105]. Next the macro issues {DOWN}, which moves the cell pointer to the next cell in the column. Now the macro issues {IF @CELLPOINTER ("contents")=upcell105} to check if the current cell content is the same as the previous cell.

If so, the macro issues {RESTART}, which clears the subroutine stack. This command is not really necessary if you manually use the macro. However, if another macro such as the macro manager calls this macro, we need to use {RESTART} if we want to stop the macro when it finds duplicated cells. When 1-2-3 encounters {RESTART}, it executes the remaining instructions in the current subroutine, but instead of returning control to the calling subroutine after it completes the current subroutine, the macro ends. If the instructions in the subroutine that follow {RESTART} transfer macro control elsewhere, the macro does not end.

Next the macro issues {BRANCH end105}, which branches to the [end105] routine that issues {ESC} to quit to the READY mode. Now you can edit the duplicate data. If the current cell's content is not the same as the previous cell's content, the macro issues {BRANCH loop105}, which routes the macro control back to the start so the macro can check the next cell in the column. Only when the macro finds two adjacent cells with the same content does the macro stop to allow you to take action.

[4] Trapping an Error in a Loop

```
          A              B              C              D              E
 1 *---A macro to TRAP errors in a loop WITHOUT breaking the loop.
 2     Let's say you want to combine 10 files to a worksheet, therefore
 3     endloop=10. If the drive door is open a message will appear.
 4     Close the drive door and try again. You can use this macro for other
 5     than file combining. Use your own ERROR ROUTINE and prompt message.
 6 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 7     range names in this column (starts with the \Z macro name)
 8 *---Hold the [ALT] key and press [Z] to activate the macro
 9 !
10 \Z           {BREAKON}
11 ERRLOOP      {FOR counter,1,endloop,1,routinea}
12 !
13 routinea     {ONERROR err1}
14 !            {your_routine}
15 !
16 err1         {GETLABEL "File doesn't exist! Press [ENTER] to continue
                . . .  ",temp}
17 !            {routinea}
18 !            {LET endloop,endloop-counter}~
19 !            {BRANCH errloop}
20 !
21 counter          1.00
22 !
23 endloop        10.00
24 !
25 temp         SSS
26 !
27 your_routine /FCCE{?}~
28 !            {PGDN}
```

This macro is for application writers, who need an error handling routine to recover from an error inside a loop without the need to quit the loop. The macro starts with the `{FOR counter,1,endloop,1,routinea}` command, which executes the [routinea] routine ten times, starting at [counter] equals 1 to [endloop] equals 10 in increments of 1. The [routinea] routine issues `{ONERROR err1}` to define [err1] as an error handling routine. Then it issues `{your_routine}` which executes the [your_routine] routine.

```
          A              B              C              D              E
27 your_routine /FCCE{?}~
28 !            {PGDN}
```

This is a routine which prompts you to combine a file `/FCCE{?}~`. For example, if you make an error in the file name and the file doesn't exist, the error handing routine [err1] is activated.

```
          A              B              C              D              E
16 err1         {GETLABEL "File doesn't exist! Press [ENTER] to continue
                . . .  ",temp}
17 !            {routinea}
18 !            {LET endloop,endloop-counter}~
19 !            {BRANCH errloop}
```

When an error occurs the [err1] routine issues:

```
{GETLABEL "File doesn't exist! Press [ENTER] to continue . . .  ",temp}
```

which displays:

```
File doesn't exist! Press [ENTER] to continue . . .
```

prompt in the panel and waits until you press ENTER. When you press ENTER, the macro issues `{routinea}` which executes [routinea].

```
          A              B              C              D              E
13 routinea      {ONERROR err1}
14 !             {your_routine}
```

The [routinea] routine issues `{ONERROR err1}` to re-define [err1] as the error handling routine. Then the macro issues `{your_routine}`, which executes [your_routine] again to allow you to type the correct file name. When the [routinea] routine is finished, the macro returns control to the [err1] routine, which issues `{LET endloop,endloop-counter}~` to decrease the value in [endloop] by one (from 10 to 9). Otherwise the macro will execute [routinea] eleven times. Every time that an error occurs the [err1] routine fixes it, but adjusts the value in [endloop].

Summary: We used the file combine routine only as an example. You can use any routine and any prompt instead.

---

[8] Key Pressed Recorder (Learn Macro for Lotus 1-2-3 2.0/2.01)

```
     A              B                C                D              E
 1 *---A macro to RECORD and EXECUTE keystrokes as they are typed. The
 2     macro allows using LOTUS 1-2-3 and simultaneously to record
 3     your keystrokes into a macro that can be used later again.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro. Start
 7     working. Your keystrokes will be recorded and placed out off the
 8     work area in the range macro1. Use Ctrl-Break to Quit.
 9                   SEE MORE DETAILS IN THE MANUAL.DOC FILE
10 \Z                {BREAKON}
11 RECORDER          {LET rel232,@INFO("release")}~{RECALC loca232}
                     {RECALC form232}
12 form232           {WINDOWSOFF}{PANELOFF}{LET here1232,@CELLPOINTER
                     ("address")}~{END}{HOME}{RIGHT}{END}{UP}macro1~
13 !                 /RNLR~{RIGHT}
14 forma232          {LET row232,1}~{LET loc232,@CELLPOINTER("address")}~
15 !                 {RECALC temp1232}{RECALC temp2232}{RECALC temp3232}
                     {GOTO}{here1232}~
16 temp1232          {WINDOWSON}{PANELON}{LET $E$1,""}
17 loop232           {GET key1232}
18 temp2232          {LET $E$1,$E$1&key1232}
19 temp3232          {IF @LENGTH($E$1)>30}{LET $E$2,""}{LET row232,row232+1}
20 !                 {key1232}
21 !                 {RECALC temp2232}{RECALC temp3232}{BRANCH loop232}
22 !
23 row232                         1
24 !
25 loc232            $E$1
26 !
27 key1232           {LEFT}
28 !
29 here1232          $A$1
30 !
31 rel232            @INFO("release")
32 !
33 loca232           address
```

This macro is one of the finest examples of Lotus macro programming skills. This macro has lost part of its importance with the introduction of the new releases of Lotus, which feature built-in macro recording; however it is invaluable for users of the older Lotus 2.0/2.01. You can work the usual way and the macro records every key press and automatically stores it in a remote area outside the worksheet area. When you terminate the recording, the macro code is stored in the remote area for later use as a macro. This macro uses string formulas extensively, to create dynamic macro code , therefore part of the code that you see is the result of these formulas. If you intend to key this macro code into Lotus 1-2-3, you have to key the formulas as they appear here, NOT the code as it appears in the main listing.

```
12 form232           +"{WINDOWSOFF}{PANELOFF}{LET here1232,@CELLPOINTER
                     (""&B33&"")}~{END}{HOME}{RIGHT}{END}{UP}MACRO1~"
14 forma232          +"{LET row232,1}~{LET loc232,@CELLPOINTER(""&B33&"")}~"
16 temp1232          +"{WINDOWSON}{PANELON}{LET "&@LEFT(B25,@LENGTH(B25)-
                     @LENGTH(@STRING(B23,0)))&@STRING(B23,0)&","""}"
18 temp2232          +"{LET "&@LEFT(B25,@LENGTH(B25)-@LENGTH(@STRING(B23,0)))
                     &@STRING(B23,0)&","&@LEFT(B25,@LENGTH(B25)-@LENGTH(@STRING
                     (B23,0)))&@STRING(B23,0)&"&key1232"&"}"
19 temp3232          +"{IF @LENGTH("&@LEFT(B25,@LENGTH(B25)-@LENGTH(@STRING
                     (B23,0)))&@STRING(B23,0)&")>30}{LET "&@LEFT(B25,@LENGTH
                     (B25)-@LENGTH(@STRING(B23,0)))&@STRING(B23+1,0)&","""}
                     {LET row232,row232+1}"
33 loca232           @IF(@LEFT(B31,1)<>"@","coord","address")
```

	A	B	C	D	E
11	RECORDER	{LET rel232,@INFO("release")}~{RECALC loca232}			
		{RECALC form232}			
12	form232	{WINDOWSOFF}{PANELOFF}{LET here1232,@CELLPOINTER			
		("address")}~{END}{HOME}{RIGHT}{END}{UP}macro1~			
13	!	/RNLR~{RIGHT}			

The macro starts with the {LET rel232,@INFO("release")} macro command, which stores the result of the @INFO("release") function in the B31 cell, [rel232]. Later, the macro uses this result to check if you are using a 2-D or a 3-D release of Lotus 1-2-3. The macro issues {RECALC loca232}{RECALC form232}, which recalculate and update the formulas in the B33 cell, [loca232], and the B14 cell, [form232]. Next the macro issues {WINDOWSOFF}{PANELOFF} which freeze the screen and panel display activities, and then {LET here1232,@CELLPOINTER("address")}, which stores the cell pointer current address in cell [here1232].

Then the macro issues {END}{HOME}{RIGHT}{END}{UP}, which move the cell pointer to the upper cell of the column right to the last occupied column of the worksheet. Now the macro writes the "macro1" text into the panel and issues the tilde "~" to write the content of the panel into the cell. Now the macro issues /RNLR~, which assigns the [macro1] range name to the cell to right to it, and then {RIGHT}, which puts the cell pointer on the cell named [macro1].

To better understand the process, let's assume that the last occupied cell in the worksheet is the AA100 cell. To find this address we can use the END HOME key sequence. Therefore {END}{HOME}{RIGHT}{END}{UP} move the cell pointer to the AB1 cell where the macro writes the "macro1" text and assigns the [macro1] name to the AC1 cell, which will become the first cell of the recorded macro. The code in the B12 cell is the result of the following formula:

```
12 form232        +"{WINDOWSOFF}{PANELOFF}{LET here1232,@CELLPOINTER
                  (""""&B33&""""")}~{END}{HOME}{RIGHT}{END}{UP}MACRO1~"
```

We can see that the "dynamic" part of the formula is the content of the B33 cell, which contains the following formula:

```
33 loca232        @IF(@LEFT(B31,1)<>"@","coord","address")
```

The formula in B33 checks the first character of the content of B31, which contains the result of the @INFO("release") function. If the character is NOT the "@" character, then you are using a 3-D Lotus release; therefore the result of the formula is the "coord" string and the resulting code of the formula in the B12 cell is:

```
12 form232        {WINDOWSOFF}{PANELOFF}{LET here1232,@CELLPOINTER
                  ("coord")}~{END}{HOME}{RIGHT}{END}{UP}macro1~
```

If "@" is the first character, you are using a 2-D Lotus release and the result of the formula in B31 is the "address" string; the resulting code of the formula in the B12 cell is:

```
12 form232        {WINDOWSOFF}{PANELOFF}{LET here1232,@CELLPOINTER
                  ("address")}~{END}{HOME}{RIGHT}{END}{UP}macro1~
```

	A	B	C	D	E
14	forma232	{LET row232,1}~{LET loc232,@CELLPOINTER("address")}~			
15	!	{RECALC temp1232}{RECALC temp2232}{RECALC temp3232}			
		{GOTO}{here1232}~			

```
16 temp1232        {WINDOWSON}{PANELON}{LET $E$1,""}
```

Now the macro issues {LET row232,1}, which stores the number "1" in cell [row232], and {LET loc232,@CELLPOINTER("address")}, which stores the address of the first cell of the macro in cell [loc232]. This line of code is the result of the following dynamic string formula:

```
14 forma232        +"{LET row232,1}~{LET loc232,@CELLPOINTER(""&B33&"")}~"
```

The "dynamic" part of the formula is B33. We saw it earlier when we studied the code of B12. Next the macro uses {RECALC temp1232}{RECALC temp2232}{RECALC temp3232}, which update the dynamic string formulas in cells [temp1232], [temp2232] and [temp3232]. To facilitate explaining the code, let's assume that the "macro1" text is in the D1 cell and the macro starts in the E1 cell. Now the macro issues {WINDOWSON}{PANELON}, which resume screen and panel display activities, and then {LET $E$1,""}, which stores the null string in the $E$1 cell. Recall that this code is the result of the following formula:

```
16 temp1232        +"{WINDOWSON}{PANELON}{LET "&@LEFT(B25,@LENGTH(B25)-
                   @LENGTH(@STRING(B23,0)))&@STRING(B23,0)&",""""}"
```

When the macro starts, this formula creates the $E$1 address because [macro1] starts at the E1 cell. When the macro records enough characters, 30 characters per cell in our macro, in the E1 cell, it starts to fill E2 and then E3 and so on until you stop recording. This formula uses the value in [row232] to create the EX address for the correct cell. Now we are going to see how the macro records the keystrokes and stores them in [macro1].

```
                A              B                C                D              E
17 loop232         {GET key1232}
18 temp2232        {LET $E$1,$E$1&key1232}
19 temp3232        {IF @LENGTH($E$1)>30}{LET $E$2,""}{LET row232,row232+1}
20 !               {key1232}
21 !               {RECALC temp2232}{RECALC temp3232}{BRANCH loop232}
```

The macro issues {GET key1232}, which waits until you press a key. When you press a key, Lotus stores it in cell [key1232]. Then the macro issues {LET $E$1,$E$1&key1232}, which adds the key to the content of E1. This command is also the result of a string formula; therefore when the macro finishes recording 30 characters in the E1 cell, the formula creates the code to look like {LET $E$2,$E$2&key1232} and so on. Let's look at the formula in B18, [temp2232].

```
18 temp2232        +"{LET "&@LEFT(B25,@LENGTH(B25)-@LENGTH(@STRING(B23,0)))
                   &@STRING(B23,0)&","&@LEFT(B25,@LENGTH(B25)-@LENGTH(@STRING
                   (B23,0)))&@STRING(B23,0)&"&key1232"&"}"
```

This is not a simple formula. It extracts the $E$1 cell address twice and then adds the content of [key1232] which holds the key that you pressed.

Next the macro issues {IF @LENGTH($E$1)>30} to check if the length of the code in E1 is more than 30 characters long. If so, the macro issues {LET $E$2,""}, which sets the next cell in the column, $E$2, to the null string, and then issues {LET row232,row232+1}, which increases the value in [row232] by one, changing it in this example, to the number 2. Recall that the macro uses the value in [row232] to produce the address where the macro stores the keys that you pressed. Again the code in B19, [temp3232] is the result of a string formula:

```
19 temp3232        +"{IF @LENGTH("&@LEFT(B25,@LENGTH(B25)-@LENGTH(@STRING
                   (B23,0)))&@STRING(B23,0)&")>30}{LET "&@LEFT(B25,@LENGTH
```

```
(B25)-@LENGTH(@STRING(B23,0)))&@STRING(B23+1,0)&",""""}
{LET row232,row232+1}"
```

Again this is not a simple formula, it differs from the previous formula by predicting the next cell in [macro1], when the content of the current cell reaches 30 characters in length. Now the macro issues `{LET row232,row232+1}` which increase the content of [row232] by one, and issues `{key1232}` to carry out the key that you pressed. This macro is built to record the keys you press and simultaneously execute them, so that you can work as usual while the macro in the background records the keys so you can uses them later as a macro to repeat the manual work. Next the macro issues `{RECALC temp2232}{RECALC temp3232}` to update the string formulas to reflect the changes, and then the macro issues `{BRANCH loop232}`, which routes the macro control back to B17, [loop232], to capture the next key you press. To stop recording you need to press Ctrl-Break.

To find the recorded macro, press the END HOME RIGHT END UP RIGHT direction keys or press F5 and type "macro1" and press ENTER.

[9] Word Processor with Search and Replace (1-2-3 2.0/2.01)

	A	B	C	D	E
1	*---A WORD PROCESSOR macro with a full featured SEARCH & REPLACE				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	*---Press {ESC} to return to main menu				
6	!				
7	!				
8	!				
9	!				
10	\Z	{BREAKON}			
11	WORDPROC	{LET num2280,@STRING(@CELLPOINTER("width"),0)}~			
		{GETLABEL "Number of characters in a line ? ",num1280}~			
		{setwidth280}			
12	!	{MENUBRANCH menu1280}			
13	!				
14	menu1280	Typing Re-justify Search & Replace Quit			
15	!	Type your Re-adjust thFull featured SEARQuit the word pr			
16	!	{loopp1280{GETLABEL "N{RESTART}{search12{WINDOWSOFF}{PAN			
17	!	{menubranc{BRANCH just{MENUBRANCH menu1280}			
18	!	{MENUBRANCH menu1280}			
19	!				
20	!				
21	loopp1280	{EDIT}{GET key1280}{IF key1280="{ESC}"}{ESC 2}~			
		{MENUBRANCH menu1280}			
22	!	{IF key1280="~"}{BRANCH justify280}			
23	!	{IF key1280="{LEFT}"#OR#key1280="{RIGHT}"#OR#key1280=			
		"{HOME}"#OR#"{BS}"}{key1280}{?}~{BRANCH justify280}			
24	!	{key1280}{BRANCH loopp1280}			
25	!				
26	key1280	{ESC}			
27	!				
28	setwidth280	/WCS			
29	num1280	50			
30	!	~			
31	!				
32	setwidth1280	/WCS			
33	num2280	30			
34	!	~{QUIT}			
35	!				
36	justify280	{WINDOWSOFF}{UP}{IF @CELLPOINTER("type")="b"}{DOWN 2}			
		{BRANCH justify1280}			
37	!	{DOWN}{END}{UP}/RJ~{END}{DOWN}{WINDOWSON}			
		{BRANCH loopp1280}			
38	!				
39	justify1280	{IF @CELLPOINTER("type")="b"}{UP}/RJ~{WINDOWSON}			
		{BRANCH loopp1280}			
40	!	{DOWN}{IF @CELLPOINTER("type")<>"b"}{UP}{END}{DOWN}			
		{WINDOWSON}{BRANCH loopp1280}			
41	!	{UP}{WINDOWSON}{BRANCH loopp1280}			
42	search1280	{LET lfind280,0}{LET flag280,0}{WINDOWSOFF}{PANELOFF}			
		/RNCWhich range ?~~/RNDWhich range ?~/RNC{WINDOWSON}			
		{PANELON}Which range ?~{BS}{BS}{?}~{GOTO}Which range ?~			
		{prompt280}			
43	!	{LET counter1280,0}~			
44	!	{FOR counter1280,0,@COLS(Which range ?)-1,1,labels1280}			
45	!	{GOTO}Which range ?~/RNDWhich range ?~			
46	!				
47	counter1280	1			
48	counter1a280	0			
49	labels1280	{IF @UPPER(match1280)="A"}{FOR counter1a280,0,@ROWS			
		(Which range ?)-1,1,cont2280}~{RIGHT}{UP @ROWS			
		(Which range ?)}{LET counter1a280,0}~			
50	!	{IF @UPPER(match1280)="F"}{FOR counter1a280,0,@ROWS			
		(Which range ?)-1,1,search2280}~{RIGHT}{UP @ROWS			

```
                (Which range ?)}{LET counter1a280,0}~
51 !
52 prompt280     {GETLABEL "Type the OLD string/number to be replaced: "
                ,dummy1280}~
53 !            {GETLABEL "Type the NEW string/number : ",dummy280}~
                {IF @CELLPOINTER("type")="b"}{cont1280}
54 !            {GETLABEL "Match (exact) Y/N: ",match280}~
55 !            {GETLABEL "<F>ind & replace / replace <A>ll ",match1280}~
56 !
57 !
58 cont2280      {windowsOFF}{PANELOFF}{IF @CELLPOINTER("type")="b"}
                {BRANCH cont1280}
59 !            {IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~{LET flag280,
                1}~{LET lfind280,0}~{BRANCH cont280}
60 !            {EDIT}{HOME}'~{LET flag280,0}~{LET lfind280,0}~
61 cont280      {PANELOFF}{RECALC dummy2280}~{RECALC dummy3280}~
                {RECALC dummy4280}~{RECALC err2280}~{RECALC err3280}~
62 !            {IF @ISERR(err2280)<>1#AND#@UPPER(match280)="Y"}
                {LET dum280,dummy3280}~{PANELON}{LET lfind280,err2280+
                @LENGTH(dummy280)}~/Cdum280~~{BRANCH cont280}
63 !            {IF @ISERR(err3280)<>1#AND#@UPPER(match280)="N"#AND#
                @UPPER(dummy280)<>@UPPER(dummy1280)}{LET dum280,dummy4280}
                ~{PANELON}{RECALC err3280}~{LET lfind280,err3280+@LENGTH
                (dummy280)}~/Cdum280~~{BRANCH cont280}
64 !            {IF @ISERR(err3280)<>1#AND#@UPPER(match280)="N"#AND#
                @UPPER(dummy280)=@UPPER(dummy1280)}{LET dum280,dummy4280}~
                {PANELON}{LET lfind280,err3280+1}~/Cdum280~~
                {BRANCH cont280}
65 !            {IF flag280=1}{EDIT}{HOME}{DEL}{DEL}~{BRANCH cont1280}
66 !            {EDIT}{HOME}{DEL}~
67 cont1280     {DOWN}
68 !
69 lfind280             0
70 dummy280     SS
71 dummy1280    DD
72 dummy2280            0
73 dummy3280    dummy2280dummy280dummy2280
74 dummy4280    dummy2280dummy280dummy2280
75 flag280              0
76 match280     N
77 match1280    F
78 err2280            ERR
79 err3280            ERR
80 findd280           ERR
81 findd1280           31
82 key280       S
83 !
84 search2280   {WINDOWSOFF}{PANELOFF}{RECALC dummy2280}~
                {RECALC dummy3280}~{RECALC dummy4280}~{RECALC err2280}~
                {RECALC err3280}~
85 !            {LET flag280,0}~{LET lfind280,0}~
                {LET previous280,@CELLPOINTER("address")}~
86 !            {IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~
                {RECALC dummy2280}~{RECALC dummy3280}~{RECALC dummy4280}~
                {LET flag280,1}~
87 loop280      {WINDOWSOFF}{IF @UPPER(match280)="Y"#OR#flag280=1}
                {RECALC err2280}~{RECALC findd280}~{continue1280}
88 !            {WINDOWSOFF}{IF @UPPER(match280)="N"}{RECALC err3280}~
                {continue2280}
89 !            {PANELON}{RECALC backk1280}~{back1280}{DOWN}
90 !
91 continue2280 {IF @ISERR(err3280)<>1#OR#err3280=0}{RECALC dummy2280}~
                /RVdummy2280~dum280~{PANELON}{GOTO}dum280~{WINDOWSON}
                {RECALC err3280}{EDIT}{HOME}{RIGHT err3280+1}{BEEP}
                {GET key280}~{BRANCH cvv280}
92 !
93 continue1280 {IF @ISERR(err2280)<>1#OR#err2280=0}{RECALC dummy2280}~
                /RVdummy2280~dum280~{PANELON}{GOTO}dum280~{WINDOWSON}
                {RECALC findd280}{EDIT}{HOME}{RIGHT findd280+1}{BEEP}
                {GET key280}~{BRANCH cvv280}
94 !
```

```
 95 cvv280          {WINDOWSOFF}{PANELOFF}{RECALC err2280}~{RECALC err3280}~
                    {IF @UPPER(key280)="R"#AND#@UPPER(match280)="Y"}
                    /RVerr2280~findd1280~{BRANCH loop1280}
 96 !               {WINDOWSOFF}{PANELOFF}{IF @UPPER(key280)="R"#AND#@UPPER
                    (match280)="N"}/RVerr3280~findd1280~{BRANCH loop1280}
 97 !               {IF @UPPER(match280)="Y"}{WINDOWSOFF}{PANELOFF}/RVerr2280~
                    findd1280~
 98 !               {IF @UPPER(match280)="N"}{WINDOWSOFF}{PANELOFF}/RVerr3280~
                    findd1280~
 99 loop1280        {IF @UPPER(key280)="R"}{LET dum280,@LEFT(dum280,findd1280)
                    &dummy280&@RIGHT(dum280,@LENGTH(dum280)-findd1280-@LENGTH
                    (dummy1280))}~{PANELON}{LET lfind280,findd1280+1}~
                    {loop280}
100 !               {IF @UPPER(key280)<>"R"#AND#@UPPER(key280)<>"Q"}
                    {LET lfind280,findd1280+1}~{loop280}
101 !               {IF @UPPER(key280)="Q"}{RECALC backk1280}~{back1280}{QUIT}
102 !               {RECALC backk280}~{back280}{DOWN}
103 !
104 !
105 back280         {WINDOWSOFF}{PANELOFF}/Cdum280~
106 backk280        C108
107 !               ~{GOTO}
108 previous280     $A$163
109 !               ~
110 !               {IF flag280=1}{EDIT}{HOME}{DEL}{DEL}~
111 !
112 back1280        {WINDOWSOFF}{PANELOFF}{GOTO}
113 backk1280       C108
114 !               ~
115 !               {IF flag280=1}{EDIT}{HOME}{DEL}{DEL}~
116 !
117 !
118 !
119 !
120 !
121 !
122 dum280          RRR RRRRRR HFSSF SSSSH HHDDH HHDDH F G F GFFFF F
123 !
124 !
125 !
126 !
127 !                                   Choose one option when you here the beep
128 !
129 !
130 !                                                   Press <R> to Replace
131 !
132 !                                                   Press <S> to Skip
133 !
134 !                                                   Press <Q> to Quit
135 !
136 !                                                   Ctrl Break
```

This macro is a Word Processor macro with a full featured Search and Replace. It allows you to use Lotus 1-2-3 as a simple word processor with the option to justify the text to any line width and allows you to use Search and Replace. However, we are not going to study the Search and Replace code because it is identical to the code of the Search and Replace macro SCRHREPD.WK1.

This macro uses a custom menu with four menu options which cannot be displayed clearly without overlapping on the screen or on a page. Therefore we show every menu option separately. If you intend to key this macro into Lotus 1-2-3, you have to key the menu code as it appears here into the B14..E18 range, NOT the code as shown in the main listing. You can find a full list of all the cell contents including string formulas at the end of this section.

```
Typing
Type your document and press [ENTER] when you are finished or here
```

```
    a BEEP
{loopp1280}
{MENUBRANCH menu1280}

Re-justify
Re-adjust the line width
{GETLABEL "Number of characters in a line ? ",num1280}~{setwidth280}
{BRANCH justify280}
{MENUBRANCH menu1280}

Search & Replace
Full featured SEARCH & REPLACE including multiple occurrences in a
    cell
{RESTART}{search1280}{WINDOWSON}
{MENUBRANCH menu1280}

Quit
Quit the word processor
{WINDOWSOFF}{PANELOFF}{BRANCH setwidth1280}{QUIT}
```

To key this macro into Lotus 1-2-3, you have to key in the following formulas in B72, B73, B74, B78, B79, B80, 106 and B113, NOT the results of the formulas as they appear in the main listing of the macro.

```
 72 dummy2280      @CELLPOINTER("contents")
 73 dummy3280      @LEFT(B72,@FIND(B71,B72,B69))&$B$70&@RIGHT(B72,@LENGTH
                   (B72)-@FIND(B71,B72,B69)-@LENGTH(B71))
 74 dummy4280      @LEFT(B72,@FIND(@UPPER(B71),@UPPER(B72),B69))&$B$70&
                   @RIGHT(B72,@LENGTH(B72)-@FIND(@UPPER(B71),@UPPER(B72),
                   B69)-@LENGTH(B71))
 78 err2280        @FIND(B71,B72,B69)
 79 err3280        @FIND(@UPPER(B71),@UPPER(B72),B69)
 80 findd280       @FIND(B71,B72,B69)
106 backk280       @FIND(@UPPER(B71),@UPPER(B72),B69)
113 backk1280      @FIND(@UPPER(B71),@UPPER(B72),B69)
```

	A	B	C	D	E
11	WORDPROC	{LET num2280,@STRING(@CELLPOINTER("width"),0)}~ {GETLABEL "Number of characters in a line ? ",num1280}~ {setwidth280}			
12	!	{MENUBRANCH menu1280}			

The macro starts with `{LET num2280,@STRING(@CELLPOINTER("width"),0)}`, which stores the current column's width as an alphanumeric label in cell [num2280]. Then the macro issues `{GETLABEL "Number of characters in a line ? ",num1280}`, which displays the "`Number of characters in a line ?` " prompt message in the panel. Type the line width and press ENTER to store your response as a label in cell [num1280]. Now the macro issues `{setwidth280}` starting the [setwidth280] routine. If you type the number 50 for the line width, Lotus stored it in B29, [num1280], which becomes part of the [setwidth280] routine.

	A	B	C	D	E
28	setwidth280	/WCS			
29	num1280	50			
30	!	~			

The routine issues `/WCS50~`, which changes the current column width to 50. This is a simple example of dynamic code programming where your response to `{LET}` command is stored in the middle of a routine and becomes part of it before the macro processes it. When the macro finishes this routine, the macro returns control to the next command after `{setwidth280}` which is `{MENUBRANCH menu1280}` and starts the [menu1280] custom menu. The first menu

option is [**T**yping]:

```
Typing
Type your document and press [ENTER] when you are finished or here
    a BEEP
{loopp1280}
{MENUBRANCH menu1280}
```

When you choose this menu option, the macro issues `{loopp1280}`, which starts the [loopp1280] routine:

	A	B	C	D	E
21	loopp1280	{EDIT}{GET key1280}{IF key1280="{ESC}"}{ESC 2}~ {MENUBRANCH menu1280}			
22	!	{IF key1280="~"}{BRANCH justify280}			
23	!	{IF key1280="{LEFT}"#OR#key1280="{RIGHT}"#OR#key1280= "{HOME}"#OR#"{BS}"}{key1280}{?}~{BRANCH justify280}			
24	!	{key1280}{BRANCH loopp1280}			

The routine starts with `{EDIT}`,which enters the EDIT mode, and `{GET key1280}` which halts the macro and waits until you press a key. Lotus stores that key in cell [key1280]. Next the macro issues a series of `{IF}` commands to check which key you pressed. The first is `{IF key1280="{ESC}"}`which checks if you pressed ESC. If so, the macro issues `{ESC 2}` and `{MENUBRANCH menu1280}` to re-activate the custom menu. You cannot use the ESC key to quit.

