ForCALC

Omnibus™

spreadsheet for the HP48SX/GX

User's guide

This manual was produced with Ventura Publisher® and a Canon BJ300® Bubble Jet Printer.

© is a typeface created by ForCALC.

Printed in Italy
First Edition

Contents

Omnibus user's guide

Acknowledgments

ForCALC gratefully acknowledges Dr. C. Patton and Dr. P. Swadener of the HP Corvallis division for their support and encouragement, Mr. M. Papazzoni for his help in the testing process and all the people around the world who trusted in this project.

Introduction

Omnibus is the programmable spreadsheet created by ***ForCALC*** for the HP48 family of computers. In addition to the spreadsheet program, the card provides a rich set of programmable commands and functions, plus a couple of programs exploiting Omnibus sophisticated user-interface.

This product required a great effort in terms of development and debugging. We do not grant it is perfect, because perfect software does not exist, but we grant that we will support you in case of problems.

The key of the success of Omnibus is in your hands. Sharing your experience with other customers is mandatory. By filling in the enclosed card and returning it to ***ForCALC***, you gain a chance to enter in touch with other users and you will receive updated information about our products too.

Requirements

Omnibus is a complex application and, at the same time, a programming environment with its own rules and idiosyncrasies. Learning a correct approach to the problems you must deal with is essential.

This manual is structured in several chapters. The first chapter leads you to through the options and features Omnibus offers without assuming any particular skill. The only indispensable requirements are the curiosity and a good understanding of HP48's workings and RPL principles. If you are a neophyte of the HP48 world, we suggest to become acquainted with the HP48 itself before diving into Omnibus waters.

Once you have read and understood the information given in the first chapter, you are ready to start working on your own. Keep into consideration that there are five additional chapters covering Omnibus in any aspect. The index may also help you to locate certain topics.

The chapter titled *Practical examples*, at the end of the manual, teaches you how to build well structured worksheets and, at the same time, shows you how to solve typical problems in finance and statistics.

Amongst the many powerful features of Omnibus is the ability to switch back and forth from other applications like the *Equation writer* or the *interactive stack*, therefore a basic knowledge of their usage is recommended, yet not indispensable. For further details about these topics, you should consult the *HP48 User's Guide*.

Overview

Omnibus has been designed to cooperate with the software developed by others. This design strategy gives you a lot of advantages respect to stand-alone programs:

- ☑ *Brand new functions and programs (like those explained throughout this manual) can be added at any time.*
- ☑ *Software developed by other parties can run inside Omnibus.*
- ☑ *Programming Omnibus is like programming the HP48. You don't have to learn yet another language, but only a few keywords.*
- ☑ *Applications written for Omnibus share only one consistent and flexible user-interface.*

Of course we made of our best to keep the highest degree of compatibility with third-parties products, but we *cannot* guarantee that *any* software will run without adaptation. Most of the products have been designed as stand-alone modules with no possibility of interaction with other programs. Notwithstanding Omnibus should be able to work properly with almost any other RPL program without the need of modifying either.

Omnibus concentrates a lot of capabilities in a small library thanks to the presence of the ***SmartROM***. Omnibus is an evolution of the Symbolic Matrix Writer Library that was formerly provided with the SmartROM card. The upgrade from the Symbolic Matrix Writer to Omnibus is painless because the data structure and the usage are 100% compatible, furthermore you can configure the spreadsheet to work like the Symbolic Matrix Writer.

The powerful utilities provided with the package should give you an idea of the potential of Omnibus. They are written taking advantage of the language extension provided with the package and are customizable by the user.

Introducing the concept of spreadsheet

In the following pages you will be introduced to the world of Omnibus. You will learn that what your HP48 can do is really much more than what you expected when you bought it.

What can really do Omnibus ?

Omnibus combines the functions of spreadsheet, agenda and data bank in one highly integrated application program based on list processing.

Despite of this verbose definition, Omnibus lets you manage data with unprecedented ease so that, in effect, you can forget at all what the internal data representation is.

If you are new to the world of the spreadsheets, you must first become acquainted with the concept of spreadsheet.

Early spreadsheets were designed to make statistical and financial analysis easier, more effective and interactive. Beside this capability, today's spreadsheets borrow a lot of features from the world of databases, and Omnibus represents the implementation of this software running on the HP48.

Basically the structure of a spreadsheet consists in a two-dimensional table subdivided in *cells*. A straight line of cells is called *row* if taken horizontally, and *column* if vertically. The

spreadsheets of the last generation lets you work with 3D data models, but for 'normal' purposes two dimensions are still enough.

The spreadsheet allows you to link cells each other by means of mathematical relations or other criteria. As soon as you change a value in a cell, all the cells whose values depend on it will be automatically recalculated.

For instance, take a look at this chained expression:

0 Previous+1 Previous+1 ... Previous+1

This is the easiest method to create a succession of natural numbers (0,1,2,3...n) starting from 0. The relevant property of a spreadsheet is that by changing the initial value, chained values change as well, so that by storing 5 in place of 0, the succession becomes (5,6,7,...n+5).

This property become even more interesting when you add links with other expressions to the original chain.

0	Previous+1	Previous+1	...	Previous+1
Upper^2	Upper^2	Upper^2	...	Upper^2

In this way you get two linked successions (naturals and square of naturals) in a straightforward fashion.

0 1 2 3 ... n
0 1 4 9 ... n^2

The way in which a cell is referenced is one of the aspect that makes the difference between Omnibus and other spreadsheets. Usually spreadsheets use numeric counters for rows and

alphabetical counters for columns, so that the upper-left cell is uniquely identified by the pair **A1**. This scheme is quite rigid and the legibility of the expressions is poor.

Omnibus employs numerical counters for both dimensions but also provides a set of special identifiers that greatly simplifies the most frequently used reference patterns. You will discover that you can address a cell in many different ways, each one optimized for a specific task. Since the way Omnibus treats cells is like that shown in the examples above, the legibility of the formulae results greatly improved and more intuitive.

Beyond the spreadsheet

As mentioned earlier, Omnibus is not just a spreadsheet. You can easily create databanks, phone-lists, inventories, price-lists and so on. You can quickly retrieve a piece of information thanks to the programmable search and retrieval functions. Note that you can even store graphics objects into cells and display or print them with a single keypress.

Omnibus improves the management of time information by adding housekeeping objects and commands to deal with times and dates. The powerful calendar application integrated with the built-in appointment system of the HP48 recognizes holidays of more than twenty countries and makes Omnibus the unique product of his class (and also of higher classes) with such ability.

The power of Omnibus is not limited to the built-in features: being fully integrated with the HP48 operating system and language, Omnibus recognizes and accepts any software extension. More functions, more programs, more power at minimal cost with a unique and consistent user-interface.

Several examples are given throughout the manual; they will show you most exciting features of Omnibus and are conceived to represent a template for the creation of any other function or application. Since they span from business applications to electronics, we hope you will appreciate the effort of showing a broad range of possibilities rather than repeating a course of statistics or finance.

Installation of the card

The card can be installed in either port of every version of HP48, including the HP48GX. Of course, you cannot run it on models like the HP48S or the HP48G because they are not expandable.

If you install the card in slot 2 of a HP48GX, you will notice a degradation in the performance. This is due to the overhead in the management of objects stored in the supplementary ports numbered 2 through 33. Moreover, in order to run any program stored on the card, there must be enough free memory to contain it, plus the additional space required by the program to work. When you employ the card this way, there should be at least 60K bytes of RAM free.

To install the card in the slot, follow the procedures described in the paragraph *Installing memory cards* in the *HP48 User's guide*.

All the libraries contained in the card will be attached to the HOME directory with the exception of the library of messages. To install this library with the preferred language, please follow the instructions given in the next chapter.

Glossary

This brief glossary collects terms used throughout the manual that may sound new to somebody.

Anchor cell A cell which shall be used as reference for subsequent operations.

Cell The place where you keep an object inside Omnibus.

Cell mode When Omnibus is running and the command line is *not* active.

Command mode When Omnibus is running and the command line is active.

Event A list made out of date and time. Date and time can be either real numbers or custom objects.

File A *worksheet* with embedded configuration data. This object is always displayed as Library data on the stack.

Function A program or expression that can be refereced by name in algebraic objects, whose syntax is:

NAME(arg1,arg2,...,argn).

Holidays planner An application named **HP** which allows you to create the holidays database.

Hot key A key whose action is performed immediately. Key assignment is performed by the utility **HOTASN**.

IconFlag An icon-based menu that allows you to change the configuration flags of Omnibus.

Idle mode When the HP48 is not running a program or editing objects in the command line. The HP48 is idle also when *cell mode* is active.

Mini help A brief explanation of the menu commands of Omnibus, given in the language chosen during the installation. It is activated pressing [↰] [▼] in cell mode.

Operator A *function* with a special syntax like:

arg1 **NAME** arg2 *(infix notation)*
arg1 **NAME** *(postfix notation)*
NAME arg1 *(prefix notation)*.

Page The area of cells that fits the display window.

Reference The link between two cells.

RPL The programming language of the HP48 series of calculators.

Worksheet The set of cells being loaded into Omnibus.

Zoom menu An internal menu of Omnibus that allows you to scroll one page at a time.

Working with Omnibus

This chapter teaches you how to:

- ☑ *install and configure Omnibus;*
- ☑ *run the spreadsheet;*
- ☑ *maintain worksheets;*
- ☑ *create references;*
- ☑ *design macro functions;*

Configuring the software

The examples and the procedures described in the manual work on the assumption you have installed the software properly. The installation of Omnibus is subdivided in two phases:

1 plugging the card in the slot;

2 running the installation program.

The installation program is named WELCOME and is stored on the card. WELCOME is the last label of the port menu.

► To run it, just press the menu key and respond to the prompts of the program.

The first time you run WELCOME, you will be asked to choose one language. Using the arrow keys [▲] [▼] move the cursor on the desired language and confirm your selection by pressing [ENTER]. If you try to abort the language selection with [ON], WELCOME will restart the program until you choose one language.

At this point, WELCOME will inform you that the calculator must be turned off.

- Press any key and after that the HP48 has been turned off, wake it up again.
- Run WELCOME again, henceforth respond **Y**es to all the questions.

You will be asked to select one country.

If you quit the program by pressing [ON], WELCOME will skip over the calendar creation. However, at this stage, it is recommendable to select a country.

Later, you will be asked to select the time format. You can leave it as it is by pressing [ON] or choose a different format by pressing [ENTER]. The marker is placed near the current format.

At this point, if a country has been selected, begins the creation of the calendar.

Then WELCOME asks whether you prefer to assign some applications to keys: Respond **Y**es and proceed with the configuration.

Finally the program asks whether you want to assign shortcuts: choose **Y**es.

The program terminates and displays the flags configuration menu.

☞ The first time you install the card, you had better to leave the configuration of flags as it is.

WELCOME presents many alternatives that may seem worthless to the neophite. You will appreciate them at a later time. Every time you want to bring Omnibus back to a known condition, you can run the installation program. If you deem that some configuration is valid, you can skip over it.

If you want to change the current language, you must call the utility LANG.

If you arrived at this stage without errors, you have completed successfully the installation of Omnibus.

A first glance at Omnibus.

► Go to the port menu and run CALEND.

If you remember the key to which you assigned this application, press it right now.

the calendar application comes up quickly with the cursor on the name of the current month; now if you press [EVAL] the calendar appears and the cursor is placed on the current day.

Move the cursor around and finally place it on a day.

► Press [EVAL] again.

The name of the day and the current date will appear in the bottom line of the screen. The name of the day as well as the the name of the month shown earlier are in the language you selected.

► Press an arrow key and move around for a while.

If you keep the arrow key pressed, you will notice a fast repeating movement of the cursor.

► Press [→] and hold down [4].

The current time and date will appear in the status line. This hot key is always assigned, even when you are typing things into the command line.

► To exit, hit [ON] twice.

You terminated succesfully the first round with Omnibus.

Running the spreadsheet

Unlike the HP MatrixWriter, Omnibus can be started inside a user-program. This capability allows programmers to design powerful applications in which users may enter data or make selections in a comfortable and protected environment. Many utilities stored on the card have been written using Omnibus as a *data browser*.

The calendar program, you ran a while ago, is an example of such capability. In the chapter titled *Time management* you will find a detailed description of the features provided by the calendar and we encourage you to jump there soon if you are curios to learn more about this topic.

Omnibus is started by the command **MATWRT**. The name of the command is inherited by the early release of the library and has been maintained for compatibility.

► {{}} **MATWRT**;

When you enter this command, Omnibus is started from scratch.

If this command causes the error Can't edit null array, ***clear flags 34, 35 and 36 and try again.***

The number of columns in the display depends on the current setting in the HP MatrixWriter.

► Check the rightmost label of the menu and press the menu key until a white dot appears (EDIT).

► press 1 SPC 2 SPC 3 ENTER.

The numbers shall fill in the cells vertically.

Omnibus can be employed to create numeric or symbolic arrays in a format suitable for SmartROM's commands.

You can configure flags to operate Omnibus in this mode by default. In this case Omnibus expects only numbers and symbolic expressions as input. Entering objects like strings, lists, dates or any other forbidden object will be interpreted as an error condition; a warning beep will be issued and the offending object will be substituted with a question mark (?).

► write "HELLO" ENTER

You should hear a double beep and a questionmark should appear in the cell.

☞ To change the operating mode of Omnibus you need to switch flag 33 on.

► press NXT until you reach the last page of the main menu.
► choose CONFI and turn the menu labeled [☐] into [☑]

When flag 33 is on, Omnibus accepts *any* object type.

► move the cursor back to the cell containing the questionmark;
► press ↰ CMD ENTER;

Now the string HELLO should appear.

Since user flags are merely identified through positive numbers in the range 1-64, we provided a graphical interface that simplifies the configuration of the program. This icon-based user-interface is always referred to as *IconFlag*.

To get a brief explanation of each menu entry, press ↰ ▼. The message is given in the currently selected language. This information is *missing* if the library containing the localized messages has been removed, detached or not installed as described earlier.

☞ User flags numbered 32 trough 45 have been allocated for use with Omnibus. The scope of these flags is limited to the environment of Omnibus. System flags affect the HP48 globally and Omnibus as well. IconFlag collects all the flags which exert a direct influence on Omnibus.

➤ quit Omnibus by pressing ON.

Data types

MATWRT accepts and returns three different types of input objects such as numeric arrays, double lists and custom library data (also referred to as *files*). The argument may contain only data to work on, or configuration parameters for Omnibus (in addition).

The array type can be either one-dimensional or two-dimensional, real valued or complex valued. When you need to store values in a compact and efficient format, the arrays are the best choice. On the other hand you cannot add comments or save configuration parameters because the array structure is tailored for raw numbers. Note also that Omnibus returns always two-dimensional arrays.

A double list (a list of lists) is a data structure that allows the coexistence of objects of different kinds at the same time. Formally a double list is merely a list object, but Omnibus is able to recognize and reject an invalid list. It is very unlikely you will ever be concerned with the making of such objects by hand, since Omnibus is the best tool around for this purpose. A brief explanation about the structure of the double list is given in the SmartROM's command reference manual, under the entry **MATWRT**.

You will find that the double list structure, empty or not, is frequently referred to as the *worksheet* throughout the rest of the manual.

When you start Omnibus using {{}} as argument, the program loads default configuration settings.

Default values are stored permanently in the *HP48* at specific locations and are shared between Omnibus and the HP MatrixWriter. System and user flags are shared with any application that makes use of them. This means that when you change the column width inside the MatrixWriter, the default column width of Omnibus will change as well. Conversely when you narrow or widen columns inside Omnibus, the change does not affect the MatrixWriter, because Omnibus saves and restores every time the previous configuration.

In this manner unsupported configuration parameters do not interfer with the built-in MatrixWriter. On the other hand you can change the default values programmatically.

The command **SetWidth** changes the default column width permanently. This command can be executed inside Omnibus too, but in this case, the original column width will be restored upon exit.

The range of values allowed by Omnibus is greater than that permitted by the HP MatrixWriter hence you must pay attention when changing this parameter.

To be sure that the column width falls within the valid range, issue the following command:

n **SetWidth**

where n is a real, integer number in the interval 1-5.

The configuration of Omnibus is also affected by the state of certain *user flags*. Omnibus saves and restores the state

of the flags, so that, under normal circumstances, the execution of the program does not interfer with other applications. When you load a library data object, the current configuration of the *HP48* is worthless because Omnibus employes the configuration stored in the *file*. Conversely, when you start Omnibus from scratch, the current configuration is loaded and certain combinations of flags may prevent the program from running.

When you encounter the message Can't edit null array, ***you have at least one of the user flags 34, 35 and 36 on. Turn these flags off and retry the operation.***

By default Omnibus does not save the accessory information together with data. You can enable the configuration saving option by turning flag 33 on.

Since you may prefer to resume working from where you left off, Omnibus implements a special data structure called *Library data* for saving configuration information and user data in a unique object. Omnibus records the cursor position, the state of flags, the number of columns and other parameters plus the current worksheet into a single data structure.

Therefore when you work with such objects, you can disregard the current configuration of the *HP48*.

If you installed SmartROM's custom menu by executing **CSTMENU**, *you will find that the last label of the second page contains the following spreadsheet icon*  . *By pressing the corresponding key you can start Omnibus from scratch. If you press*  , *Omnibus loads the object on the first level of the stack, provided that it is a valid argument.*  *automatically creates a backup*

copy of the input worksheet which will be discarded only if you confirm the changes by pressing ENTER.

When working in low memory conditions, this procedure may take too much room than that actually available. In order to accomplish this operation you need at least three times the space occupied by the object you are working on.

Storing a worksheet

Once you have created a worksheet, you can store it into a directory assigning the name you prefer.

To load again the worksheet into Omnibus, you must first recall the object to the stack, then execute the command **MATWRT** or perform the keystroke described earlier.

We suggest you to choose some convention in the assignment of names, so that you can easily distinguish Omnibus worksheets from other variables. Alternatively you could collect Omnibus files into a specific directory or even create a whole branch of directories dedicated to Omnibus.

Anyhow remember that Omnibus files (library data objects) can be easily selected by means of the statement:

26 **TVARS**

that returns a collection of global names, taken from the current directory, containing Library data objects. With the exception of variables like MHpar and PTpar, which are created by HP applications, all other variables should be Omnibus files. Passing an invalid library data object to Omnibus results in a Bad Argument Type error.

Throughout this section we will talk of remote computers or host computers; in the majority of cases this means

IBM PC and compatibles, notwithstanding the procedures described may apply to Amiga, Macintosh or Unix systems connected to your calculator through the Kermit protocol.

Uploading a worksheet to a host computer

The procedures described in this section are meant for making a backup copy of your data on a remote computer. The suggestions are given on the assumption that you know how to make a reliable connection between your *HP48* and the remote system.

☞ In order to share the data of the worksheet with an application running on the remote computer, you must convert the data into an ASCII string. This operation is not the same as doing a backup copy. See on page 195 for the details.

As long as a worksheet has been stored in a directory, you can upload it to a host computer using the Kermit transfer protocol[1].

When you choose a name for a worksheet, you must take into account the restrictions imposed on file names by the remote file system.

☞ The file name extension may be not supported on some remote file systems. Moreover, remote file names may be shorter than those accepted on the *HP48*, that allows names up to 255 characters long.

1 We preferred to speak about the Kermit protocol because it is implemented on both models of *HP48*, but you could use Xmodem on the *HP48GX*.

A short name (6-8 letters), consisting of uppercase letters only, will be accepted by most file systems.

If a name is too long to fit the requirements of the file system, it may be truncated. If the file being transferred conflicts with the name of an existing file, depending on your settings, it may happen one of the following things:

- the connection breaks due to an error
- the new file is discarded
- the old file is overwritten
- the new file is renumbered

In either case what you get is different from what you expected, therefore you must be aware of the drawbacks your decision implies.

For the details concerning file transfers, please refer to the RPL commands **SEND**, **RECV**, **RECN** in the the *HP48 User's guide*. You may also run the utility SENDMNGR, stored on the card, that allows you to select the files to upload, eventually renaming some of them.

A worksheet created with Omnibus must be transferred using the binary transmission mode. Failure to do so, may result in the partial or total loss of the data being transferred.

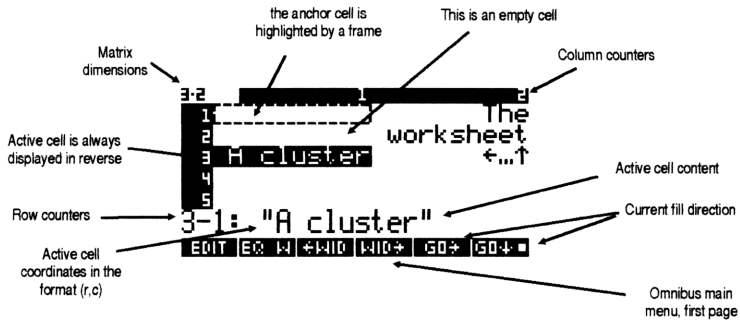
Note that the binary transmission mode saves tranfer time and disk space. Since the binary transmission mode uses *HP48*'s internal representation, you cannot retrieve data from a backup file using an ASCII file editor. If you need to transfer data in ASCII format, you must export the worksheet as a string; see the description of the utility EXPORT given on page 195.

Downloading a worksheet

Downloading a worksheet saved on a remote computer is as easy as transferring any other file to the *HP48*. Please refer to the appropriate section of the *HP48 User's guide* for further details.

Keep flag -51 (fraction mode) clear to avoid name conflicts.

When this flag is set, period marks used in DOS file names extensions are converted into semicolons and cause a parsing error. In order to prevent such annoying problem, keep it clear.



Basic operations

To save time later, we suggest you to become familiar with the definitions given throughout this section because these terms are used a couple of times in the rest of the manual. Figure 0 allows you to visualize all the new terms being introduced.

As you can see, the graphic user-interface of the built-in Matrix Writer has been maintained, though we added a lot of features. Most of these features have been implemented by redefining the keyboard and extending the main menu. Although we made of our best to keep a

logical link between the original key label and the new definition, the position of a few keys is arbitrary: please check the layouts of the keyboards given at the end of the manual to locate the desired key.

The cell underneath the cursor is displayed in reverse. This cell is named *active cell.*

The active cell is the place where a new object goes when you enter it from the command line. The content of the active cell is echoed in the bottom line of the display using big characters and its position *row-column* is prepended to the text. Note that you can switch the sixth row of cells on or off by toggling the menu key / in the second page of the menu . When the sixth row is on, the echo of the active cell is suppressed.

By means of the arrow keys [▲] [◄] [▼] [►] you can move the pointer from cell to cell across the worksheet. Prepending [↵] to an arrow key, you can leap at the worksheet boundary. At this point you can move the cursor only one cell farther, provided that the worksheet has not been locked up.

When you reach the boundaries of the display window, the worksheet is scrolled if more items are available, until you reach the line[1] beyond the physical end of the worksheet otherwise a beep is issued. To know exactly how large is the worksheet, watch at the upper left corner of the display where the dimension of the matrix are shown with small characters. Under normal circumstances the pointer can be moved up to one line

1 The term *line* means indifferently a row or a column.

beyond the last defined line; in this case the active cell will be a *hollow* cell. However watching the picture above you may see something called *empty* cell; Empty cells differ from hollow cells because their space has been allocated. Empty cells consume a minimal amount of memory and they are spawned every time you place an object into a new line[1].

By pressing [↰][▼] you can view a short commentary about the main menu functions currently available. For each page of the menu there is a corresponding screen of reminders. [↰][▼] works also for Omnibus sub-menus.

In addition to the active cell, you can also set an *anchor cell* by pressing the key [·] on a non-hollow cell, provided that *cell mode*[2] is enabled. As soon as you press this key, a dashed frame surrounds the anchor cell, which is also the active cell at the time being. If you move the cursor, the frame remains "anchored" to the original cell.

The anchor cell serves for many purposes:

- combined with the position of the cursor delimits a rectangular region of cells to form a cluster;
- allows you to copy the content of the anchor cell elsewhere;
- combined with the position of the cursor determines a horizontal/vertical/diagonal direction for the automatic reference builder;
- combined with the position of the cursor restricts sort/arrange commands to a range of cells;

1 When the symbolic matrix writer emulation is enabled a new line is filled with zeros rather than with empty cells

2 See later on for a definition of cell mode

- allows you to quickly return to the cell with the function GO→A of the sort menu.

When you extract a portion of a worksheet to the stack, you deal with something called *cluster*. A cluster can be either a whole worksheet (in the form of a list) imported from somewhere or a clipping of the current one. In effect there is no practical difference between a cluster and a worksheet but there is a logical distinction because the latter usually contains something that cannot be used as it is.

The anchor cell remains active until you explicitly delete it by pressing DEF. However its position may become invalid due to the deletion of a certain number of rows or columns, in that case its position is ignored or rejected with a beep.

There are several operations involving the anchor cell, either in *cell mode* or in *command mode*. Generally, when the command line is active, the anchor cell acts as a reference point for the operation itself, as for the function ▪, that copies the object stored in the anchor cell into the command line, at cursor's position. If cell mode is active instead, ▪ sets the active cell as anchor cell, eventually deleting the frame from the old location.

The fact a key behaves differently depending on the context is one of the exclusive characteristics of Omnibus. Managing effectively this capabality may save you time and work. The next two paragraphs will introduce you to the primary working environments of Omnibus.

Cell mode

Cell mode is always the start-up mode. When this mode is active the keyboard is mapped as shown on the back cover of this manual; for an exhaustive description of

each key, please read the section *Keyboard and menu operations* beginning on page 69.

The [ON] key quits the application without saving changes.

The [ENTER] key quits Omnibus returning the current worksheet to the stack.

Command mode

When you press the tick mark ['], [α] [α] or a numeric key like [0], the command line becomes active and you can type values, functions or commands and execute them by pressing [ENTER]. This means that you can create or purge variables, change directory or start other applications without leaving Omnibus.

Omnibus command mode gives you almost the same freedom as the *HP48* standard command mode does, that is you can view it like a command shell. The main difference lies in the fact that objects entered from the command line fill spreadsheet cells instead of stack levels. The stack becomes a clipboard for exchanging data between Omnibus and external applications.

When you need to push objects on the stack, press [↰] [=] *instead of* [ENTER].

Errors in command mode

If an error occurs while you are in command mode, the error message is displayed in the status area, a beep is issued and cell mode is restored. The only exception is represented by *syntax errors* which are handled directly by the editor that displays the offending text in reverse color.

Editing the command line

All editing function are fully functional and the editing menu can be recalled by pressing the key [↰] [EDIT] once command mode has been activated. Remember that you will need to press [↱] [MATRIX] to restore the main menu of Omnibus.

Leaving command mode

To quit from command mode, just press [ON]. In this fashion the contents of the command line are discarded and cell mode is restored. If the command line took more than one line of text, the worksheet is automatically redrawn and cell mode is restored.

Managing Omnibus

We will now take a look at the different working characteristic of Omnibus.

As stated in the first paragraphs of this chapter, Omnibus can be configured to accomplish many different tasks; since the configuration is controlled mainly through global flags[1], we will explain how each flag affects its workings.

In order to assist you in the configuration of Omnibus, we created a special interface, based on icons, that greatly simplifies the interaction with the program. Henceforth you will often find the icon beside the number of the flag being discussed. The icon represents how the label will appear after that the procedure has been carried out.

☞ Not all the flags of the *HP48* have got a related icon because some of them do not influence Omnibus.

From an abstract point of view we may summarize flags usage in the following statement: system flags affect the way Omnibus shows you the data while user flags determine functional changes. Both flag types have been collected into a unique icon-driven menu called **IconFlag**.

1 *Global* means a piece of information that can be accessed across multiple environments, whose lifetime exceeds the lifetime of the environment where it was created

IconFlag **IconFlag** is a command of the spreadsheet library returning to the stack a list of menu definitions suitable for the execution of commands like **MENU** or **TMENU**. In this fashion this interactive menu is made available to external programs, even if Omnibus is not running. You can expand, reduce or change it by storing the list, in a global variable called ICONFLAG which will override the default version.

While Omnibus is running, **IconFlag** can be accessed through the menu key CONFI, in the last page of the main menu, provided that this function has not been locked up by an application program.

☞ Generally, when a program inhibits the access to the configuration menu, it means that it is important to preserve certain modes.

This is the case, for instance, of the language browser (LANG) stored on the card.

When **IconFlag** is accessed through CONFI you can recall the mini help describing the function of each entry by pressing [↰] [▼]. Outside of Omnibus this capability is not available.

This function is lost also when you override the default menu with a custom version, as described in the next paragraph.

Customizing IconFlag When you create the variable ICONFLAG in the home directory, **IconFlag** is overridden and you cannot access it until you move ICONFLAG to a subdirectory off the current path or purge it. Remember also that storing an invalid menu definition in ICONFLAG will result in the impossibility of accessing the submenu CONFI. For

instance, if you store a real number in ICONFLAG, CONFI beeps and the access to the menu is denied.

☞ When the variable ICONFLAG cannot be found in the search path, CONFI will automatically use the internal version.

IconFlag collects system and user flags in one menu. Some icons refer to single flags, while others control a whole group. When an icon is assigned to a group of flags you can cycle through all the possible states of the icon by repeatedly pressing the corresponding key.