The second is `{IF key1280="~"}`, which checks if you pressed ENTER (the tilde "~" macro command is equivalent to the ENTER key). If so, the macro uses `{BRANCH justify280}`, which routes macro control to the [justify280] routine, which justifies the text in the paragraph to the column width. Before taking a look at the [justify280] routine, let's continue with the rest of the `{IF}` commands. The third is:

```
{IF key1280="{LEFT}"#OR#key1280="{RIGHT}"#OR#key1280="{HOME}"
    #OR#"{BS}"}
```

This checks if you used the LEFT, RIGHT, HOME or the BACKSPACE keys. If so, the macro issues `{key1280}`to executes the code in the [key1280] routine (`{RIGHT}`, `{LEFT}`, `{HOME}` or `{BS}`). Then the macro issues `{?}`, which allows you to use any key but ENTER. When you press ENTER, the macro issues the tilde "~" and then `{BRANCH justify280}`,which routes macro control to [justify280] justifying the text in the paragraph to the column width.

If all of the conditions are false, the macro issues `{key1280}`, which executes the key in [key1280], and then issues `{BRANCH loopp1280}`, which routes macro control to the beginning of the [loopp1280] routine to allow you to continue to type text into the column. Now let's look at the [justify280] routine.

	A	B	C	D	E
36	justify280	{WINDOWSOFF}{UP}{IF @CELLPOINTER("type")="b"}{DOWN 2} {BRANCH justify1280}			
37	!	{DOWN}{END}{UP}/RJ~{END}{DOWN}{WINDOWSON} {BRANCH loopp1280}			

The routine starts with `{WINDOWSOFF}` which freeze[4] screen display activity, and then issues `{UP}` and `{IF @CELLPOINTER("type")="b"}` to check if the cell above is blank. If so, the macro issues `{DOWN 2}` and `{BRANCH justify1280}` which starts the [justify1280] routine.

```
     A              B                C              D              E
39 justify1280    {IF @CELLPOINTER("type")="b"}{UP}/RJ~{WINDOWSON}
                  {BRANCH loopp1280}
40 !              {DOWN}{IF @CELLPOINTER("type")<>"b"}{UP}{END}{DOWN}
                  {WINDOWSON}
                  {BRANCH loopp1280}
41 !              {UP}{WINDOWSON}{BRANCH loopp1280}
```

The routine starts with `{IF @CELLPOINTER("type")="b"}` to check if the current cell is blank. If so, the macro issues `{UP}` because there is only one cell of text and then `/RJ~`, which justifies the text in the cell to the column width. Then it issues `{WINDOWSON}{BRANCH loopp1280}`, which resume screen display activity, and route macro control back to the [loopp1280] routine to allow you to continue to type. Now the macro issues `{DOWN}` and `{IF @CELLPOINTER("type")<>"b"}` to check if the cell below is not empty.

If so, the macro issues `{UP}`. Because there is at least one occupied cell below the current cell, the macro issues `{END}{DOWN}`, which move the cell pointer to the last contiguous populated cell. Then the macro issues `{WINDOWSON}{BRANCH loopp1280}` which resume screen display activity and route macro control back to the [loopp1280] routine to allow you to continue to type. Whenever you press ENTER, the macro justifies the text and moves to the last contiguous occupied cell. When there is only one cell with text, after justification, the cell pointer stays on that cell.

When the [justify1280] routine is finished, the macro returns control to the [justify280] routine

```
     A              B                C              D              E
36 justify280     {WINDOWSOFF}{UP}{IF @CELLPOINTER("type")="b"}{DOWN 2}
                  {BRANCH justify1280}
37 !              {DOWN}{END}{UP}/RJ~{END}{DOWN}{WINDOWSON}
                  {BRANCH loopp1280}
```

If the cell above was not empty the macro continues with `{DOWN}{END}{UP}`, which put the cell pointer on the first cell in the paragraph with text, and then issues `/RJ~{END}{DOWN}` which justify the contiguous paragraph. Next the macro issues `{WINDOWSON}{BRANCH loopp1280}`, which resume screen display activity and route macro control back to the [loopp1280] routine to allow you to continue to type. When you press ESC, the macro routes control back to the custom menu and issues `{MENUBRANCH menu1280}` to re-start the custom menu. The second menu option is [**R**e-justify]:

```
Re-justify
Re-adjust the line width
{GETLABEL "Number of characters in a line ? ",num1280}~{setwidth280}
{BRANCH justify280}
{MENUBRANCH menu1280}
```

The macro issues `{GETLABEL "Number of characters in a line ? ",num1280}` to prompt you for the text line width, and then issues the `{setwidth280}` routine command to change the column width. Then the macro issues `{BRANCH justify280}`, which we have already explained. When the [justify280] routine is finished, the macro issues `{MENUBRANCH menu1280}` to re-start the custom menu. The third menu option is [**S**earch & Replace]:

```
Search & Replace
Full featured SEARCH & REPLACE including multiple occurrences in a
    cell
{RESTART}{search1280}{WINDOWSON}
```

```
{MENUBRANCH menu1280}
```

When you choose this option, the macro issues {RESTART}{search1280} to start the search and replace [search1280] routine. The code starting from the [search1280] and down is exactly the same as the code in the Search and Replace macro SCRHREPD.WK1, therefore if you are interested see this macro for the code analysis. The last menu option is [**Q**uit]:

```
Quit
Quit the word processor
{WINDOWSOFF}{PANELOFF}{BRANCH setwidth1280}{QUIT}
```

The macro issues {WINDOWSOFF}{PANELOFF} to freeze screen and panel display activities and then {BRANCH setwidth1280}, which starts the [setwidth1280] routine, which resets the column width back to the width before the macro started. When the routine is finished, the macro issues {QUIT} and quits.

	A	B	C	D	E
32	setwidth1280	/WCS			
33	num2280	30			
34	!	~{QUIT}			

This is a simple /WCS30~ macro. The macro stored the original column width in B33, [num2280], when it started. Now the macro uses it to reset the column to the original width. This is a second example of dynamic macro code.

To key this macro into Lotus 1-2-3, we show the full list of all the cell contents and formulas.

```
A1: U [W14] '*---A WORD PROCESSOR macro with a full featured SEARCH &
            REPLACE
A2: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W14] '    range names in this column (starts with the \Z macro name)
A4: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W14] '*---Press {ESC} to return to main menu
A6: [W14] '!
A7: [W14] '!
A8: [W14] '!
A9: [W14] '!
A10: U [W14] '\Z
B10: [W9] '{BREAKON}
A11: U [W14] 'WORDPROC
B11: (,2) [W9] '{LET num2280,@STRING(@CELLPOINTER("width"),0)}~
               {GETLABEL "Number of characters in a line ? ",num1280}~
               {setwidth280}
A12: [W14] '!
B12: [W9] '{MENUBRANCH menu1280}
A13: [W14] '!
A14: [W14] 'menu1280
B14: [W9] 'Typing
C14: [W10] 'Re-justify
D14: [W9] 'Search & Replace
E14: 'Quit
A15: [W14] '!
B15: [W9] 'Type your document and press [ENTER] when you are finished or
          here a BEEP
C15: [W10] 'Re-adjust the line width
D15: [W9] 'Full featured SEARCH & REPLACE including multiple occurrences
          in a cell
E15: 'Quit the word processor
A16: [W14] '!
B16: [W9] '{loopp1280}
C16: (,2) [W10] '{GETLABEL "Number of characters in a line ? ",num1280}~
```

```
          {setwidth280}
D16: [W9] '{RESTART}{search1280}{WINDOWSON}
E16: '{WINDOWSOFF}{PANELOFF}{BRANCH setwidth1280}{QUIT}
A17: [W14] '!
B17: [W9] '{MENUBRANCH menu1280}
C17: [W10] '{BRANCH justify280}
D17: [W9] '{MENUBRANCH menu1280}
A18: [W14] '!
C18: [W10] '{MENUBRANCH menu1280}
A19: [W14] '!
A20: [W14] '!
A21: [W14] 'loopp1280
B21: (,2) [W9] '{EDIT}{GET key1280}{IF key1280="{ESC}"}{ESC 2}~
               {MENUBRANCH menu1280}
A22: [W14] '!
B22: (,2) [W9] '{IF key1280="~"}{BRANCH justify280}
A23: [W14] '!
B23: [W9] '{IF key1280="{LEFT}"#OR#key1280="{RIGHT}"#OR#key1280="{HOME}"
          #OR#"{BS}"}{key1280}{?}~{BRANCH justify280}
A24: [W14] '!
B24: [W9] '{key1280}{BRANCH loopp1280}
A25: [W14] '!
A26: [W14] 'key1280
B26: [W9] '{ESC}
A27: [W14] '!
A28: [W14] 'setwidth280
B28: [W9] '/WCS
A29: [W14] 'num1280
B29: [W9] '50
A30: [W14] '!
B30: [W9] '~
A31: [W14] '!
A32: [W14] 'setwidth1280
B32: [W9] '/WCS
A33: [W14] 'num2280
B33: [W9] '30
A34: [W14] '!
B34: [W9] '~{QUIT}
A35: [W14] '!
A36: [W14] 'justify280
B36: (,2) [W9] '{WINDOWSOFF}{UP}{IF @CELLPOINTER("type")="b"}{DOWN 2}
               {BRANCH justify1280}
A37: [W14] '!
B37: [W9] '{DOWN}{END}{UP}/RJ~{END}{DOWN}{WINDOWSON}{BRANCH loopp1280}
A38: [W14] '!
A39: [W14] 'justify1280
B39: (,2) [W9] '{IF @CELLPOINTER("type")="b"}{UP}/RJ~{WINDOWSON}
               {BRANCH loopp1280}
A40: [W14] '!
B40: [W9] '{DOWN}{IF @CELLPOINTER("type")<>"b"}{UP}{END}{DOWN}{WINDOWSON}
          {BRANCH loopp1280}
A41: [W14] '!
B41: [W9] '{UP}{WINDOWSON}{BRANCH loopp1280}
A42: U [W14] 'search1280
B42: [W9] '{LET lfind280,0}{LET flag280,0}{WINDOWSOFF}{PANELOFF}/RNC
          Which range ?~~/RNDWhich range ?~/RNC{WINDOWSON}{PANELON}
          Which range ?~{BS}{BS}{?}~{GOTO}Which range ?~{prompt280}
A43: [W14] '!
B43: [W9] '{LET counter1280,0}~
A44: [W14] '!
B44: [W9] '{FOR counter1280,0,@COLS(Which range ?)-1,1,labels1280}
A45: [W14] '!
B45: [W9] '{GOTO}Which range ?~/RNDWhich range ?~
A46: [W14] '!
A47: [W14] 'counter1280
B47: [W9] 1
A48: [W14] 'counter1a280
B48: [W9] 0
A49: [W14] 'labels1280
B49: [W9] '{IF @UPPER(match1280)="A"}{FOR counter1a280,0,@ROWS
          (Which range ?)-1,1,cont2280}~{RIGHT}{UP @ROWS(Which range ?)}
```

```
          {LET counter1a280,0}~
A50: [W14] '!
B50: [W9] '{IF @UPPER(match1280)="F"}{FOR counter1a280,0,@ROWS
          (Which range ?)-1,1,search2280}~{RIGHT}{UP @ROWS(Which range ?)}
          {LET counter1a280,0}~
A51: [W14] '!
A52: [W14] 'prompt280
B52: [W9] '{GETLABEL "Type the OLD string/number to be replaced: ",
          dummy1280}~
A53: [W14] '!
B53: [W9] '{GETLABEL "Type the NEW string/number : ",dummy280}~
           {IF @CELLPOINTER("type")="b"}{cont1280}
A54: [W14] '!
B54: [W9] '{GETLABEL "Match (exact) Y/N: ",match280}~
A55: [W14] '!
B55: [W9] '{GETLABEL "<F>ind & replace / replace <A>ll ",match1280}~
A56: [W14] '!
A57: [W14] '!
A58: [W14] 'cont2280
B58: [W9] '{windowsOFF}{PANELOFF}{IF @CELLPOINTER("type")="b"}
          {BRANCH cont1280}
A59: [W14] '!
B59: [W9] '{IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~{LET flag280,1}~
          {LET lfind280,0}~{BRANCH cont280}
A60: [W14] '!
B60: [W9] '{EDIT}{HOME}'~{LET flag280,0}~{LET lfind280,0}~
A61: [W14] 'cont280
B61: [W9] '{PANELOFF}{RECALC dummy2280}~{RECALC dummy3280}~
          {RECALC dummy4280}~{RECALC err2280}~{RECALC err3280}~
A62: [W14] '!
B62: [W9] '{IF @ISERR(err2280)<>1#AND#@UPPER(match280)="Y"}{LET dum280,
          dummy3280}~{PANELON}{LET lfind280,err2280+@LENGTH(dummy280)}~
          /Cdum280~~{BRANCH cont280}
A63: [W14] '!
B63: [W9] '{IF @ISERR(err3280)<>1#AND#@UPPER(match280)="N"#AND#@UPPER
          (dummy280)<>@UPPER(dummy1280)}{LET dum280,dummy4280}~{PANELON}
          {RECALC err3280}~{LET lfind280,err3280+@LENGTH(dummy280)}~
          /Cdum280~~{BRANCH cont280}
A64: [W14] '!
B64: [W9] '{IF @ISERR(err3280)<>1#AND#@UPPER(match280)="N"#AND#@UPPER
          (dummy280)=@UPPER(dummy1280)}{LET dum280,dummy4280}~{PANELON}
          {LET lfind280,err3280+1}~/Cdum280~~{BRANCH cont280}
A65: [W14] '!
B65: [W9] '{IF flag280=1}{EDIT}{HOME}{DEL}{DEL}~{BRANCH cont1280}
A66: [W14] '!
B66: [W9] '{EDIT}{HOME}{DEL}~
A67: [W14] 'cont1280
B67: [W9] '{DOWN}
A68: [W14] '!
A69: [W14] 'lfind280
B69: [W9] 0
A70: [W14] 'dummy280
B70: [W9] 'SS
A71: [W14] 'dummy1280
B71: [W9] 'DD
A72: [W14] 'dummy2280
B72: [W9] @CELLPOINTER("contents")
A73: [W14] 'dummy3280
B73: [W9] @LEFT(B72,@FIND(B71,B72,B69))&$B$70&@RIGHT(B72,@LENGTH(B72)-@FIND
          (B71,B72,B69)-@LENGTH(B71))
A74: [W14] 'dummy4280
B74: [W9] @LEFT(B72,@FIND(@UPPER(B71),@UPPER(B72),B69))&$B$70&@RIGHT(B72,
          @LENGTH(B72)-@FIND(@UPPER(B71),@UPPER(B72),B69)-@LENGTH(B71))
A75: [W14] 'flag280
B75: [W9] 0
A76: [W14] 'match280
B76: [W9] 'N
A77: [W14] 'match1280
B77: [W9] 'F
A78: [W14] 'err2280
B78: [W9] @FIND(B71,B72,B69)
```

```
A79: [W14] 'err3280
B79: [W9] @FIND(@UPPER(B71),@UPPER(B72),B69)
A80: [W14] 'findd280
B80: [W9] @FIND(B71,B72,B69)
A81: [W14] 'findd1280
B81: [W9] 31
A82: [W14] 'key280
B82: [W9] 'S
A83: [W14] '!
A84: [W14] 'search2280
B84: [W9] '{WINDOWSOFF}{PANELOFF}{RECALC dummy2280}~{RECALC dummy3280}~
          {RECALC dummy4280}~{RECALC err2280}~{RECALC err3280}~
A85: [W14] '!
B85: [W9] '{LET flag280,0}~{LET lfind280,0}~{LET previous280,@CELLPOINTER
          ("address")}~
A86: [W14] '!
B86: [W9] '{IF @CELLPOINTER("type")="v"}{EDIT}{HOME}''~{RECALC dummy2280}~
          {RECALC dummy3280}~{RECALC dummy4280}~{LET flag280,1}~
A87: [W14] 'loop280
B87: [W9] '{WINDOWSOFF}{IF @UPPER(match280)="Y"#OR#flag280=1}
          {RECALC err2280}~{RECALC findd280}~{continue1280}
A88: [W14] '!
B88: [W9] '{WINDOWSOFF}{IF @UPPER(match280)="N"}{RECALC err3280}~
          {continue2280}
A89: [W14] '!
B89: [W9] '{PANELON}{RECALC backk1280}~{back1280}{DOWN}
A90: [W14] '!
A91: [W14] 'continue2280
B91: [W9] '{IF @ISERR(err3280)<>1#OR#err3280=0}{RECALC dummy2280}~
          /RVdummy2280~dum280~{PANELON}{GOTO}dum280~{WINDOWSON}
          {RECALC err3280}{EDIT}{HOME}{RIGHT err3280+1}{BEEP}{GET key280}~
          {BRANCH cvv280}
A92: [W14] '!
A93: [W14] 'continue1280
B93: [W9] '{IF @ISERR(err2280)<>1#OR#err2280=0}{RECALC dummy2280}~
          /RVdummy2280~dum280~{PANELON}{GOTO}dum280~{WINDOWSON}
          {RECALC findd280}{EDIT}{HOME}{RIGHT findd280+1}{BEEP}
          {GET key280}~{BRANCH cvv280}
A94: [W14] '!
A95: [W14] 'cvv280
B95: [W9] '{WINDOWSOFF}{PANELOFF}{RECALC err2280}~{RECALC err3280}~
          {IF @UPPER(key280)="R"#AND#@UPPER(match280)="Y"}/RVerr2280~
          findd1280~{BRANCH loop1280}
A96: [W14] '!
B96: [W9] '{WINDOWSOFF}{PANELOFF}{IF @UPPER(key280)="R"#AND#@UPPER
          (match280)="N"}/RVerr3280~findd1280~{BRANCH loop1280}
A97: [W14] '!
B97: [W9] '{IF @UPPER(match280)="Y"}{WINDOWSOFF}{PANELOFF}/RVerr2280~
          findd1280~
A98: [W14] '!
B98: [W9] '{IF @UPPER(match280)="N"}{WINDOWSOFF}{PANELOFF}/RVerr3280~
          findd1280~
A99: [W14] 'loop1280
B99: [W9] '{IF @UPPER(key280)="R"}{LET dum280,@LEFT(dum280,findd1280)&
          dummy280&@RIGHT(dum280,@LENGTH(dum280)-findd1280-@LENGTH
          (dummy1280))}~{PANELON}{LET lfind280,findd1280+1}~{loop280}
A100: [W14] '!
B100: [W9] '{IF @UPPER(key280)<>"R"#AND#@UPPER(key280)<>"Q"}{LET lfind280,
           findd1280+1}~{loop280}
A101: [W14] '!
B101: [W9] '{IF @UPPER(key280)="Q"}{RECALC backk1280}~{back1280}{QUIT}
A102: [W14] '!
B102: [W9] '{RECALC backk280}~{back280}{DOWN}
A103: [W14] '!
A104: [W14] '!
A105: [W14] 'back280
B105: [W9] '{WINDOWSOFF}{PANELOFF}/Cdum280~
A106: [W14] 'backk280
B106: [W9] +B108
A107: [W14] '!
B107: [W9] '~{GOTO}
```

```
A108: [W14] 'previous280
B108: [W9] '$A$163
A109: [W14] '!
B109: [W9] '~
A110: [W14] '!
B110: [W9] '{IF flag280=1}{EDIT}{HOME}{DEL}{DEL}~
A111: [W14] '!
A112: [W14] 'back1280
B112: [W9] '{WINDOWSOFF}{PANELOFF}{GOTO}
A113: [W14] 'backk1280
B113: [W9] +B108
A114: [W14] '!
B114: [W9] '~
A115: [W14] '!
B115: [W9] '{IF flag280=1}{EDIT}{HOME}{DEL}{DEL}~
A116: [W14] '!
A117: [W14] '!
A118: [W14] '!
A119: [W14] '!
A120: [W14] '!
A121: [W14] '!
A122: [W14] 'dum280
B122: [W9] 'RRR RRRRRR HFSSF SSSSH HHDDH HHDDH F G F GFFFF F
A123: [W14] '!
A124: [W14] '!
A125: [W14] '!
A126: [W14] '!
A127: [W14] '!
D127: [W9] 'Choose one option when you here the beep
A128: [W14] '!
A129: [W14] '!
A130: [W14] '!
E130: U 'Press <R> to Replace
A131: [W14] '!
A132: [W14] '!
E132: U 'Press <S> to Skip
A133: [W14] '!
A134: [W14] '!
E134: U 'Press <Q> to Quit
A135: [W14] '!
A136: [W14] '!
E136: U 'Ctrl Break
```

[9] Calendar

```
     A              B              C              D              E              F
 1 *---A Calendar macro. The screen displays four months at a time
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}{WINDOWSOFF}{PANELOFF}{GOTO}current4093~{LEFT}
                  {END}{DOWN}{FOR counts093,1,16,1,wid093}{WINDOWSON}
                  {PANELON}
11 CALANDER       {LET current1093,@NOW-@DAY(@NOW)+16}~{CALC}
                  {MENUBRANCH menu1093}
12 !
13 menu1093       Previous    Next        Quit
14 !              Shows previoShows next fQuit the macro
15 !              {LET current{LET current{WINDOWSOFF}{PANELOFF}{FOR count
16 !              {MENUBRANCH {MENUBRANCH menu1093}
17 !
18 current1093         Jan-93
19 current2093         Feb-93
20 current3093         Mar-93
21 current4093         Apr-93
22 !
23 wid093         /WCS4~{RIGHT}
24 !
25 unwid093       /WCS9~{RIGHT}
26 !
27 counts093              17
28 !
```

```
     A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q
29 !    Sun MonTue Wed Thu Fri Sat     Sun Mon Tue Wed Thu Fri Sat
30 !
31 !           January 1993                    February 1993
32 !   ~~~~~~~~~~~~~~~~~~~~~~~~~~~    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
33 !                           1   2          1   2   3   4   5   6
34 !      3  4   5   6   7   8   9        7   8   9  10  11  12  13
35 !     10 11  12  13  14  15  16       14  15  16  17  18  19  20
36 !     17 18  19  20  21  22  23       21  22  23  24  25  26  27
37 !     24 25  26  27  28  29  30       28
38 !     31
39 !
40 !           March 1993                      April 1993
41 !    ~~~~~~~~~~~~~~~~~~~~~~~~~~~    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
42 !         1   2   3   4   5   6                      1   2   3
43 !      7  8   9  10  11  12  13        4   5   6   7   8   9  10
44 !     14 15  16  17  18  19  20       11  12  13  14  15  16  17
45 !     21 22  23  24  25  26  27       18  19  20  21  22  23  24
46 !     28 29  30  31                   25  26  27  28  29  30
47 !
48 !    _________________________      _____________________________
```

This macro displays a calendar of four months simultaneously. You can scroll back and forth and display the next four months or the previous four months using a custom menu. Notice that we have split the code text into two parts, beginning of row number 29, and changed the column width to display the calendar the way it is displayed when you start the macro. This is quite a complicated macro and every date which appears on the display is the result of a long and complicated formula. If you intend to key this macro into Lotus 1-2-3, use the full list of cell formulas and contents as they appear at the end of this section. This macro also uses a custom menu which cannot be displayed without overlapping on the screen or on this page, therefore we show the code for each menu option separately.

```
Previous
Shows previous four months
{LET current1093,current1093-124}~{CALC}
{MENUBRANCH menu1093}

Next
Shows next four months
{LET current1093,current1093+124}~{CALC}
{MENUBRANCH menu1093}

Quit
Quit the macro
{WINDOWSOFF}{PANELOFF}{FOR counts093,1,16,1,unwid093}{HOME}{END}
    {DOWN}
```

	A	B	C	D	E	F
10	\Z	{BREAKON}{WINDOWSOFF}{PANELOFF}{GOTO}current4093~{LEFT}				
		{END}{DOWN}{FOR counts093,1,16,1,wid093}{WINDOWSON}				
		{PANELON}				
11	CALANDER	{LET current1093,@NOW-@DAY(@NOW)+16}~{CALC}				
		{MENUBRANCH menu1093}				

The macro starts with the {WINDOWSOFF}{PANELOFF} commands which freeze screen and panel display activities, and then it issues {GOTO}current4093~, which moves the cell pointer to cell [current4093]. Next the macro issues {LEFT}{END}{DOWN}, which move the cell pointer to the last row of the macro and position the calendar on the screen as it appears in the macro listing, beginning with the 29th row. Now the macro issues {FOR counts093 ,1,16,1,wid093}, which executes the [wid093] routine sixteen (16) times.

	A	B	C	D	E	F
23	wid093	/WCS4~{RIGHT}				

The [wid093] routine issues /WCS4~{RIGHT} to change the column width to four characters wide and move the cell pointer to the next column. The macro executes this routine sixteen times to set the appearance of the calendar in the second part of the macro beginning with the 29th row of the macro. Next the macro issues {WINDOWSON}{PANELON}, which resume the screen and panel display activities. Next the macro issues {LET current1093 ,@NOW-@DAY(@NOW)+16}, which inserts the date of the 16th of the current month into cell [current1093]. The reason is: the next three months are calculated by adding 31 days; therefore it is best to start with the middle of the month to avoid skipping a month such as February. The macro continues with {CALC}, which recalculates the worksheet and updates all the date and string formulas based on the content of [current1093]. Then the macro issues {MENUBRANCH menu1093}, which starts the [menu1093] custom menu. The first menu option is [**P**revious]:

```
Previous
Shows previous four months
{LET current1093,current1093-124}~{CALC}
{MENUBRANCH menu1093}
```

The macro issues {LET current1093,current1093-124}, which decreases the date value in [current1093] by 124, days which is four months ago, and then {CALC}, which recalculates the worksheet and updates all the date and string formulas based on the content of [current1093]. The calendar will display the previous four months. The second menu option is [**N**ext]:

```
Next
```

```
Shows next four months
{LET current1093,current1093+124}~{CALC}
{MENUBRANCH menu1093}
```

The macro issues {LET current1093,current1093-124}, which increases the date value in [current1093] by 124 days, which is four months ahead, and then {CALC}, which recalculates the worksheet and updates all the date and string formulas based on the content of [current1093]. The calendar will display the next four months. The third menu option is [**Q**uit]:

```
Quit
Quit the macro
{WINDOWSOFF}{PANELOFF}{FOR counts093,1,16,1,unwid093}{HOME}{END}
    {DOWN}
```

The macro issues {WINDOWSOFF}{PANELOFF}, which freeze screen and panel display activities. Now the macro issues {FOR counts093,1,16,1,unwid093}, which executes the [unwid093] routine sixteen (16) times.

	A	B	C	D	E	F
25	unwid093	/WCS9~{RIGHT}				

The [unwid093] routine issues /WCS9~{RIGHT} to change the column width to nine character wide and move the cell pointer to the next column. The macro executes this routine sixteen times to set the macro back to normal column width.

To explained how the macro calculates all the dates, months, names, etc. let's begin with the formula in the B19 cell:

```
+B18+31
```

This formula adds 31 days to the date in the B18 cell and displays the second month of the four. The formula in the B20 cell is:

```
+B18+62
```

This adds 62 days to the date in the B18 cell and displays the third month of the four. The formula in the B21 cell is:

```
+B18+93
```

This adds 93 days to the date in the B18 cell and displays the fourth month of the four. Now the macro "knows" for which months to display the calendar.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
29	!	Sun	Mon	Tue	Wed	Thu	Fri	Sat		Sun	Mon	Tue	Wed	Thu	Fri	Sat	
30	!																
31	!				January 1993								February 1993				
32	!	~~~~~~~~~~~~~~~~~~~~~~~~~~~								~~~~~~~~~~~~~~~~~~~~~~~~~~~							
33	!						1	2			1	2	3	4	5	6	
34	!	3	4	5	6	7	8	9		7	8	9	10	11	12	13	
35	!	10	11	12	13	14	15	16		14	15	16	17	18	19	20	
36	!	17	18	19	20	21	22	23		21	22	23	24	25	26	27	
37	!	24	25	26	27	28	29	30		28							
38	!	31															
39	!																
40	!				March 1993								April 1993				
41	!	~~~~~~~~~~~~~~~~~~~~~~~~~~~								~~~~~~~~~~~~~~~~~~~~~~~~~~~							

```
42 !          1   2   3   4   5   6                       1   2   3
43 !      7   8   9  10  11  12  13       4   5   6   7   8   9  10
44 !     14  15  16  17  18  19  20      11  12  13  14  15  16  17
45 !     21  22  23  24  25  26  27      18  19  20  21  22  23  24
46 !     28  29  30  31                  25  26  27  28  29  30
47 !
48 !   ____________________________    ____________________________
```

The four month calendar display, as shown here, is totally formula dependent. The "January 1993", "February 1993", "March 1993" and "April 1993" titles are the result of string formulas. The dates display in B33..H38, J33..P38, B42..H47 and J42..P47 are the result of approximately 140 formulas.

Let's look at the formulas behind the "January 1993", "February 1993", "March 1993" and "April 1993" titles, starting with the formula in the D31 cell which creates "January 1993" in the macro listing is:

```
@CHOOSE(@MONTH($B$18)-1,"January","February","March","April","May"
    ,"June","July","August","September","October","November",
    "December")&" "&@STRING(@YEAR($B$18)+1900,0)
```

This formula uses the `@CHOOSE` Lotus function to extract the month name and then add to it the string of the year. If B18 contains the number "33974.803206" then the `@CHOOSE` formula returns `January` and the `@YEAR($B$18)` function returns `93`; therefore the `@STRING (@YEAR($B$18)+1900,0)` formula returns the `1993` string and the result is `January 1993`. Let's look at the formula which creates the date in the B33 cell.

```
@IF(@IF(@MOD(@INT($B$18-@DAY($B$18)+1),7)=0,7,@MOD(@INT($B$18-
    @DAY($B$18)+1),7))=1,1,"")
```

This formula looks for the 1st day of the month; therefore it only allows the number 1 or the null "" character. The formula in C33 is:

```
@IF(@IF(@MOD(@INT($B$18-@DAY($B$18)+1),7)=0,7,@MOD(@INT($B$18-
    @DAY($B$18)+1),7))=2,1,@IF(B33>0,B33+1,""))
```

In addition, it checks if B33 contains a number greater than zero. If so, it returns a number greater than the number in B33 by 1. For example: if B33 returns the null "" string, this formula will return only 1 or null "" as the previous cell. However, if the previous cell contain the number 1 than this formula returns 2. The five formulas in D33, E33, F33, G33 and H33 are the same as C33 because the first day of the month may be in any one of these seven cells. To summarize: if Sunday is the First day of the month, then B33 returns 1 and the rest follow (C33=2, D33=3, E33=4, F33=5, G33=6 and H33=7). For example, if the First day of the month is Wednesday, then B33, C33, D33 return the null "" string and E33 returns 1 and (F33=2, G33=3 and H33=3).

To accept the dates up to the 28th of the month is as easy as increasing each cell by one. For example, the formula in B34 is `+H33+1`. Therefore all the formulas in the B34..H36 range are the same, they add 1 to the previous cell. The last seven formulas in the B37..H37 range are different because the last day of the month may be Sunday to Saturday. The first formula in the B37 cell is:

```
@IF(@DAY(@INT($B$18-@DAY($B$18))+1+H36)<H36,"",@DAY(@INT($B$18-
    @DAY($B$18))+1+H36))
```

This formula checks if the day of the current date is less than the previous date in B36. If so, this

is the First day of the next month and the date in B36 is the last day of the current month. Therefore the formula will return the null "" string. Otherwise it increases the day in B36 by 1. The formulas in C37..H37 are the same. The other four tables are the same; therefore you are on your own.

To key this macro into Lotus 1-2-3, we show the full list of all the cell contents in the macro.

```
A1: U [W15] '*---A Calendar macro. The screen displays four months at a
            time
A2: [W15] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W15] '    range names in this column (starts with the \Z macro name)
A4: [W15] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W15] '!
A6: [W15] '!
A7: [W15] '!
A8: [W15] '!
A9: [W15] '!
A10: U [W15] '\Z
B10: [W12] '{BREAKON}{WINDOWSOFF}{PANELOFF}{GOTO}current4093~{LEFT}{END}
           {DOWN}{FOR counts093,1,16,1,wid093}{HOME}{END}{DOWN}{WINDOWSON}
           {PANELON}
A11: U [W15] 'CALANDER
B11: [W12] '{LET current1093,@NOW-@DAY(@NOW)+16}~{CALC}{MENUBRANCH menu1093}
A12: [W15] '!
A13: [W15] 'menu1093
B13: [W12] 'Previous
C13: [W12] 'Next
D13: [W4] 'Quit
A14: [W15] '!
B14: [W12] 'Shows previous four months
C14: [W12] 'Shows next four months
D14: [W4] 'Quit the macro
A15: [W15] '!
B15: [W12] '{LET current1093,current1093-124}~{CALC}
C15: [W12] '{LET current1093,current1093+124}~{CALC}
D15: [W4] '{WINDOWSOFF}{PANELOFF}{FOR counts093,1,16,1,unwid093}{HOME}{END}
          {DOWN}
A16: [W15] '!
B16: [W12] '{MENUBRANCH menu1093}
C16: [W12] '{MENUBRANCH menu1093}
A17: [W15] '!
A18: [W15] 'current1093
B18: (D3) U [W12] 33974.803206
A19: [W15] 'current2093
B19: (D3) U [W12] +B18+31
A20: [W15] 'current3093
B20: (D3) U [W12] +B18+62
A21: [W15] 'current4093
B21: (D3) U [W12] +B18+93
A22: [W15] '!
A23: [W15] 'wid093
B23: [W12] '/WCS4~{RIGHT}
A24: [W15] '!
A25: [W15] 'unwid093
B25: [W12] '/WCS9~{RIGHT}
A26: [W15] '!
A27: [W15] 'counts093
B27: [W12] 17
A28: [W15] '!
A29: [W15] '!
B29: U [W12] 'Sun
C29: U [W12] 'Mon
D29: U [W4] 'Tue
E29: U [W4] 'Wed
F29: U [W4] 'Thu
G29: U [W4] 'Fri
H29: U [W4] 'Sat
```

```
J29: U [W4] 'Sun
K29: U [W4] 'Mon
L29: U [W4] 'Tue
M29: U [W4] 'Wed
N29: U [W4] 'Thu
O29: U [W4] 'Fri
P29: U [W4] 'Sat
A30: [W15] '!
A31: [W15] '!
D31: U [W4] @CHOOSE(@MONTH($B$18)-1,"January","February","March","April",
            "May","June","July","August","September","October","November"
            ,"December")&" "&@STRING(@YEAR($B$18)+1900,0)
L31: U [W4] @CHOOSE(@MONTH($B$19)-1,"January","February","March","April",
            "May","June","July","August","September","October","November"
            ,"December")&" "&@STRING(@YEAR($B$19)+1900,0)
A32: [W15] '!
B32: U [W12] \~
C32: U [W12] \~
D32: U [W4] \~
E32: U [W4] \~
F32: U [W4] \~
G32: U [W4] \~
H32: U [W4] \~
J32: U [W4] \~
K32: U [W4] \~
L32: U [W4] \~
M32: U [W4] \~
N32: U [W4] \~
O32: U [W4] \~
P32: U [W4] \~
A33: [W15] '!
B33: [W12] @IF(@IF(@MOD(@INT($B$18-@DAY($B$18)+1),7)=0,7,@MOD(@INT($B$18-
           @DAY($B$18)+1),7))=1,1,"")
C33: [W12] @IF(@IF(@MOD(@INT($B$18-@DAY($B$18)+1),7)=0,7,@MOD(@INT($B$18-
           @DAY($B$18)+1),7))=2,1,@IF(B33>0,B33+1,""))
D33: [W4] @IF(@IF(@MOD(@INT($B$18-@DAY($B$18)+1),7)=0,7,@MOD(@INT($B$18-
          @DAY($B$18)+1),7))=3,1,@IF(C33>0,C33+1,""))
E33: [W4] @IF(@IF(@MOD(@INT($B$18-@DAY($B$18)+1),7)=0,7,@MOD(@INT($B$18-
          @DAY($B$18)+1),7))=4,1,@IF(D33>0,D33+1,""))
F33: [W4] @IF(@IF(@MOD(@INT($B$18-@DAY($B$18)+1),7)=0,7,@MOD(@INT($B$18-
          @DAY($B$18)+1),7))=5,1,@IF(E33>0,E33+1,""))
G33: [W4] @IF(@IF(@MOD(@INT($B$18-@DAY($B$18)+1),7)=0,7,@MOD(@INT($B$18-
          @DAY($B$18)+1),7))=6,1,@IF(F33>0,F33+1,""))
H33: [W4] @IF(@IF(@MOD(@INT($B$18-@DAY($B$18)+1),7)=0,7,@MOD(@INT($B$18-
          @DAY($B$18)+1),7))=7,1,@IF(G33>0,G33+1,""))
J33: [W4] @IF(@IF(@MOD(@INT($B$19-@DAY($B$19)+1),7)=0,7,@MOD(@INT($B$19-
          @DAY($B$19)+1),7))=1,1,"")
K33: [W4] @IF(@IF(@MOD(@INT($B$19-@DAY($B$19)+1),7)=0,7,@MOD(@INT($B$19-
          @DAY($B$19)+1),7))=2,1,@IF(J33>0,J33+1,""))
L33: [W4] @IF(@IF(@MOD(@INT($B$19-@DAY($B$19)+1),7)=0,7,@MOD(@INT($B$19-
          @DAY($B$19)+1),7))=3,1,@IF(K33>0,K33+1,""))
M33: [W4] @IF(@IF(@MOD(@INT($B$19-@DAY($B$19)+1),7)=0,7,@MOD(@INT($B$19-
          @DAY($B$19)+1),7))=4,1,@IF(L33>0,L33+1,""))
N33: [W4] @IF(@IF(@MOD(@INT($B$19-@DAY($B$19)+1),7)=0,7,@MOD(@INT($B$19-
          @DAY($B$19)+1),7))=5,1,@IF(M33>0,M33+1,""))
O33: [W4] @IF(@IF(@MOD(@INT($B$19-@DAY($B$19)+1),7)=0,7,@MOD(@INT($B$19-
          @DAY($B$19)+1),7))=6,1,@IF(N33>0,N33+1,""))
P33: [W4] @IF(@IF(@MOD(@INT($B$19-@DAY($B$19)+1),7)=0,7,@MOD(@INT($B$19-
          @DAY($B$19)+1),7))=7,1,@IF(O33>0,O33+1,""))
A34: [W15] '!
B34: [W12] +H33+1
C34: [W12] +B34+1
D34: [W4] +C34+1
E34: [W4] +D34+1
F34: [W4] +E34+1
G34: [W4] +F34+1
H34: [W4] +G34+1
J34: [W4] +P33+1
K34: [W4] +J34+1
L34: [W4] +K34+1
M34: [W4] +L34+1
```

```
N34: [W4] +M34+1
O34: [W4] +N34+1
P34: [W4] +O34+1
A35: [W15] '!
B35: [W12] +H34+1
C35: [W12] +B35+1
D35: [W4] +C35+1
E35: [W4] +D35+1
F35: [W4] +E35+1
G35: [W4] +F35+1
H35: [W4] +G35+1
J35: [W4] +P34+1
K35: [W4] +J35+1
L35: [W4] +K35+1
M35: [W4] +L35+1
N35: [W4] +M35+1
O35: [W4] +N35+1
P35: [W4] +O35+1
A36: [W15] '!
B36: [W12] +H35+1
C36: [W12] +B36+1
D36: [W4] +C36+1
E36: [W4] +D36+1
F36: [W4] +E36+1
G36: [W4] +F36+1
H36: [W4] +G36+1
J36: [W4] +P35+1
K36: [W4] +J36+1
L36: [W4] +K36+1
M36: [W4] +L36+1
N36: [W4] +M36+1
O36: [W4] +N36+1
P36: [W4] +O36+1
A37: [W15] '!
B37: [W12] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+H36)<H36,"",@DAY(@INT($B$18-
          @DAY($B$18))+1+H36))
C37: [W12] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+B37)<B37,"",@IF(@VALUE(B37)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+B37)))
D37: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+C37)<C37,"",@IF(@VALUE(C37)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+C37)))
E37: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+D37)<D37,"",@IF(@VALUE(D37)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+D37)))
F37: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+E37)<E37,"",@IF(@VALUE(E37)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+E37)))
G37: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+F37)<F37,"",@IF(@VALUE(F37)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+F37)))
H37: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+G37)<G37,"",@IF(@VALUE(G37)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+G37)))
J37: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+P36)<P36,"",@DAY(@INT($B$19-
          @DAY($B$19))+1+P36))
K37: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+J37)<J37,"",@IF(@VALUE(J37)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+J37)))
L37: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+K37)<K37,"",@IF(@VALUE(K37)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+K37)))
M37: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+L37)<L37,"",@IF(@VALUE(L37)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+L37)))
N37: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+M37)<M37,"",@IF(@VALUE(M37)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+M37)))
O37: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+N37)<N37,"",@IF(@VALUE(N37)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+N37)))
P37: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+O37)<O37,"",@IF(@VALUE(O37)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+O37)))
A38: [W15] '!
B38: [W12] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+H37)<H37,"",@IF(@VALUE(H37)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+H37)))
C38: [W12] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+B38)<B38,"",@IF(@VALUE(B38)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+B38)))
D38: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+C38)<C38,"",@IF(@VALUE(C38)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+C38)))
E38: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+D38)<D38,"",@IF(@VALUE(D38)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+D38)))
```

```
F38: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+E38)<E38,"",@IF(@VALUE(E38)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+E38)))
G38: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+F38)<F38,"",@IF(@VALUE(F38)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+F38)))
H38: [W4] @IF(@DAY(@INT($B$18-@DAY($B$18))+1+G38)<G38,"",@IF(@VALUE(G38)=0
          ,"",@DAY(@INT($B$18-@DAY($B$18))+1+G38)))
J38: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+P37)<P37,"",@IF(@VALUE(P37)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+P37)))
K38: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+J38)<J38,"",@IF(@VALUE(J38)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+J38)))
L38: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+K38)<K38,"",@IF(@VALUE(K38)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+K38)))
M38: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+L38)<L38,"",@IF(@VALUE(L38)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+L38)))
N38: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+M38)<M38,"",@IF(@VALUE(M38)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+M38)))
O38: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+N38)<N38,"",@IF(@VALUE(N38)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+N38)))
P38: [W4] @IF(@DAY(@INT($B$19-@DAY($B$19))+1+O38)<O38,"",@IF(@VALUE(O38)=0
          ,"",@DAY(@INT($B$19-@DAY($B$19))+1+O38)))
A39: [W15] '!
A40: [W15] '!
D40: U [W4] @CHOOSE(@MONTH($B$20)-1,"January","February","March","April","May","June"
            ,"July","August","September","October","November","December")&" "&@STRING
            (@YEAR($B$20)+1900,0)
L40: U [W4] @CHOOSE(@MONTH($B$21)-1,"January","February","March","April","May","June"
            ,"July","August","September","October","November","December")&" "&@STRING
            (@YEAR($B$21)+1900,0)
A41: [W15] '!
B41: U [W12] \~
C41: U [W12] \~
D41: U [W4] \~
E41: U [W4] \~
F41: U [W4] \~
G41: U [W4] \~
H41: U [W4] \~
J41: U [W4] \~
K41: U [W4] \~
L41: U [W4] \~
M41: U [W4] \~
N41: U [W4] \~
O41: U [W4] \~
P41: U [W4] \~
A42: [W15] '!
B42: [W12] @IF(@IF(@MOD(@INT($B$20-@DAY($B$20)+1),7)=0,7,@MOD(@INT($B$20-
           @DAY($B$20)+1),7))=1,1,"")
C42: [W12] @IF(@IF(@MOD(@INT($B$20-@DAY($B$20)+1),7)=0,7,@MOD(@INT($B$20-
           @DAY($B$20)+1),7))=2,1,@IF(B42>0,B42+1,""))
D42: [W4] @IF(@IF(@MOD(@INT($B$20-@DAY($B$20)+1),7)=0,7,@MOD(@INT($B$20-
          @DAY($B$20)+1),7))=3,1,@IF(C42>0,C42+1,""))
E42: [W4] @IF(@IF(@MOD(@INT($B$20-@DAY($B$20)+1),7)=0,7,@MOD(@INT($B$20-
          @DAY($B$20)+1),7))=4,1,@IF(D42>0,D42+1,""))
F42: [W4] @IF(@IF(@MOD(@INT($B$20-@DAY($B$20)+1),7)=0,7,@MOD(@INT($B$20-
          @DAY($B$20)+1),7))=5,1,@IF(E42>0,E42+1,""))
G42: [W4] @IF(@IF(@MOD(@INT($B$20-@DAY($B$20)+1),7)=0,7,@MOD(@INT($B$20-
          @DAY($B$20)+1),7))=6,1,@IF(F42>0,F42+1,""))
H42: [W4] @IF(@IF(@MOD(@INT($B$20-@DAY($B$20)+1),7)=0,7,@MOD(@INT($B$20-
          @DAY($B$20)+1),7))=7,1,@IF(G42>0,G42+1,""))
J42: [W4] @IF(@IF(@MOD(@INT($B$21-@DAY($B$21)+1),7)=0,7,@MOD(@INT($B$21-
          @DAY($B$21)+1),7))=1,1,"")
K42: [W4] @IF(@IF(@MOD(@INT($B$21-@DAY($B$21)+1),7)=0,7,@MOD(@INT($B$21-
          @DAY($B$21)+1),7))=2,1,@IF(J42>0,J42+1,""))
L42: [W4] @IF(@IF(@MOD(@INT($B$21-@DAY($B$21)+1),7)=0,7,@MOD(@INT($B$21-
          @DAY($B$21)+1),7))=3,1,@IF(K42>0,K42+1,""))
M42: [W4] @IF(@IF(@MOD(@INT($B$21-@DAY($B$21)+1),7)=0,7,@MOD(@INT($B$21-
          @DAY($B$21)+1),7))=4,1,@IF(L42>0,L42+1,""))
N42: [W4] @IF(@IF(@MOD(@INT($B$21-@DAY($B$21)+1),7)=0,7,@MOD(@INT($B$21-
          @DAY($B$21)+1),7))=5,1,@IF(M42>0,M42+1,""))
O42: [W4] @IF(@IF(@MOD(@INT($B$21-@DAY($B$21)+1),7)=0,7,@MOD(@INT($B$21-
          @DAY($B$21)+1),7))=6,1,@IF(N42>0,N42+1,""))
P42: [W4] @IF(@IF(@MOD(@INT($B$21-@DAY($B$21)+1),7)=0,7,@MOD(@INT($B$21-
```

```
          @DAY($B$21)+1),7))=7,1,@IF(O42>0,O42+1,""))
A43: [W15] '!
B43: [W12] +H42+1
C43: [W12] +B43+1
D43: [W4] +C43+1
E43: [W4] +D43+1
F43: [W4] +E43+1
G43: [W4] +F43+1
H43: [W4] +G43+1
J43: [W4] +P42+1
K43: [W4] +J43+1
L43: [W4] +K43+1
M43: [W4] +L43+1
N43: [W4] +M43+1
O43: [W4] +N43+1
P43: [W4] +O43+1
A44: [W15] '!
B44: [W12] +H43+1
C44: [W12] +B44+1
D44: [W4] +C44+1
E44: [W4] +D44+1
F44: [W4] +E44+1
G44: [W4] +F44+1
H44: [W4] +G44+1
J44: [W4] +P43+1
K44: [W4] +J44+1
L44: [W4] +K44+1
M44: [W4] +L44+1
N44: [W4] +M44+1
O44: [W4] +N44+1
P44: [W4] +O44+1
A45: [W15] '!
B45: [W12] +H44+1
C45: [W12] +B45+1
D45: [W4] +C45+1
E45: [W4] +D45+1
F45: [W4] +E45+1
G45: [W4] +F45+1
H45: [W4] +G45+1
J45: [W4] +P44+1
K45: [W4] +J45+1
L45: [W4] +K45+1
M45: [W4] +L45+1
N45: [W4] +M45+1
O45: [W4] +N45+1
P45: [W4] +O45+1
A46: [W15] '!
B46: [W12] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+H45)<H45,"",@DAY(@INT($B$20-
           @DAY($B$20))+1+H45))
C46: [W12] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+B46)<B46,"",@IF(@VALUE(B46)=0
           ,"",@DAY(@INT($B$20-@DAY($B$20))+1+B46)))
D46: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+C46)<C46,"",@IF(@VALUE(C46)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+C46)))
E46: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+D46)<D46,"",@IF(@VALUE(D46)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+D46)))
F46: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+E46)<E46,"",@IF(@VALUE(E46)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+E46)))
G46: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+F46)<F46,"",@IF(@VALUE(F46)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+F46)))
H46: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+G46)<G46,"",@IF(@VALUE(G46)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+G46)))
J46: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+P45)<P45,"",@DAY(@INT($B$21
          -@DAY($B$21))+1+P45))
K46: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+J46)<J46,"",@IF(@VALUE(J46)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+J46)))
L46: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+K46)<K46,"",@IF(@VALUE(K46)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+K46)))
M46: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+L46)<L46,"",@IF(@VALUE(L46)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+L46)))
N46: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+M46)<M46,"",@IF(@VALUE(M46)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+M46)))
```

```
O46: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+N46)<N46,"",@IF(@VALUE(N46)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+N46)))
P46: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+O46)<O46,"",@IF(@VALUE(O46)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+O46)))
A47: [W15] '!
B47: [W12] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+H46)<H46,"",@IF(@VALUE(H46)=0
           ,"",@DAY(@INT($B$20-@DAY($B$20))+1+H46)))
C47: [W12] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+B47)<B47,"",@IF(@VALUE(B47)=0
           ,"",@DAY(@INT($B$20-@DAY($B$20))+1+B47)))
D47: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+C47)<C47,"",@IF(@VALUE(C47)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+C47)))
E47: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+D47)<D47,"",@IF(@VALUE(D47)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+D47)))
F47: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+E47)<E47,"",@IF(@VALUE(E47)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+E47)))
G47: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+F47)<F47,"",@IF(@VALUE(F47)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+F47)))
H47: [W4] @IF(@DAY(@INT($B$20-@DAY($B$20))+1+G47)<G47,"",@IF(@VALUE(G47)=0
          ,"",@DAY(@INT($B$20-@DAY($B$20))+1+G47)))
J47: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+P46)<P46,"",@IF(@VALUE(P46)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+P46)))
K47: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+J47)<J47,"",@IF(@VALUE(J47)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+J47)))
L47: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+K47)<K47,"",@IF(@VALUE(K47)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+K47)))
M47: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+L47)<L47,"",@IF(@VALUE(L47)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+L47)))
N47: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+M47)<M47,"",@IF(@VALUE(M47)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+M47)))
O47: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+N47)<N47,"",@IF(@VALUE(N47)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+N47)))
P47: [W4] @IF(@DAY(@INT($B$21-@DAY($B$21))+1+O47)<O47,"",@IF(@VALUE(O47)=0
          ,"",@DAY(@INT($B$21-@DAY($B$21))+1+O47)))
A48: [W15] '!
B48: U [W12] '_____
J48: U [W4] '_____
```

[4] Learn Macro

```
     A                B                C                D                E
 1 *---A macro to start LEARN mode and range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Expand and highlight the LEARN range and press [ENTER]
 6 *---Press [ALT-F5] and press [RETURN] to end LEARN
 7 !
 8          * * *     A LOTUS 2.2/2.3/2.4 MACRO ONLY    * * *
 9 !
10 \Z               {BREAKON}
11 LEARN22          {WINDOWSOFF}{PANELOFF}/RNCLearn range ?~~/RND
                    Learn range ?~/RNC{WINDOWSON}{PANELON}Learn range ?~
                    {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Learn range ?~
12 !                /WLRLearn range ?~
13 !                Press [ALT-F5] and press [RETURN] to start the LEARN
                    mode:{?}{ESC}~
14 !                {GOTO}Learn range ?~/RNDLearn range ?~
```

This macro uses of the LEARN mode available with Lotus 1-2-3 beginning version 2.2. and simplifies the process. The macro starts with the `{WINDOWSOFF}{PANELOFF}` commands which suppress the screen and panel display activity. Then the macro uses the "safe technique" to paint the range for the LEARN code, and simultaneously assigns the [Learn range ?] name to the same range, which acts as a prompt. When the macro finishes it issues `{GOTO}Learn range ?` that places the cell pointer on the top cell of the [Learn range ?]. Now the macro issues `/WLRLearn range ?~` that assigns the [Learn range ?] range for the LEARN code. Next the macro displays:

```
Press [ALT-F5] and press [RETURN] to start the LEARN mode:
```

in the panel to remind you that you have to press the `ALT-F5` key combination to enter the LEARN mode. Lotus types the prompt to the panel because it cannot find any macro command in the prompt text. To hold the prompt in the panel, the macro issues `{?}`. Press the `ALT-F5` key combination and press ENTER to start the LEARN mode. We could not use `{GET key}` macro command as we do in most of the macros in ***Super Power*** when we want to display a message in the panel, because you have to press the `ALT-F5` key combination and ENTER while `{GET key}` allows you to use only one key.

When you press ENTER the macro issues `{ESC}` to clear the panel before the prompt is written to the current cell. Last, the macro moves the cell pointer to the beginning of the LEARN range using `{GOTO}Learn range ?~` and issues `/RNDLearn range ?~` to clear the range name from the worksheet, which leaves the worksheet as clean as possible and as close to the condition it was before the macro started

[6] Document the Worksheet

```
     A                B                C                D                E
 1 *---A macro to DOCUMENT the spreadsheet
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6 !
 7 !
 8 !
 9 !
10 \Z             {BREAKON}
11 DOCSHEET       {MENUBRANCH menu1104}
12 !
13 menu1104       As displayed  Formulas  132 Width  240 Width  Quit
14 !              Print spreadshPrint spreSelect 132 Select 240 Quit the m
15 !              {WINDOWSOFF}{p{windowsof{LET widthp{LET widthp104,"240"}~
16 !              /PPCAA{WINDOWS/PPCAA{WIN{MENUBRANCH{MENUBRANCH menu1104}
17 !              {IF widthp104={IF widthp104="132"}/PPOS {ESC}\027@\015\0
18 !              {IF widthp104={IF widthp104="240"}/PPOS {ESC}\027@\015\0
19 !              H {ESC}@|As DiH {ESC}@|Cell Formulas|Page #~
20 !              OAQGPOS {ESC}\OCQGPOS {ESC}\027@~{ESC 6}
21 !              {MENUBRANCH me{MENUBRANCH menu1104}
22 !
23 widthp104      132
```

This macro allows you to document the worksheet on his printer. The macro can print the document in 132 characters line width on an 80 column printer and 240 characters line width on a wide printer. The macro uses a custom menu which cannot be displayed clearly on one page without overlapping. Therefore we show the code of each menu item separately:

```
As displayed
Print spreadsheet as displayed (choose 132 or 240 depends on
    printer)
{WINDOWSOFF}{PANELOFF}{CALC}
/PPCAA{WINDOWSON}{PANELON}R{HOME}.{END}{HOME}{?}~{WINDOWSOFF}
    {PANELOFF}{ESC 6}
{IF widthp104="132"}/PPOS {ESC}\027@\015\027\069\027\071~MR132~
{IF widthp104="240"}/PPOS {ESC}\027@\015\027\069\027\071~MR240~
H {ESC}@|As Displayed|Page #~
OAQGPOS {ESC}\027@~{ESC 6}
{MENUBRANCH menu1104}

Formulas
Print spreadsheet's cell formulas (choose 132 or 240 depends on
    printer)
{WINDOWSOFF}{PANELOFF}{CALC}
/PPCAA{WINDOWSON}{PANELON}R{HOME}.{END}{HOME}{?}~{WINDOWSOFF}
    {PANELOFF}{ESC 6}
{IF widthp104="132"}/PPOS {ESC}\027@\015\027\069\027\071~MR132~
{IF widthp104="240"}/PPOS {ESC}\027@\015\027\069\027\071~MR240~
H {ESC}@|Cell Formulas|Page #~
OCQGPOS {ESC}\027@~{ESC 6}
{MENUBRANCH menu1104}

132 Width
Select 132 characters per line (for 80 columns printer)
{LET widthp104,"132"}~
{MENUBRANCH menu1104}

240 Width
Select 240 characters per line (for wide carriage printer)
{LET widthp104,"240"}~
{MENUBRANCH menu1104}
```

```
Quit
Quit the macro
```

If you intend to key the code of this macro into Lotus 1-2-3, you have to key the code of the custom menu options as they appear here, NOT the partial code as it appears in the main listing of the macro. To further assist you, we bring the full cell contents list at the end of this section. The first menu option is [**A**s displayed]:

```
As displayed
Print spreadsheet as displayed (choose 132 or 240 depends on
    printer)
{WINDOWSOFF}{PANELOFF}{CALC}
/PPCAA{WINDOWSON}{PANELON}R{HOME}.{END}{HOME}{?}~{WINDOWSOFF}
    {PANELOFF}{ESC 6}
{IF widthp104="132"}/PPOS {ESC}\027@\015\027\069\027\071~MR132~
{IF widthp104="240"}/PPOS {ESC}\027@\015\027\069\027\071~MR240~
H {ESC}@|As Displayed|Page #~
OAQGPOS {ESC}\027@~{ESC 6}
{MENUBRANCH menu1104}
```

The macro issues `{WINDOWSOFF}{PANELOFF}`, which freeze screen and panel display activities. The macro issues `{CALC}`, which updates all the formulas in the worksheet. The macro continues with the `/PPCAA` macro keys which clear all the previous print settings and align the page. Then the macro issues `{WINDOWSON}{PANELON}` which resume screen and panel display activities and `R{HOME}.{END}{HOME}` to define the whole worksheet as the area to print. Next the macro issues `{?}`, which allows you to expand or shrink the range to print. When you finish defining the printing range and press ENTER, the macro issues the tilde "`~`", which is the same as ENTER, and issues `{WINDOWSOFF}{PANELOFF}` and the `{ESC 6}` to return to the READY mode.

Now the macro issues `{IF widthp104="132"}` to check if you are using a printer which can print 132 compressed characters to a line. If so, the macro issues `/PPOS {ESC}\027@ \ 015\027\069\027\071~` macro command which assigns a new set string for printing (this setup string is for Epson compatible printer, you need to replace it with other string if you use a different printer). Next the macro issues the `MR132~` macro command which sets the right margin to 132. If the condition was false the macro issues the `{IF widthp104="240"}` macro command to check if you use a printer which can print 132 compressed characters line. If so, the macro issues `/PPOS {ESC}\027@\015\027\069\027\071~` which assigns a new setup string for printing (the space between the `/PPOS` and `{ESC}` is important). Next the macro issues `MR240~` which sets the right margin to 240.

Now the macro issues the "`H`" macro key for headers and continues with a space character and then `{ESC}` to clear any previous header and continues with `@|As Displayed|Page #~` macro code. The "`@`" character instructs Lotus to print the date at the left side of the page. The `|As Displayed|` instructs Lotus to print the text between the two bars in the middle of the page, and the "`#`" character instructs Lotus to print the page number after the "`Page `" string on the right side of the page. Therefore, for example the header of the first printed page will looks like:

```
25-Jan-93                    As Displayed                    Page 1
```

Last the macro issues `OAQGPOS {ESC}\027@~{ESC 6}`, which print the range and reset the setup string back to normal (`\027@`) and returns to the READY mode. Next the macro issues `{MENUBRANCH menu1104}`, which restarts the custom menu again.