User flags Here is the list of the user and system flags playing a key role in the environment; beside each flag number you can see the related icon(s) of IconFlag. Flags are listed following the order of the icons as they appear in the menu.

Flag 38 has got no associated icon because its purpose is to prevent the user from accessing the menu CONFI. When flag 38 is set, CONFI just beeps.

☞ The default state of all flags is *clear*.

Icons	Flag	when set	when clear
	38	denies the access to CONFI	allows the access to CONFI
[]	37	locks recalculation	unlocks recalculation
[][]	32	recalculates cells	does not recalculate cells
[][]	33	any object in a cell (file mode)	only symbolic values (symbolic mode)
[][]	34	locks matrix rows	unlocks matrix rows
[][]	35	locks matrix columns	unlocks matrix columns
[][]	36	inhibits editing	allows editing

[Abc] [ABC]	40	case sensitive	ignore case
[$1] [$21]	62	Annuity due	Ordinary annuity
[⊠] [↺?]	39	Circular reference checked	Circular reference not checked
[▭] [▭]	60	Imperial units	SI units
[∠rad]	-17	Radians	Degrees or grads
[∠50] [∠45]	-18	Grads	Degrees
[∠∠] [▭]	-16	Polar coordinates	Rectangular coordinates
[a=1] [a=?]	-3	Numeric value	Symbolic value
[3,14] [π]	-2	Numeric constant	Symbolic constant
[x=1] [x=±1]	-1	Principal solution	General solution
[24:00] [12 PM]	-41	24 hour	12 hour
[1 JAN] [JAN 1]	-42	DD.MM.YY	MM/DD/YY
	-45...-48	decimal digits	
[▭] [$¥£] [▭] [▭]	-49...-50	STD → FIX → SCI → ENG	
[0,11] [0.11]	-51	Radix comma	Radix point
[▭] [▭]	-56	Beep disabled	Beep enabled
[▭] [▭]	-57	Alarm beep disabled	Alarm beep enabled
[▭] [▭]	-58	Verbose messages suppressed	Verbose messages displayed
[öé¢] [oec]	41	Sensitive to accented characters	Insensitive to accented characters
[▭] [ ]	42	Shows calendar enhancements	Clears calendar enhancements
[▭] [▭]	43	Translates system objects	Does not translate system objects
[A▭] [A▭] [▭]	-59 and 45	Tagged objects display mode	

When all user flags are clear, Omnibus behaves like the built-in Matrix Writer. The main difference is represented by the possibility of storing names or expressions in the cells and by the presence of a larger menu. Note also that you cannot create symbolic vectors but only symbolic matrices.

MatrixWriter mode

While flag 33 is clear ([⊟]), you cannot enter an object whose type is not listed in the following set:

- ☑ *Real numbers*
- ☑ *Complex numbers*
- ☑ *Global names*
- ☑ *Local names*
- ☑ *Algebraic expressions*
- ☑ *Units*

If you try to enter a sequence of items among those there is one of illegal type, the offending object is removed and its position is preserved by storing the '?' as a place holder in the destination cell. *In this fashion valid objects fill the desired cells even if most objects are invalid.* Note also that, if the beeper is enabled, every time an invalid object is detected, a double beep is emitted. For instance if flag 33 is clear ([⊟]) and you try to enter a string in a cell, Omnibus beeps and discards the string, whose place is taken by '?'.

The first times you work with Omnibus, you may often encounter the situation just described. Keep in mind that Omnibus records into the Last Command buffer the commands typed in the command line.

To succesfully enter the object refused by Omnibus, you need to:

- ➤ set flag 33 ([⊟]);
- ➤ move the cursor back to the original cell;
- ➤ recall the content of the command line through [↰] [CMD];
- ➤ press [ENTER].

This procedure works only if the Last Command buffer is active.

To change the status of flag 33 either press the corresponding menu key from the CONFI menu or set the flag with the command **SF**.

Omnibus and the objects

Once you have set flag 33, you can enter any object type in any cell. Since Omnibus is able to handle more object types than your *HP48* usually does, it is important to learn how and when a certain object is worth using.

Strings

Strings are suitable for displaying comments, telephone numbers, addresses, names or whatever text is likely to be looked for. Strings are displayed without the surrounding quotes and left aligned. A string split on several lines should not exceed 7 lines of height and 22 characters of width. Exceeding charaters and lines are automatically clipped.

Global names

Global names and local names should be reserved to variables and program names. Global names are treated like strings during a string search. Tick marks surrounding global or local names are always omitted.

Real numbers

Numbers (real or complex) are shown with the accuracy specified by the current display mode (**STD**, **FIX**, **SCI**, **ENG**), however, only most significant digits are shown due to the cell constraints. To change the notation of real numbers, jump to the third page of the menu CONFI, where you will find two icons (in the last two places): the former determines the type of notation while the latter controls the number of significant digits after the period. The fraction mark mode flag affects the way the separators are represented. To change it press the key labeled [□,11] or [□.11]. To represent dates or times you should resort to the specific objects described later on.

Algebraic expressions

Algebraic expressions are suitable for generic computations as well as for building cell references. You

can edit an expression either using the Equation Writer, typing the functions into the command line or applying hardwired functions (immediate function keys). System flag -2 ([3.14] [π]) may affect the numerical result.

Units

Units are worth using when you need to carry out computations where the dimensional information can vary or when you want to highlight the magnitude of the quantity. The tick marks surrounding units are always removed to improve readability.

The status of the flag 62, represented by the icons [▪▪] [▫▫], affects the computations only if the HP Solve card is plugged in and if you are working with equations extracted from the Equation library.

The status of the flag 62, represented by the icons [▪▪] [▫▫], affects the computations only if you are working with equations extracted from the Equation library application [EQ LIB].

It is worth noting that in the *HP48* there is the possibility of creating custom units. This capability allows you to create units for currencies, making financial computations easier if funds of different countries are involved.

To create units for dollars and british pounds, do the following:

- Store 1_? into a variable named '$'
- Store 1.75_$ into a variable named '£'
- To convert the amount of 1500_$ into pounds, issue the command 1_£ CONVERT

It is worth noting that you can choose the fund you prefer as a reference value. In the example above, we chose the

dollar. The conversion mechanism works only if you store 1_? into the reference unit. As a natural consequence, if you execute the command **UBASE** on a unit object like 1000_£, you will get 1750_? and not 1750_$. To achieve this result, you must convert 1750_? in dollars by means of **CONVERT**.

Tagged objects

Tagged objects serve for a twofold purpose. They maintain all the characteristics of the ancestor object while they provide a symbolic identifier that can be located dynamically. The tag is treated like a string during string search. Flags -59 and 45 rule the appearance of tagged objects; you can choose among three options: tag and taggee, tag only, taggee only. This capability allows you to show only the piece of information pertaining to the user. When the last option is in effect, string search skips over tags and compares the taggee with the search key.

Dates and times

Date and time objects require a special handling because they are not standard objects but an implementation of Omnibus. To enter a date object, type the real number representing the date followed by the command **R→DT**; time objects require the command **R→T**. A date or time object appears as External on the stack, making impossible to decipher its actual value. They are displayed in a readable form only when stored in cells. In this case, the appearance of the object depends on the state of time and date format flags, on the current column width and, finally, on the calendar enhancements flag. Dates and times can be employed in all the sorting and searching functions. For more information about time management issues, see the related chapter.

Characters

Character objects are used to create repeating patterns. The character pattern is always as wide as the column's is

wide. This feature can save considerable space when adding text motifs bounding logical areas in the worksheet. To enter a character object, type the real number representing the decimal ASCII code or a binary integer object and issue the command →**Char**. Note also that a character object appears as Char_ddd when you press [EVAL].

System binaries

System binary objects can be used to display library messages. This feature can be turned on ([▦]) or off ([✎]) at will. To enter a system binary object, type a real number or a binary integer followed by the command →**SYS**.

Programs

Programs (macros) allows the greatest flexibility. You can resort to a program when standard objects do not fit your needs or you want to achieve a special effect. It is a good practice to avoid developing large programs inside cells because small worksheets work faster. A program behaving like a function is the best choice over all[1]. See also the restrictions given on page 42.

1 A program behaving like a function must be necessarily called by name

Recalculation

Flag 32 rules the automatic recalculation of the cells. When flag 32 is clear ([ ]) all the cells containing references to other cells are displayed in their symbolic form. As soon as you set the flag ([ ]), all the cells in the display *containing global names, algebraic expressions or programs* are automatically evaluated. The time required to update the cells depends on the length of the formulae, on the total number of cells and, last but not least, on the complexity of the references. *Less cells are in the display, less time the recalculation will last.*
A single column display mode takes always less time for recalculating than a multiple column mode.

It is not always true that the recalculation is the most time consuming mode. A long expression may take indeed more time to be converted into text rather than be executed. For further details, see later on, under the paragraph *Keyboard and menu operations* on page 69.

When flag 32 is set [ ] the recalculation takes place, that is all the algebraic expressions, programs and global names are evaluated and the results are displayed in their respective cells. The content of the active cell is always shown in its symbolic form in the bottom line, truncated to the first 19 characters or less.

Evaluating the active cell

You can force the recalculation of the active cell by pressing the [EVAL] key. In this fashion, only the cells linked to it are evaluated. In this case you had better to turn the recalculation off ([ ]) first. The time required to get the value is affected only by the complexity of the references and by the duration of the computations.

The purpose of EVAL is twofold:

- recalculate a cell at a time;
- display the result in the best environment.

There are situations where this option is mandatory, because the object returned is too large to fit in a cell: arrays, pictures, worksheets, lists, multi-line strings, expressions and so on. In all these circumstances, EVAL switches automatically to the best fitting environment where you can freely inspect the result of the operation.

You should always resort to EVAL *when performing the What If analysys because at the end of the computation you get an array of values that can be explored in a separate environment. The array is also stored in the reserved variable* ΣDAT *for later use.*

The following table summarizes how certain objects are treated by EVAL.

Object type	Appearance
Real numbers, complex numbers and units	shown using the current format
Arrays and worksheets	inside Omnibus environment
Graphic objects and PICT	graphically with scrolling if needed
Expressions	inside the Equation Writer
Times and dates	using current time and date format
Characters	Char_ddd
Strings	on multiple lines if needed, up to 7 lines by 22 characters each
Tagged objects	Depending on the status of flags -59 and 45. It is possible to hide or show the tag or the object or display both (default).
Any other object	converted into a string and handled likewise

If you want to permanently substitute the expression with its value, you can resort to the key [→NUM]. *Keep into account that the substitution cannot be recovered. If you want to store the resulting object elsewhere, you may use the menu key* [↱] **→STK**, *that pushes the result of the evaluation on the stack.*

Locking out the recalculation

There are cases where you may want to inhibit the evaluation of programs because the intent of the application is to just show the names. This is the case of the last three application programs described at the end of this chapter. Omnibus allows you to lock out the keys involved in the recalculation of a cell like [EVAL] and [→NUM]. When the recalculation is locked out, the first icon of IconFlag is [⇨⇦] regardless of the current settings of flag 32.

When you press [EVAL], instead of executing the object contained in the cell, Omnibus takes it as it is and displays it in the best environment, as explained earlier.

Tuning the recalculation

The recalculation mechanism is disjoined by the state of the system flag -3 (symbolic result). In general you had better to keep flag -3 clear ([a=?], default setting) to avoid the occurrence of run-time errors due to the presence of undefined names in expressions, unless you explicitly want this kind of information. If you never experienced how the *HP48* responds when flag -3 is set, we suggest you to try it using a simple expression like 'X+1' where 'X' is an undefined variable. You will see that Omnibus shows you the string Undefined name instead of the symbolic expression result.

The most remarkable effect of setting flag -3 occurs when you use Omnibus as symbolic matrix writer. Once you have created the symbolic matrix, any mathematical

operation you try will cause an error if any symbolic value is undefined at evaluation time.

Flag -2 has got a meaning alike: it determines how symbolic constants are treated during expression evaluation. Flag -2 is less critical than flag -3 because usually it does not cause errors. If you prefer that constants are always evaluated, set flag -2 ([3.14]), otherwise you will always get a symbolic expression in terms of the constant. There are some situations in which setting flag -2 is mandatory:

- When you need to collect items in a real or complex array
- When you need to sort data using run-time values
- When you must perform some operation where only pure numerical values are acceptable (take the mean, find the minimum and so on).

Note that flag -2 does not affect the way the constants stored in the HP Solve Equation Library card are shown, because their value can be obtained only by evaluating the function **CONST**. Therefore, in order to view the value of an expression containing such constants, you must press [EVAL] or switch the recalculation on.

Evaluating programs

Although you can store a very complex program in a cell, we suggest you to avoid whenever possible such habitude for three good reasons.

Foremost is debugging: Omnibus disables the debugging capability so that you cannot destroy environment settings. Without a debugging tool it is hard to develop a large program. However if the program does not contain references to cells you can still run it outside the spreasheet environment and debug it successfully. Once you have obtained a working program, you can store it into a global name and use the name instead of the raw

program. Moreover if you need to call the program elesewhere, the named version *saves a lot of memory* and improves readability.

Secondly there are speed issues whose most important factor is represented by the conversion of the object into its string representation. A large program would take a considerable time to be converted, while a global name is immediately displayed.

Last but not least, there is a practical consideration: understanding what the program does is easier if you assign a descriptive name rather than if you try to interpret its contents time after time.

Remember that switching the context to a subdirectory may prevent you from accessing a program or a function stored elsewhere unless the variable is on the current path.

If you are going to create several subdirectories for storing worksheets, it can be useful to keep general macro definitions in the upper directories so that they can be shared.

Errors while recalculating

There are many circumstances under which the recalculation of a cell may result in an error. For instance you may refer to a non-existing location or make meaningless operations. In all these cases the error is trapped and the result of the evaluation is a string containing the error message.

When the recalculation of a cell causes an error, any other cell referencing it will return an error too.

A well-behaving program must always return one and only one object as result of its evaluation.

This is the rule-of-thumb which will save you time during the development of a program. Algebraic expressions are a typical case of a program taking several arguments from the stack (or none at all) and returning a single result (no matter which result). *Returning none, two or more objects is allowed*, but Omnibus always collects everything in a list (eventually an empty list) to guarantee stable input/output conditions. The following scheme summarizes how it works.

No result	One result	More than one result	Error
An empty list is returned	The object is returned 'as is'	A list containing all the objects is returned	The error message is returned as result

When an error condition is detected the error is forwarded to the user. However Omnibus traps any error occurring during the recalculation and returns the appropriate error message in form of a string. The error trapping is accomplished at the display stage, that is at the highest possible level, so that the user has got a chance to trap the error on his own.

The error trapping technique consists in a properly designed IFERR...THEN...ELSE...END *construct placed inside a program object.*

For most applications the error trapping capability of Omnibus should be enough.