Note: Notice the SPACE character after /PPOS and before {ESC}. Typing the SPACE character assures that one {ESC} is enough to clear the previous setup string when an old setup string exists. However if a previous setup string does not exist, {ESC} will return Lotus to the previous menu. Typing any character as a temporary setup string insures that {ESC} will clear the setup string whether a previous one exists or not. We have used the same trick to assign the header and to reset the setup string back to normal.

The second menu option is [**F**ormulas]:

```
Formulas
Print spreadsheet's cell formulas (choose 132 or 240 depends on
    printer)
{WINDOWSOFF}{PANELOFF}{CALC}
/PPCAA{WINDOWSON}{PANELON}R{HOME}.{END}{HOME}{?}~{WINDOWSOFF}
    {PANELOFF}{ESC 6}
{IF widthp104="132"}/PPOS {ESC}\027@\015\027\069\027\071~MR132~
{IF widthp104="240"}/PPOS {ESC}\027@\015\027\069\027\071~MR240~
H {ESC}@|Cell Formulas|Page #~
OCQGPOS {ESC}\027@~{ESC 6}
{MENUBRANCH menu1104}
```

This menu code is almost identical to the previous one except that it prints cell formulas. The third menu option is [**1**32 Width]:

```
132 Width
Select 132 characters per line (for 80 columns printer)
{LET widthp104,"132"}~
{MENUBRANCH menu1104}
```

When you choose this option, the macro issues {LET widthp104,"132"}, which stores the 132 string in the cell named [widthp104] and then issues the {MENUBRANCH menu1104} macro command which restarts the custom menu again to allow you to continue with the printing job. The fourth menu option is [**2**40 Width]:

```
240 Width
Select 240 characters per line (for wide carriage printer)
{LET widthp104,"240"}~
{MENUBRANCH menu1104}
```

This menu option is the same as [132 Width] except that this time the macro stores the 240 string in cell [widthp104]. The last menu option is [**Q**uit]:

```
Quit
Quit the macro
```

When you choose this menu option, the macro reaches an empty cell and quits.

To help you key this macro into Lotus 1-2-3, we show the full list of all the cell contents.

```
A1: U [W14] '*---A macro to DOCUMENT the spreadsheet
A2: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W14] '    range names in this column (starts with the \Z macro name)
A4: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A5: [W14] '!
A6: [W14] '!
A7: [W14] '!
A8: [W14] '!
A9: [W14] '!
A10: U [W14] '\Z
```

```
B10: [W14] '{BREAKON}
A11: U [W14] 'DOCSHEET
B11: [W14] '{MENUBRANCH menu1104}
A12: [W14] '!
A13: [W14] 'menu1104
B13: [W14] 'As displayed
C13: [W15] 'Formulas
D13: [W15] '132 Width
E13: '240 Width
F13: 'Quit
A14: [W14] '!
B14: [W14] 'Print spreadsheet as displayed (choose 132 or 240 depends on
          printer)
C14: [W15] 'Print spreadsheet's cell formulas (choose 132 or 240 depends
          on printer)
D14: [W15] 'Select 132 characters per line (for 80 columns printer)
E14: 'Select 240 characters per line (for wide carriage printer)
F14: 'Quit the macro
A15: [W14] '!
B15: [W14] '{WINDOWSOFF}{PANELOFF}{CALC}
C15: [W15] '{WINDOWSOFF}{PANELOFF}{CALC}
D15: [W15] '{LET widthp104,"132"}~
E15: '{LET widthp104,"240"}~
A16: [W14] '!
B16: [W14] '/PPCAA{WINDOWSON}{PANELON}R{HOME}.{END}{HOME}{?}~{WINDOWSOFF}
          {PANELOFF}{ESC 6}
C16: [W15] '/PPCAA{WINDOWSON}{PANELON}R{HOME}.{END}{HOME}{?}~{WINDOWSOFF}
          {PANELOFF}{ESC 6}
D16: [W15] '{MENUBRANCH menu1104}
E16: '{MENUBRANCH menu1104}
A17: [W14] '!
B17: [W14] '{IF widthp104="132"}/PPOS {ESC}\027@\015\027\069\027\071~MR132~
C17: [W15] '{IF widthp104="132"}/PPOS {ESC}\027@\015\027\069\027\071~MR132~
A18: [W14] '!
B18: [W14] '{IF widthp104="240"}/PPOS {ESC}\027@\015\027\069\027\071~MR240~
C18: [W15] '{IF widthp104="240"}/PPOS {ESC}\027@\015\027\069\027\071~MR240~
A19: [W14] '!
B19: [W14] 'h {ESC}@|As Displayed|Page #~
C19: [W15] 'h {ESC}@|Cell Formulas|Page #~
A20: [W14] '!
B20: [W14] 'OAQGPOS {ESC}\027@~{ESC 6}
C20: [W15] 'OCQGPOS {ESC}\027@~{ESC 6}
A21: [W14] '!
B21: [W14] '{MENUBRANCH menu1104}
C21: [W15] '{MENUBRANCH menu1104}
A22: [W14] '!
A23: [W14] 'widthp104
B23: [W14] '132
```

[0] View Data in the Worksheet

	A	B	C	D	E
1	*---A macro to let you move the pointer across the worksheet to view				
2	your data and then come back to the current cell				
3	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
4	range names in this column (starts with the \Z macro name)				
5	*---Hold the [ALT] key and press [Z] to start the macro				
6	*---Move around the worksheet to watch your data, when you are				
7	finished press the [ESC] key TWICE				
8	!				
9	!				
10	\Z	{BREAKON}			
11	DATAVIEW	+			

This one character macro is not a joke. It is simple but powerful, and allows you to walk all over the worksheet to look for data and then press {ESC} TWICE to return to the beginning. When you start the macro, Lotus thinks that you want to write a formula; instead you can use the direction keys to move the pointer cell to any place in the worksheet and back easily.

[1] Use the Numeric Key Pad to Insert Data Along a Column

```
     A          B          C          D          E
 1 *---Use the numeric key pad to enter data in a column.
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 *---Press the NUMLOCK key then place the cell pointer at the first
 6     cell and start data entry, every time you press the ENTER key the
 7     cell pointer moves down one cell. Use [Ctrl Break] to quit.
 8 !
 9 !
10 \Z              {BREAKON}
11 NUMLOCKD        {BREAKON}
12 loop240         {?}~{DOWN}
13 !               {BRANCH loop240}
```

This macro allows you to use the numeric key pad and ENTER to rapidly insert data along a column. The `{?}` halts the macro execution and allows you to use the numeric keys. When you press ENTER, the macro issues the tilde "`~`" to write the content of the panel into the cell. Next the macro issues `{DOWN}` to move the cell pointer to the next cell in the column. Finally the macro issues `{BRANCH loop240}` to loop back to B12, [loop240], and wait for the next entry.

[1] Use the Numeric Pad to Insert Data Across a Row

	A	B	C	D	E
1	*---Use the numeric key pad to enter data in a row.				
2	*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the				
3	range names in this column (starts with the \Z macro name)				
4	*---Hold the [ALT] key and press [Z] to activate the macro				
5	*---Press the NUMLOCK key then place the cell pointer at the first				
6	cell and start data entry, every time you press the ENTER key the				
7	cell pointer moves right one cell. Use [Ctrl Break] to quit.				
8	!				
9	!				
10	\Z	{BREAKON}			
11	NUMLOCKR	{BREAKON}			
12	loop241	{?}~{RIGHT}			
13	!	{BRANCH loop241}			

This macro allows you to use the numeric key pad and ENTER to rapidly insert data across a row. The `{?}` halts the macro execution and allows you to use the numeric keys. When you press ENTER, the macro issues the tilde "`~`" to write the content of the panel into the cell. Next the macro issues `{RIGHT}` to move the cell pointer to the next cell in the row. Finally the macro issues `{BRANCH loop241}` to loop back to B12, [loop241], and wait for the next entry.

[6] Labels Entry Macro

```
     A                B                C                D                E
 1 *---A macro to always accept data as a label. If you type a number the
 2     macro automatically adds "'" to change it into a label.
 3 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 4     range names in this column (starts with the \Z macro name)
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 *---The Next_sheet and Previous_sheet are only for RELEASE 3 and up
 7 !
 8         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 9                    IT WILL WORK IN LOTUS 2.0 AND UP
10 \Z             {BREAKON}
11 LABLMCRO       {LET rel125,@INFO("release")}~
12 !              {MENUBRANCH menu1125}{BRANCH lablmcro}
13 !
14 menu1125       Row      Column     Next_sheet  Previous_sheet  Quit
15 !              Insert daInsert dataInsert data Insert data as lQuit the
16 !              {rowa125}{columna125{IF @LEFT(RE{IF @LEFT(rel125{BRANCH r
17 !                                  {MENUBRANCH {MENUBRANCH menu1125}
18 !
19 rowa125        Insert data and press [RETURN] or [ESC] to quit:
                  {GET key1125}{ESC}{IF key1125="{ESC}"}{ESC 6}{RETURN}
                  {PANELON}{MENUBRANCH menu1125}
20 !              {IF key1125="~"}{RIGHT}{RETURN}
21 !              {IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125=
                  "{UP}"#OR#key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#
                  key1125="{PGUP}"}{key1125}{BRANCH rowa125}
22 !              {IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}{PANELOFF}
                  /RE~~{PANELON}{EDIT}'
23 !              {key1125}{?}~{RIGHT}{BRANCH rowa125}
24 !
25 !
26 columna125     Insert data and press [RETURN] or [ESC] to quit:
                  {GET key1125}{ESC}{IF key1125="{ESC}"}{ESC 6}{RETURN}
                  {PANELON}{MENUBRANCH menu1125}
27 !              {IF key1125="~"}{DOWN}{RETURN}
28 !              {IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125=
                  "{UP}"#OR#key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#
                  key1125="{PGUP}"}{key1125}{BRANCH columna125}
29 !              {IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}{PANELOFF}
                  /RE~~{PANELON}{EDIT}'
30 !              {key1125}{?}~{DOWN}{BRANCH columna125}
31 !
32 !
33 next125        Insert data and press [RETURN] or [ESC] to quit:
                  {GET key1125}{ESC}{IF key1125="{ESC}"}{ESC 6}{RETURN}
                  {PANELON}{MENUBRANCH menu1125}
34 !              {IF key1125="~"}{NS}{RETURN}
35 !              {IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125=
                  "{UP}"#OR#key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#
                  key1125="{PGUP}"#OR#key1125="{NS}"}{key1125}
                  {BRANCH next125}
36 !              {IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}{PANELOFF}
                  /RE~~{PANELON}{EDIT}'
37 !              {key1125}{?}~{NS}{BRANCH next125}
38 !
39 !
40 previous125    Insert data and press [RETURN] or [ESC] to quit:
                  {GET key1125}{ESC}{IF key1125="{ESC}"}{ESC 6}{RETURN}
                  {PANELON}{MENUBRANCH menu1125}
41 !              {IF key1125="~"}{PS}{RETURN}
42 !              {IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125=
                  "{UP}"#OR#key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#
                  key1125="{PGUP}"#OR#ket1125="{PS}"}{key1125}
                  {BRANCH previous125}
43 !              {IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}{PANELOFF}
                  /RE~~{PANELON}{EDIT}'
44 !              {key1125}{?}~{PS}{BRANCH previous125}
```

```
45 !
46 key1125        {ESC}
47 !
48 ret125
49 !
50 rel125
```

This macro allows you to enter all the data you write into Lotus 1-2-3 as labels even if you write numbers. The macro automatically senses that you inserted a number and precedes the number with an apostrophe "'". When you start the macro, it displays a custom menu which allows you to insert the data along a column, across a row, or through the sheets of the worksheet with a 3-D release. Because the custom menu cannot be displayed on the screen or the paper without overlapping, we show every menu option separately. If you intend to key this macro into 1-2-3, you have to key the code as it appears here, NOT the code as it appears in the main listing. To further assist you, we show a full list of all the cell contents and formulas at the end of this section.

```
Row
Insert data as labels across a row
{rowa125}{MENUBRANCH menu1125}

Column
Insert data as labels across a column
{columna125}{MENUBRANCH menu1125}

Next_sheet
Insert data as labels across the next sheets (RELEASE 3 ONLY)
{IF @LEFT(rel125,1)<>"@"}{next125}
{MENUBRANCH menu1125}

Previous_sheet
Insert data as labels across the previous sheets (RELEASE 3 ONLY)
{IF @LEFT(rel125,1)<>"@"}{previous125}
{MENUBRANCH menu1125}

Quit
Quit the macro
{BRANCH ret125}
```

The macro starts with the `{LET rel125,@INFO("release")}~` macro commands which store the result of the `@INFO("release")` function in the B50 cell, [rel125]. Later, the macro uses it to decide if you are using a 2-D or a 3-D Lotus release. Next the macro issues `{MENUBRANCH menu1125}` which activates the [menu1125] custom menu. The first menu option is [**R**ow]:

```
Row
Insert data as labels across a row
{rowa125}{MENUBRANCH menu1125}
```

This allows you to insert the labels across a row. The macro issues `{rowa125}`, which starts the [rowa125] routine:

```
     A          B                C          D          E
19 rowa125      Insert data and press [RETURN] or [ESC] to quit:
                {GET key1125}{ESC}{IF key1125="{ESC}"}{ESC 6}{RETURN}
                {PANELON}{MENUBRANCH menu1125}
20 !            {IF key1125="~"}{RIGHT}{RETURN}
21 !            {IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125=
                "{UP}"#OR#key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#
                key1125="{PGUP}"}{key1125}{BRANCH rowa125}
22 !            {IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}{PANELOFF}
                /RE~~{PANELON}{EDIT}'
```

```
23 !             {key1125}{?}~{RIGHT}{BRANCH rowa125}
```

This routine writes the "Insert data and press [RETURN] or [ESC] to quit:" text into the panel as a prompt message and then issues {GET key1125}, which halts macro execution to allow you to read the message in the panel. When you press a key, the macro stores the key code in cell [key1125] and then issues {ESC}, which clears the message from the panel before Lotus writes it to the current cell. Now the macro starts a series of {IF} commands, which monitor the keys that you use. First, the macro issues {IF key1125= "{ESC}"} to check if you pressed ESC. If so, the macro issues {ESC 6} to return to the READY mode, and then issues {RETURN}{PANELON}{MENUBRANCH menu1125} to end the routine, resume panel display activity, and re-activate the [menu1125] custom menu

Next the macro issues {IF key1125="~"} to check if you pressed ENTER. If so, the macro issues {RIGHT}, which moves the cell pointer to the next cell in the row. The next condition is:

```
{IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125="{UP}"#OR#
    key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#key1125="{PGUP}"}
```

It checks if of you pressed one of the direction keys RIGHT, LEFT, UP, DOWN, PGDN and PGUP. If so, the macro issues {key1125} which activates the [key1125] routine. Cell [key1125] holds the code of the key that you pressed. Therefore the {key1125} routine command executes the key that you pressed. If you pressed DOWN, the macro first store {DOWN} in cell [key1125] and then issues the {key1125}, which executes the DOWN key. Last, the macro issues {BRANCH rowa125} which routes macro control back to the beginning of the routine.

The next is {IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}, which checks if you pressed a number key. The macro uses the @CODE function to calculate the ASCII code of the key that you pressed. If the code is greater than 47 or smaller than 58, it means that you pressed a number. If so, the macro issues {PANELOFF} to freeze panel display activity and /RE~~ to erase the current cell content. Then the macro issues {PANELON}{EDIT} to allow panel activity and enter the EDIT mode. Next the macro writes the apostrophe "'" into the panel.

Now the macro issues {key1125}, which executes the [key1125] routine. However the [key1125] routine contains the number that you pressed; therefore the {key1125} routine command writes the number after the apostrophe "'". Last the macro issues the tilde "~" to write the content of the panel into the current cell. For example: if you pressed the number 5, the macro stores it in the cell named [key1125]. Because it is a number, the macro types the apostrophe "'" into the panel and then the {key1125} routine command writes the number 5 into the panel after the apostrophe "'". The result is always a label with the apostrophe "'" prefix. Now the macro issues {?} to allow you to continue write to the panel.

When you press ENTER, the macro issues {RIGHT}, which writes the content of the panel into the current cell and moves the cell pointer to the next cell in the row, and then issues {BRANCH rowa125}, which starts the routine from the beginning, so you can type new data in the new cell. When you press {ESC} to quit the routine, the macro returns control to the {MENUBRANCH menu1125} command, which follows the {key1125} routine command, and re-activates the custom menu. The second menu option is [**C**olumn]:

```
Column
Insert data as labels across a column
```

```
{columna125}{MENUBRANCH menu1125}
```

The macro issues the {columna125} routine command, which starts the [columna125] routine. This routine is identical to the [rowa125] routine except that each time you press ENTER the macro issues {DOWN} to move the cell pointer one cell down to the next cell in the column. The third menu option is [**N**ext_sheet]:

```
Next_sheet
Insert data as labels across the next sheets (RELEASE 3 ONLY)
{IF @LEFT(rel125,1)<>"@"}{next125}
{MENUBRANCH menu1125}
```

This menu option is valid only for the 3-D spreadsheet, which can hold more than one sheet. When you choose this menu option, the macro first issues {IF @LEFT(rel125,1)<>"@"}, which checks if the first character of the content of the cell named [rel125] is not equal to the "@" character. If so, you are using a 3-D Lotus release. Therefore the macro issues {next125} which activates the [next125] routine. This routine is identical to the [rowa125] routine except that each time you press ENTER, the macro issues {NS} to move the cell pointer to the next sheet. The next menu option is [**P**revious_sheet]:

```
Previous_sheet
Insert data as labels across the previous sheets (RELEASE 3 ONLY)
{IF @LEFT(rel125,1)<>"@"}{previous125}
{MENUBRANCH menu1125}
```

This menu option is identical to the [Next_sheet] menu option except that each time you press ENTER, the macro issues {PS} to move the cell pointer to the pervious sheet (out of the screen).

The last menu option is [**Q**uit]:

```
Quit
Quit the macro
{BRANCH ret125}
```

The macro issues {BRANCH ret125} which routes macro execution control to the empty routine named [ret125] and quits.

The following is a full list of all the cell contents and formulas to help you key this macro code into Lotus 1-2-3.

```
A1: U [W14] '*---A macro to always accept data as a label. If you type a
          number the
A2: U [W14] '    macro automatically adds "'" to change it into a label.
A3: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A4: [W14] '    range names in this column (starts with the \Z macro name)
A5: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A6: [W14] '*---The Next_sheet and Previous_sheet are only for RELEASE 3
          and up
A7: [W14] '!
A8: U [W14] '       THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
A9: U [W14] '                    IT WILL WORK IN LOTUS 2.0 AND UP
A10: U [W14] '\Z
B10: [W9] '{BREAKON}
A11: U [W14] 'LABLMCRO
B11: [W9] '{LET rel125,@INFO("release")}~
A12: [W14] '!
B12: [W9] '{MENUBRANCH menu1125}{BRANCH lablmcro}
A13: [W14] '!
```

```
A14: [W14] 'menu1125
B14: [W9] 'Row
C14: [W11] 'Column
D14: 'Next_sheet
E14: [W10] 'Previous_sheet
F14: 'Quit
A15: [W14] '!
B15: [W9] 'Insert data as labels across a row
C15: [W11] 'Insert data as labels across a column
D15: 'Insert data as labels across the next sheets (RELEASE 3 ONLY)
E15: [W10] 'Insert data as labels across the previous sheets (RELEASE 3 ONLY)
F15: 'Quit the macro
A16: [W14] '!
B16: [W9] '{rowa125}{MENUBRANCH menu1125}
C16: [W11] '{columna125}{MENUBRANCH menu1125}
D16: '{IF @LEFT(rel125,1)<>"@"}{next125}
E16: [W10] '{IF @LEFT(rel125,1)<>"@"}{previous125}
F16: '{BRANCH ret125}
A17: [W14] '!
D17: '{MENUBRANCH menu1125}
E17: [W10] '{MENUBRANCH menu1125}
A18: [W14] '!
A19: [W14] 'rowa125
B19: [W9] 'Insert data and press [RETURN] or [ESC] to quit: {GET key1125}
     {ESC}{IF key1125="{ESC}"}{ESC 6}{RETURN}{PANELON}
     {MENUBRANCH menu1125}
A20: [W14] '!
B20: [W9] '{IF key1125="~"}{RIGHT}{RETURN}
A21: [W14] '!
B21: [W9] '{IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125="{UP}"#OR#
     key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#key1125="{PGUP}"}{key1125}
     {BRANCH rowa125}
A22: [W14] '!
B22: [W9] '{IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}{PANELOFF}/RE~~
     {PANELON}{EDIT}'
A23: [W14] '!
B23: [W9] '{key1125}{?}~{RIGHT}{BRANCH rowa125}
A24: [W14] '!
A25: [W14] '!
A26: [W14] 'columna125
B26: [W9] 'Insert data and press [RETURN] or [ESC] to quit: {GET key1125}
     {ESC}{IF key1125="{ESC}"}{ESC 6}{RETURN}{PANELON}
     {MENUBRANCH menu1125}
A27: [W14] '!
B27: [W9] '{IF key1125="~"}{DOWN}{RETURN}
A28: [W14] '!
B28: [W9] '{IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125="{UP}"#OR#
     key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#key1125="{PGUP}"}{key1125}
     {BRANCH columna125}
A29: [W14] '!
B29: [W9] '{IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}{PANELOFF}/RE~~
     {PANELON}{EDIT}'
A30: [W14] '!
B30: [W9] '{key1125}{?}~{DOWN}{BRANCH columna125}
A31: [W14] '!
A32: [W14] '!
A33: [W14] 'next125
B33: [W9] 'Insert data and press [RETURN] or [ESC] to quit: {GET key1125}
     {ESC}{IF key1125="{ESC}"}{ESC 6}{RETURN}{PANELON}
     {MENUBRANCH menu1125}
A34: [W14] '!
B34: [W9] '{IF key1125="~"}{NS}{RETURN}
A35: [W14] '!
B35: [W9] '{IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125="{UP}"#OR#
     key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#key1125="{PGUP}"#OR#
     key1125="{NS}"}{key1125}{BRANCH next125}
A36: [W14] '!
B36: [W9] '{IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}{PANELOFF}/RE~~
     {PANELON}{EDIT}'
A37: [W14] '!
B37: [W9] '{key1125}{?}~{NS}{BRANCH next125}
```

```
A38: [W14] '!
A39: [W14] '!
A40: [W14] 'previous125
B40: [W9] 'Insert data and press [RETURN] or [ESC] to quit: {GET key1125}
          {ESC}{IF key1125="{ESC}"}{ESC 6}{RETURN}{PANELON}
          {MENUBRANCH menu1125}
A41: [W14] '!
B41: [W9] '{IF key1125="~"}{PS}{RETURN}
A42: [W14] '!
B42: [W9] '{IF key1125="{RIGHT}"#OR#key1125="{LEFT}"#OR#key1125="{UP}"#OR#
          key1125="{DOWN}"#OR#key1125="{PGDN}"#OR#key1125="{PGUP}"#OR#
          ket1125="{PS}"}{key1125}{BRANCH previous125}
A43: [W14] '!
B43: [W9] '{IF @CODE(key1125)>=48#AND#@CODE(key1125)<=57}{PANELOFF}/RE~~
          {PANELON}{EDIT}'
A44: [W14] '!
B44: [W9] '{key1125}{?}~{PS}{BRANCH previous125}
A45: [W14] '!
A46: [W14] 'key1125
B46: [W9] '{ESC}
A47: [W14] '!
A48: [W14] 'ret125
A49: [W14] '!
A50: [W14] 'rel125
```

[6] Combine Two Range in the Same Worksheet

```
     A              B          C          D          E          F          G
 1 *---A macro to COMBINE two ranges inside the worksheet. This macro
 2     works exactly as file extract and combined together. This macro uses
 3     the disk, therefore the disk should not be write protected.
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Place the cell pointer on the upper left cell of the range to
 7     changed.
 8 *---Hold the [ALT] key and press [Z] to activate the macro
 9                      IT WILL WORK IN LOTUS 2.0 AND UP
10 \Z             {BREAKON}
11 RANGCOMB       {LET rel028,@INFO("release")}~{RECALC loc028}
                  {RECALC form028}
12 form028        {WINDOWSOFF}{PANELOFF}{LET here028,@CELLPOINTER("coord")}~
                  /FXFtemp#1~~R{ESC}/FEWtemp#1~Y
13 !              {WINDOWSOFF}{PANELOFF}/RNCCombined range?~~/RND
                  Combined range?~/RNC{PANELON}Combined range?~{BS}{BS}
                  {WINDOWSON}{?}~{WINDOWSOFF}{GOTO}{here028}~
14 !              {MENUCALL menu1028}
15 !
16 menu1028       Formulas  Values
17 !              Combine raCombine range's current formula values only
18 !              /FXFtemp#1/fxVtemp#1~Combined range?~R{ESC}
19 !              {menubranc{MENUBRANCH menu2028}
20 !              {menubranc{MENUBRANCH menu1028}
21 !
22 menu2028       Copy      Add      Subtract
23 !              Replace ceAdd valueSubtract values from combined range
24 !              {GOTO}{her{GOTO}{he{GOTO}{here028}~/FCSEtemp#1~
25 !              /FEWtemp#1/FEWTEMP#/FEWtemp#1~Y/RNDCombined range?~{WIN
26 !
27 here028        $BA:$A$1
28 !
29 rel028         3.00.00
30 !
31 loc028         coord
```

Lotus allows you to combine an outside file or range into the current worksheet, but does not allow you to combine two ranges in the worksheet, itself. This macro does! It works by extracting and combining them. It uses the disk, so the disk should not be write protected. When you start the macro, it displays a custom menu similar to the standard Lotus menu, which allows you to combine the data as formulas. The macro contains a second custom menu which allows you to copy, add or subtract the combined range. Because the custom menus cannot be displayed on the screen without overlapping, we show every menu option separately. If you intend to key this macro into Lotus 1-2-3, you have to key the code as it appears here, NOT the code as it appears in the main listing. You will find a full list of all the cell contents and formulas at the end of this section. The first custom menu contains the following two menu options:

```
Formulas
Combine range formulas
/FXFtemp#1~Combined range?~R{ESC}
{MENUBRANCH menu2028}
{MENUBRANCH menu1028}

Values
Combine range's current formula values only
/FXVtemp#1~Combined range?~R{ESC}
{MENUBRANCH menu2028}
{MENUBRANCH menu1028}
```

The second custom menu contains the following three menu options:

```
Copy
Replace cells in current range with cells from combined range
{GOTO}{here028}~/FCCEtemp#1~
/FEWtemp#1~Y/RNDCombined range?~{WINDOWSON}{WINDOWSOFF}

Add
Add values from combined range to values in current range
{GOTO}{here028}~/FCAEtemp#1~
/FEWtemp#1~Y/RNDCombined range?~{WINDOWSON}{WINDOWSOFF}

Subtract
Subtract values from combined range from values in current range
{GOTO}{here028}~/FCSEtemp#1~
/FEWtemp#1~Y/RNDCombined range?~{WINDOWSON}{WINDOWSOFF}
```

This macro contains two dynamic string formulas which create the correct macro code depending on whether you are using a 2-D or a 3-D Lotus release. If you intend to key this macro code into Lotus 1-2-3, you need to key the following two formulas in the B12 and the B31 cell, NOT the code as it appears in the main listing, which is the result of these formulas.

```
12 form028        +"{WINDOWSOFF}{PANELOFF}{LET here028,@CELLPOINTER
                  (""""&B31&""")}~/FXFtemp#1~~R{ESC}/FEWtemp#1~Y"
31 loc028         @IF(@LEFT(B29,1)<>"@","coord","address")
```

The macro starts with `{LET rel028,@INFO("release")}~` to store the result of the `@INFO("release")` function in cell [rel028]. Later, the macro uses this cell to decide if you are using a 2-D or a 3-D Lotus release. Next, the macro issues `{RECALC loc028} {RECALC form028}`, which update the dynamic formulas in cells [loc028] and [form028]. Now the macro issues `{WINDOWSOFF}{PANELOFF}`, which freeze screen and panel display activity. Next the macro issues `{LET here028,@CELLPOINTER("coord")}~` to store the cell pointer location in cell [here028].

The code uses the `@CELLPOINTER("coord")` function, but the "`coord`" attribute exists only in a 3-D Lotus release. The B12 cell, [form028], contains a string formula, which produces the correct code for the Lotus release you are using. Therefore, the code as it appears here means that the macro was used in a 3-D release when it was copied to ***Super Power***. Let's see how the formula produces the code.

```
12 form028        +"{WINDOWSOFF}{PANELOFF}{LET here028,@CELLPOINTER
                  (""""&B31&""")}~/FXFtemp#1~~R{ESC}/FEWtemp#1~Y"
```

When we look at the formula in B12, we can see that the "dynamic" part is the content of B31, [loc028]; however [loc028], itself, contains a formula:

```
31 loc028         @IF(@LEFT(B29,1)<>"@","coord","address")
```

This cell contains the result of the `@INFO("release")` function. The formula in B31, [loc028], checks the first character of the content of [rel028]. If it is an "`@`", you are using a 3-D Lotus release and the formula returns the "`coord`" string, otherwise you are using a 2-D Lotus release and the formula returns the "`address`" string. This result is fed into the formula in B12, [form028], to create the correct code for the respective Lotus release. Now we can continue to look at the code in B12.

The macro issues `/FXFtemp#1~~` to extract the current cell and name the file as "TEMP#1.WK1". Then issues the "`R`" for **R**eplace just in case such a file already exists. If such a

name does not exist, Lotus writes the "R" character to the panel, and the macro issues {ESC} to clear the panel. Now, the macro issues /FEWtemp#1~Y to erase the TEMP#1.WK1 file. The whole process was to ensure that there is no file with the TEMP#1.WK1 name before the macro continues, because Lotus does not have direct means of verifying if a file exists. The macro first creates such a file to make sure that it exists and then erases it.