Saving and restoring settings

If you want preserve the configuration of Omnibus while running a program that alters flags settings, you can surround the body of the program with the paired commands **SaveModes**/**RstModes**.

It is recommendable that you follow the scheme suggested below when using these commands.

« **SaveModes**
IFERR *program* **THEN**
error clause
RstModes
ELSE
else clause
RstModes »

Restrictions to programs

There are a few guidelines to follow when running external programs inside Omnibus, mainly due to the necessity of preserving the integrity of the data on the screen.

Keep in mind that there is a substantial difference between a program run from the command line and a program run inside a cell:

☞ A program run from the command line can alter flags, the display and other settings. In every case the display will be redrawn after the end of the user program according to the new settings.

☞ A program run inside a cell cannot change flags and other settings because they are automatically saved and restored by Omnibus. If it corrupts the display, it is up to the user to fix the problem.

A program prompting the user with questions or displaying messages in the desktop area, will corrupt the display when run inside a cell.

When a program corrupts the display area, it must be adapted to cooperate with Omnibus. By embedding the program between the commands **SaveDesktop** and **RstDesktop** you effectively save and restore the picture in the display and make the program usable inside a cell.

Interactive programs are not meant for use with Omnibus, but with the aforementioned workarounds you will be able to run most programs. See also the notes for the commands **SaveModes**/**RstModes**, explained earlier.

SaveDesktop requires 1K bytes of free memory to save the current screen bitmap. In low memory conditions this may lead to an insufficient memory error.

A program drawing a function in the graphic environment (PICT) does not corrupt the screen, therefore it is not necessary to embed it within **SaveDesktop** and **RstDesktop**. Keywords like **DRAW** or **ERASE** can be used freely. The example given on page 204 treats the problem of integrating graphics functions with the spreadsheet.

BARPLOT, SCATRPLOT ***and*** **HISTPLOT** ***do corrupt the screen***[1] ***when executed by programs inside cells.***

1 There is a workaround for these situations: substitute the command with the

A program displaying a message in the bottom line of the screen does not require special handling because line is automatically refreshed[1] by Omnibus.

Changing the current directory may cause errors.

It is a good practice to put in the HOME directory the utilities shared by programs which are located in different subdirectories, so that they can be recalled from anywhere.

Interrupting cell recalculation

When the recalculation of a cell seems to last too long, you can interrupt it by pressing ON once. The cell will display the message Interrupted and the recalculation will continue with the next cell.

The *What If analysis* can be stopped at any time by pressing ON, however the content of ΣDAT is always erased.

A cell pointing to a cell that has been interrupted will likely return an error. The type of error returned depends on the command originating it.

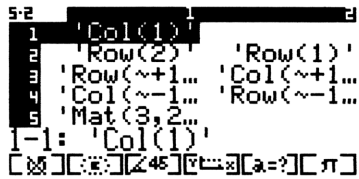
Once you have stopped the evaluation of a cell, in order to yield a value for it, you need to restart the process. If more cells depend on that value, you must switch the recalculation off and on again, alternatively you can evaluate the cell by pressing EVAL.

procedure ERASE *plottype* DRAW, where plottype is the command setting the type of statistic plot.

1 When the optional sixth row is on, this is not true. In this case you must disable the sixth row or embed the program as explained earlier

Circular references

An expression that cannot be resolved, due to an endless loop among cell references, is referred to as a *circular reference*. Circular references can be of varying complexity: the simplest form consists in a cell referencing itself. You can encounter very complex paths turning into circular references; the figure on this page shows you the topology of common circular references. In particular you may see that the cell in the lower left corner points to a cell inside a ring; Omnibus is able to detect even this kind of circular reference.



Circular references would be potentially dangerous if no caution were taken because they could lock up the *HP48*, forcing the user to reset the computer. However Omnibus implements two different levels of protection: *manual* and *automatic*.

Manual protection

You can press ON to stop the recalculation of a cell. Note that this method works even if a cell does not contain a circular reference but only a function taking a long time to return a value. When you break the recalculation the message Interrupted is returned as result.

Automatic protection

You can set flag 39 (⌐⊠¬) to automatically detect the presence of circular references.

Since circular references can be easily created while moving rows or columns, you had better to disable the

recalculation before editing the worksheet. When flag 39 is set (□⊠□), each time a circular reference is detected, the recalculation is stopped and the error message Circular reference is returned. When flag 39 (□↻?□) is off, you must press [ON] to stop the computation.

Locking matrix dimensions

By locking up the rows (columns), you can force the user to expand the matrix only by columns (or rows). See under **CONFORM?** and **EQUAL?** in chapter 1 of the SmartROM user's guide for examples on this topic. When you lock up both dimensions, the input matrix is editable but its shape cannot be changed. As a practical consequence, you cannot enter from the command line a number of objects greater than the current matrix can hold. Note also that *the number of input objects depends on the current state of the fill direction* and on the current cell position. In fact if the current fill direction is set to *none*, you can enter an unlimited number of objects because the cell pointer is not updated after each entry.

Dimension locking is recommendable when you call Omnibus to edit an existing template. Consider the following problem:

- write a program asking the user for a new PRTPAR variable while running a program.

You can provide the user with default values he can change at will. Moreover you can tag the cells with labels to make things easier to understand.

```
«
SaveModes
4 SetWidth
32 CF 33 SF 34 SF 35 SF
PRTPAR
DUP TYPE 5
IF ≠ THEN
```

```
DROP { 1.8 "" 80 "<013><010>" }
END
{ "Delay" "Remapping" "Width" "End of line" }
{ →TAG } LVOP
1 →LIST TRNSP IF MATWRT THEN
   TRNSP   1 GET { DTAG } LOP1
   'PRTPAR' STO
END
RstModes »
```

The strings <013> and <010> represent the *ASCII code* respectively of the *Carriage Return* and the *Line Feed* characters. To enter the carriage return character from the keyboard you must have the SmartROM custom menu installed. In this case you can press the following keys while the cursor is placed between the double quotes:

▶ CST NXT #CHR 0 1 3 → .

Disabling editing functions

Omnibus allows you to prevent the user from changing what is contained in a worksheet by means of a keyboard lock. Since you can start a session of Omnibus either using the default configuration or restoring the configuration stored with the data, you must be aware of the practical difference this fact implies.

When you load a worksheet in form of list, the default configuration is used. In this case, if you want to inhibit the editing functions inside the spreadsheet, you must set user flag 36 ([⊞⊟]) before starting up.

... 36 **SF** ...

This caution is normally taken by programs calling Omnibus as a data browser, with the intent of protecting the data of the worksheet against accidental erasure. In this case flag 36 is set after that the existing configuration has been saved with **SaveModes**. The configuration is restored upon termination of the program with **RstModes**.

Usually, flag 38 is set as well to prevent the user from accessing IconFlag and changing the settings inside the application. Always set both flags if you want to restrict the active keyboard to arrow keys only.

By setting flag 36 and 38, you effectively prevent the user from changing the configuration of Omnibus, because the only active keys are some menu functions and cursor movement keys. As long as flag 36 is set, you cannot add, delete or modify any cell inside the spreadsheet.

When flag 36 is set, you cannot start Omnibus from scratch; you can only load existing worksheets.

This approach works fairly well when the worksheet must be created on-the-fly, depending on user's needs.

If you deal with fixed worksheets instead, you had better to embed the configuration settings in a library data object. In this fashion all the configuration isseus are handled by Omnibus and you are sure that everything will be as you left off.

When you load a worksheet in form of library data object, the original configuration is recalled, therefore the current setting of flag 36 (as well the state of all other flags) is uninfluential.

If you wish to make a worksheet protected against editing, first enter IconFlag, then set flag 33 (⌨) and flag 38, finally switch flag 36 to [⇄]. If you press [ENTER], a library data object will be returned.

When the editing lock is active, you can still access the Equation Writer and edit an expression, but the changes are discarded.

Suspending the command line

By toggling flag 36, you can actually suspend and resume the editing functions of the command line. As soon as you set flag 36, the command line becomes hidden and you can move the cursor to another cell. If you press EDIT or ↰ EDIT, the object stored in the cell is copied at the cursor position. By clearing flag 36 you can resume the editing.

When the command line is suspended, ENTER and ON do not terminate Omnibus, but clear the command line.

The possibility of moving around the cursor while editing an object allows you to cut and paste text taken from other cells. You can also tell Omnibus to create references between two cells, but this is the argument of the next section.

Referencing cells

A cell reference is a function returning the value of another cell. The task of the spreadsheet is to update the values of the references as soon as the cells on which they depend are changed by the user.

There are functions returning the value of an operation performed on a group of cells rather than on a single one. These functions do not require an extensive specification of all the cells involved in the operation because they are able to recognize the valid range by looking at the type of objects and at their relative position.

Omnibus implements a cell reference method somewhat unusual respect to the standards of other spreadsheets. First of all you may easily notice that Omnibus uses a *pair of coordinates* to determine a cell location instead of a letter/number encoding. This notation implies a simpler structure of the spreadsheet and is more consistent with the RPL conventions. Moreover, being the *HP48* a math-oriented computer, you have the possibility to define subscript functions. For instance you can define a function returning the sum of all elements belonging to odd columns in a straightforward fashion:

```
Σ(k=0,FLOOR(MaxCol/2),Row(2*k+1))
```

and place it in the last column of every row.

We must add that Omnibus is conceived to prevent the user from creating chaotic worksheets. A subtle hierarchy force the user to build up ordered worksheet and *discourages the creation of 'zigzag' references*. The tiny display of the *HP48* does not bear exceptions to this rule.

The underlying hierarchy

Many functions work on the assumption that data precede the formulae either in the vertical or in the horizontal versus, using the top-to-bottom or left-to-right convention. When data are arranged in this fashion, some functions assume a particularly short and efficient format. After all this is what the 99% of users usually do, so that we privileged this referencing scheme respect to others. Notwithstanding you can define scrambled cell references though we recommend to avoid them whenever possible. All these special functions are explained in the next paragraphs.

Explicit references

Omnibus allows you to create cell references in several ways. There is not a best one, but it depends on what you want to get: some are faster, some are shorter, some other are more comfortable to use.

A cell can be referenced by specifying its absolute location, in this fashion: **Mat**(3,3); **Cell**(2,10).

The function **Mat** always evaluates cell contents when flag 32 is set ([▭]). In order to prevent the evaluation, the function **Cell** has been provided. **Cell** inhibits the recalculation of the cell and it is handy to check the type of the object stored in a cell, regardless of the state of the recalculation flag.

There is not much to say about explicit references because they are extremely intuitive and quite rigid. In brief, explicit references are preferrable when your data structure is likely to remain unchanged forever. On the other hand this method is intrinsecally fast, although not dramatically faster than others.

Mixed references

A special mechanism lets you create references linked to the current cell location: **Mat**(~,1), **Mat**(~-1,~-1), **Mat**(**MaxRow**,**MaxCol**), **Cell**(2,~-1). The '~' identifier acts

as a jolly and its actual value is computed dynamically during the recalculation (and varies from cell to cell). It means either *current row* or *current column*.

Implicit references

When both counters depend on the current location we may speak of an *implicit reference*. When only one counter is linked, we had rather to speak of a *mixed reference*.

'~' (tilde) can be specified indifferently as a row counter or as a column counter allowing a great flexibility to the user. Of course more complex is the expression involving tilde, more time it requires to be evaluated, but, strictly speaking, this difference is perceptible when dealing with dozens of terms: evaluating '~' or '~-1' does not affect the speed significantly.

As a practical consequence of the circual reference detection scheme, you cannot evaluate **Mat**(~,~), as you can easily proof by yourself.

Symbolic references

By tagging a cell with a string you can locate the cell by means of the tag. The functions **Ctag?** and **Rtag?** locate a tag within the current column or row.

The tag string must be a valid name, that is a string not containing punctuation characters like tick marks, spaces, commas and so on.

The search is always *case sensitive* regardless of the state of flag 40. The argument of the function must match the tag exactly, thus you cannot search for an item using the name like a substring. The search begins from the bottom of the row or column and proceeds backwards; if you have a column (or row) with two identical tags, only that appearing last will be found.

☞ In general there should *never* be two identical tags on the same line.

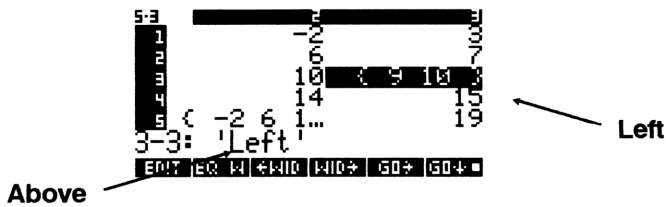
Row and Col Two more functions use implicit pointers: they are **Row**(x) and **Col**(y). In practice **Row**(x) is equal to **Mat**(~,x) and **Col**(y) is equal to **Mat**(y,~). Since it happens very often that arguments stay on the same row or column **Row**(x) and **Col**(y) represent the ideal solution to this kind of problem.

You cannot evaluate '**Row**(~)' or '**Col**(~)', while you may enter '**Row**(~-1)' or '**Col**(~+1)'. Similarly a cell containing an expression like the following turns out to be a circular reference:

:Circular:'2***Col**(**Ctag?**(Circular))'

Retrieving matrix bounds The functions **MaxRow** and **MaxCol** return current matrix bounds. Knowing the actual dimensions of the worksheet is necessary if you write a program which iterates an operation over the whole matrix.

Collector functions Two functions, taking no arguments, let you deal with lists. We are speaking of **Above** and **Left**. **Above** returns



the list of the objects lying in the same column of the current cell and preceding it (exclusive). The counterpart, working with rows, is **Left** which returns a list as well. As we will see a bit later, **Above** and **Left** do not return a list of calculated values but merely collect the objects

preceding themselves, deferring the evaluation to the calling routine. This is necessary to guarantee that internal pointers are updated correctly.

If you place **Left** or **Above** on the physical border of the worksheet, an empty list will be returned.

Above and **Left** are conceived to work in conjunction with other commands which span the list, determine valid objects, extract elements and pass the information on to routines which perform the desired task.

To copy a row (a column) of data to the stack just move the cursor to the farmost place with [Shift][→] ([Shift][↓]), *then type* **Left** *(or* **Above***)* [Alt][=].

Carray and Rarray

Above and **Left** work well when you want to collect *all* the objects regardless of their type. **Carray** and **Rarray** are designed to return a one-dimensional array whose elements can be either complex or real numbers only. **Carray** and **Rarray** require a symbolic argument: a left arrow (←) or an up arrow (↑). This symbolic range is resolved only at evaluation time; in brief, the function first looks for the closest contiguous range of values or expressions and, if one is found, evaluates each symbolic expression; thereafter it collects the closest set of numerical values into the array and returns it as result. When no values are found, the error message Bad Argument Type is issued. **Carray** is more tolerant than **Rarray** because it allows to mix real and complex numbers while **Rarray** does not.