	A	B	C	D	E	F	G
13	!	{WINDOWSOFF}{PANELOFF}/RNCCombined range?~~/RND					
		Combined range?~/RNC{PANELON}Combined range?~{BS}{BS}					
		{WINDOWSON}{?}~{WINDOWSOFF}{GOTO}{here028}~					
14	!	{MENUCALL menu1028}					

Now the macro uses the "safe technique" to prompt you to paint the range to combine and simultaneously assigns the meaningful [Combined range?] name to the same range which appears as a message prompt in the panel. Next the macro issues the {GOTO}{here028}~ indirect macro command to send the cell pointer back to its point of origin, and issues {MENUCALL menu1028} which starts the [menu1028] custom menu. The first menu item in the [menu1028] custom menu is [**F**ormulas]:

```
Formulas
Combine range formulas
/FXFtemp#1~Combined range?~R{ESC}
{MENUBRANCH menu2028}
{MENUBRANCH menu1028}
```

The macro issues /FXFtemp#1~Combined range?~ to extract the [Combined range?] to the disk as formulas and name it "TEMP#1.WK1". Again the macro issues R{ESC} to account for both cases when the file name is new, or already exists. Next, the macro issues {MENUBRANCH menu2028} which starts the second custom menu named [menu2028]. The second menu option is [**V**alues]. The code for this option is similar to the previous one.

Note: The second {MENUBRANCH menu1028} is to assure that if you press ESC when the [menu2028] custom menu is active, the macro returns to the [menu1028] custom menu as it works when you use the standard Lotus menus.

The first menu option of the [menu2028] custom menu is [**C**opy]:

```
Copy
Replace cells in current range with cells from combined range
{GOTO}{here028}~/FCCEtemp#1~
/FEWtemp#1~Y/RNDCombined range?~{WINDOWSON}{WINDOWSOFF}
```

The macro issues {GOTO}{here028}~ which sends the cell pointer back to its point of origin (the cell where the cell pointer was before the macro started). Then the macro issues /FCCEtemp#1~ which combines the entire TEMP#1.WK1 file. Next the macro issues /FEWtemp#1~Y which erases the TEMP#1.WK1 file from the disk, and last /RNDCombined range?~ which deletes the [Combined range?~] temporary range name. The other two menu options are basically the same; therefore we are not going to study them. The {WINDOWSON} {WINDOWSOFF} macro commands refresh the screen display to show the new changes to the cell and then freeze it again.

The following is a full list of all the cell contents and formulas to help you key this macro code into Lotus 1-2-3.

```
A1: U '*---A macro to COMBINE two ranges inside the worksheet. This macro
A2: U '    works exactly as file extract and combined together. This macro
           uses
A3: U '    the disk, therefore the disk should not be write protected.
A4: '*---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
A5: '    range names in this column (starts with the \Z macro name)
A6: '*---Place the cell pointer on the upper left cell of the range to
A7: '    changed.
A8: '*---Hold the [ALT] key and press [Z] to activate the macro
A9: U '                    IT WILL WORK IN LOTUS 2.0 AND UP
A10: U '\Z
B10: '{BREAKON}
A11: U 'RANGCOMB
B11: '{LET rel028,@INFO("release")}~{RECALC loc028}{RECALC form028}
A12: 'form028
B12: U +"{WINDOWSOFF}{PANELOFF}{LET here028,@CELLPOINTER(""""&B31&""")}~
       /FXFtemp#1~~R{ESC}/FEWtemp#1~Y"
A13: '!
B13: '{WINDOWSOFF}{PANELOFF}/RNCCombined range?~~/RNDCombined range?~
     /RNC{PANELON}Combined range?~{BS}{BS}{WINDOWSON}{?}~{WINDOWSOFF}
     {GOTO}{here028}~
A14: '!
B14: '{MENUcall menu1028}
A15: '!
A16: 'menu1028
B16: 'Formulas
C16: 'Values
A17: '!
B17: 'Combine range formulas
C17: 'Combine range's current formula values only
A18: '!
B18: '/FXFtemp#1~Combined range?~R{ESC}
C18: '/fXVtemp#1~Combined range?~R{ESC}
A19: '!
B19: '{MENUBRANCH menu2028}
C19: '{MENUBRANCH menu2028}
A20: '!
B20: '{MENUBRANCH menu1028}
C20: '{MENUBRANCH menu1028}
A21: '!
A22: 'menu2028
B22: 'Copy
C22: 'Add
D22: 'Subtract
A23: '!
B23: 'Replace cells in current range with cells from combined range
C23: 'Add values from combined range to values in current range
D23: 'Subtract values from combined range from values in current range
A24: '!
B24: '{GOTO}{here028}~/FCCEtemp#1~
C24: '{GOTO}{here028}~/FCAEtemp#1~
D24: '{GOTO}{here028}~/FCSEtemp#1~
A25: '!
B25: '/FEWtemp#1~Y/RNDCombined range?~{WINDOWSON}{WINDOWSOFF}
C25: '/FEWtemp#1~Y/RNDCombined range?~{WINDOWSON}{WINDOWSOFF}
D25: '/FEWtemp#1~Y/RNDCombined range?~{WINDOWSON}{WINDOWSOFF}
A26: '!
A27: 'here028
B27: '$BA:$A$1
A28: '!
A29: 'rel028
B29: '3.00.00
A30: '!
A31: 'loc028
B31: U @IF(@LEFT(B29,1)<>"@","coord","address")
```

[5] Runkey Macro for Lotus 2.0/2.01

```
     A              B                C                D              E
 1 *---A macro to ACTIVATE any macro/routine without using the [ALT] key,
 2     all range names are displayed on screen and by pointing and pressing
 3     the [ENTER] key the macro is activated (like RUNKEY in symphony)
 4 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 5     range names in this column (starts with the \Z macro name)
 6 *---Hold the [ALT] key and press [Z] to activate the macro
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                      IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z               {BREAKON}
11 RUNKEY2          {LET rel158,@INFO("release")}~{RECALC loc158}
                    {RECALC form158}{RECALC hhh1158}
12 form158          +"{err1158}{LET back1158,@CELLPOINTER("coord")}~
                    {WINDOWSOFF}{PANELOFF}{GOTO}{NAME}{WINDOWSON}{PANELON}
                    {NAME}{GET key158}{IF key158=""{ESC}""}{ESC 8}
                    {BRANCH key158}"
13 !                {IF key158="~"}{BRANCH hhh1158}
14 !                {key158}{?}~{BRANCH hhh1158}
15 !
16 hhh1158          ~{LET routin158,@CELLPOINTER("coord")}~{back158}
                    {DISPATCH routin158}~
17 !
18 err1158          {ONERROR err2158}
19 !
20 err2158          {BEEP}INVALID MACRO NAME! PRESS ANY KEY TO CONTINUE ...
                    {GET key158}{ESC}{BRANCH \Z}
21 !
22 routin158        $A:$B$11
23 !
24 back158          {GOTO}
25 back1158         $A:$A$1
26 !                ~
27 !
28 key158           ~
29 !
30 rel158
31 !
32 loc158           coord
```

Notice that the code in B12, B16 and B32 of the macro the result of the following dynamic string formulas. If you intend to key this macro into 1-2-3, you have to key the formulas, NOT the code shown in the main listing.

```
12 form158          +"{err1158}{LET back1158,@CELLPOINTER("""&B32&""")}~
                    {WINDOWSOFF}{PANELOFF}{GOTO}{NAME}{WINDOWSON}{PANELON}
                    {NAME}{GET key158}{IF key158=""{ESC}""}{ESC 8}
                    {BRANCH key158}"
16 hhh1158          +"~{LET routin158,@CELLPOINTER("""&B32&""")}~{back158}
                    {DISPATCH routin158}~"
32 loc158           @IF(@LEFT(B30,1)<>"@","coord","address")
```

Before the 2.2 version of Lotus 1-2-3 was released, you could only assign up to 26 macro letters, the number of letters in the English alphabet, plus \0, the auto exec macro. With the introduction of version 2.2 a new option has been added that allows a macro to have any name, limited to the same rules of up to 15 character per name as with range names. Now you can assign a meaningful name to a macro. To activate a macro, press the ALT-F3 key combination, which displays the list of all the range names in the worksheet. Highlight and point to the name and press ENTER. This macro imitates this new option in Lotus versions 2.0/2.01. To activate the \Z macro, press the classic ALT-Z key combination, and the macro displays a full screen list of all the range names in the worksheet. Then move the highlight and point to the name of the

macro and press ENTER.

	A	B	C	D	E
11	RUNKEY2	{LET rel158,@INFO("release")}~{RECALC loc158} {RECALC form158}{RECALC hhh1158}			

To check if you are using a 3-D or a 2-D Lotus release, the macro issues {LET rel158, @INFO("release")}~, which store the result of the @INFO("release") function in cell [rel158] for later use. The macro issues {RECALC loc158}{RECALC form158}{RECALC hhh1158} to update the dynamic code formulas in B12, B16 and B32 cells named [form158], [hhh1158] and [loc158] respectively. The formulas in these cells are integral part of the macro code; therefore the macro updates them before it processes them. We will study these formulas when we reach their code. The first formula is in B12, [form158].

```
12 form158         +"{err1158}{LET back1158,@CELLPOINTER("""&B32&""")}~
                   {WINDOWSOFF}{PANELOFF}{GOTO}{NAME}{WINDOWSON}{PANELON}
                   {NAME}{GET key158}{IF key158=""{ESC}""}{ESC 8}
                   {BRANCH key158}"
```

Before starting with the detailed investigation of the result of this formula, note that the formula uses the content of B32, [loc158] to construct the code, so let's look at it first.

```
32 loc158          @IF(@LEFT(B30,1)<>"@","coord","address")
```

The result of this formula also depends on the content of B30, [rel158]. The result of the @INFO("release") function is stored in [rel158] and the macro uses @IF(@LEFT(rel158,1)<>"@" to check if you are using a 3-D Lotus release. The macro uses the @LEFT(rel158,1) Lotus string function to check if the first character of the cell content is unequal to the "@" character. If so, you are using a 3-D release, but if it is, you are using a 2-D Lotus release.

The result of this formula can be the "address" string if you are using a 2-D release or the "coord" string if you are using a 3-D release. Therefore the result code of the formula in B12 can take the following two forms. If you are using a 2-D release the form is:

	A	B	C	D	E
12	form158	+"{err1158}{LET back1158,@CELLPOINTER("address")}~ {WINDOWSOFF}{PANELOFF}{GOTO}{NAME}{WINDOWSON}{PANELON} {NAME}{GET key158}{IF key158=""{ESC}""}{ESC 8} {BRANCH key158}"			

If you are using a 3-D release the form is:

	A	B	C	D	E
12	form158	+"{err1158}{LET back1158,@CELLPOINTER("coord")}~ {WINDOWSOFF}{PANELOFF}{GOTO}{NAME}{WINDOWSON}{PANELON} {NAME}{GET key158}{IF key158=""{ESC}""}{ESC 8} {BRANCH key158}"			

The release doesn't matter since the explanation for both forms is the same. The macro starts with {err1158}; this is an error handling routine that the macro activates to prevent the macro from stopping if an error occurs. Such an error can happen if you point to a range name that is not really the name of a macro and therefore can contain elements that Lotus will consider as errors. Next, the macro uses {LET back1158,@CELLPOINTER("coord")}~ to store the cell pointer's current position in cell [back1158], which will allow the macro to return the cell pointer to this location. To avoid unnecessary screen activity, the macro issues {WINDOWSOFF}

{PANELOFF}.

To display a full screen list of range and macro names, the macro uses {GOTO}{NAME} {NAME} and {WINDOWSON}{PANELON} to release screen and panel display activities. Their use combined with the previous {PANELOFF}{WINDOWSOFF} serves only to improve the macro's appearance. In other cases where many calculations and operations are needed, it can save up to 50% of the process time.

To pause and wait until you respond, the macro issues {GET key158}. When you press any key, the macro regains control and stores the key in cell [key158]. If you press ESC, the macro issues {ESC 8} to clear the prompt from the panel and return to READY mode before any other command will write the prompt to the current cell. The macro uses {BRANCH key158} to branch to the [key158] routine, which holds the key that you pressed, in this case {ESC}, and executes it one more time and quits.

	A	B	C	D	E
13	!	{IF key158="~"}{BRANCH hhh1158}			
14	!	{key158}{?}~{BRANCH hhh1158}			

To check if you pressed ENTER, the macro issues {IF key158="~"}. If so, the macro uses {BRANCH hhh1158} to branch to the [hhh1158] routine. If you press any other key, the macro uses {key158} to activate this key. Recall that [key158] holds the key you pressed, therefore the macro executes the key as you meant to. The macro issues {?} to halt macro execution and pause until you point to the macro name and press ENTER. When you press ENTER, the macro uses {BRANCH hhh1158} to branch to the [hhh1158] routine.

Note: the {?} macro command is not the safest approach to control user input, because you can actually use all the keys and functions of Lotus 1-2-3 once {?} is issued.

16	hhh1158	+"~{LET routin158,@CELLPOINTER(""&B32&"")}~{back158} {DISPATCH routin158}~"

Again this formula can have two results, the first result is:

	A	B	C	D	E
16	hhh1158	~{LET routin158,@CELLPOINTER("address")}~{back158} {DISPATCH routin158}~			

or the second result is:

	A	B	C	D	E
16	hhh1158	~{LET routin158,@CELLPOINTER("coord")}~{back158} {DISPATCH routin158}~			

Now the macro issues tilde "~" to finish the {GOTO}{NAME}{NAME} sequence that put the cell pointer on the range whose name you picked. Next it issues {LET routin158, @CELLPOINTER("coord")}~ to store the macro address in cell [routin158]. To send the cell pointer back to its point of origin, the macro issues {back158}.

	A	B	C	D	E
24	back158	{GOTO}			
25	back1158	$A:$A$1			
26	!	~			

The [back158] routine is a second example of the use of dynamic code in this macro. At the beginning of the macro, the cell pointer position (address or coord) was stored in cell [back1158]. Now the macro uses it as part of the code of the [back158] routine, which is a simple {GOTO}$A:$A$1~ macro command, where the $A:$A$1 was the position of the cell pointer before the macro started, the "A1" cell in the "A" sheet of a 3-D worksheet. To actually execute the macro you chose, the macro uses the indirect {DISPATCH} macro command. The macro issues {DISPATCH routin158} to start the routine/macro from the address written in [routin158] which is similar to the indirect @@(cell1) function that returns the value of the cell whose address is written in the [cell1] cell.

To summarize: **(1)** Issue {GOTO}{NAME}{NAME}{?} to display a full screen list of all the range names in the worksheet. **(2)** When you choose the macro name and press ENTER, the tilde "~" finishes the GOTO process, which puts the cell pointer on the first cell of the chose macro. **(3)** Store the macro's cell position in another cell. **(4)** Issue the {DISPATCH} command to start the macro whose address/name appears as the parameter in the {DISPATCH parameter} command.

	A	B	C	D	E
18	err1158	{ONERROR err2158}			
19	!				
20	err2158	{BEEP}INVALID MACRO NAME! PRESS ANY KEY TO CONTINUE ...			
		{GET key158}{ESC}{BRANCH \Z}			

The only routine left to discuss is the [err1158] error handling routine. To make Lotus alert for a possible error, the {err1158} routine command must be issued. The [err1158] routine issues {ONERROR err2158} to tell Lotus to activate the [err2158] routine in case of an error. When an error occurs, Lotus starts the [err2158] routine that uses {BEEP} to notify you that an error occurred. Following the beep, the macro displays:

```
INVALID MACRO NAME! PRESS ANY KEY TO CONTINUE ...
```

in the panel. Because there is no valid Lotus command in this line of text, Lotus just types the text to the panel which appears as a prompt. To halt the macro wait until you press a key, the macro issues {GET key158}. When you press a key, it is stored in [key158], and the macro resumes control. The macro immediately issues {ESC} that clears the prompt from the panel. Then the macro issues {BRANCH \Z} to branch the macro to its beginning and start all over again. Without {ESC} after {GET key158}, the prompt text would have be written to the current cell and may override the cell's content.

[7] Write Check Amount in Words

```
     A                B                C                D                E
 1 *---A macro to write CHECK'S amount in WORDS up to $99,999,999
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Position the cell pointer where you want the amount to be written
 5 *---Hold the [ALT] key and press [Z] to activate the macro
 6 !
 7         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 8                    IT WILL WORK IN LOTUS 2.0 AND UP
 9 !
10 \Z             {BREAKON}
11 CHECKAMT       {LET rel414,@INFO("release")}~{RECALC loc414}
                  {RECALC form414}
12 form414        {WINDOWSOFF}{PANELOFF}{LET here1414,@CELLPOINTER
                  ("address")}~
                  {GOTO}wordsa414~/RNCwordsa414~.{DOWN 8}{RIGHT}~
13 !              {GOTO}ones414~/RNCones414~~/RNDones414~/RNCones414~.{END}
                  {DOWN}~{GOTO}{here1414}~
14 !              {WINDOWSON}{PANELON}{GETNUMBER "Enter check's amount
                  (NO commas) in $ ",summ414}~{CALC}
15 !              {word414}~
16 !
17 here1414       $B$6
18 !
19 rel414         @INFO("release")
20 !
21 loc414         address
22 !
23 word414        TWENTY MILLION ONE AND 00/100 DOLLARS
24 summ414             20000001
25 wordsa414                     AND 00/100 DOLLARS
26 !                            1 ONE
27 !                            0
28 !                            0
29 !                            0
30 !                            0
31 !
32 !                            0
33 !                  20000000
34 !                             MILLION
35 !                  20000000 TWENTY
36 !
37 ones414                      0
38 !                            1 ONE
39 !                            2 TWO
40 !                            3 THREE
41 !                            4 FOUR
42 !                            5 FIVE
43 !                            6 SIX
44 !                            7 SEVEN
45 !                            8 EIGHT
46 !                            9 NINE
47 !                           10 TEN
48 !                           11 ELEVEN
49 !                           12 TWELVE
50 !                           13 THIRTEEN
51 !                           14 FOURTEEN
52 !                           15 FIFTEEN
53 !                           16 SIXTEEN
54 !                           17 SEVENTEEN
55 !                           18 EIGHTEEN
56 !                           19 NINETEEN
57 !                           20 TWENTY
58 !                           30 THIRTY
59 !                           40 FOURTY
60 !                           50 FIFTY
61 !                           60 SIXTY
```

```
62 !                          70 SEVENTY
63 !                          80 EIGHTY
64 !                          90 NINETY
```

This macro translates a number that you write with digits into words. You can use this macro to write check's in amounts of up to 99,999,999.99. Place the cell pointer on the cell where you want the amount written in words and start the macro that prompts you to enter a number, which the macro writes as words in the current cell. This macro is heavily based on dynamic string formulas which take the number that you enter in digits apart, and translate it into words. To key this macro into Lotus 1-2-3, we show a complete list of cell contents at the end of this section.

When you insert a number, the macro stores it in the B24 cell named [summ414]. The text that you see in the B25..C35 range is the result of dynamic string formulas that take the number in B24 and break it to create the building blocks which the macro assembles back in the B23 cell as the same number expressed in words. Do not attempt to key the text that you see in the B25..C35 into Lotus 1-2-3. Instead, key the formulas as they appear in the list at the end of this section. The B37..C64 range is a lookup table that the formulas in the B25..B35 cell use to translate number expressed as digits into numbers in words.

```
     A              B                C                D                E
11 CHECKAMT       {LET rel414,@INFO("release")}~{RECALC loc414}
                  {RECALC form414}
12 form414        {WINDOWSOFF}{PANELOFF}{LET here1414,@CELLPOINTER
                  ("address")}~
                  {GOTO}wordsa414~/RNCwordsa414~.{DOWN 8}{RIGHT}~
13 !              {GOTO}ones414~/RNCones414~~/RNDones414~/RNCones414~.{END}
                  {DOWN}~{GOTO}{here1414}~
14 !              {WINDOWSON}{PANELON}{GETNUMBER "Enter check's amount
                  (NO commas) in $ ",summ414}~{CALC}
15 !              {word414}~
```

The macro starts with the `{LET rel414,@INFO("release")}~` commands which store the result of the `@INFO("release")` function in B38, [rel414]. Later, the macro uses this result to check if you are using a 2-D or a 3-D Lotus release. Next the macro issues `{RECALC loc414}{RECALC form414}`, which recalculate and update the string formulas in B21, [loc414], and B12, [form414]. The code in, B12 is the result of the following string formula:

```
12 form414        +"{WINDOWSOFF}{PANELOFF}{LET here1414,@CELLPOINTER
                  (""&B21&"")}~{GOTO}wordsa414~/RNCwordsa414~.{DOWN 8}
                  {RIGHT}~"
```

We can see that this formula uses the content of B21 to create the resultant code. Let's look at this code, which also includes a formula:

```
21 loc414         @IF(@LEFT(B19,1)<>"@","coord","address")
```

This formula uses the result of the `@INFO("release")` function, which was stored in the B19 cell. If "`@`" is the first character of the content of the B19 cell, you are using a 2-D version of 1-2-3, and the result of the formula is the "`address`" label. Otherwise you are using a 3-D version of 1-2-3 and the result of the formula is the "`coord`" label. Therefore the result of the formula in the B12 cell can take two possible forms. If you are using a 2-D version, the resultant code is:

```
     A              B                C                D                E
12 form414        {WINDOWSOFF}{PANELOFF}{LET here1414,@CELLPOINTER
                  ("address")}~{GOTO}wordsa414~/RNCwordsa414~.{DOWN 8}
                  {RIGHT}~
```

For a 3-D version, the resultant code is:

	A	B	C	D	E
12	form414	{WINDOWSOFF}{PANELOFF}{LET here1414,@CELLPOINTER ("coord")}~{GOTO}wordsa414~/RNCwordsa414~.{DOWN 8} {RIGHT}~			

The code in B12, [form414], starts with {WINDOWSOFF}{PANELOFF}, which freeze screen and panel display activities. Then the macro uses {LET here1414,@CELLPOINTER ("address")}~ to store the current cell pointer location in cell [here1414]. Later, the macro will use this address to return the cell pointer to its point of origin. Next the macro issues {GOTO}wordsa414~, which moves the cell pointer to B25, [wordsa414], and then /RNC wordsa414~.{DOWN 8}{RIGHT}~ to reassign the [wordsa414] range name definition to include not only the B25 cell but the B25..C35 range. The [wordsa414] range name was placed in the macro as a position marker. We could not use a simple cell address as this marker because no one can tell where a user will place the macro.

	A	B	C	D	E
13	!	{GOTO}ones414~/RNCones414~~/RNDones414~/RNCones414~.{END} {DOWN}~{GOTO}{here1414}~			
14	!	{WINDOWSON}{PANELON}{GETNUMBER "Enter check's amount (NO commas) in $ ",summ414}~{CALC}			
15	!	{word414}~			

Next the macro issues {GOTO}ones414~, which moves the cell pointer to B37, [ones414], and then /RNCones414~~/RNDones414~/RNCones414~.{END}{DOWN}~ (the "safe technique") to assign the [ones414] range name to the B37..C64 range instead of B37 alone. Again the [ones414] range name for B37 serves as a marker for the macro to expand the [ones414] range name to include the B37..C64 range. The B37..C64 range serves as a lookup table for the macro, which contains the translation of the numbers from 1 to 20 and 30 to 90 to words. Next the macro issues the {GOTO}{here1414}~ indirect[7] macro command to return the cell pointer to its point of origin address, which was in [here1414] when the macro started. To allow you to view the screen and the panel, the macro issues {WINDOWSON}{PANELON} and then issues

```
{GETNUMBER "Enter check's amount (NO commas) in $ ",summ414}
```

which displays:

```
Enter check's amount (NO commas) in $
```

in the panel and waits for your response. When you type a number, Lotus stores it in B24, [summ414], and then issues {CALC}, which updates all the formulas in the macro. Now the macro continues with the {word414} routine command which writes (injects) the content of [word414] into the panel and then issues the tilde "~", which writes it into the current cell.

This code is only a small part of the macro. Most of the work is done behind the scenes by the formulas that split the number you entered into small parts and translate them into words, and re-assemble the whole puzzle into a meaningful sentence that describes the number in words. The starring function is the @MOD(number,10) Lotus function that makes it possible to remove the reminder of the number divided by 10. The formula in B26 is a good example:

```
@MOD(@MOD(@MOD(@MOD(@MOD(@MOD(@INT(B24),1000000),100000),10000)
    ,1000),100),10)
```

The formula in B26 extracts the units remainder of the number. If you enter 99999999.56, then B26 returns the digit `9`. Therefore C26 which contains the following formula:

```
@IF(B26<>0,@IF(B27>19,@VLOOKUP(B26,B37..C64,1)&" ",@IF(B27>19
    #OR#B27<10,@VLOOKUP(B26,B37..C64,1)&" ","")),"")
```

returns the "`NINE`" string. The rest of the formulas in the B27..C35 range extract the other parts of the numbers (the tenth, hundreds, thousands, ten thousands, hundred thousands, millionths and ten millionths). The formulas in the B26..C35 range use the `@LOOKUP` function to translate the numbers to words. Notice that the formulas "know" how to translate numbers between 11 and 19 correctly. We are not going to research these formulas further. ***Super Power*** is about the macro language. Study these formulas carefully if you would like to fully understand how this macro works. The C25 cell contains the following formula:

```
@IF(B24-@INT(B24)<>0,+"AND "&@STRING(@ROUND(+B24-@INT(B24),2)*100
    ,0)&"/100 DOLLARS","AND 00/100 DOLLARS")
```

This formula builds the cents if the number also includes cents. If you enter 123.56, then this formula returns the `AND 56/100 DOLLARS` string. Otherwise it returns the `AND 00/100 DOLLARS` string. The B25 cell contains the following formula:

```
@IF(B30<19999#AND#B30>9999,C33&C32&C31&C28&C27&C26&C25,C35&C34&
    C33&C32&C30&C29&C28&C27&C26&C25)
```

This formula assembles all the parts of the numbers in the C25..C35 range to create the complete sentence that expresses the number in words. Here is the complete list of the cell contents in this macro.

```
A1: U [W14] '*---A macro to write CHECK'S amount in WORDS up to 99,999,999
A2: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A3: [W14] '    range names in this column (starts with the \Z macro name)
A4: [W14] '*---Position the cell pointer where you want the amount to be
          written
A5: [W14] '*---Hold the [ALT] key and press [Z] to activate the macro
A6: [W14] '!
A7: U [W14] '       THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
A8: U [W14] '                    IT WILL WORK IN LOTUS 2.0 AND UP
A9: [W14] '!
A10: U [W14] '\Z
B10: [W14] '{BREAKON}
A11: U [W14] 'CHECKAMT
B11: [W14] '{LET rel414,@INFO("release")}~{RECALC loc414}{RECALC form414}
A12: [W14] 'form414
B12: U [W14] '+"{WINDOWSOFF}{PANELOFF}{LET here1414,@CELLPOINTER
            (""""&B21&""")}~{GOTO}wordsa414~/RNCwordsa414~.{DOWN 8}{RIGHT}~"
A13: [W14] '!
B13: [W14] '{GOTO}ones414~/RNCones414~~/RNDones414~/RNCones414~.{END}{DOWN}~
           {GOTO}{here1414}~
A14: [W14] '!
B14: [W14] '{WINDOWSON}{PANELON}{GETNUMBER "Enter check's amount (NO commas)
          in $ ",summ414}~{calc}
A15: [W14] '!
B15: [W14] '{word414}~
A16: [W14] '!
A17: [W14] 'here1414
B17: [W14] '$B$6
A18: [W14] '!
A19: [W14] 'rel414
B19: [W14] '@INFO("release")
A20: [W14] '!
```

```
A21: [W14] 'loc414
B21: U [W14] '@IF(@LEFT(B19,1)<>"@","coord","address")
A22: [W14] '!
A23: [W14] 'word414
B23: [W14] @IF(B30<19999#AND#B30>9999,C33&C32&C31&C28&C27&C26&C25,C35&C34&
           C33&C32&C30&C29&C28&C27&C26&C25)
A24: [W14] 'summ414
B24: [W14] 20000001
A25: [W14] 'wordsa414
C25: U '@IF(B24-@INT(B24)<>0,+"AND "&@STRING(@ROUND(+B24-@INT(B24),2)*100,0)
       &"/100 DOLLARS","AND 00/100 DOLLARS")
A26: [W14] '!
B26: U [W14] @MOD(@MOD(@MOD(@MOD(@MOD(@MOD(@INT(B24),1000000),100000),10000)
             ,1000),100),10)
C26: U @IF(B26<>0,@IF(B27>19,@VLOOKUP(B26,B37..C64,1)&" ",@IF(B27>19#OR#B27
       <10,@VLOOKUP(B26,B37..C64,1)&" ","")),"")
A27: [W14] '!
B27: U [W14] @MOD(@MOD(@MOD(@MOD(@MOD(@INT(B24),1000000),100000),10000),
             1000),100)-(B26)
C27: U @IF(B27<>0,@IF(B27<20,@VLOOKUP(B27+B26,B37..C64,1)&" ",@IF(B27<>0
       ,@VLOOKUP(B27,B37..C64,1)&" ","")),"")
A28: [W14] '!
B28: U [W14] (@MOD(@MOD(@MOD(@MOD(@INT(B24),1000000),100000),10000),1000)
             -B27-B26)
C28: U @IF(B28<>0,@VLOOKUP(B28/100,B37..C64,1)&" HUNDRED ","")
A29: [W14] '!
B29: U [W14] @MOD(@MOD(@MOD(@INT(B24),1000000),100000),10000)-(B28)-(B27)-
             (B26)
C29: U @IF(B29<>0,@VLOOKUP((B29)/1000,B37..C64,1)&" THOUSAND ",(@IF((B29=0)
       #AND#(B30<>0#OR#(B32<>0)),"THOUSAND ","")))
A30: [W14] '!
B30: U [W14] (@MOD(@MOD(@INT(B24),1000000),100000)-(B29)-(B28)-(B27)-(B26))
C30: U @IF(B30<>0,@VLOOKUP((B30)/1000,B37..C64,1)&" ","")
A31: [W14] '!
C31: U @IF(B30<>0#AND#B30<19999,@VLOOKUP((B30+B29)/1000,B37..C64,1)&
       " THOUSAND ","")
A32: [W14] '!
B32: U [W14] @MOD(@INT(B24),1000000)-(B30)-(B29)-(B28)-(B27)-(B26)
C32: U @IF(B32<>0,@VLOOKUP(B32/100000,B37..C64,1)&" HUNDRED ","")
A33: [W14] '!
B33: U [W14] @INT(B24)-(B32)-(B30)-(B29)-(B28)-(B27)-(B26)
C33: U @IF(B33<>0#AND#B33<20000000,@VLOOKUP(B33/1000000,B37..C64,1)&
       " MILLION ","")
A34: [W14] '!
B34: U [W14] @IF(B33>20000000,@MOD(@INT(B24),10000000)-(B30)-(B29)-(B28)-
             (B27)-(B26)-(B32),"")
C34: U @IF(B34<>0,@VLOOKUP(B34/1000000,B37..C64,1)&" MILLION ",@IF(B34=0
       #AND#B35>=20000000,"MILLION ",""))
A35: [W14] '!
B35: [W14] +B33-B34
C35: U @IF(B33>=20000000,@VLOOKUP(B33/1000000,B37..C64,1)&" ","")
A36: [W14] '!
A37: [W14] 'ones414
B37: [W14] 0
C37: '
A38: [W14] '!
B38: [W14] 1
C38: 'ONE
A39: [W14] '!
B39: [W14] 2
C39: 'TWO
A40: [W14] '!
B40: [W14] 3
C40: 'THREE
A41: [W14] '!
B41: [W14] 4
C41: 'FOUR
A42: [W14] '!
B42: [W14] 5
C42: 'FIVE
A43: [W14] '!
```

```
B43: [W14] 6
C43: 'SIX
A44: [W14] '!
B44: [W14] 7
C44: 'SEVEN
A45: [W14] '!
B45: [W14] 8
C45: 'EIGHT
A46: [W14] '!
B46: [W14] 9
C46: 'NINE
A47: [W14] '!
B47: [W14] 10
C47: 'TEN
A48: [W14] '!
B48: [W14] 11
C48: 'ELEVEN
A49: [W14] '!
B49: [W14] 12
C49: 'TWELVE
A50: [W14] '!
B50: [W14] 13
C50: 'THIRTEEN
A51: [W14] '!
B51: [W14] 14
C51: 'FOURTEEN
A52: [W14] '!
B52: [W14] 15
C52: 'FIFTEEN
A53: [W14] '!
B53: [W14] 16
C53: 'SIXTEEN
A54: [W14] '!
B54: [W14] 17
C54: 'SEVENTEEN
A55: [W14] '!
B55: [W14] 18
C55: 'EIGHTEEN
A56: [W14] '!
B56: [W14] 19
C56: 'NINETEEN
A57: [W14] '!
B57: [W14] 20
C57: 'TWENTY
A58: [W14] '!
B58: [W14] 30
C58: 'THIRTY
A59: [W14] '!
B59: [W14] 40
C59: 'FOURTY
A60: [W14] '!
B60: [W14] 50
C60: 'FIFTY
A61: [W14] '!
B61: [W14] 60
C61: 'SIXTY
A62: [W14] '!
B62: [W14] 70
C62: 'SEVENTY
A63: [W14] '!
B63: [W14] 80
C63: 'EIGHTY
A64: [W14] '!
B64: [W14] 90
C64: 'NINETY
```