The directory FINANCE.DIR contains two worksheets showing a powerful use of **Rarray**; see BTP.WKS and CAPITAL.WKS.

AsIfRow and AsIfCol

These functions allow you to temporarily change current pointers and evaluate an expression as if it were in a different row or column.

Typically **AsIfCol** and **AsIfRow** are employed in conjunction with functions like **Sum** or **Rarray** whose results depend on the cell where they are located.

Suppose you want to calculate the sums of two different groups of values stored along the same column and separated logically, without intermediate steps: since **Sum** determines the range of values to sum by looking at the nearest group of objects preceding itself, in the direction specified, it would be impossible to make the calculation without allocating a cell near to the first set of values. Thanks to **AsIfCol** instead, it is possible to simulate the presence of **Sum** in a cell actually occupied by another object and perform the computation correctly.

Current row and current column

Three functions are provided for returning current row and column: they are respectively **ThisRow** (↑↓), **ThisCol** (←→) and **Current** which returns both values as complex number.

☞ The term current refers to the cell being evaluated rather than the cell highlighted (the active cell).

This distinction falls away when the program or command is issued from the command line. In fact if you assign the value '**ThisRow**' to the cell (3,1), as soon as you set flag 32 () you will see a 3 in that cell even if the active cell is somewhere. Moreover if you add or delete some row above it, the value will change accordingly. Conversely, if you make (3,1) active cell and you enter **ThisRow** *omitting the tick marks*, you will push an instance of '**ThisRow**' into the cell, whose value is

identical, yet absolutely fixed, because current cell and active cell coincide only in that particular situation.

Note also that writing '**Mat**(↑↓,c)' is exactly the same as writing '**Mat**(~,c)'. The placeholder '~' prevents mistakes and saves space. On the other hand '~' cannot be used outside of cell-functions because its value would be ambiguous.

More functions

Current and **Anchor** are functions returning complex numbers as cell locations. **Current** returns current cell location (and is equal to the sequence ↑↓ ←→ **R→C**) while **Anchor** returns the current anchor point (see [.] or [·]). The pair **Current Anchor** allows you to identify a rectangular region of cells to which apply certain operations.

Anchor can be employed as search command to quickly return the cursor to the anchor cell in a large worksheet. The proper usage is:

Anchor →NUM

at the [↰] GOTO prompt.

Active and Value

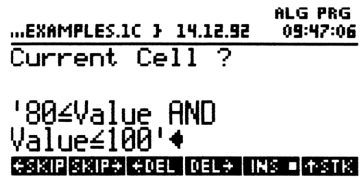
Active and **Value** are functions designed to work within *search functions*. There are some cases where they can help you as well but we will come back on this matter a bit later.

Searching objects

Search functions are the expressions created inside the [↰] GOTO utility to locate a cell satisfying certain conditions. For instance say we want to locate the numbers in the range 80<=x<=100. The search function can be written as shown in the picture on the next page.

Value differs from **Active** because evaluates active cell contents. In effect we may say that **Value** is the counterpart of **Mat** while **Active** resembles to **Cell**.

Active is suitable for searching primitive objects, that is objects we don't need to evaluate, like real numbers, strings or dates. **Value** is slightly slower but lets you find hidden values (results of expressions or programs) and, in virtue of this property, is generally slower.



The rule-of-thumb is the following:

- to look for a numerical value (real, complex, unit), you must use a *search function*;
- to look for non-numerical value (date, time, string, list and so on), you must use a *search program*.

Repeating the last GOTO

Omnibus *remembers the last item* supplied to the GOTO utility, provided it is a string, an algebraic expression or a program. In this fashion, by pressing GOTO, you can repeat the search in a single key press.

A string made out of a single character is automatically converted into a character object and employed to search for an item given the initial letter, therefore substrings must be at least two characters long.

When dealing with search functions always keep in mind the following important factors:

- the search is limited to the current row or column;
- The current fill direction setting determines the current search direction. If no direction is selected, GOTO beeps;

- Strings and characters are subject to various levels of case sensitivity.
- The equality sign is == and not =.

Using = causes the early termination of the search due to an invalid function format. You can also employ* SAME *in search programs.

Alternatively to search functions, you may specify a complex number as an absolute cell location, a row (or column) relative index, a string or a global name.

If you frequently employ a program or function, you should take into consideration the possibility of recording it permanently. See the paragraph More about shortcuts later on.

In the case of strings and numbers though, if the items are in alphabetical order, no matter if ascending or descending, you had better to use SEMRC, that employes a binary search algorithm.

When you execute a binary search (SEMRC), keep in mind that:

- The match is always case sensitive;
- a substring can be found only if it is the beginning portion of a larger string;
- if a match is found, the cursor is placed on the corresponding cell, otherwise the cursor is left on the nearest cell.

Quick search Omnibus allows you to move the cursor from cell to cell given the initial character. To begin the search, just press [α] followed by the key of the corresponding letter until you reach the desired item. The search ends when you reach the bottom of the line.

If upper/lower case distinction is important, check for the label [Abc] in the first page of IconFlag. To search for a lowercase character, you must press [α][←] and the letter.

☞ [α][→][ ] keystrokes implement a special function. They can be configured to act as quick search keys for special characters, but, by default, they are not assigned.

By analogy, you can enable or disable accented characters sensivity by means of the menu key labeled [öéç] or [oec], in the [CONFI] menu.

Note, however, that when the accent sensitivity is enabled ([öéç]), you may be not able to reach items beginning with special characters, unless you defined that character as a shortcut, as explained on page 100. In fact [α][←][7] or [α][→][8], normally beep and do nothing.

If you frequently need to search for an initial character which is not comprised in the set of the alphabetic letters, you had better to declare it as a shortcut.

Assigning a shortcut is fairly easy:

- Put the desired object on the stack
- Run the utility SHORT, stored on the card.

☞ Keep in mind that shift keys are ignored.

For instance to assign the character "0" to the key [α][→][0],

- Convert the string "0" into a character object, 0 →Char;
- Run SHORT, :&:SHORT EVAL;
- Press the key [0].

More about shortcuts

Shortcuts are the keystrokes beginning with [α][→], and they are reserved to the user. By default they do not exert any action in *cell mode*.

You can record up to 46 definitions, that is the total number of the keys in the keyboard minus the shift-keys [α][←][→].

The purpose of a shortcut is to assign to a key a function, a program or a search key, supplied by the user *with the intent of moving the cursor*. Typically functions and programs are those that have been created for use with GOTO, but you must be aware that there can be other alternatives.

An example of a special application called RPLMAN is given in the following section.

A summary of the operations required to declare an object as a shortcut and the list of valid objects are given on page 100.

The backup object PLANE6, stored on the card, contains default shortcuts definitions. The map of keys allocated by PLANE6 is given on page 101. The instructions for using PLANE6 assignments are also given there.

If you chose **Y**es to all the questions during the installation of Omnibus, PLANE6 should have been already installed. To check for the presence of default assignements, try pressing [α][→][VAR]. A screen containing system information should appear.

☞ This keystroke holds only while Omnibus is running.

If a double tone is played, PLANE6 has not been installed.

Using built-in functions

As mentioned earlier, Omnibus has the ability to use standard RPL functions for computing values. In virtue of this property, you can freely mix built-in function with user-defined functions or third-parties extensions effortlessly. The *HP48* is well known for its huge set of math and engineering functions so that we will not embark ourselves in the challenge of explaining how to use each function. We had rather to explain some general rules to achieve the best results and performance.

Applying functions to cells

The *HP48* keyboard contains a few keys hard-wired to common math functions like [SIN] [1/x] [LN] and so on. Omnibus allows you to apply these functions directly to the active cell, without intermediate steps. For this reason keys hard-wired to functions are called *immediate function keys.*

You can also combine a cell with an object taken from the stack, using operators like [+] [−] [×] [÷] or [y^x].

To push a value on the stack, press [↰] [=] *instead of* [ENTER].

Of course the object in the cell and the optional object on the stack must be compatible with the algebraic object structure being created otherwise they would be rejected with a beep. This means, for instance, that you cannot apply [×] given an array and a number, although such operation is allowed in a program.

*If you put the array in the cell (m,n), you could write the expression 'Cell(m,n)*x' and get the dot product of the array as result.*

When the recalculation is enabled ([▪▪▪▪▪]), you can watch the value of the expression in the cell, if any. Note that this operating mode is very similar to an RPN calculator,

with the difference that you are also recording the functions applied step by step.

Managing expressions

Since it may happen to mistakenly apply a wrong function or exchange the order of the arguments, there is the possibility to undo the last function or swap the order of the arguments (for binary functions only).

The key [←] rewrites the expression in the active cell deleting the last function applied. If the last function applied is *unary*, [←] deletes only the function and keeps its argument, that becomes the new expression. If the last function applied is binary instead, [←] deletes the function and the left-hand argument; the right-hand argument becomes the new expression.

If you want to delete a binary operator and its right-hand argument, you must first swap the operands with [SWAP] and then press [←]. Of course [SWAP] does not work with unary functions.

Functions with more than two arguments cannot be handled this way, but their occurrence in a complex expression is very rare, so that this is not a real problem.

The *HP48* has got a huge set functions and, surely, more functions than keys. When a function is not comprised in the set of the immediate functions, you must type it into the command line (verbatim or by means of a typing aid) or edit the whole expression inside the Equation Writer environment.

Since you may need to apply the function to a large expression which is already in cell, Omnibus allows you to minimize your efforts.

For instance, say you want to replace the expression in the active cell with the following:

'(Col(1)+Col(2)) % expr'

where *expr* is the original expression.

▶ ' ( ) +COL ( ) 1 ▶ + +COL
( ) 2 ▶ ▶ SPC α ↱ TAN SPC EDIT ENTER

Replacing objects

The approach shown above works well when you are working on a single expression, but there can be situations where you need to replace an expression in the whole worksheet or in a selected portion.

This is a special case of a general problem that can be solved with the help of some utilities stored on the card.

The main program, handling the replacement of the objects inside the spreadsheet, is called RPLMAN (Replacement manager). This program prompts the user with a list of four options, then calls a specific utility to carry out the task.

RPLMAN has been conceived for a twofold purpose:

1 to make the replacement easy;

2 to demonstrate a powerful usage of the programming capabilities of Omnibus.

The source code of RPLMAN is printed starting from page 173. In the subsequent pages you can find the source code of the utilities invoked by RPLMAN too.

In the next paragraphs we will assume you have assigned RPLMAN to the key α ↱ EEX, but you are free to choose another key.

☞ If you ran WELCOME to install Omnibus, may be you have already assigned shortcuts. To verify the assignment, press α ↱ VAR. The system information screen should appear.

If the assignment has not been made, you can do it as follows:

- write :&:PLANE6 RCL **StoShortcuts**

or for a customized assignment:

- open a list and write { :&:RPLMAN EVAL };
- write :&:SHORT EVAL;
- press the key [EEX]. Shifts keys are ignored.

And now let's create a worksheet like the following:

	:Discount%:30
	:VAT%:8.5
1000	'Col(2) %A (Col(1) %D Row(1))'
2000	'Col(2) %A (Col(1) %D Row(1))'
3000	'Col(2) %A (Col(1) %D Row(1))'
4000	'Col(2) %A (Col(1) %D Row(1))'
5000	'Col(2) %A (Col(1) %D Row(1))'
6000	'Col(2) %A (Col(1) %D Row(1))'

Replacing expressions

Now suppose you want to replace the expression

'Col(2) %A (Col(1) %D Row(1))'

with

'AsIn(3,2)'

in all the cells but (3,2).

- Fix the anchor cell (the upper left corner of a cluster) in (4,2), [.];
- move the cursor to the opposite corner (8,2), [→][▼];
- push the cluster on the stack, [EEX];
- run RPLMAN, [α][→][EEX];

- choose Algebraic expressions;
- enter the expression 'Col(2) %A &' ENTER;
- enter the expression 'Asln(3,2)' ENTER;
- press ENTER at the next question;

RPLMAN calls the utility EXPRPL that begins the replacement. Finally it shows a message where it explains what happened.

If any expression has been changed, EXPRPL tells that the operation can be acknowledged by pressing the key → EEX. This keystroke physically substitutes the portion of the worksheet determined by the position of the active cell and by the dimensions of the cluster.

It is possible to make several changes to the same cluster and a single acknowledgment. Since the cluster being modified is on the stack, you need only to call RPLMAN *as many times as required. When you are done, press* → EEX.

If EXPRPL could not find the expression specified, you may retry the operation or drop the cluster off the stack. To remove the cluster from the stack, you have the following alternatives:

- issue the command **DROP** from the command line
- enter the interactive stack environment and press ←.

EXPRPL employs the keyword ↑MATCH to recognize and substitute expressions. This fact implies that you can enter generalized expressions as well.

Consider the problem of replacing the expression 'Col(1)' with 'Col(2)' and 'Col(2)' with 'Col(3)'. The replacement must be accomplished in a single pass to avoid confusion and must affect only the expressions containing 'Col(1)' or 'Col(2)'.

The trick consists in generalizing the expressions while restricting the replacement.

The expression to replace is

'Col(&)'

the substitute is

'Col(&+1)'

and the restriction is:

'&<3'

The symbol '&' is referred to as a *wildcard.*

The last expression should return a boolean value (either 1 or 0), which determines whether the replacement must be accomplished (1) or not (0).

☞ EXPRPL substitutes any occurrence of the expression specified, even if it is a sub-expression.

Other replacements

As you have seen, RPLMAN offers four alternatives.

- Algebraic expressions

explained earlier; they are handled by the utility EXPRPL.

- Strings

if the object in the cell is a string, uses the search string as a substring, otherwise, if the object in the cell is a program or list, it looks for an exact match. The search is always case sensitive. The operation is performed by the utility STRRPL.

- Generic objects

the objects must match exactly. The search key is compared with the content of each cell. You can easily

replace empty cells with a default value. For instance, to fill up empty cells with zero:

► Type [α] [α] **FALSE** [ENTER]
► Type [0] [ENTER]

The operation is performed by the utility OBJRPL.

■ User defined

Allows you to apply a generic transformation to the objects in the cells. The program must take one object from the stack and return at least one object. If more objects are returned, they are automatically collected into a list.

User-defined transformation are very useful to apply an operation to a group of cells.

For instance, you can delete the tag from tagged objects,

« DTAG »

change the type of objects,

« 0 R→C »

« →Str »

« UBASE UVAL »

collect expressions,

« COLCT »

transform expressions,

« 100 / »

User-defined transformations are handled by the utility USRRPL.

For all the four utilities holds the following property :

☞ if the operation attempted on a cell is not valid, the error is trapped and the procedure continues with the next cell.

Fatal errors may occur only when:

- an insufficient memory condition is detected.
- the user-defined transformation drops too many objects from the stack.