[10] The Macro Manager

```
     A                B                C                D                E
 1 *---A MACRO MANAGER to activate any macro without first combining it
 2     to the worksheet. The macro manager combines the macro, defines
 3     the range names and activates the macro. When the macro is done
 4     the MACRO MANAGER deletes the range names and erases the macro.
 5 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 6     range names in this column (starts with the \Y name)
 7 *---Place the cell pointer at upper left cell of the range to be
 8     processed
 9 *---Hold the [ALT] key and press [Y] to activate the macro
10 !
11 \Y             {RESTART}{BRANCH SMALLMGR}
12 !
13 trash@247
14 !
15 erra@247       {ONERROR errb@247}
16 !
17 errb@247       {BEEP}INVALID MACRO NAME! PRESS ANY KEY TO CONTINUE ...
                  {GET keyw@247}{ESC}{back#247}{BRANCH \Y}}
18 !
19 remove@247     {IF @CELLPOINTER("type")="b"}{GOTO}\Y~{LEFT}{UP 10}/RND\Y~
                  /RNDtrash@247~/RNDerra@247~/RNDerrb@247~/RNDremove@247~
                  /RNDrngnam1@247~/RNC!~~/RND!~/RE.{END}{DOWN}{RIGHT 7}~
                  {QUIT}
20 !              {WINDOWSON}{IF @CELLPOINTER("contents")="!"}{DOWN}
                  {BRANCH remove@247}
21 !              {LET rngnam1@247,@CELLPOINTER("contents")}~/RND
22 rngnam1@247    execute3@247
23 !              ~{DOWN}
24 !              {IF @CELLPOINTER("type")="b"}{GOTO}\Y~{LEFT}{UP 10}/RND\Y~
                  /RNDtrash@247~/RNDerra@247~/RNDerrb@247~/RNDremove@247~
                  /RNDrngnam1@247~/RE.{END}{DOWN}{RIGHT 7}~{QUIT}
25 !              {BRANCH remove@247}
26 !
27 SMALLMGR       {WINDOWSOFF}{PANELOFF}{LET back#1247,@CELLPOINTER
                  ("address")}~{rangerase@247}{MENUBRANCH menu#1247}
                  {BRANCH \Y}
28 !
29 BACK#247       {GOTO}
30 BACK#1247      $AA$1
31 !              ~
32 !
33 namedel#1247   {WINDOWSOFF}{GOTO}execute3@247~{LEFT}{DOWN 10}
34 lop#1247       {DOWN}{IF @CELLPOINTER("type")="b"}{GOTO}execute3@247~
                  {rangerase@247}{back#247}{WINDOWSON}{BRANCH \Y}
35 !              {IF @CELLPOINTER("contents")="!"}{BRANCH lop#1247}
36 !              {IF @CELLPOINTER("type")="b"}{GOTO}execute3@247~
                  {rangerase@247}{back#247}{WINDOWSON}{BRANCH \Y}
37 !              {LET rangname@247,@CELLPOINTER("contents")}~/RND
38 rangname@247   rel210
39 !              ~{BRANCH lop#1247}
40 !
41 execute@247    {err#1247}{WINDOWSOFF}{GOTO}execute3@247~{LEFT}{DOWN 2}
                  /FCCE
42 execute1@247   \Z
43 !              ~{BRANCH execute2@247}
44 !
45 execute4@247   {err#1247}{WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~{LEFT}
                  {DOWN 2}/FCCE{PANELON}{WINDOWSON}{NAME}{?}{WINDOWSOFF}~
                  {PANELOFF}{BRANCH execute2@247}
46 !
47 execute2@247   {WINDOWSOFF}{PANELOFF}{DOWN 9}/RNLR{END}{DOWN}~/RNC!~~
                  /RND!~{back#247}{execute3@247}{namedel#1247}
48 !
49 rangerase@247 {WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~{LEFT}{DOWN 2}
                  {IF @CELLPOINTER("type")<>"b"}/RE{END}{DOWN}{RIGHT 8}~
50 !              {back#247}
```

```
51 !
52 menu#1247      Default-dir Macro_run  Runkey  View  List  Erase_mgr  Quit
53 !              Set the defaShort cut tExecute View tDisplaRemove the Quit
54 !              {WINDOWSON}{{WINDOWSOFF{RUNKEY#{GOTO}/FLO{W{GOTO}small{QUI
55 !              {MENUBRANCH MENU#1247} {MENUBRANCH ME{MENUBRANCH MENU#1247
56 !
57 err#1247       {ONERROR warning@247}
58 !
59 warning@247    {BEEP}{GETLABEL "Check your default directory ! , Press
                  [ENTER] to continue",trash@247}~{BRANCH \Y}
60 !
61 filename@247   dddd
62 !
63 describ@247    z1
64 !
65 runkey#1247    {erra@247}{LET back#1247,@CELLPOINTER("address")}~
                  {WINDOWSOFF}{PANELOFF}{GOTO}{NAME}{WINDOWSON}{PANELON}
                  {NAME}{GET keyw@247}{IF keyw@247="{ESC}"}{ESC 8}{back#247}
                  {BRANCH \y}
66 !              {IF keyw@247="~"}{BRANCH hhh#1247}
67 !              {keyw@247}{?}~{BRANCH hhh#1247}
68 !
69 hhh#1247       ~{LET routin#1247,@CELLPOINTER("address")}~{back#247}
                  {DISPATCH routin#1247}~
70 !
71 routin#1247    $B$11
72 !
73 keyw@247       {END}
74 !
75 view1@247      Use ARROWS to move or <E>xecute or Press [RETURN] to
                  finish{GET keyw@247}{ESC}
76 !              {IF keyw@247="{RIGHT}"#OR#keyw@247="{LEFT}"#OR#keyw@247=
                  "{UP}"#OR#keyw@247="{DOWN}"#OR#keyw@247="{PGDN}"#OR#
                  keyw@247="{PGUP}"}{keyw@247}{BRANCH view1@247}
77 !              {IF @UPPER(keyw@247)="E"}{LET execute1@247,"\Z"}~
                  {err#1247}{WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~
                  {LEFT}{DOWN 2}{BRANCH execute2@247}
78 !              {rangerase@247}{back#247}{WINDOWSON}{BRANCH \y}
79 !
80 execute3@247  {WINDOWSON}{PANELON}{back#247}{DISPATCH execute1@247}
                  {RETURN}
81 !
```

This macro allows you to start any macro in the default directory using point and shoot, and to work from a custom menu. The macros in the default directory have to be built according to the macro structure rules. This is only one of seven macro managers which are included in the SUPER MACRO LIBRARY package. We show it here because it is the shortest. Because this macro contains a custom menu which cannot fit on one page, we show every menu option separately. If you intend to key this macro into Lotus 1-2-3, you have to key the full code as it appears here, NOT as it is shown in the main macro list. To make it easier, we also show a complete list of all the cell contents of the macro at the end of this section.

```
Default-dir
Set the default directory to the directory where the macros are.
{WINDOWSON}{PANELON}/WGDD{?}~{WINDOWSOFF}{PANELOFF}{ESC 6}
{MENUBRANCH menu#1247}


Macro_run
Short cut to run a macro. Make sure the macros are in the default
    directory
{WINDOWSOFF}{PANELOFF}{LET back#1247,@CELLPOINTER("address")}~
    {LET execute1@247,"\Z"}~{BRANCH execute4@247}


Runkey
Execute any routine in the current worksheet (Point and Shoot)
{runkey#1247}
```

```
{MENUBRANCH menu#1247}

View
View the macro and operating instruction
{GOTO}execute3@247~{LEFT}{DOWN 2}/FCCE{WINDOWSON}{PANELON}{NAME}
     {?}~{WINDOWSOFF}{PGDN}{UP 20}{WINDOWSON}{view1@247}

List
Display names of all files in current directory to verify macro
    existance
/FLO{WINDOWSON}{PANELON}{NAME}{NAME}{?}{ESC 8}~
{MENUBRANCH menu#1247}

Erase_mgr
Remove the macro manager from the spreadsheet
{GOTO}SMALLMGR~{LEFT}{BRANCH remove@247}

Quit
Quit the macro manager
{QUIT}
```

	A	B	C	D	E
11	\Y	{RESTART}{BRANCH SMALLMGR}			

The macro starts with the {RESTART} command, which clears the macro stack, and then issues {BRANCH SMALLMGR}, which routes macro control to the [SMALLMGR] routine.

	A	B	C	D	E
27	SMALLMGR	{WINDOWSOFF}{PANELOFF}{LET back#1247,@CELLPOINTER ("address")}~{rangerase@247}{MENUBRANCH menu#1247} {BRANCH \Y}			

The routine starts with {WINDOWSOFF}{PANELOFF}, which freeze screen and panel display activities, and then {LET back#1247,@CELLPOINTER("address")}, which stores the current cell pointer location in the B30 cell, [back#1247]. The macro will use this address to return to this location. Next the macro issues the {rangerase@247} routine command which starts the [rangerase@247] routine.

	A	B	C	D	E
49	rangerase@247	{WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~{LEFT}{DOWN 2} {IF @CELLPOINTER("type")<>"B"}/RE{END}{DOWN}{RIGHT 8}~			
50	!	{back#247}			

This routine is used to erase the area where the manager combines the macro you choose to execute. Before the manager combines a new macro, it first checks if the area down to the B80 cell, [execute3@247], is clear. If not, the macro erases it. The routine starts with {WINDOWSOFF}{PANELOFF}, which freeze screen and panel display activities, and then issues {GOTO}execute3@247~{LEFT}{DOWN 2}, which position the cell pointer on the A82 cell. Next the macro issues {IF @CELLPOINTER("type")<>"B"}, which checks if the A82 cell is occupied. If so, the macro issues /RE{END}{DOWN}{RIGHT 8}~, which erases all the contiguous data downward and eight columns to the right. One of the rules for how to build a macro that will work with this macro manager is that all the cells in the first column of the macro must be occupied to create a contiguous column, to allow the macro manager to use {END}{DOWN} to select the whole text of the combined macro. Now, the macro issues the {back#247} routine command, which starts the [back#247] routine.

	A	B	C	D	E
29	back#247	{GOTO}			

```
30 back#1247      $AA$1
31 !              ~
```

This is a very simple routine which sums up to the {GOTO}$AA$1~ command, which sends the cell pointer to AA1. The macro previously used {LET back#1247,@CELLPOINTER ("address")} in B27, [SMALLMGR], to store the $AA$1 address in B30, [back#1247]. This is one example of how we can use dynamic macro codes where the macro changes the code before reaching it. When this routine is finished, the macro routes back to the [SMALLMGR] routine.

```
          A              B                C              D              E
27 SMALLMGR       {WINDOWSOFF}{PANELOFF}{LET back#1247,@CELLPOINTER
                  ("address")}~{rangerase@247}{MENUBRANCH menu#1247}
                  {BRANCH \Y}
```

Now the macro issues the {MENUBRANCH menu#1247} macro command which activates the [menu#1247] custom menu. The first menu item is [**D**efault-dir]

```
Default-dir
Set the default directory to the directory where the macros are.
{WINDOWSON}{PANELON}/WGDD{?}~{WINDOWSOFF}{PANELOFF}{ESC 6}
{MENUBRANCH menu#1247}
```

The macro manager looks for the macros in the default directory only so use this menu option to define the default directory to the same directory where the macros are. When you start this menu option, the macro issues {WINDOWSON}{PANELON}, which resume screen and panel display activities to allow you view the prompts in the panel and enter the default directory. Next the macro issues /WGDD, which start the process to assign the default directory and then {?}, which halts the macro and waits for your entry. When you enter the default directory or accept the current one and press ENTER, the macro issues the tilde "~" which finishes the process. Then the macro issues {WINDOWSOFF}{PANELOFF} macro commands to freeze screen and panel display activities and then {ESC 6} to return to the READY mode. Next, the macro issues {MENUBRANCH menu#1247}, which re-starts the [menu#1247] custom main menu, to allow you to continue to use the macro manager. The second menu item is [**M**acro_run]:

```
Macro_run
Short cut to run a macro. Make sure the macros are in the default
    directory
{WINDOWSOFF}{PANELOFF}{LET back#1247,@CELLPOINTER("address")}~
    {LET execute1@247,"\Z"}~{BRANCH execute4@247}
```

When you know the name of the macro you want to start, use this menu option. It issues {WINDOWSOFF}{PANELOFF} to freeze screen and panel display activities and then {LET back#1247,@CELLPOINTER("address")}, which stores the current cell pointer location in B30, [back#1247]. Later, the macro will use this address to return to the same location. Next, the macro issues {LET execute1@247,"\Z"}, which stores the \Z label in B42, [execute1@247], and then {BRANCH execute4@247}, which routes macro control to the [execute4@247] routine.

```
          A              B                C              D              E
45 execute4@247   {err#1247}{WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~{LEFT}
                  {DOWN 2}/FCCE{PANELON}{WINDOWSON}{NAME}{?}{WINDOWSOFF}~
                  {PANELOFF}{BRANCH execute2@247}
```

The [execute4@247] routine starts with the {err#1247} routine command, which starts the [err#1247] error handling routine. We will come back to the error handling routine later. The macro continues with {WINDOWSOFF}{PANELOFF}, which freeze screen and panel display activities and then {GOTO}execute3@247~{LEFT}{DOWN 2}, which temporarily move the cell pointer to the A82 cell, the first empty cell under the macro manager. Now, the macro issues /FCCE to start the / **F**ile **C**ombine process. Then, the macro issues {PANELON}{WINDOWSON}, which resume screen and panel display activities. Next, it issues {NAME}, which displays a full screen list of all the *.WK* macros in the default directory, follow by {?}, which allows you to highlight the macro name and press ENTER.

To you it looks like that you chose the macro to execute, which is partially correct, however what the macro really does is combine the macro to the current worksheet. But just before the macro manager combines the macro that you chose, it issues {WINDOWSOFF} to suppress screen activity, and only then issues the tilde "~" to tell Lotus to combine the macro. Next the macro issues {BRANCH execute2@247}, which starts the [execute2@247] routine.

	A	B	C	D	E
47	execute2@247	{WINDOWSOFF}{PANELOFF}{DOWN 9}/RNLR{END}{DOWN}~/RNC!~~ /RND!~{back#247}{execute3@247}{namedel#1247}			

Again the macro issues {WINDOWSOFF}{PANELOFF} and then issues {DOWN 9}, which moves the cell pointer to the cell containing the \Z label in the combined macro. The newly combined macro has to conform to the same macro structure as the macros in ***Super Power***. Now the macro issues /RNLR{END}{DOWN}~, which assigns all the range names in the combined macro exactly as you do, manually, from the keyboard. Because there is a good chance that the first column of the combined macro contains the exclamation mark, there is a possibility that one of the new range names is the [!] range name, which is not quite legal. Therefore the macro continues and issues /RNC!~~/RND!~ to safely delete the [!] range name. Next the macro issues {back#247}; which activates the [back#247] routine and returns the cell pointer to its point of origin. Now, the macro issues {execute3@247}, which starts the [execute3@247] routine.

	A	B	C	D	E
80	execute3@247	{WINDOWSON}{PANELON}{back#247}{DISPATCH execute1@247} {RETURN}			

The routine starts with {WINDOWSON}{PANELON} resuming screen and panel display activities to allow you to respond to the macro prompts. Next the macro issues {back#247} activating the [back#247] routine, which returns the cell pointer to its origin location. Now the macro issues {DISPATCH execute1@247} to start the macro whose name is written in B42, [execute1@247]. At this point, [execute1@247] contains the \Z label, therefore {DISPATCH execute1@247} starts the \Z macro in the combined macro. Recall that almost all the macros in ***Super Power*** start with the [\Z] range name. When you are finished with the combined macro, the macro processor returns control to the next {RETURN}, which returns macro control to the calling routine ,[execute2@247].

	A	B	C	D	E
47	execute2@247	{WINDOWSOFF}{PANELOFF}{DOWN 9}/RNLR{END}{DOWN}~/RNC!~~ /RND!~{back#247}{execute3@247}{namedel#1247}			

Now that the [execute1@247 routine is finished, the macro issues {namedel#1247}, which starts the [namedel#1247] routine erasing the combined macro and deleting its associated range

names.

	A	B	C	D	E
33	namedel#1247	{WINDOWSOFF}{GOTO}execute3@247~{LEFT}{DOWN 10}			
34	lop#1247	{DOWN}{IF @CELLPOINTER("type")="b"}{GOTO}execute3@247~			
		{rangerase@247}{back#247}{WINDOWSON}{BRANCH \Y}			
35	!	{IF @CELLPOINTER("contents")="!"}{BRANCH lop#1247}			
36	!	{IF @CELLPOINTER("type")="b"}{GOTO}execute3@247~			
		{rangerase@247}{back#247}{WINDOWSON}{BRANCH \Y}			
37	!	{LET rangname@247,@CELLPOINTER("contents")}~/RND			
38	rangname@247	rel210			
39	!	~{BRANCH lop#1247}			

The routine starts with {WINDOWSOFF} to freeze screen activity. Then the macro issues {GOTO}execute3@247~{LEFT}{DOWN 10}{DOWN}, which move the cell pointer to the combined macro area. Then the macro issues {IF @CELLPOINTER("type")="b"} to check if the cell is blank. If so, there are no range names in this macro, therefore the macro issues {GOTO}execute3@247~ to return the cell pointer to [execute3@247], and then issues {rangerase@247}, which erases the combined macro text.

	A	B	C	D	E
49	rangerase@247	{WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~{LEFT}{DOWN 2}			
		{IF @CELLPOINTER("type")<>"b"}/RE{END}{DOWN}{RIGHT 8}~			
50	!	{back#247}			

The routine starts with {WINDOWSOFF}{PANELOFF} to freeze screen and panel display activities. Then the macro issues {GOTO}execute3@247~, which moves the cell pointer to [execute3@247], and then {DOWN 2} to move the cell pointer to the beginning of the combined macro. Next, the macro issues {IF @CELLPOINTER("type")<>"b"} to check that the current cell is not empty. If so, the macro issues /RE{END}{DOWN}{RIGHT 8}~, which erase the text from the current cell and downward and nine columns wide to make sure that all the combined macro text is erased. This is exactly why all the macros that work with the macro manager have to have a contiguous first column to allow the macro manager to use {END}[DOWN} to highlight its code. When the macro manager finishes erasing the combined macro, it issues {back#247} to return to its point of origin, and returns control to the [namedel#1247] routine.

	A	B	C	D	E
33	namedel#1247	{WINDOWSOFF}{GOTO}execute3@247~{LEFT}{DOWN 10}			
34	lop#1247	{DOWN}{IF @CELLPOINTER("type")="b"}{GOTO}execute3@247~			
		{rangerase@247}{back#247}{WINDOWSON}{BRANCH \Y}			
35	!	{IF @CELLPOINTER("contents")="!"}{BRANCH lop#1247}			
36	!	{IF @CELLPOINTER("type")="b"}{GOTO}execute3@247~			
		{rangerase@247}{back#247}{WINDOWSON}{BRANCH \Y}			
37	!	{LET rangname@247,@CELLPOINTER("contents")}~/RND			
38	rangname@247	rel210			
39	!	~{BRANCH lop#1247}			

Next the macro issues {BRANCH \Y} which re-starts the macro manager to allow you to use more macros. If the cell pointer was not blank, then the combined macro probably contains range names. The macro previously deleted the [!] range name, therefore the macro issues {IF @CELLPOINTER("contents")="!"} to check if the current cell contains the "!" label. If so, the macro issues {BRANCH lop#1247}, to route macro control back to the [lop#1247] routine, which moves the cell pointer one cell down and again checks if it is blank. If the cell does not contain the exclamation mark "!", the macro again checks if the cell is blank. If so, the macro repeats the code that we have previously seen at the beginning of the [lop#1247] routine.

If the cell contains any other label, the macro issues {LET rangname@247,@CELLPOINTER ("contents")}, which copies the current cell's content into B38, [rangname@247]. The macro issues the tilde "~" to force Lotus to update the content of B38 immediately, because the macro processes the code in B38 immediately. The macro now issues /RND which start the / **R**ange **N**ame **D**elete process. Then it comes to B38, which holds the same label as the label in the current cell, which is also the range name for the cell to right of the current cell. Therefore, the macro deletes the range name. For example: if the current cell is the AA10 cell, which contains the "rel210" label, then the B38 cell contains the same label and the BB10 cell has the [rel210] range name. Therefore /RNDrel210~ deletes the [rel210] range name. The macro continues with {BRANCH lop#1247}, which routes the macro control back to the [lop#1247] routine to process the next cell in the column. The macro continues to delete all the range names of the combined macro until it reaches a blank cell.

When the macro finishes deleting all the range names, the [namedel#1247] routine ends and the macro issues {BRANCH lop#1247}, which routes macro control back to the beginning and displays the main menu.

We still need to look at the error handling routine. When the macro issued {erra@247}, it activated {ONERROR errb@247} which assigns the [errb@247] as the error handling routine.

```
                A               B               C               D               E
15 erra@247         {ONERROR errb@247}
16 !
17 errb@247         {BEEP}INVALID MACRO NAME! PRESS ANY KEY TO CONTINUE ...
                    {GET keyw@247}{ESC}{back#247}{BRANCH \Y}
```

If an error occurs, the macro starts with a {BEEP} to worn you, then types "INVALID MACRO NAME! PRESS ANY KEY TO CONTINUE ..." into the panel, and then issues {GET keyw@247} to pause the macro to allow you to read the message and respond. When you press ENTER, the macro issues {ESC}, which clears the message from the panel and then {back#247} which returns the cell pointer to its point of origin. Last, the macro issues {BRANCH \Y}, which activates the main menu again. This concludes the code of the second menu option. The third menu option is [**R**unkey]:

```
Runkey
Execute any routine in the current worksheet (Point and Shoot)
{runkey#1247}
{MENUBRANCH menu#1247}
```

This allows you to start any routine/macro in the worksheet using point and shoot at the macro/routine name. You can even enter a cell address to start from there to run a macro. When you choose this menu option, the macro issues {runkey#1247}, which starts the [runkey#1247] routine.

```
                A               B               C               D               E
65 runkey#1247      {erra@247}{LET back#1247,@CELLPOINTER("address")}~
                    {WINDOWSOFF}{PANELOFF}{GOTO}{NAME}{WINDOWSON}{PANELON}
                    {NAME}{GET keyw@247}{IF keyw@247="{ESC}"}{ESC 8}{back#247}
                    {BRANCH \y}
66 !                {IF keyw@247="~"}{BRANCH hhh#1247}
67 !                {keyw@247}{?}~{BRANCH hhh#1247}
```

The routine starts with {erra@247}, which assigns [errb@247] as the error handling routine, as we have seen. Next the macro issues {LET back#1247,@CELLPOINTER("address")}, which stores the current cell pointer location address in cell [back#1247]. Next, the macro

issues {WINDOWSOFF}{PANELOFF} to freeze screen and panel display activities, and then {GOTO}, which starts the GOTO cell movement process followed by {NAME}{WINDOWSON}{PANELON}{NAME}, which resume screen and panel display activities. This time, it displays a full screen list of all the range names in the worksheet because of the two {NAME} macro commands.

The macro continues with {GET keyw@247}, which pauses macro execution and waits until you press a key. Lotus stores the key in cell [keyw@247] and then issues {IF keyw@247="{ESC}"} macro command. If so, the macro issues {ESC 8} to return to the READY mode. Then it issues {back#247} to return to its point of origin and issues {BRANCH \Y}, which re-starts the macro manager and displays the main menu. Next the macro issues {IF keyw@247="~"}, which checks if you just pressed ENTER. If so, the macro issues {BRANCH hhh#1247}, which starts the [hhh#1247] routine.

	A	B	C	D	E
69	hhh#1247	~{LET routin#1247,@CELLPOINTER("address")}~{back#247}			
		{DISPATCH routin#1247}~			

The routine starts with the tilde "~" macro command, which finishes the GOTO command and moves the cell pointer to the cell name the screen cursor was highlighting when you pressed the enter key. Then the macro issues {LET routin#1247,@CELLPOINTER ("address")}, which stores the current cell pointer address in [routin#1247] and dynamically alters the code of the macro. Next the macro issues the tilde "~" and forces Lotus to immediately update the content of [routin#1247] to reflect the result of the last {LET} macro command. Next the macro issues the familiar {back#247} to return to the previous location before the last GOTO movement. Now the macro issues the magic {DISPATCH routin#1247} macro command to instruct Lotus to run the macro code which starts in the cell whose address/name is written in [routin#1247], which is the address of the cell that you typed or the address the cell pointer moved to, because of the previous GOTO.

If you press any other key, the macro issues {keyw@247}, which injects the key into the panel and then {?}, which allows you to continue. For example: if you pressed "A" because you wanted to insert an address such as the AA20 address, the macro injects the "A" character to the panel and then issues {?}, which allows you to continue to type the AA20 address. However, if you pressed DOWN because you wanted to point to a range name, the macro executes the {DOWN} command and moves the cursor pointer one row down. Then the macro issues {?}, which allows you to continue to use the direction keys. When you press ENTER, the macro issues the tilde "~" and finishes the GOTO process. Next, the macro issues {BRANCH hhh#1247}, which routes macro control to the [hhh#1247] routine as we have just seen.

This beautiful technique is actually a GOTO command, which displays a full screen list of all the range names in the worksheet, and then moves the cell pointer to the macroor routine or address that you want to execute. It seem that you choose a macro or routine to run. Actually, you instruct Lotus to move the cell pointer to that address, which the macro processes to dynamically insert it as a part of the code, and then uses the {DISPATCH} command to start the macro code from that address. The fourth menu option is [View]:

```
View
View the macro and operating instruction
{GOTO}execute3@247~{LEFT}{DOWN 2}/FCCE{WINDOWSON}{PANELON}{NAME}
     {?}~{WINDOWSOFF}{PGDN}{UP 20}{WINDOWSON}{view1@247}
```

This menu option allows you to view the macro description and code before you choose to execute it. When you choose this option, the macro issues {GOTO}execute3@247~{LEFT}{DOWN 2}, which moves the cell pointer to the place reserved for the combined macro as we have seen in the second menu option, and then uses /FCCE{WINDOWSON}{PANELON}{NAME}{? to start the / **F**ile **C**ombine process and display a full screen list of all the macros in the default directory.

When you select the macro and press ENTER, the manager combines the macro and issues {PGDN}{UP 20}, which bring the first row of the combined macro (the current row) to the first row of the screen (see the TOPSCREN.WK1 macro). Next the macro issues {WINDOWSON} to resume screen display activity and then {view1@247} which starts the [view1@247] routine.

	A	B	C	D	E
75	view1@247	Use ARROWS to move or <E>xecute or Press [RETURN] to finish{GET keyw@247}{ESC}			
76	!	{IF keyw@247="{RIGHT}"#OR#keyw@247="{LEFT}"#OR#keyw@247="{UP}"#OR#keyw@247="{DOWN}"#OR#keyw@247="{PGDN}"#OR#keyw@247="{PGUP}"}{keyw@247}{BRANCH view1@247}			
77	!	{IF @UPPER(keyw@247)="E"}{LET execute1@247,"\Z"}~{err#1247}{WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~{LEFT}{DOWN 2}{BRANCH execute2@247}			
78	!	{rangerase@247}{back#247}{WINDOWSON}{BRANCH \y}			

The routine writes:

```
Use ARROWS to move or <E>xecute or Press [RETURN] to finish
```

into the panel and then issues {GET keyw@247} to pause the macro to allow you to read the message and waits until you press a key. Lotus stores the key in cell [keyw@247] and then the macro manager issues {ESC} to clear the message from the panel before Lotus writes it into the current cell. To allow you to scroll and view the combined macro code and instructions, the macro issues a series of {IF} commands to check the keys you pressed. You can only use the direction keys or execute the macro. You are not allowed to use other keys. Let's see how the macro does it.

The first {IF} condition in the B76 cell checks if you pressed RIGHT, LEFT, UP, DOWN, PGDN or PGUP. If so, the macro issues {keyw@247}, which executes the key that you pressed. For example: if you pressed DOWN, Lotus stored the "{DOWN}" string in [keyw@247]. When the macro issues {keyw@247}, Lotus executes {DOWN}, the only command in the [keyw@247] routine, which moves the cell pointer one cell down. Then the macro issues {BRANCH view1@247}, which routes macro control to the beginning of the [view1@247] routine to allow you to continue to scroll through the macro code.

The second {IF} condition in the B77 cell checks if you pressed the "E" or the "e" character. The macro uses the @UPPER function to allow you to use either upper case or lower case characters. If so, the macro issues GOTO}execute3@247~{LEFT}{DOWN 2}, which place the cell pointer on the first cell of the combined macro. Then the macro issues {BRANCH execute2@247}, which we have already seen in the [**M**acro_run] option from the [execute4@247] routine. In this menu option, there is no need to call the [execute4@247] routine because the manager already combined the macro to allow you to view it. The [execute2@247] routine executes the macro and deletes the range names and the macro's text when you finish using the macro.

If you press any other key, the macro issues the {rangerase@247} routine command to erase the code of the combined macro. Next the macro issues the {back#247} routine command to return the cell pointer to its origin location before you chose the [**V**iew] option, and then issues the {WINDOWSON} command to resume the screen display activity, and last issues {BRANCH \ y}, which re-starts the macro manager and displays the main menu. The fifth menu option is [**L**ist]:

```
List
Display names of all files in current directory to verify macro
    existance
/FLO{WINDOWSON}{PANELON}{NAME}{NAME}{?}{ESC 8}~
{MENUBRANCH menu#1247}
```

When you choose this menu option, the macro issues /FLO, which is the standard / **F**ile **L**ist **O**ther menu options to display all the files in the current default directory. Then the macro issues {WINDOWSON}{PANELON} to allow you to view the screen and panel changes followed by {NAME}{NAME}, which display the file names in a full screen list format. Then the macro issues {?} to pause the macro and allow you to use the direction keys to scroll through the file names to look if the file/macro exists. When you press ENTER, the macro issues the {ESC 8} and the tilde "~" to return to the READY mode. Next the macro issues {MENUBRANCH menu#1247} to display the main menu. The sixth menu option is [**E**rase_mgr]:

```
Erase_mgr
Remove the macro manager from the spreadsheet
{GOTO}SMALLMGR~{LEFT}{BRANCH remove@247}
```

A good macro manager has to delete all the range names of the macros it combines, as well as all deleting all the range names it creates and erasing its code when you are finished. This menu option allows you to erase the macro manager's code and range names to leave the worksheet as it was before you combined the macro manager. When you choose this menu option, the macro issues the {GOTO}SMALLMGR~{LEFT}, which move the cell pointer to the A27 cell. Next the macro issues {BRANCH remove@247}, which routes macro control to the [remove@247] routine.

	A	B	C	D	E
19	remove@247	{IF @CELLPOINTER("type")="b"}{GOTO}\Y~{LEFT}{UP 10}/RND\Y~			
		/RNDtrash@247~/RNDerra@247~/RNDerrb@247~/RNDremove@247~			
		/RNDrngnam1@247~/RNC!~~/RND!~/RE.{END}{DOWN}{RIGHT 7}~			
		{QUIT}			
20	!	{WINDOWSON}{IF @CELLPOINTER("contents")="!"}{DOWN}			
		{BRANCH remove@247}			
21	!	{LET rngnam1@247,@CELLPOINTER("contents")}~/RND			
22	rngnam1@247	execute3@247			
23	!	~{DOWN}			
24	!	{IF @CELLPOINTER("type")="b"}{GOTO}\Y~{LEFT}{UP 10}/RND\Y~			
		/RNDtrash@247~/RNDerra@247~/RNDerrb@247~/RNDremove@247~			
		/RNDrngnam1@247~/RE.{END}{DOWN}{RIGHT 7}~{QUIT}			
25	!	{BRANCH remove@247}			
26	!				

Let's start with the code in the B20 cell. The macro issues {WINDOWSON}, which allows you to see the screen activity when the manager deletes the range names in the "A" column of the macro. Next the macro issues {IF @CELLPOINTER("contents")="!"} to check if the current cell contains the exclamation mark "!". If so, the macro issues {DOWN} and moves the cell pointer to the next cell in the column. Next the macro issues {BRANCH remove@247} to

loop back to the start of the [remove@247] routine to delete the next range name. Recall that the macro deleted the "!" range name when it started, which is the reason why the manager ignores it now.

Now the macro issues `{LET rngnam1@247,@CELLPOINTER("contents")}`, which copies the content of the current cell into B22, [rngnam1@247]. To force Lotus to immediately update the worksheet to reflect the result of the last `{LET}` command, the macro issues the tilde "`~`". The new text in B22 becomes part of the code that the macro is about to process. For example, if the current cell contains the "`execute3@247`" label, then the last `{LET}` command copied the same label into B22, [rngnam1@247]. The macro continues with `/RNDexecute3@247~`, which deletes the [execute3@247] range name. This is a fine example of how we can use dynamic code to change the code before the macro processes it. Every time the cell pointer moves to the next cell, the routine copies the content of the cell to B22 and then issues the / **R**ange **N**ame **D**elete commands followed by the content of the B22 cell which is the name to delete.

Next the macro issues `{DOWN}`, which moves the cell pointer to the next cell to process. Next the macro issues `{IF @CELLPOINTER("type")="b"}`, which checks if the current cell is blank. If so, the cell pointer reached the end of the macro manager's code. Therefore the macro issues `{GOTO}\Y~{LEFT}{UP 10}`, which move the cell pointer to the upper left cell of the macro manager, and then issues:

```
/RNDtrash@247~/RNDerra@247~/RNDerrb@247~/RNDremove@247~/RND rngnam1@247~
```

to delete the rest of the range names in the macro manager. Next the macro issues `/RE.{END}{DOWN}{RIGHT 7}~{QUIT}` to erase all the code of macro manager and quits. Next the macro issues `{BRANCH remove@247}` to branch to the [remove@247] routine and then repeats the code in B24 again in B19 cell. You probably ask why? to our surprise, the first `{IF @CELLPOINTER("type")="b"}` in B24 does not recognize the blank cell, therefore the next one recognizes it. If you key this macro into Lotus 1-2-3, try it and replace the code in the B19 cell with the `{}` do nothing command and see for yourself. The last macro command is [**Q**uit].

```
Quit
Quit the macro manager
{QUIT}
```

When you choose this menu option, the macro issues the standard `{QUIT}` and quits. Here we can afford to use the `{QUIT}` macro command because we want to quit the macro manager. Here is the complete list of cell contents in the macro to assist you in keying this macro into 1-2-3.

```
A1: U [W14] '*---A MACRO MANAGER to activate any macro without first
            combining it
A2: U [W14] '    to the worksheet. The macro manager combines the macro,
            defines
A3: U [W14] '    the range names and activates the macro. When the macro
            is done
A4: U [W14] '    the MACRO MANAGER deletes the range names and erases the
            macro.
A5: [W14] '*---Use the /Range Name Label Right [End] [Down] [ENTER] to
          define the
A6: [W14] '    range names in this column (starts with the \Y name)
A7: [W14] '*---Place the cell pointer at upper left cell of the range to be
A8: [W14] '    processed
A9: [W14] '*---Hold the [ALT] key and press <Y> to activate the macro
A10: [W14] '!
```

```
A11: U [W14] '\Y
B11: [W10] '{RESTART}{BRANCH SMALLMGR}
A12: [W14] '!
A13: [W14] 'trash@247
A14: [W14] '!
A15: [W14] 'erra@247
B15: [W10] '{ONERROR errb@247}
A16: [W14] '!
A17: [W14] 'errb@247
B17: [W10] '{BEEP}INVALID MACRO NAME! PRESS ANY KEY TO CONTINUE ...
           {GET keyw@247}{ESC}{back#247}{BRANCH \Y}}
A18: [W14] '!
A19: [W14] 'remove@247
B19: [W10] '{IF @CELLPOINTER("type")="b"}{GOTO}\Y~{LEFT}{UP 10}/RND\Y~/RND
           trash@247~/RNDerra@247~/RNDerrb@247~/RNDremove@247~/RND
           rngnam1@247~/RNC!~~/RND!~/RE.{END}{DOWN}{RIGHT 7}~{QUIT}
A20: [W14] '!
B20: [W10] '{WINDOWSON}{IF @CELLPOINTER("contents")="!"}{DOWN}
           {BRANCH remove@247}
A21: [W14] '!
B21: [W10] '{LET rngnam1@247,@CELLPOINTER("Contents")}~/RND
A22: [W14] 'rngnam1@247
B22: [W10] 'execute3@247
A23: [W14] '!
B23: [W10] '~{DOWN}
A24: [W14] '!
B24: [W10] '{IF @CELLPOINTER("type")="b"}{GOTO}\Y~{LEFT}{UP 10}/RND\Y~/RND
           trash@247~/RNDerra@247~/RNDerrb@247~/RNDremove@247~
           /RNDrngnam1@247~/RE.{END}{DOWN}{RIGHT 7}~{QUIT}
A25: [W14] '!
B25: [W10] '{BRANCH remove@247}
A26: [W14] '!
A27: U [W14] 'SMALLMGR
B27: [W10] '{WINDOWSOFF}{PANELOFF}{LET back#1247,@CELLPOINTER("address")}~
           {rangerase@247}{MENUBRANCH menu#1247}{BRANCH \Y}
A28: [W14] '!
A29: [W14] 'back#247
B29: [W10] '{GOTO}
A30: [W14] 'back#1247
B30: [W10] '$A$1
A31: [W14] '!
B31: [W10] '~
A32: [W14] '!
A33: [W14] 'namedel#1247
B33: [W10] '{WINDOWSOFF}{GOTO}execute3@247~{LEFT}{DOWN 10}
A34: [W14] 'LOP#1247
B34: [W10] '{DOWN}{IF @CELLPOINTER("type")="b"}{GOTO}execute3@247~
           {rangerase@247}{back#247}{WINDOWSON}{BRANCH \Y}
A35: [W14] '!
B35: [W10] '{IF @CELLPOINTER("contents")="!"}{BRANCH lop#1247}
A36: [W14] '!
B36: [W10] '{IF @CELLPOINTER("type")="b"}{GOTO}execute3@247~{rangerase@247}
           {back#247}{WINDOWSON}{BRANCH \Y}
A37: [W14] '!
B37: [W10] '{LET rangname@247,@CELLPOINTER("contents")}~/RND
A38: [W14] 'rangname@247
B38: [W10] 'rel210
A39: [W14] '!
B39: [W10] '~{BRANCH lop#1247}
A40: [W14] '!
A41: [W14] 'execute@247
B41: [W10] '{err#1247}{WINDOWSOFF}{GOTO}execute3@247~{LEFT}{DOWN 2}/FCCE
A42: [W14] 'execute1@247
B42: U [W10] '\Z
A43: [W14] '!
B43: [W10] '~{BRANCH execute2@247}
A44: [W14] '!
A45: [W14] 'execute4@247
B45: [W10] '{err#1247}{WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~{LEFT}
           {DOWN 2}/FCCE{PANELON}{WINDOWSON}{NAME}{?}{WINDOWSOFF}~{PANELOFF}
           {BRANCH execute2@247}
```

```
A46: [W14] '!
A47: [W14] 'execute2@247
B47: [W10] '{WINDOWSOFF}{PANELOFF}{DOWN 9}/RNLR{END}{DOWN}~/RNC!~~/RND!~
           {back#247}{execute3@247}{namedel#1247}
A48: [W14] '!
A49: [W14] 'rangerase@247
B49: [W10] '{WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~{LEFT}{DOWN 2}
           {IF @CELLPOINTER("type")<>"B"}/RE{END}{DOWN}{RIGHT 8}~
A50: [W14] '!
B50: [W10] '{back#247}
A51: [W14] '!
A52: [W14] 'menu#1247
B52: [W10] 'Default-dir
C52: [W9] 'Macro_run
D52: [W12] 'Runkey
E52: 'View
F52: 'List
G52: 'Erase_mgr
H52: 'Quit
A53: [W14] '!
B53: [W10] 'Set the default directory to the directory where the macros are.
C53: [W9] 'Short cut to run a macro. Make sure the macros are in the default
          directory
D53: [W12] 'Execute any routine in the current worksheet (Point and Shoot)
E53: 'View the macro and operating instruction
F53: 'Display names of all files in current directory to verify macro
     existance
G53: 'Remove the macro manager from the spraedsheet
H53: 'Quit the macro manager
A54: [W14] '!
B54: [W10] '{WINDOWSON}{PANELON}/WGDD{?}~{WINDOWSOFF}{PANELOFF}{ESC 6}
C54: [W9] '{WINDOWSOFF}{PANELOFF}{LET back#1247,@CELLPOINTER("address")}~
          {LET execute1@247,"\Z"}~{BRANCH execute4@247}
D54: [W12] '{runkey#1247}
E54: '{GOTO}execute3@247~{LEFT}{DOWN 2}/FCCE{WINDOWSON}{PANELON}{NAME}{?}~
     {WINDOWSOFF}{PGDN}{UP 20}{WINDOWSON}{view1@247}
F54: '/FLO{WINDOWSON}{PANELON}{NAME}{NAME}{?}{ESC 8}~
G54: '{GOTO}SMALLMGR~{LEFT}{BRANCH remove@247}
H54: '{Quit}
A55: [W14] '!
B55: [W10] '{MENUBRANCH menu#1247}
D55: [W12] '{MENUBRANCH menu#1247}
F55: '{MENUBRANCH menu#1247}
A56: [W14] '!
A57: [W14] 'ERR#1247
B57: [W10] '{ONERROR warning@247}
A58: [W14] '!
A59: [W14] 'warning@247
B59: [W10] '{BEEP}{GETLABEL "Check your default directory ! , Press [ENTER]
           to continue",trash@247}~{BRANCH \Y}
A60: [W14] '!
A61: [W14] 'filename@247
B61: [W10] 'dddd
A62: [W14] '!
A63: [W14] 'describ@247
B63: [W10] 'z1
A64: [W14] '!
A65: [W14] 'runkey#1247
B65: [W10] '{erra@247}{LET back#1247,@CELLPOINTER("address")}~{WINDOWSOFF}
           {PANELOFF}{GOTO}{NAME}{WINDOWSON}{PANELON}{NAME}{GET keyw@247}
           {IF keyw@247="{ESC}"}{ESC 8}{back#247}{BRANCH \y}
A66: [W14] '!
B66: [W10] '{IF keyw@247="~"}{BRANCH hhh#1247}
A67: [W14] '!
B67: [W10] '{keyw@247}{?}~{BRANCH hhh#1247}
A68: [W14] '!
D68: [W12] ^
A69: [W14] 'hhh#1247
B69: [W10] '~{LET routin#1247,@CELLPOINTER("address")}~{back#247}
           {DISPATCH routin#1247}~
A70: [W14] '!
```

```
A71: [W14] 'routin#1247
B71: [W10] '$B$11
A72: [W14] '!
A73: [W14] 'keyw@247
B73: [W10] '{END}
A74: [W14] '!
A75: [W14] 'view1@247
B75: [W10] 'Use ARROWS to move or <E>xecute or Press [RETURN] to finish
           {GET keyw@247}{ESC}
A76: [W14] '!
B76: [W10] '{IF KEYW@247="{RIGHT}"#OR#keyw@247="{LEFT}"#OR#keyw@247="{UP}"
           #OR#keyw@247="{DOWN}"#OR#keyw@247="{PGDN}"#OR#keyw@247="{PGUP}"}
           {keyw@247}{BRANCH view1@247}
A77: [W14] '!
B77: [W10] '{IF @UPPER(keyw@247)="E"}{LET execute1@247,"\Z"}~{err#1247}
           {WINDOWSOFF}{PANELOFF}{GOTO}execute3@247~{LEFT}{DOWN 2}
           {BRANCH execute2@247}
A78: [W14] '!
B78: [W10] '{rangerase@247}{back#247}{WINDOWSON}{BRANCH \y}
A79: [W14] '!
A80: [W14] 'execute3@247
B80: [W10] '{WINDOWSON}{PANELON}{back#247}{DISPATCH execute1@247}{RETURN}
A81: U [W14] '!
```

Lotus 1-2-3 Version 4 for Windows 3.X

General

With the introduction of Lotus 1-2-3 Version 4 for Windows 3.X, we can use commands such as the {MENU-INSERT}, {MENU-CREATE}, and {MENU-RESET} macro commands to create a mouse aware custom pull down menus. We can use the {CHOOSE-FILE} command and allow the user to use the mouse to select a file from a file list dialog box and more.

These commands were already available in Lotus 1-2-3 Version 1.1 for Windows 3.X with the SMARTPAK add-in. However, it did not offer a macro command that prompts and allows the user to use the mouse to select a range while the macro is running. The new macro command in 123W4 that makes it possible is the {GET-RANGE} macro command. We are not going to explain all the new macros in 123W4, but only the commands that you can use to make the macros in ***Super Power*** mouse aware.

Selecting a Range with the Mouse

Many macros in ***Super Power*** prompt the user to select a range. Therefore, to demonstrate how to use the {GET-RANGE} macro command to make the macros mouse aware, let us look at an updated version of the @ABSRANG macro.

```
     A              B                C              D              E
 1 *---A macro to CALCULATE the @ABS of all values in a 3-D range
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7       IT WILL WORK IN LOTUS 2/2.01/2.2/2.3/3/3.1/3.1+/123W1.0 to 4
 8 !
 9 !
10 \Z              {BREAKON}
11 @ABSRANG        {BRANCH win4422}
12 conta422        {WINDOWSOFF}{PANELOFF}/RNCWhich range ?~~/RND
                   Which range ?~/RNC{WINDOWSON}{PANELON}Which range ?~
                   {BS}{BS}{?}~{WINDOWSOFF}{GOTO}Which range ?~
                   {LET counterb422,0}~
13 cont422         {LET hereabs422,@CELLPOINTER("address")}~
                   {LET counter1422,0}
14 !               {FOR counter1422,0,@COLS(Which range ?)-1,1,labels1422}
15 !               {LET rel422,@INFO("release")}~{IF @LEFT(rel422,1)<>"@"}
                   {GOTO}{hereabs422}~{LET counterb422,counterb422+1}~
                   {IF counterb422<@SHEETS(Which range ?)}{NS}{GOTO}
                   {hereabs422}~{BRANCH cont422}
16 !               {GOTO}Which range ?~/RNDWhich range ?~
17 !
18 counter1422              4
19 counter1a422             6
20 labels1422      {RIGHT}{LET here422,@CELLPOINTER("address")}~{LEFT}
                   {FOR counter1a422,0,@ROWS(Which range ?)-1,1,
                   labels1a422}~{IF counter1422<@COLS(Which range ?)-1}
                   {GOTO}{here422}~{LET counter1a422,0}~
21 !
22 here422         $E$34
23 !
24 labels1a422     {IF @CELLPOINTER("type")="v"}{rnd422}
25 !               {DOWN}
26 !
27 rnd422          {EDIT}){HOME}@ABS({END}
28 !
29 counterb422               1
30 hereabs422      $A$34
31 !
32 rel422
33 !
34 range422        A:AF1..A:AG8
35 !
36 win4422         {LET rel422,@COSH(0.5)}~{IF @LEFT(rel422,1)="@"}
                   {BRANCH ret422}
37 !               {CHOOSE-ONE choices422;results422;To use the mouse
                   select OK To use Classic mode select CANCEL;You are
                   using 123W4!}~
38 !               {IF @CELL("contents",results422)=0}{BRANCH conta422}
39 !
40 ret422
41 !
42 choices422       Mouse System
43 !
44                 {GET-RANGE "SELECT THE RANGE";range422;;;}{IF @CELL(
                   "type",range422)="b"}{BRANCH ret422}
45 !               {WINDOWSOFF}/RNCwhich range ?~~/RNDwhich range ?~
                   /RNCwhich range ?~{range422}~{GOTO}Which range ?~
```

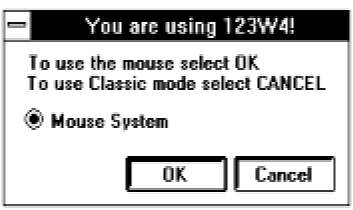
```
                      {LET counterb422,0}~
46 !                  {BRANCH cont422}
47 results422               1
```

The first change in the macro code is the {BRANCH win4422} macro command in the B11 cell that activates the [win4422] routine in the B36 cell.

```
            A              B                C               D              E
36 win4422            {LET rel422,@COSH(0.5)}~{IF @LEFT(rel422,1)="@"}
                      {BRANCH ret422}
37 !                  {CHOOSE-ONE choices422;results422;To use the mouse
                      select OK To use Classic mode select CANCEL;You are
                      using 123W4!}~
38 !                  {IF @CELL("contents",results422)=0}{BRANCH conta422}
```

The [win4422] routine starts with the {LET rel422,@COSH(0.5)}~ macro command, which stores the result of the @COSH(0.5) function in the B32 cell named [rel422]. The @COSH(0.5) function is a new function which is available only in 123W Version 4. Therefore if you are not using 123W Version 4, the result in the [rel422] cell is the "@" character. If you are using 123W Version 4, the result in the [rel422] cell is a number.

The macro continues with the {IF @LEFT(REL422,1)="@"} command. If the condition is true, the macro issues the {BRANCH ret422} command which branches to an empty routine and quits. If you are using 123W Version 4, it issues the {CHOOSE-ONE choices422;results422;To use the mouse select OK To use Classic mode select CANCEL;You are using 123W4!}~ command which displays the following dialog box:



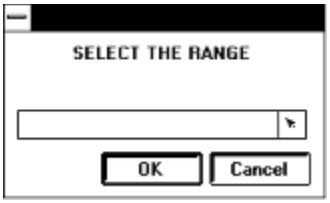
When you select OK, the macro stores the number "1" in the B48 cell, [results422]. If you select CANCEL the macro stores the number "0" in the cell [results422]. If you are a veteran Lotus 1-2-3 user, you can select the CANCEL and use the CLASSIC mode. If you are new to Lotus 1-2-3, select OK and the macro will allow you to use the mouse to select the range to process. We used the {CHOOSE-ONE} macro command to display the "Mouse System" choice. You can also use this command to display a list of choices.

When you select an option, the macro issues the {IF @CELL("contents",results422)=0} command. If the condition is true, you selected to use the CLASSIC mode. Therefore, the macro issues the {BRANCH conta422} command and branches to the [conta422] routine, which is the same as in the @ABSRANG macro. If the condition is false, the macro routes to the "Mouse System" option in the [choices422] range.

```
            A              B                C               D              E
42 choices422         Mouse System
43 !
44                    {GET-RANGE "SELECT THE RANGE";range422;;;}{IF @CELL(
                      "type",range422)="b"}{BRANCH ret422}
45 !                  {WINDOWSOFF}/RNCwhich range ?~~/RNDwhich range ?~
```

```
                    /RNCwhich range ?~{range422}~{GOTO}Which range ?~
                    {LET counterb422,0}~
46 !                {BRANCH cont422}
```

The `{GET-RANGE "SELECT THE RANGE";range422;;;}` command displays the following dialog box:

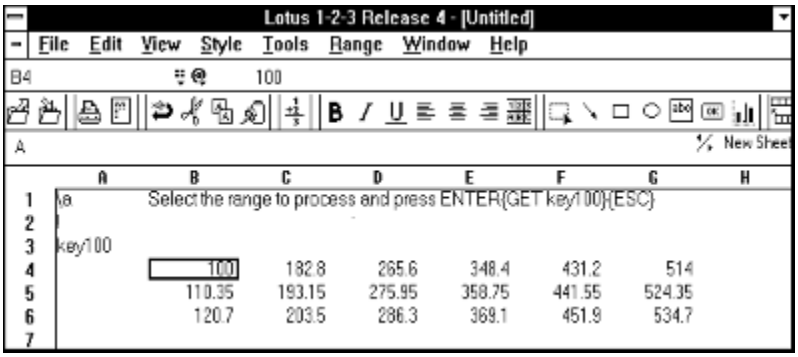


You can use the mouse to select (highlight) the range to process. The rest of the code is the same as in the @ABSRANG macro.

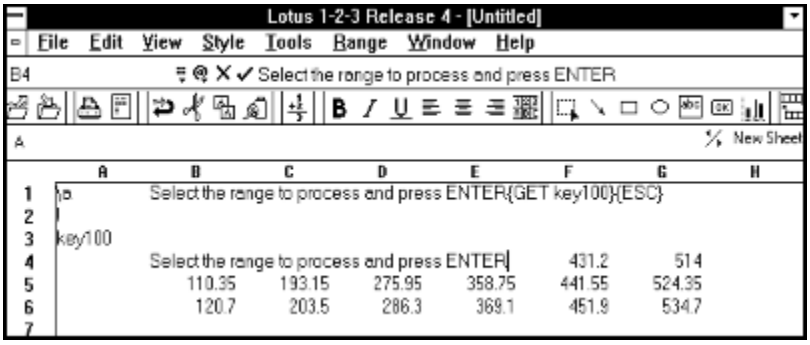
In-Cell Editing

123W Version 4 allows in-cell editing, which displays the cell's content simultaneously in the panel and in the worksheet when you enter the EDIT mode. This clutters the screen and becomes an esthetic problem. Therefore, if you are using the panel to manipulate and display text, remember to write the `{WINDOWSOFF}` macro command before any command that prints text to the panel.

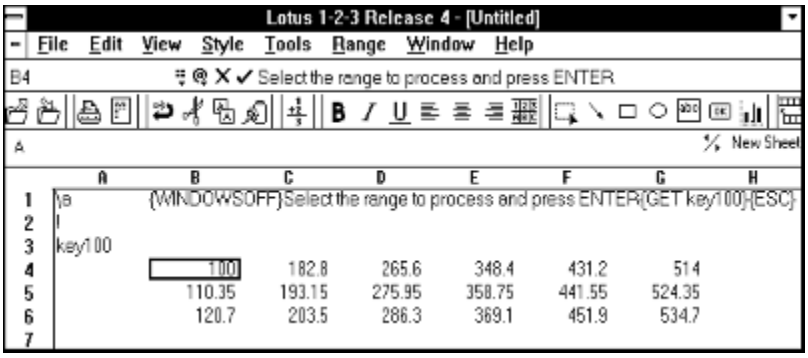
Let us look at the following [\a] macro and the data table in the B4..G7 range.



When you start the macro, Lotus displays the "`Select the range to process and press ENTER`" prompt in the panel and simultaneously displays it in the B4 cell, which hides part the data in the C4..E4 range. The following picture displays the problem.

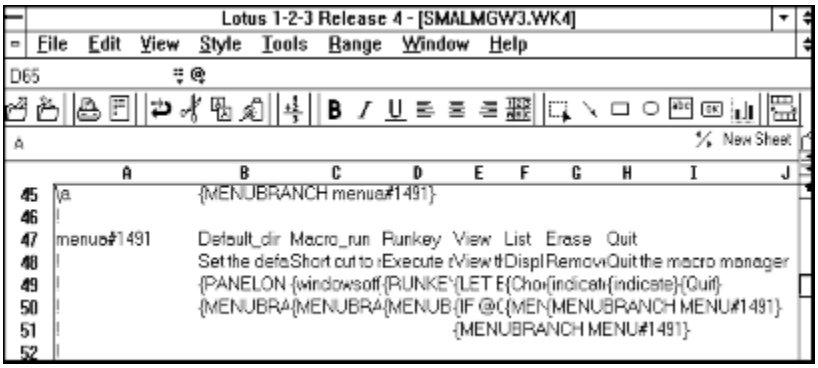


When the macro contains the `{WINDOWSOFF}` macro command before the prompt, it freezes the screen activity before the macro prints the "`Select the range to process and press ENTER`" prompt into the panel. The following picture shows the effect of the `{WINDOWSOFF}` macro command, which leaves a clean screen when the "`Select the range to process and press ENTER`" prompt appears in the panel.