Keyboard and menu operations

This chapter covers the following topics:

- ☑ *menu functions;*
- ☑ *keyboard functions;*
- ☑ *programmable commands and functions;*

The first section describes menu entries and keys, whose function has been redefined in the new environment. Keys maintaining the standard definition are not treated.

☞ The dummy key [ ] means *any* key.

Menu keys usually perform different actions depending on the current mode, either *cell* mode or *command* mode. The reminders that appear on the screen when you press [↰][▼] refer to *cell mode.*

☞ The alpha lock flag (system flag -60) does not affect Omnibus. To lock the alphabetic keyboard you must always press [α][α].

The alphabetic keyboard serves for a twofold purpose:

1 if you press [α] or [α][↰] followed by a letter (***A...Z***), while in cell mode, you start the search of text beginning with that letter. This function, called *search by character*, is described on page 100. [α][↱][ ] keystrokes are reserved to *shortcuts*, described on page 100.

2 if you press [α][α], while in cell mode, you switch to command mode and, at the same time, lock the alphabetic keyboard. Greek letters can be obtained

by preceding the latin letter with [→] instead of [←]. Refer to the alphabetic keyboard map printed on the *HP48 User's guide* to find the position of letters.

However there are characters that cannot be typed in directly; in this case you have the following options:

In the *HP48GX* you can enter such characters by means of the *Character Map* application. To start the character map, press [→] [CHARS] *while in command mode.* To paste the selected character with the text in the command line, press [ECHO].

In the *HP48SX* there are no provisions by default. Run the utility CMAP that emulates the character map of the *HP48GX*. To paste the selected character with the text in the command line, press [ECHO]. We suggest you to assign this application to a user key by means of the utility HOTASN, as described on page 159. Remember to begin the assignment keystroke with [α], so that the application can be run inside the editor.

Alternatively:

SmartROM's custom menu[1] ([CST]) contains the function [#CHR] that allows you to enter a character, *given its ASCII code*. Usually this is the fastest way to enter a character if you know in advance the code of the character.

1 To install the SmartROM's custom menu, execute the command **CSTMENU**. If you use the custom menu to hold your own menu definitions, try merging the menus.

Editing the active cell

EDIT Checks the type of object stored in the active cell and chooses the best editing environment automatically. **EDIT** beeps if the active cell is empty or flag 36 is set[1].

The following table summarizes the treatment reserved to objects:

Object type	Editing environment
Real and complex numbers, binary integers, strings, lists, programs, global names, tagged objects	Copied into the command line.
Algebraic expressions and units	Passed to the Equation Writer.
Dates, times, characters, system binaries	Copied into the command line as real numbers, followed by the conversion command if flag 43 is clear ([→#]) . If flag 43 is set ([→"]) characters are converted into strings.
Graphic objects, **PICT**	Passed to the graphic editor.
Omnibus files, arrays and symbolic matrices	Passed to Omnibus recursively.
FALSE and **TRUE** place holders	**EDIT** beeps.

To force the editing of an object in the command line use the keystroke [←] [EDIT].

1 There is an exception to this rule. See later on

EDIT works either in cell mode or in command mode. This means you can copy the contents of the active cell into the command line at cursor's position. For instance, say you want to change the active cell from '**Row**(~-1)' to '(1-**Row**(~-1))/**Row**(~-1)':

► press ['];

this keystroke opens the command line and sets algebraic mode;

► press [↰] [( )] [1] [−];
► press EDIT, then [▶] [÷]
► finally press EDIT and [ENTER].

There are situations where EDIT becomes ECHO. If you set flag 36 ([⇧⌫]) after running Omnibus while the command line is on, ECHO allows you to insert the content of the active cell with the text in the command line at cursor's position.

☞ In this circumstances, the command line is hidden and you can move the cell pointer from cell to cell without disturbing the editor.

To resume typing, just toggle flag 36 off.

When flag 36 is set and the command line is disabled, ECHO beeps.

ECHO is affected by flag 43 ([↵⌫] or [↵⏎]), exactly like EDIT.

See also the entry [↰] [EDIT].

[↰] EDIT Builds a reference to the active cell and pastes it at cursor's position. The type of reference

created depends upon the presence or absence of the anchor cell. When the anchor cell is aligned with the active cell, the reference will be either **Row**(c) or **Col**(r), otherwise it will be explicit, of the type **Mat**(r,c).

The reference just created should not be used inside the active cell, because it is circular; this kind of function is required when you are building an expression containing references to several cells, without making calculations by hand. See also the entry [↶] [ ].

[→] EDIT Builds a reference to the active cell. The type of reference created is always implicit and requires the presence of the anchor cell. When the anchor cell is aligned with the active cell, the reference will be either **Row(~+δc)** or **Col(~+δr)**, otherwise it will be implicit, of the type **Mat(~+δr,~+δc)**. If the anchor cell is missing or invalid, a beep is issued.

Since the reference created would be circular if used inside the active cell, this kind of operation is performed when you are building an expression that must be stored elsewhere. See also the entry [→] [ ].

EQW Passes the contents of the cell to the Equation Writer for editing. Works only if the item is an algebraic expression, unit or number (either real or complex).

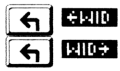
The Equation Writer is started from scratch when the cell is empty, hollow or an invalid object type is detected.

Within the Equation Writer environment you can rearrange the expression according to algebraic rules. By pressing [ON] you exit from the environment and restore the original value in the cell, while pressing [ENTER] you put the expression in the active cell. However, if flag 36 is set, you can still access the Equation Writer but any modification you apply to the expression is discarded.

Adjusting column width

Narrows or widens columns. The total number of cells shown in the display affects recalculation time.

Cells in the display are recalculated when flag 32 is set ([▭]), while hidden cells are recalculated only if they are referenced by cells in the display, therefore less cells are shown, less time the recalculation lasts.



Writes the command **SetWidth** into the command line. Works if flag 36 is clear.

Sets default fill/search direction. When you enter an object from the command line, Omnibus automatically shifts the cell pointer to the next cell. This capability lets you fill out a worksheet by entering several objects at a time.

☞ You can disable the fill direction. In this case the cursor position does not change after each entry. If you try to enter more than one object at a time, each entry will be overwritten by the next one.

The way in which Omnibus fills up cells depends on the shape of the matrix and on the default fill direction. The mechanism works on the assumption that people usually create uniform matrices, that is worksheets with straight

bounds. Should you find it annoying, you can switch it off, as explained below. In this case, however, you will be forced to enter just one object at a time.

In order to disable the fill direction, press the key whose label is checked with the dot.

It may happen you want to replace an object stored in the last cell of a line. In this case, if the fill direction coincides with the line direction, Omnibus will wrap around the last element, scrolling the display if required. By switching off the default fill direction, you can avoid this side-effect.

☞ GO→ and GO↓ determine the *search direction* of all the sequential search functions like GOTO and α or α ↰ keystrokes. When the fill/search direction is set to *none* (GO→ and GO↓), GOTO accepts only absolute cell pointers, in form of complex numbers, while the search by character is aborted with a beep.

Managing rows and columns

+ROW *Cell mode*
Shifts down all the rows below the current cell and inserts a blank row. Works if flags 34(⊏⇓⊐ or ⊏⇨⊐) and 36 ([⌨]) are both clear. If flag 33 is clear ([≡]), the new row will contain zeros otherwise it will contain empty cells.

Command mode
Copies the word **Row** into the command line at cursor's position.

↰ +ROW Removes current row. Works if flags 34(⊏⇓⊐ or ⊏⇨⊐) and 36 ([⌨]) are both clear.

ROW→ Rolls right all the elements of the current row, at the right side of the current column, with wrap around. Works if flag 36 is clear ([ ]).

☞ If the recalculation is on and the protection against circular references is disabled, you may lock up the *HP48* if any explicit reference is pointing to an element in the row being moved. It is a good practice to disable the recalculation or enable the circular reference protection beforehand. Note however that you can always unlock the *HP48* by pressing ON once.

← ROW→ Rolls left all the elements of the current row, at the right side of the current column, with wrap around. Works if flag 36 is clear ([ ]). See the note above.

ROW↑ Rolls up all the rows below the current row (inclusive) with wrap around. Works if flag 36 is clear ([ ]).

← ROW↑ Rolls down all the rows below the current row (inclusive) with wrap around. Works if flag 36 is clear ([ ]).

+COL *Cell mode*
Shifts right all the columns at the right side of the current cell and inserts a blank column. Works if flags 35 (or) and 36 ([ ]) are both clear. If flag 33 is clear ([ ]), the new column will contain zeros otherwise it will contain empty cells.

Command mode
copies the word **Col** into the command line at cursor's position.

[←] [+COL] Removes current column. Works if flags 35([⇒] or [⇐]) and 36 ([⌨]) are both clear.

[COL+] Rolls right all the columns at the right side of the current column with wrap around. Works if flag 36 is clear ([⌨]).

[←] [COL+] Rolls left all the columns at the right side of the current column with wrap around. Works if flag 36 is clear ([⌨]).

[COL↑] Rolls up the elements of the current column, below the current row, with wrap around. Works if flag 36 is clear ([⌨]).

[←] [COL↑] Rolls down the elements of the current column, below the current row, with wrap around. Works if flag 36 is clear ([⌨]).

Managing the stack

[→STK] Copies the current cell onto the stack. The object is copied unevaluated, regardless of the status of the recalculation flag.

When you quit Omnibus, all the objects on the stack, if any, are removed and placed in a *clipboard*. The command **Clipboard** returns a list to the stack containing those objects.

You can also pass objects from one spreadsheet to another, in this fashion:

- run Omnibus;
- type the name of the source-worksheet in the command line;
- evaluate it with [↰] [=];
- execute the command **MATWRT DROPN**;
- copy the objects to the stack (clipboard) with [→STK] or extract clusters with [EEX];
- quit the current session of the spreadsheet;
- split the clipboard with [↰] [↑STK];
- press [←] to copy each object in the destination cell or press [↱] [EEX] to replace a cluster of cells.

[↰] [→STK] Pushes a cell reference onto the stack. A cell reference can be of different formats, '**Mat**(r,c)', '**Row**(c)' or '**Col**(r)', depending on the presence and position of the anchor cell. If no anchor cell is active, the reference will be of type **Mat**. **Row** and **Col** are created only if the anchor cell is aligned.

[↱] [→STK] Pushes the value of the active cell onto the stack. This keystroke allows you to put values into the clipboard for later use. The item is always evaluated, regardless of the current recalculation setting, but the recalculation must be unlocked (flag 37 must be clear).

[↑STK] Enters the *interactive stack* environment. This function is active in either modes [1] If the stack is empty, a warning message appears. See the

1 If command mode is active, you can copy stack items to the command line but you cannot change stack contents.

HP48 User's guide for more information about the interactive stack.

Expands clipboard contents on the stack and starts the interactive stack application. This menu key is active either in cell mode or in command mode. See the note 1. The clipboard is not erased until you exit from the current session of the spreadsheet.

Copying rows and columns

Adds a copy of the current row at the bottom of the worksheet. Works if flags 34 (,) and 36 ([]) are both clear. To duplicate the current row and keep it tight, press .

Adds a copy of all the rows below the current row at the bottom of the worksheet. Works if flags 34 (or) and 36 ([]) are both clear.

Adds a copy of the current column after the rightmost column. To duplicate the current column and keep it tight, press . Works if flags 35(or) and 36 ([]) are both clear.

Adds a copy of all the columns at the right side of the current column after the rightmost column. Works if flags 35 (or) and 36 ([]) are both clear.

Working with the anchor

Cell mode
Fixes the anchor cell. This menu key is

equivalent to [·]. Current anchor location can be obtained as a complex number by evaluating the function **Anchor**. The complex numbers returned by **Current Anchor** represent the opposite corners of a cluster.

Command mode
Pastes the object stored in the anchor cell with the text in the command line. This feature is handy for copying and modifying an object that must be stored in the active cell.

In both modes the anchor cell must be in a valid location otherwise [·] beeps.

[←] [·] Builds a reference to the anchor cell and pastes it at cursor's position. Works only in command mode. The type of reference created depends upon the position of the anchor cell, that is required. When the anchor cell is aligned with the active cell, the reference will be either **Row**(c) or **Col**(r), otherwise it will be explicit, of the type **Mat**(r,c).

The reference created is suitable for storing in the active cell. This menu function builds a reference whose path is the reverse of that built by [←] EDIT.

[→] [·] Builds a reference to the anchor cell and pastes it at cursor's position. The type of reference created is always implicit and requires the presence of the anchor cell. When the anchor cell is aligned with the active cell, the reference will be either **Row**(~+δc) or **Col**(~+δr),

otherwise it will be implicit, of the type **Mat(~+δ_r,~+δ_c)**. If the anchor cell is missing or invalid, a beep is issued.

The reference created is suitable for storing in the active cell. This menu function builds a reference whose path is the reverse of that built by [→] EDIT.

[ ] Copies anchor cell into the active cell. Works if flag 36 is clear ([⌨]).

[←] [ ] Builds a reference to the anchor cell and stores it into the active cell. Works if flag 36 is clear ([⌨]). The type of reference created is the same as for [←] [ ].

[→] [ ] Builds a reference to the anchor cell and stores it into the active cell. Works if flag 36 is clear ([⌨]). The type of reference created is the same as for [→] [ ].

Scrolling pages

ZOOM Zoom menu. In the first page there are six labels for eight functions which scroll the display, one *page* at a time, in any direction. The term page refers to the area of cells that fits the display window.

[↑] Scrolls the page up.

[←][↑] Scrolls the page down.

[↗] [↖] Scrolls the page diagonally.
[↘] [↙]

[←] Scrolls the page left.

[←] [→] Scrolls the page right.

Activating the sixth row

EXTRA NORM

Toggles the sixth row on or off. The flag that controls this setting is number 44. When flag 38 is set (access to CONFIG forbidden), you cannot change the current setting.

Filling up a cluster

Copies the object stored in the anchor cell into the area delimited by the anchor cell and the active cell. Works if flag 36 is clear ([ ]). This function allows you to fill a cluster with clones of the anchor cell. This capability turns out to be quite useful to replicate a formula or program along a row or a column.

Sorting and searching objects

SORT

Enters the *sorting and searching* sub-menu. This menu is split on two pages: the first page is dedicated to functions which change the order of the objects; the second page lists functions which move the cursor. Since the search/fill toggles affect the working of all these functions, you will find the labels GO→ and GO↓ in the last two places of this menu.

To restore the main menu you can press [→] [MENU].

All the sorting functions require the presence of a valid anchor cell in order to work. The anchor cell must be aligned otherwise the warning message Ambiguous command is displayed. If the active cell coincides with the anchor cell or the anchor cell is either invalid or undefined, a beep is issued.

SORT sorts isomorphic objects. SORT works with date and time objects as well as strings, global names, real numbers, integers, units and character objects. Flag 36 must be clear ([⌨]).

FSORT allows you to sort a range of cells by value. Expressions, programs, global names and local names are evaluated prior to start up the sort engine. If an expression cannot be resolved, the sort is aborted. Expressions, programs or global names must all return objects of the same type otherwise the process will be aborted. Flags 36 and 37 must be clear.

SKEY allows you to sort data using multiple keys. To define a sort key see the next entry. You can have an arbitrary number of keys and specify the order you prefer. You cannot enter the same key twice and a value out of range will cause a run time error. SKEY does not work with expressions. If no vector of keys is currently defined, the warning message No vector available is shown. Of course the i+1-th key will be taken into consideration only in case of tie between the i-th elements. Take into account that more keys you define, more time the sorting process will last.

[←] SKEY writes the command **SetKey** into the command line. It works only in command mode, because the vector should have been already typed in. The vector of keys is maintained in memory until you scratch it with **DelKey** or **WipeInfo**. To recall the vector to the stack use **GetKey**. Redundant values are forbidden.

[→] SKEY copies the current value of the vector of keys into the command line. Works either in cell or in command mode, provided that flag 36 is clear ([⌨]).

REV reverses the order of the objects along the current path. The path is determined by the reciprocal position of the active cell and the anchor cell, which must be aligned. When the path is undefined or ambiguous the warning message Ambiguous command appears.

ORDER rearranges the objects given a previously defined vector of positions. If no vector of positions is currently defined, the warning message No vector available is displayed. A vector of positions is a one-dimensional array where each value appears only once and the magnitude of each value cannot exceed the size of the array. This implies that all the rows (or columns) to reorder must be contiguous. The values are always relative to the lowest subscript: for instance, say you have delimited the range of rows from 3 to 7 and you want to exchange the third row with the sixth and the fourth with the seventh; the resulting vector would be [4 5 3 1 2].

← ORDER enables command mode and writes the command **SetOrder** into the command line. It works only in command mode. The vector is maintained in memory until you explicitly scratch it with **DelOrder** or **WipeInfo**. To recall the vector to the stack use **GetOrder**. A check is made to ensure that only valid positions are entered.

→ ORDER copies the current value of the vector of positions into the command line. Works either in cell or in command mode.

Searching objects

Omnibus allows you to search for objects in two different ways:

1 by looking at each cell sequentially along a line, using an arbitrary criterion. This method works well for objects in random order.

2 by applying a binary algorithm along a line where there are objects of the same type which have been ordered beforehand. The type of object you can employ in this search is restricted to a given set (see later on).

In both cases the direction of the search is determined by the fill/search direction flag. See also GO→ and GO↓.

It is worth noting that using a string as search key may produce different results, depending on the algorithm employed:

1 In the case of a sequential search the string is used as a sub-string, therefore the operation would terminate when the first string containing that sub-string is encountered. When a match cannot be found, the cell pointer is not moved and a double beep is issued. Flags 40 and 41, controlling the sensitivity to letter case and accented characters may influence the result.

2 In the case of a binary search, the status of flags is ignored and the strings are compared by their lexicographic value[1]. The search terminates when the closest string has been located. Settings of flags 40 and 41 are ignored.

SEARC Starts the binary search using the current search item. To define the binary search item,

1 The value of a string beginning with "Z" is lesser than that of a string beginning with "a"

see the next paragraph. The direction is established by GO→ and GO↓.

↰ SEARC Asks the user to enter the binary search key. The list of valid objects is given below:

- ☑ *real numbers*
- ☑ *binary integers*
- ☑ *strings*
- ☑ *global names (treated as strings)*
- ☑ *system binaries*
- ☑ *dates*
- ☑ *times*
- ☑ *characters*
- ☑ *tagged objects falling in the range above*

Omnibus takes the key specified by the user and looks for a set of contiguous objects of the same type. In order to begin the search process, the cursor must be inside the group of the objects being considered. The boundaries of the search area are automatically detected by Omnibus. At the end of the search, the cursor is placed either on the object that matches exactly the key or on the closest object. In the latter case, you can press +ROW or +COL to add a new line, preserving the existing order. As you can guess, in this way you can easily expand an ordered database without sorting the records.

GOTO Repeats the search using the last search function. To define or modify a search function see ↰ GOTO. The search begins after the current location and continues towards higher row/column numbers.

GOTO Asks the user to enter a cell location or a search item, then updates cursor's position if a match is found otherwise emits a beep.

Several choices are available. You can supply :

- *real numbers*
- *complex numbers*
- *strings*
- *characters*
- *expressions*
- *programs*
- *a sequence of commands returning one of the aforementioned objects*

A *real number* specifies a location relative to the current row or column (depending on the state of the fill direction). If the location lies outside the physical boundaries of the worksheet, a double beep is issued.

A *complex number* uniquely identifies a cell. GOTO jumps to the desired location if it lies inside the boundaries of the worksheet otherwise it beeps.

A *string* is treated as a sub-string and its presence is checked in strings, global names, messages and tags. If the tag is applied to a string object or to a global name, only the tag is matched, unless you have configured Omnibus to ignore tags ([ ]). Flag 40 determines whether the search is case sensitive ([Abc]) or insensitive ([ABC]). Flag 41 determines whether the search is accent sensitive ([öéç]) or insensitive ([oec]). The search is *restricted* to the current row or column depending on the state of the fill (search) direction and the versus is always towards higher row/column numbers.

A *character* is treated like a string but the match is restricted to the initial character of the target object. All the considerations made for strings apply to characters as well. To enter a character you must type a string containing only one letter. Take into account that the keystrokes [α] *A..Z* or [α] [↰] *A..Z* are provided as shortcuts for this operation.

When a function is supplied, the result of the function should be numeric and, best of all, boolean. This means that equalities or inequalites should be preferred. Again, the search is restricted to the current row or column and the versus is always towards higher row/column numbers. Special functions are provided to alleviate pains: see **Active** and **Value**. Note also that the recalculation does not affect the search.

Programs are suitable for searching objects like dates, times, binary integers and arrays. In this case it is up to you to write the procedure for comparing the objects. For instance say you want to find the first location after January 1-st 1993:

« **Value** 1.011993 **After?** »

The object used by GOTO could come as result of a sequence of commands. For instance, say you want to jump to the center of a large worksheet, leaving to Omnibus the task of doing all the necessary computations:

MaxRow MaxCol R→C 2 /

this sequence of commands returns a complex number which is handled as an absolute cell location, so that GOTO effectively jumps into the middle of the worksheet.

When an invalid search function or program is supplied (a program returning more than one object as result or no

result at all), the search is aborted and the error Invalid user function is issued. If the function or program makes a comparison between incompatible objects, the error is trapped and the search continues with the next object. Finally, if the program or function drops more objects than the stack holds, the error message Missing Stack Marker is issued and the search terminates.

Jumping to the anchor cell

GOEX This menu function allows you to move the cursor back to the anchor cell. If the anchor is missing or invalid the operation is canceled and a beep is issued.

Changing system settings

CONFI Enters the flag configuration sub-menu also called IconFlag. The meaning of each entry is fully described in the paragraph titled *IconFlag* starting from page 29. Works if flag 36 ([⌨]) and 38 (configuration lock) are both clear. When you enter CONFI the main menu of Omnibus is saved in the *Last Menu buffer*, therefore you can press [→] [MENU] to restore it.

Keyboard operations

As for menu entries, there are keys working differently in cell or in command mode. When the specification is missing, the entry works in cell mode only.

Keys having associated math functions are called *immediate functions*. For instance, [SIN] is a unary immediate function while [+] is a binary immediate function. Functions like the sigma, the integral and the derivative are not implemented as immediate functions.

Immediate functions apply themselves to the value contained in the cell to form a new expression. This means that you can see the actual value of the expression

only if recalculation is set or by pressing [EVAL] to force the evaluation.

Exiting from Omnibus

[ENTER] *Cell mode*

If pressed while the command line is empty, returns the current matrix to the stack and closes the application. [ENTER] beeps if the worksheet is empty.

Command mode

Executes the commands typed in the command line. In this fashion you can enter several elements into cells, according with the current status of the fill direction.

When the fill direction is set to *none* (GO→ and GO↓), you must enter one object at a time, otherwise only the last object will be held. Dimension locking flags (34 and 35) may restrict the number of objects you can enter. Flag 36 may prevent you from entering objects at all. See on page 29 for the details about flag status.

Note also that when flag 33 is clear ([ ]), any invalid object is substituted with the place holder '?' and a double beep is issued.

See also the entry [←] [=].

Calling the Equation Writer

[←] [ENTER]

This keystroke calls the Equation Writer. The Equation Writer is always started from scratch. To pass the content of the active cell to the Equation Writer use EQ↓ instead. If you press [ENTER] the expression is stored into the active cell, while it is discarded if you press [ON]. Works if flag 36 is clear.

Restoring the main menu

[→] [MENU]

This keystroke restores the main menu of Omnibus. The main menu of Omnibus is not saved in the Last Menu buffer when you recall system menus by pressing [PRG], [CST] or [←] [MEMORY], therefore this keystroke represents the only way to bring it back.

Quitting Omnibus

[ON]

Cell mode
abandons the current environment and brings you back to the topmost application (eventually to the *HP48* prompt when all applications have been terminated).

Refreshing the display

[ON]

Sometimes the display is frozen to give you the time to read a message or watch a result of an operation. To refresh the display area you can press [ON]. This situation occurs when you press [←] [▼], for instance.

Clearing the command line

[ON]

Command mode
clears the command line and restores cell mode. The active cell remains unchanged. If you split the text across two or more rows, the worksheet must be redrawn entirely and this may take a while.

[◄] [▲]
[▼] [►]

Moves the pointer (active cell) to the next cell. When preceded by [→] allows you to jump quickly to the boundary. The movement can be restricted by locking flag 34 or flag 35 or both ([⇩] [⇒] or [ALPHA]). In any case you cannot move the pointer beyond the first undefined row or column.

	CST	Retains its definition. Note however that you can customize CST in order to provide typing aids or menu driven extensions. Works if flag 36 is clear.
Switching to command mode	'	Enables command mode, sets algebraic entry mode and puts the cursor between two expression delimiters (tick marks). During algebraic entry mode, when you press operator keys like (+ ÷ $\sqrt{x}$ y^x), the operator is copied into the command line without adding leading and trailing spaces; the parentheses are automatically added to hard-wired functions and to library based functions.
Evaluating a cell	EVAL	Recalculates the active cell and shows the result in the most appropriate format. The recalculation takes place regardless of the status of flag 32. If the evaluation lock, flag 37, is set (⇦), EVAL merely displays the object in the cell, in the most appropriate format; this means that a program, an algebraic expression, or a global name are not evaluated. The following table lists the treatment reserved to different object types when the evaluation lock is disabled, that is after that recalculation takes place.

Type of result	How the object is rendered
Real and complex numbers, binary integers	Shown in the bottom line of the display using the current display mode.
Strings	Shown in the bottom line of the display, truncated to 22 characters, if no linefeed is found. Split across several lines if some linefeed is found, but always truncated to 22 characters per line, up to 7 lines.
Algebraic expressions	Passed through the Equation Writer and treated as a graphic object.
Date objects	The name of the day is shown in the currently selected language together with the date, using the current date display format (1.JAN or JAN 1), in the bottom line of the display.
Time objects	Shown in the bottom line of the display using the current time display format (12:00 PM or 24:00).
Character objects	Shown as Char_ddd, where ddd is the decimal code of the character.
System binaries	Returns the corresponding message, if any, provided that flag -58 is clear (CLEAR), otherwise the raw number is displayed using the current base format.

Type of result	How the object is rendered
Graphic objects, **PICT**	Shown graphically. In the case of graphic objects larger than 131x64 pixels, Omnibus would automatically enable display scrolling through the arrow keys [◄][▲][▼][►]. To leave the graphic environment and return to the spreadsheet, [ON] must be pressed.
Omnibus files, arrays and symbolic matrices	Passed to Omnibus recursively. Flag 36 is automatically set ([⊏⇕⊐]) upon entry.
FALSE and **TRUE** place-holders	Displays either **FALSE** or **TRUE**.
Any other object	Converted into text and treated as a string.

[EVAL] allows you to preview a value minimizing recalculation times and it is mainly used when the recalculation is off ([⊏⇄⊐]). [EVAL] is also helpful for reading full display messages or lists of objects. Works if recalculation lock flag is clear (flag 37). The result remains in the display until a key is hit.

[→NUM] Applies →**NUM** to the active cell. It is a handy shortcut to replace a formula or program with its result. Works if flag 36 and 37 are both clear. If an error occurs, the string containing the error message is returned.

Obtaining information about menu functions

[←] [▼] Mini help. It gives information in the current language about the menu functions available in cell mode. If no language library has been installed, the space for the messages remains blank. To install a language, follow the instructions given beginning on page 11.

Managing expressions

[←] [SWAP] Exchanges the topmost operands of an algebraic expression. Works if flag 36 is clear and the item is an expression. If the topmost operator of the expression is not binary, a beep is issued.

Immediate functions

Immediate functions are mathematical functions hard-wired to keys. When you press one of these keystrokes, Omnibus forms an algebraic expression by combining the object in the active cell with the desired function. If the function requires two arguments, Omnibus takes the object lying on the top of the stack as second argument; if the stack is empty a warning message appears. Of course immediate functions accept as arguments only the objects which can be included in symbolic expressions and reject others with a beep.

By setting flag 36 you can disable immediate functions.

[SIN] Applies **SIN** to the active cell.

[←] [ASIN] Applies **ASIN** to the active cell.

[COS] Applies **COS** to the active cell.

[←] [ACOS] Applies **ACOS** to the active cell.

[TAN] Applies **TAN** to the active cell.

[←] [ATAN] Applies **ATAN** to the active cell.

[$\sqrt{x}$] Applies $\sqrt{}$ to the active cell.

[←] [x^2] Applies **SQ** to the active cell.

[→] [$\sqrt[x]{y}$] Applies **XROOT** to the active cell.

[y^x] Applies ^ to the active cell.

[←] [10^x] Applies **ALOG** to the active cell.

[→] [LOG] Applies **LOG** to the active cell.

[$1/x$] Applies **INV** to the active cell.

[←] [e^x] Applies **EXP** to the the active cell.

[→] [ATAN] Applies **LN** to the active cell.

[+/-] Applies **NEG** to the active cell.

[÷] Divides the value on the stack by the active cell. To swap the arguments press [←] [SWAP].

[×] Multiplies the value on the stack by the active cell.

[−] Subtracts the active cell from the value on the stack. To swap the arguments press [←] [SWAP].

[+] Adds the value on the stack to the active cell.