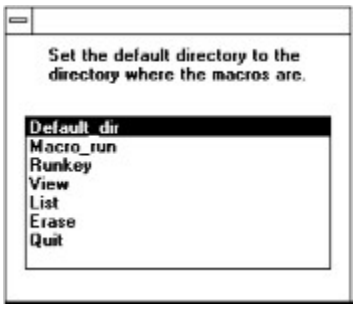

Lotus 1-2-3 Release 4 - [Untitled]
File Edit View Style Tools Range Window Help
B4
Select the range to process and press ENTER
A
New Sheet
A B C D E F G H
1 \e {WINDOWSOFF}Select the range to process and press ENTER{GET key100}{ESC}
2 |
3 key100
4 100 182.8 265.6 348.4 431.2 514
5 110.35 193.15 275.95 358.75 441.55 524.35
6 120.7 203.5 286.3 369.1 451.9 534.7
7

The Old Custom Menu Commands

When you are using the {MENUCALL} or the {MENUBRANCH} macro commands to display a custom menu, 123W4 displays a dialog box in the middle of the screen instead of a menu bar in the panel. For example let us look at the following macro:



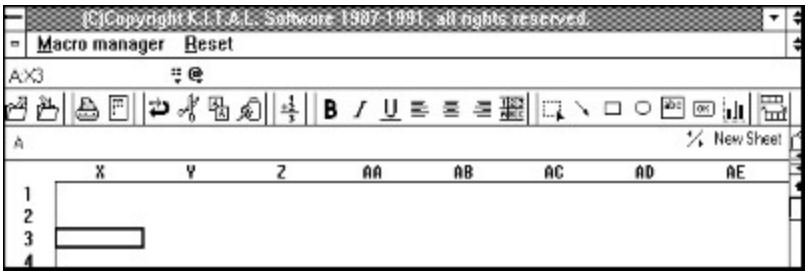
This picture displays the main custom menu of the macro manager. When the macro starts, Lotus display:



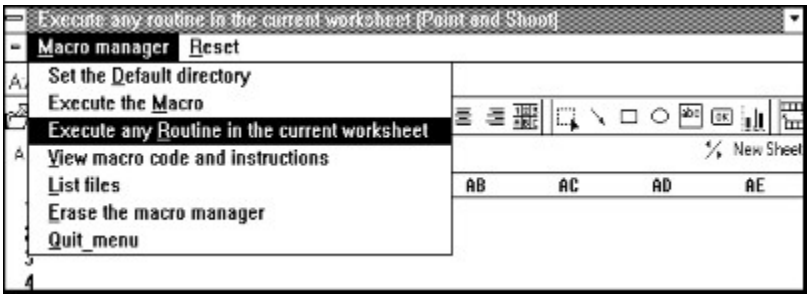
This menu box has the advantage that you can use the mouse to select a menu option. Just double click on the menu option or click on the OK button to select the menu option. You can also use the first capital letter to select a menu option and you can use the ENTER key. The disadvantage is that the menu box hides part of the screen. 123W4 allows you to drag the menu box with the mouse, but you must put the {WINDOWSON} command before the {MENUCALL} or the {MENUBRANCH} command. If you don't do it, then, when you try to drag the dialog box to clear the screen, 123W4 displays the menu box twice. If you move it again, it will display it three times on the screen and so on.

The New Custom Menu Commands

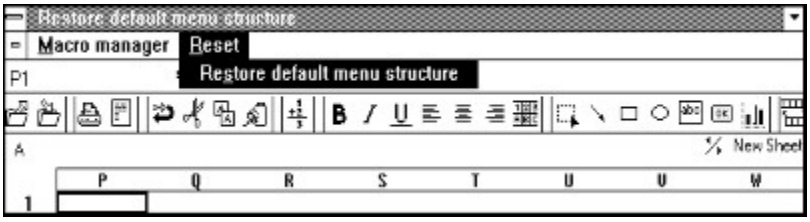
123W4 introduces new macro commands to create custom menus. Let us look at sample results of a macro that uses {MENU-CREATE}, {MENU-INSERT} and {MENU-RESET}. We used the {MENU-CREATE} macro command to replace the Lotus standard menu with the following two items menu:



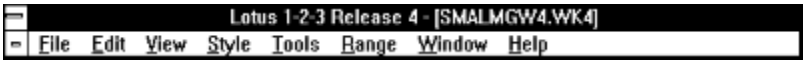
When you select the [Macro manager] menu option, the screen displays the manager's options:



When you select the [Reset] option the screen displays:



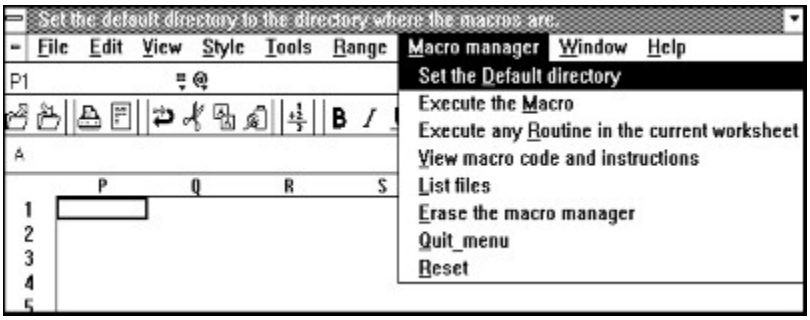
When you select the [Restore default menu structure] menu option, the macro issues the {MENU-RESET} command and Lotus 123W4 displays the standard menu:



When we use the {MENU-INSERT} to create a custom menu, Lotus 123W4 does not replace the standard menu, it adds a menu option before the [Window] menu option:

When you select the new [Macro manager] menu option, the macro issues the {MENU-INSERT}

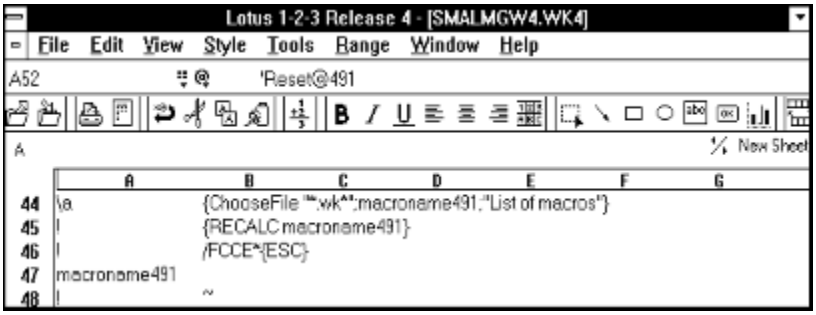
macro command and Lotus 123W4 displays:



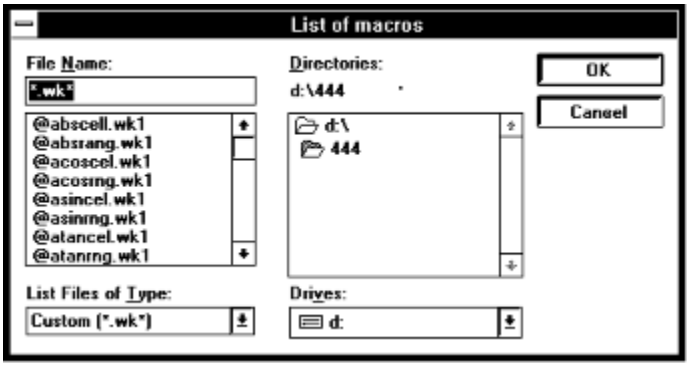
When you select the [Reset] option, the macro issues the `{MENU-RESET}` and Lotus displays the standard menu.

Using the Mouse to Select Files From a List

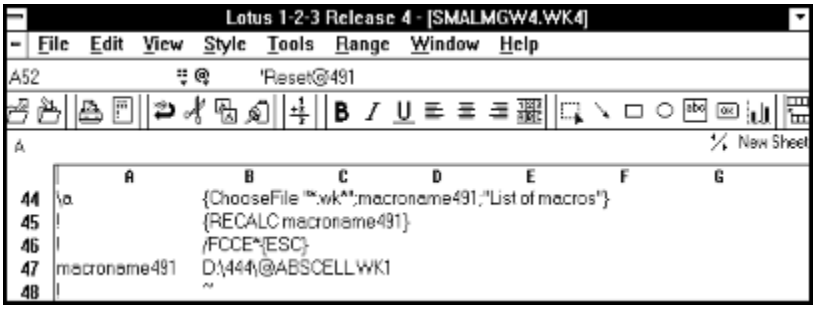
We use the `{CHOOSE-FILE}` command to allow you to select files while the macro is running. The `{CHOOSE-FILE}` command displays a dialog box that contains a list of files and waits for you to select one. Let us look at a macro that combines the file that you select from the list.



When the macro starts, Lotus displays the following dialog box:



If you select the @ABSCELL.WK1 file, the macro stores the file's name in the B47 cell, [macroname491] as shown in the following picture:



Then the macro issues the `{RECALC macroname491}` command, which updates the content of the B47 cell, [macroname491]. Last, the macro issues `/FCCE*{ESC}D:\444\@ABSCELL.WK1~` and combines the file.

ASP and Shareware

ASP OMBUDSMAN STATEMENT

This book is produced by a member of the Association of Shareware Professionals (ASP). ASP wants to make sure that the shareware principle works for you. If you are unable to resolve a shareware- related problem with an ASP member by contacting the member directly, ASP may be able to help. The ASP Ombudsman can help you resolve a dispute or problem with an ASP member, but does not provide technical support for members' products.

Please write to the ASP Ombudsman at:

ASP Ombudsman
545 Grover Road
Muskegon, MI 49442-9427
U.S.A.

or send a CompuServe message via CompuServe MAIL to ASP Ombudsman [70007,3536].

NOTE FOR SHAREWARE DISTRIBUTORS

K.I.T.A.L. Software grants you, without charge, the right to reproduce and distribute copies of the "**SHAREWARE**" version of the ***Super Power*** book on the express conditions:

1. NO fee other than a disk and handling charge (of up to $10 per disk) may be charged. The rights to receive any such financial or other benefits are exclusively reserved by K.I.T.A.L. Software.
2. NO change will be made to this license agreement or the copyright notice.
3. No change will be made to the installation and disk files, the diskette must be distributed as is, except an introduction file/s that can be added by shareware distributors.
4. If you are not an ASP approved disk vendor, you can distribute this software ONLY with K.I.T.A.L.'s written approval. Please attach your catalog.

This package comes on one 5.25" 1.2M or 3.5" 1.44M floppy diskette.

Registration

To legally try and use ***Super Power*** for more than 30 days you need to register it. If you choose to, you can order the paper bound form of ***Super Power*** too, which contains more than 1000 pages. When you register only for the electronic version that you have we send you a letter confirming your registration.

When you register for both the paper bound and the electronic versions forms of ***Super Power***, we send you only a copy of the paper bound book.

Please register me for _____ copy(ies) of ***Super Power*** electronic version which I currently have ($20 each)

_____

Please register me for _____ copy(ies) of ***Super Power*** electronic and paper bound versions and send me a copy of the paper bound book ($50 each) _____

Subtotal		$ _____
Surface Mail Shipping & Handling which may take 2 month	($7 per item)	$ _____
Air Mail Shipping & Handling which takes 8-21 days	($15 per item)	$ _____
TOTAL		$ _____

Note: Check availability of the paper bound book before sending your money.

Attach your check or cash (US currency only) and mail to:

K.I.T.A.L. Software
P.O. Box 748
Karmiel 20100, ISRAEL

For fast response call (972)-4-9987255 and order TODAY.
Please call at the evenings ISRAEL time.
All prices subject to change without notice

Appendix A - The Super Macro Library for Lotus 1-2-3

How to Order the SUPER MACRO LIBRARY for LOTUS 1-2-3?

The SUPER MACRO LIBRARY for LOTUS 1-2-3 is a large collection of more than 250 *.WK* macros that enhance and increase the power and efficiency of Lotus 1-2-3 versions 2.0 and up, and an equally large collection of over 220 *.MLB library macros for Lotus 2.2/2.3/2.4. Most of the macros are far more powerful and flexible than those found in other books, magazines and commercially available packages which you already know it if you looked at the macros in ***Super Power***. All these macros are just few key strokes away using our revolutionary Macro Library Managers (macro themselves). You need to combine only the Macro Library Manager to the worksheet, and then you use the main menu of the macro manager to start the macros directly from the disk using Point and Shoot.

The Macro Library Manager features a Quick Run mode, which displays a full screen list of all the macros in the default directory. You point to the macro name and press ENTER to start it. A Search by Keyword query allows you to find all the macros matching a keyword. When you press ENTER to start the macro, the manager automatically combines the macro, defines all the necessary range names and starts the macro. When you finish working with the macro, the manager erases it and deletes all its range names and returns to the main menu to allow you to start another macro. Of course, you can still use all the macros manually without the macro managers.

The *.MLB macros support the MACROMGR.ADN macro library manager add-in, which comes with Lotus 1-2-3 versions 2.2/2.3/2.4 and the third dimension of Lotus 1-2-3 versions 3.0/3.1/3.1+ /3.4 and Lotus 123W version 1.0/1.0a/1.1 for Windows 3/3.1. Some of the macros in this package contain new features which become available with the introduction of the new releases 2.2/2.3/2.4 to the users of Lotus 1-2-3 versions 2.0/2.01 which can now enjoy, full featured Search and Replace, and pseudo Linking.

This package contains seven Macro Managers for different cases:

- The SMALLMGR.WK1, the last macro in ***Super Power*** is about 6-7 Kbytes in size and is intended for use in memory shortage situations. This macro manager displays the full files directory list, and allows you to use point and shoot to activate the macro you need.

- The MACROMGR.WK1 file is almost 26 Kbytes in size. This manager has a Quick-Run mode, like the SMALLMGR.WK1, as well as a full featured Search by Keyword working mode, which performs a query on the files list database to look for the matched macro description. Just enter the desired string and the manager will find all the macro descriptions containing this string anywhere in the description. To move, you use the UP and DOWN direction keys. To activate the matched macro, you press ENTER. This special way to use the Lotus query to activate a macro is based on a discovery made by the developer of this package and was published in the ***PC-MAGAZINE (SPREADSHEET CLINICS) July 1988 Vol 13 page 414***.

- The SMALLMLB.WK1 and the MACROMLB.WK1 are equivalent to the previous two managers but activate the *.MLB macros instead of the *.WK1 macros, and make use of the MACROMGR.ADN add-in which you received with Lotus 1-2-3 version

2.2/2.3/2.4.

- The SMALLMG3.WK3 and the MACROMG3.WK3 are equivalent to the previous managers and activate the *.WK* macros, but they use the third dimension of the 3-D releases to keep the activated macro in a separate sheet.

- The MEW_MNGR.MLB manager is a completely new type of manager, which resides in the memory (as do all *.MLB macros) and allows you to activate the *.MLB macros in the disk using point and shoot. This is a real breakthrough in the field of *.MLB macro managers and only K.I.T.A.L. Software offers this unique approach for macro library handling. In this package we include a "compiler" macro, COMPILE.MLB that can create a manager from a simple column list of macro names which you prepare. Create as many managers as you like, it's easy. You can always use the NEW_MNGR.MLB manager, which allows you to activate all the *.MLB macros listed in the NEW_MNGR.PRN ASCII file. Or use partial lists to create smaller managers and save memory when you run short.

  Unfortunately, the MACROMGR.ADN add-in which comes with LOTUS 2.2/2.3/2.4 is limited, and does not allow the use of all the commands and techniques that are allowed in the worksheet. Therefore the number of *.MLB macros is lower than the *.WK1 macros, and you still need to use some of the *.WK* macros that we could not make work as *.MLB macros.

  All *.WK1 macros have been checked and adapted to work with the 3-D releases of Lotus 1-2-3. Few *.WK3 macros are included in the package and can be used only with the 3-D releases of Lotus 1-2-3. Many macros specifically state that they detect which Lotus release you use. These macros use the @INFO("release") function to check if you are using a 3-D release or a 2-D release of Lotus 1-2-3. If you are using a 3-D release, the macro also exploits the third dimension. Many other macros even if it is not specifically noted, also support the third dimension when they use the Lotus natural functions (like: copy, delete, move etc.). Because the 3-D releases do not support the MACROMGR.ADN add-in supplied with Lotus 2.2/2.3 and 2.4, the *.MLB macros cannot be used with a 3-D release. However, the 3-D releases can have multiple sheets and multiple files in memory, which allows us to keep macros and data in separate sheets.

  In the Super Macro Library, we have minimized the risk that two or more macros may contain the same range names. Every macro containing more than one range name was numbered by a three digit number attached to the range name. For example, a previously "loop" range name now may be "loop001" in one macro but "loop123" in other macros. The result is a foolproof package, which allows you to create large macro libraries without the danger of a range names conflict.

  If you already collected macros or own another macro library, you can change them fairly simply to work with the macro managers and enjoy the new possibilities.

Appendix B - The Scientific and Engineering Tool for Lotus 1-2-3

The SCIENTIFIC and ENGINEERING TOOL for LOTUS 1-2-3 is a completely menu driven solver and calculator which is included in the SUPER MACRO LIBRARY for LOTUS 1-2-3 as a bonus pack. It features:

1. simple arithmetic
2. functions editing, programming and calculations
3. physical units definition and conversion
4. integration and differentiation of analytic functions and data tables
5. editing and calculating statistic functions on lists of numbers
6. frequency distribution and normal distribution of data
7. curve fitting (Polynomial, Exponent, Powers, Linear)
8. root finding of non-linear one dimension equations
9. matrix and vector operations
10. simultaneous linear equation solver of up to 70 unknowns

This version works correctly inside all versions of Lotus 1-2-3, 2.0 and up, it does not work in Lotus 1-2-3 version 1.0 and 1.0a for Windows 3.0/3.1. The two previous Lotus releases for windows lost the Horizontal and the Vertical division of the screen each time the /GV or the F10 was pressed. However it works fine with Lotus 1-2-3 version 1.1 for Windows 3.0/3.1.

Statistical functions and any other functions and units are user programmable. This program can become your custom scientific handbook. You can program thousands of functions: from one variable functions to seven variable functions. If you split functions of higher numbers of variables to groups of functions of seven variables and less, you can solve more complicated problems.

Integration and differentiation is done on functions inserted from the keyboard or on data tables imported as text files. The program enables a graph of the function before and after integration or differentiation and allows you to save the numeric data tables and graphs for later use.

The distribution functions are build-in with the program and work on a list of numbers inserted from the keyboard or imported as a text file. These function produces a graph of the distribution and allows you to save the graph as a .PIC file and the data as a text file for later use.

The program already includes a large number of commonly used functions and units. You can edit them or add as much as you like. You can use this program to keep track of your worksheets, and by using a keyword instead of a function, you can activate a whole worksheet not just a function.

How to Order the Super Macro Library for Lotus 1-2-3?

Save Time and Ensure Accuracy!

Super Macro Library for Lotus 1-2-3

$50 per copy if you didn't register ***Super Power***

($35 per copy if you register or registered ***Super Power***)

Contains the latest version of the **Super Macro Library for Lotus 1-2-3** which includes all the macros in ***Super Power*** plus over 220 *.MLB macros, six other macro managers and the **Scientific & Engineering Tool for Lotus 1-2-3**, a menu driven solver and calculator, total of 1.6 Mbyte of pure macro code .

Please send _____ copy(ies) of the Super Macro Library for Lotus 1-2-3.

Check disk format: 360K 5.25" (X 3) __ , 1.2M 5.25" __ , 720K 3.5" (X2) __ ,
1.44M 3.5" __

Subtotal	$ _____
Shipping & Handling ($5 per item)	$ _____
TOTAL	$ _____

Attach your check (US currency only) and mail to K.I.T.A.L. Software. P.O. Box 748 Karmiel 20100 ISRAEL.

For fast response call (972)-4-9987255 and order TODAY.
Please call at the evenings ISRAEL time.
All prices subject to change without notice

Important Note

Sometimes it may seem that there are unnecessary commands or repetitions of the same command in the code of the macros in ***Super Power***; but from our inquiries, they are needed to cover the many versions of Lotus 1-2-3. For the macro to correctly work they are needed. In every Lotus release, there are some quirks and incompatibilities to the previous versions and different releases behave differently in the same situation. There are many examples of this. Sometimes when we look in our macros, we may forget why a specific command is there, but we are very careful not to change it before we can extensively test it in all the versions of Lotus 1-2-3.

Start Here

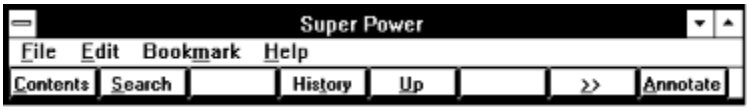
Super Power contains the equivalent of 1000 pages of a printed paper book. It is not just another book about the Lotus macro language it is completely different. See the Forward and Welcome for details.

This electronic book named 123POWER.HLP comes in the form of a Windows 3.1 help file. It was written using Word for Windows 2.0c and compiled with the Windows Help Compiler version 3.1 from Microsoft.

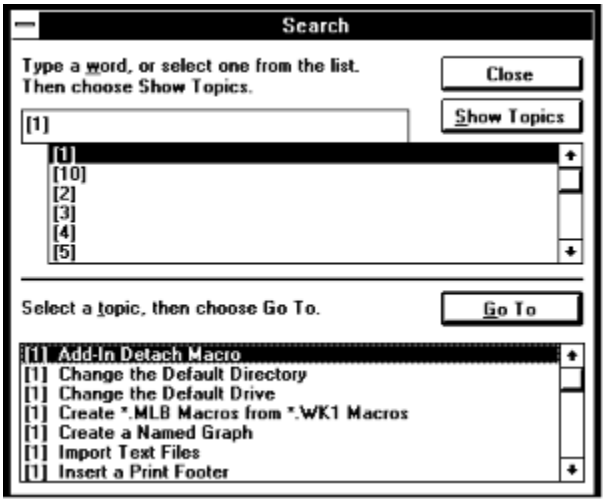
For clarity, we used different fonts and different colors for macro codes and written text. We used only fonts which come with Windows 3.1 to assure that any one who has Windows 3.1 can view ***Super Power*** correctly. For text we used Times New Roman size 11 and for macro code with used `Courier New` sizes `8` and `10`. For headings headings we used **Univers 11**,**14**,**18**.

Use the maximum window size to correctly view ***Super Power***.

Because this file is a Windows help file you can use and read it as any other help file under Windows. Therefore a green colored and underlined text indicates a jump to another topic. Just click on it with the mouse to jump to the relevant topic. The Content, Search, Back, History, Browse buttons etc. are fully operational.



For beginners, the Search button can be very useful to locate all the macros with the same level of difficulty for studying. For example, if you click on the Search button on the top of the screen and type [1] as the search string (including the brackets), windows will list all the macros of [1] level of difficulty as displayed here:



Super Power comes in two forms:

1. The first, the standard form, is a **semiautomatic** version of ***Super Power*** just as a paper bound book. You cannot activate a macro from inside a paper bound book, can you?

2. The second is a **sutomatic** version of ***Super Power*** which allows you to start Lotus and the macros that you currently read just by clicking with the mouse on the green underlined macro name.

The second form is a bonus for readers who also purchased the SUPER MACRO LIBRARY for LOTUS 1-2-3, which allows them to enjoy the fact that they have all the macros featured in ***Super Power*** and more, which also saves them many hours of keying the code into Lotus 1-2-3.

If you decide to purchase the SUPER MACRO LIBRARY for LOTUS 1-2-3 then you will also receive the **automatic** version of ***Super Power*** that allows you to launch Lotus and the macros you currently read just by clicking with the mouse on the macro name as you will see in a moment. The following is part of the code of the SWITCH.WK1 macro

```
     A              B                C              D              E
 1 *---A macro to switch 3-D and 2-D ranges places
 2 *---Use the /Range Name Label Right [End] [Down] [ENTER] to define the
 3     range names in this column (starts with the \Z macro name)
 4 *---Hold the [ALT] key and press [Z] to activate the macro
 5 !
 6         THIS MACRO AUTOMATICALLY DETECTS THE LOTUS 1-2-3 RELEASE
 7                     IT WILL WORK IN LOTUS 2.0 AND UP
 8 !
 9 !
10 \Z              {BREAKON}
11 SWITCH          {LET rel209,@INFO("release")}~{RECALC loc209}
                   {RECALC form209}
12 form209         {LET here1209,@CELLPOINTER("coord")}~
13 !               {WINDOWSOFF}{PANELOFF}/RNCFirst range ?~~/RND
                   First range ?~/RNC{PANELON}First range ?~{WINDOWSON}
```

When you click on the green underlined SWITCH text, windows starts Lotus 123W for Windows and retrieves the SWITCH.WK1 file if it is located in the Lotus default directory, so you can experiment with the actual file while you are reading the book.

To try this now:

1. Quit ***Super Power***.
2. Copy or move the 123POWER.HLP file to the Lotus 123W directory.
3. Create any file named SWITCH.WK1 and place it in the Lotus default directory. If you are not sure which is the default directory, start 123W and press / **W**orksheet **G**lobal **D**efault **D**irectory ... to see the current default directory.
4. Start ***Super Power*** again (double click on the 123POWER.HLP file name).
5. Click on the green underlined SWITCH text.

In no time you will see Lotus 123W with the SWITCH.WK1 file loaded. When you quit Lotus 1-2-3 you return to ***Super Power***.

If you do not own Lotus 123W but own a DOS version of Lotus 1-2-3 you can still launch the macros but it demands more steps:

1. Quit ***Super Power***.

2. Copy or move the 123POWER.HLP file to the Lotus 123 directory.
3. Create any file named SWITCH.WK1 and place it in the Lotus default directory. If you are not sure which is the default directory, start 123W and press / **W**orksheet **G**lobal **D**efault **D**irectory ... to see the current default directory.
4. Use a pure ASCII editor such as EDLIN to create a batch file, which contains the following command line:

   ECHO %1>C:\MACROS\SUPER.PRN

   The C:\MACROS is the Lotus default directory. If your default directory is different, types it instead of the C:\MACROS. For example, if your default directory is C:\123\DATA, the batch file should include:

   ECHO %1>C:\123\DATA\SUPER.PRN

5. Save the file with the SUPER.BAT file name in the Lotus 1-2-3 directory.
6. Create the following Lotus macro:

	A	B	C	D
1	\0	{GOTO}B3~/FITSUPER~		
2		/FR		
3				
4		~		

7. Type the macro in the A1..B4 range as you see here.
8. Place the cell pointer on the A1 cell which contains the \0 (backslash and zero) and issue: / **R**ange **N**ame **L**abel **R**ight and press ENTER to assign the [\0] range name to the B1 cell.
9. Save it to the Lotus default directory and name it as AUTO123.WK1.
10. Start ***Super Power*** again (double click on the 123POWER.HLP file name).
11. Click on the green underlined \Z text.

In no time you will see Lotus 1-2-3 with the SWITCH.WK1 file loaded. When you quit Lotus 1-2-3 you return to ***Super Power***.

The automatic book comes with the AUTO123.WK1 and the SUPER.BAT files. You need to edit the SUPER.BAT according to your default directory.

About The Author

Israel Kehaty received his B.Sc. in aeronautical engineering from the Technion Institute in Israel in 1972. He is currently working as a Senior Engineer for one of the largest companies in Israel. Israel has 20 years of experience working with computers for scientific and engineering applications. Since 1986 he has developed some shareware applications for IBM PC's and since 1993 he is a member of the Association of Shareware Professionals (ASP). Israel is also the developer of the expert system for mechanical spring design THE ULTIMATE SPRING DESIGNER. In 1986 During his sabbatical leave in Oregon state university, Israel has got acquainted with Lotus 1-2-3. Since than he is using 1-2-3 as a scientific and engineering tool for his engineering work, and develops general macros as a hobby. Israel is currently a Lotus Authorized Consultant (LAC) for the Israeli branch of Lotus Company. He has developed the SUPER MACRO LIBRARY for LOTUS 1-2-3 on which ***Super Power*** is based.

Legal Matters

Important: These macros should work in other Lotus 1-2-3 compatible spreadsheets (but not guarantied), such as QUATTRO and QUATTRO PRO from Borland International Company but not limited to. Trying to modify these macros and distribute them as originals, using minor changes such as QUATTRO-PRO's MENU EQUIVALENT commands instead of the keyboard macro keys, is an infringement of K.I.T.A.L. Software's copyright. K.I.T.A.L. Software already created and reserves the copyrights to such versions of the macros listed in ***Super Power*** which will be available to customers in the future.

SUPER POWER (THIS BOOK) AND THE ACCOMPANYING SOFTWARE IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, RESPECTING THE CONTENTS OF ***SUPER POWER*** AND THE ACCOMPANYING SOFTWARE, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES FOR THE BOOK'S QUALITY, PERFORMANCE, MERCHANABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. NEITHER ISRAEL KEHATY NOR K.I.T.A.L. SOFTWARE NOR ITS DEALERS OR DISTRIBUTORS SHALL BE LIABLE TO THE PURCHASER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS, OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY ***SUPER POWER*** AND/OR ITS ACCOMPANYING SOFTWARE.

If you purchased ***Super Power*** you are allowed to key the macro codes from ***Super Power*** into Lotus 1-2-3 and save them as Lotus files for your personal use ONLY. You are NOT ALLOWED to distribute them or use them in commercial applications that you develop and sell. The macros listed in ***Super Power*** are macros copyrighted by K.I.T.A.L. software and used in ***Super Power*** with K.I.T.A.L.'s permission.

You MUST treat this electronic book the same way you treat a paper bound book. You can use it only on ONE CPU and on ONE VIEWING MACHINE at the same time. Therefore you cannot install it on ANY KIND of a network without purchasing a site license and you cannot install it on more than one personal computer.

If you value your time and want the full power of K.I.T.A.L.'s macro library, you can use the order form for the SUPER MACRO LIBRARY for LOTUS 1-2-3 to order it directly from K.I.T.A.L. Software, instead of investing your valuable time keying the macros into Lotus 1-2-3.

Trademark Acknowledgment

{BREAKON}

Almost all the macros in ***Super Power*** start with the do nothing {BREAKON} macro command in the B10 cell. The reason is connected to the way the macro manager works. It always looks for the [\Z] range name to assign the range name for the combined macro. Often the macro loops back to the beginning of its code. If the loop is explicitly to the [\Z] range name, you will not be able to change this name to [\A] for example. Therefore the loop is always to the macro name itself such as {BRANCH LOWRCASE} instead of {BRANCH \Z}. Sometimes you may see the {BREAKON} macro command twice and it concern the way the macro is used as an *.MLB macro in the SUPER MACRO LIBRARY.