Editing an object in the command line

[↰] [EDIT] Copies cell contents to the command line and enables command mode. Works if flag 36 is clear. If the item is an algebraic expression algebraic entry mode is set. Analogously program entry mode is set when a program is being edited. Dates, time, characters and system binaries are converted into real numbers and copied into the command line, then, if flag 43 is clear ([→»]), the related conversion command is appended to the stream of characters.

Copying a cluster to the stack

[EEX] Copies the region of cells (cluster), delimited by the active and the anchor cell, to the stack (clipboard). Works in command mode. Both corners must fall within the physical boundaries of the worksheet. If the anchor cell is empty or missing beeps.

Retrieving the coordinates of the anchor cell

[↰] [EEX] Copies the coordinates of the anchor cell into the command line at the cursor position. The coordinates are separated either by a comma or by a semicolon, whichever applies to the current fraction mode setting. To change this setting open [CONFIG] and press the key whose icon is [0,11] or [0.11]. Note also that the pair of numbers is not output like a complex number; this makes easier to include the coordinates as function arguments. Of course the anchor cell must be defined at the time being. Works in command mode

Replacing a cluster — → EEX

Replaces a portion of the matrix with a sub-matrix taken from the top of the stack (clipboard). Works if flag 36 is clear. The upper left corner of the overlaid region must coincide with the active cell. If the cluster is larger than the area of cells being overlaid, the worksheet is resized accordingly, provided that flags 34 and 35 allow the operation. Hollow clusters are rejected in any case. Typically a cluster has been extracted with EEX beforehand. You can use the clipboard to pass a cluster from one worksheet to another, as described on page 77.

Deleting the anchor — DEL

Disables the anchor. To move the anchor from one location to another one, you can press [redacted] or . directly.

Copying an object from the stack — ←

Moves the object from the top of the stack to the active cell. If the active cell falls outside the physical boundaries, the worksheet is resized. If the active cell already contains an object, the new object takes the place of the old one. Works if flag 36 is clear. If the stack is empty a warning message appears.

Undoing an immediate function — ↶ ←

Deletes the topmost function and the auxiliary argument from an expression. If the function being removed is unary, it removes the function but leaves the operand intact. If the function is binary removes the operator and the left hand operand. It works like an *undo* feature

for all the immediate functions. Works if flag 36 is clear and the cell contains an expression.

Clearing a cluster — [→][←] Clears the cluster of cells delimited by the active cell and by the anchor cell. If the anchor cell does not exist or its position is invalid, [→][←] beeps. Works if flag 36 is clear. If flag 33 is set ([⌂]), [→][←] replaces the cluster with empty cells, otherwise with zeros.

Angle modes — [←][RAD] Toggles between degree angle mode and radians angle mode, thereafter the display is redrawn. Works either in cell or in command mode. To watch which mode is currently in effect, go to the second page of CONFI where you will find [∠R], [∠45] or [∠50]. *Grads* can be set through CONFI only.

Coordinate modes — [→][POLAR] Toggles between polar coordinates and rectangular coordinates, thereafter the display is redrawn. Works either in cell or in command mode. To watch which mode is currently active or to change it, go to the second page of CONFI where you will find [↳R], [↳Z] or [↳θ]. To switch between cylindrical coordinates and spherical coordinates, you must resort to CONFI.

Restoring the last menu — [→][MENU] Restores the last menu. It is used to return to the main menu from the ZOOM, SOLV and CONFI, sub-menus. See also [→][MATRIX].

Setting the anchor cell — [·] This key works as [.] in cell mode. Current anchor location can be obtained as a complex

number by evaluating the function **Anchor**. The complex numbers returned by **Current Anchor** represent the opposite corners of a cluster. The pointer must be in a valid location otherwise [.] beeps. See also the entries [←][EEX] and [▪] for the operations related to the anchor cell.

Executing commands

[→][=] This key allows you to push objects on the stack or execute commands without disturbing the spreadsheet. Once you have abandoned Omnibus, the objects left on the stack can be recalled in form of a list by executing the command **Clipboard**.

[←][=] is also used to temporarily push onto the stack the left hand argument for operators like [×] [+] [y^x].

Search by character

[α]*A..Z* These keystrokes allow you to move the cursor [α][←] to the nearest cell (in the current search *A..Z* direction) containing a string, character, message or tag whose initial character matches the letter. The search is always forward and it is affected by flags 40, 41, 45 and -59.

Shortcuts

[α][→] It is possible to assign a procedure to a key of [ ] the sixth plane. This kind of assignment, called *shortcut*, is reserved for the execution of search functions. Each shortcut can be one of the objects accepted by [GOTO], listed on page 87 plus *list objects*.

A list object represents a sequence of commands that must be executed soon, before starting the search. This

procedure should return an object falling in the range of the aforementioned objects or no object at all.

☞ Shortcuts are active only in *cell mode*.

To assign a shortcut key, run the utility SHORT, stored on the card. Shortcuts work if flag 36 is clear.

If you assign the following definition to a key, you will get the same result as executing GO+:

{ **Anchor** }

The file named PLANE6, stored on the auxiliary disk, contains a few definitions and is provided as a sample of shortcuts programming. To store it into memory, use the command **StoShortcuts**, to recall it, execute **RclShortcuts**. Shortcuts can be erased with the command **DelShortcuts**. Key assignments recorded in PLANE6 are described in the following table; each cell represents a key of the *HP48*.

PLANE6					
PRINT		SYSINFO	PRLCD		
		LCD→			
		RPLMAN			
	Char "7"	Char "8"	Char "9"		
	Char "4"	Char "5"	Char "6"		
	Char "1"	Char "2"	Char "3"		
	Char "0"	Anchor	EXPORT		

The numeration of the cells starts in the upper left corner with 1 and ends in the lower right corner with the last white cell (number 49). The cells in light grey correspond

to the shift keys. Although you can assign a definition to a shift key, there is no chance to use it.

If the procedure returns current cell's location, it is possible to execute a program or a sequence of commands that has nothing to do with cursor movement. In this fashion you have a whole plane of keys available for customization.

Keep in mind that the command **Current** returns the active cell location, therefore you could provide a dummy cursor movement in all the cases where you need to leave some object on the stack.

For instance, in the following case:

{ LCD→ Current }

LCD→ returns a snapshot of the screen to the stack in form of a graphic object. Since the shortcut handler program expects an object to use a search key, by adding **Current** you preserve the grob from being discarded as an invalid object.

Numeric keypad

Numeric keys [0]...[9] enable command mode implicitly.

All other keys retain their standard definition.

Programmable commands

The following chapter lists all the RPL keywords provided with Omnibus in alphabetical order. Commands dealing with time management are not treated here but in a specific section of the chapter titled *Time management*, starting from page 151.

Stack diagrams This is the template for a generic entry:

NAME *(reminder)*

n	...	2	1	→	m	...	2	1
arg_n	...	arg_2	arg_1		$result_m$	...	$result_2$	$result_1$

Type The class to which the command belongs (command, function, operator). Functions and operators are usable inside algebraics while commands are not. The difference between functions and operators is in the algebraic syntax:

'**Function**(X,Y,...)' ↔ 'X **operator** Y'

Operators provided by Omnibus are all binary.

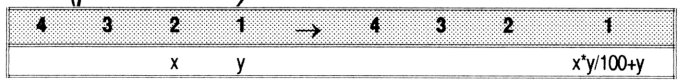
Scope Range of action of the command.

Remarks In this paragraph are summarized the notes regarding the command.

See also List of related commands in alphabetical order.

Example This paragraph (if any) ends the description.

%A *(percent add)*



Type Operator

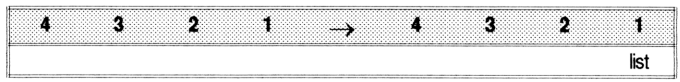
Scope Anywhere

Remarks Adds the percentage x of y to y. When Omnibus is plugged in you can write 'X **%A** Y'.

See also **%CH, %D, %T, %**

Example '5 %A 20' gives 21

Above



Type Function

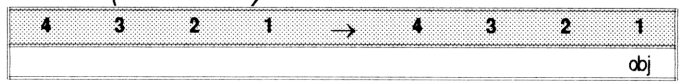
Scope Inside Omnibus

Remarks Returns a list of all the objects above the active cell (belonging to the current column) without evaluating them.

See also **AsIfCol, AsIfRow, Left**

Example

Active *(Active cell)*



Type Function

Scope Inside Omnibus

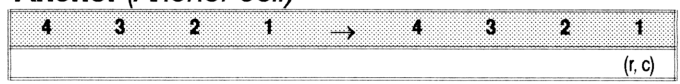
Remarks Returns the content of the active cell unevaluated. It is mainly employed in search functions because it represents what is contained in the target cell.

See also **Value**

Example Searching for empty cells:

- ◀ ↰ GOTO
- ◀ « **Active** FALSE SAME » ENTER

Anchor *(Anchor cell)*



Type Function

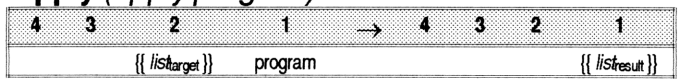
Scope Inside Omnibus

Remarks Returns the anchor cell location as complex number. If the anchor cell is missing, it returns (0,0).

See also **Current**

Example See on page 100.

Apply *(Apply program)*



Type Command

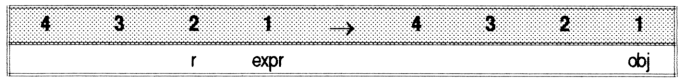
Scope Anywhere

Remarks Applies the program to each element of the symbolic matrix.

Example

```
{{ 1 2 3 } { 4 5 6 }}
« INV →Q » Apply
{{ 1 '1/2' '1/3' } { '1/4' '1/5' '1/6' }}
```

AsIfCol



Type Function

Scope Inside Omnibus

Remarks Evaluates the expression as if it were in the row specified of the same column. The execution of functions like **Above**, **Sum**, **Rarray**, etc. is influenced by **AsIfCol**. Any expression containing implicit references is affected as well.

See also **AsIfRow, AsIn**

Example See the example for **AsIfRow**.

AsIfRow

4	3	2	1	→	4	3	2	1
		c	expr					obj

Type Function

Scope Inside Omnibus

Remarks Evaluates the expression as if it were in the column specified of the same row. The execution of functions like **Above**, **Sum**, **Rarray**, etc. is influenced by **AsIfCol**. Any expression containing implicit references is affected as well.

See also **AsIfCol, AsIn**

Example

AsIn

4	3	2	1	→	4	3	2	1
		r	c					obj

Type Function

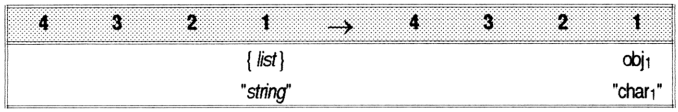
Scope Inside Omnibus

Remarks Returns the value of the expression stored in the location specified as if it were executed in the current cell.

See also **AsIfCol, AsIfRow**

Example See the example on page 219.

CAR



Type Function

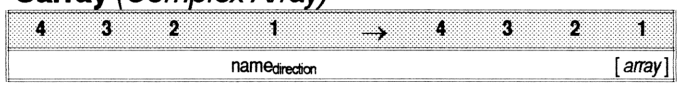
Scope Anywhere

Remarks Returns the first element of a list or the first character of a string. If the input object is null, it is returned as it is.

See also **CDR**

Example { 1 2 3 } **CAR** → 1
"123" **CAR** → "1"

Carray *(Complex Array)*



Type Function

Scope Inside Omnibus

Remarks Returns an array of complex or real values extracted by the current row or column depending on the direction. The direction must be either an up arrow ↑ or a left arrow ←. **Carray** dinamically detects wrong object types and collects the nearest cluster of complex or real values into a complex or real array. The array is always one-dimensional. A real array is returned only if complex numbers have not been found.

See also **Rarray**

Example

CDR

4	3	2	1	→	4	3	2	1
			{ *list* }					{ *list*remainder }
			"*string*"					"*string*remainder"

Type Command

Scope Anywhere

Remarks Returns all the but the first object of a list or all but the first character of a string. If the input object is null, it is returned as it is.

See also **CAR**

Example { 1 2 3 } **CDR** → { 2 3 }
"123" **CDR** → "23"

Cell

4	3	2	1	→	4	3	2	1
		r	c					obj

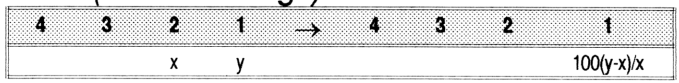
Type Function

Scope Inside Omnibus

Remarks Returns the content of cell specified unevaluated. You can specify the tilde '~' as current row or current column.

See also **Mat**

%CH *(Percent change)*



Type Binary operator

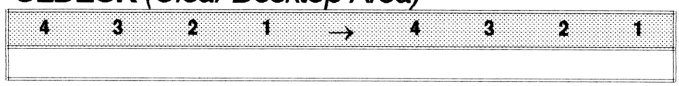
Scope Anywhere

Remarks Returns the percentual change between x and y. Replaces the built-in function with an operator that improves legibility. When Omnibus is plugged in you can write 'X **%CH** Y' instead of '**%CH**(X,Y)'.

See also **%A, %D, %T, %**

Example '5 %CH 20' gives 300

CLDESK *(Clear Desktop Area)*



Type Command

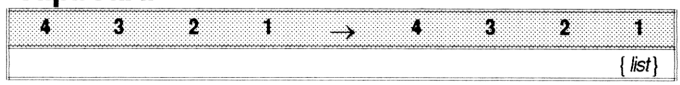
Scope Anywhere

Remarks Clears the stack area of the display (also called *desktop area* by Omnibus). This area is beneath the status area and above the menu row. The command does not perform **FREEZE** when it is executed by the command line.

Example CLDESK 2 FREEZE

clears the desktop area and freezes the display until a key is hit.

Clipboard



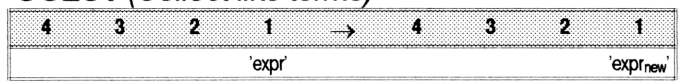
Type Command

Scope Anywhere

Remarks Returns a list containing the objects left on the stack during the previous call to **MATWRT** (Omnibus).

See also **Scratch, WipeInfo**

COLCT *(Collect like terms)*



Type Command

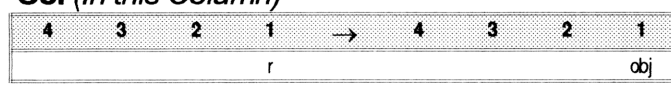
Scope Anywhere

Remarks Collects like terms. It works like the built-in command but it handles special cases involving spreadsheet functions.

Example

```
{{ '10 %D Row(1)' '19 %A Row(2)' 'Row(3)+5' }}
« { 'Row(&)' 'Row(&+1)' '&>1' } ↓MATCH DROP
COLCT » Apply
{{ '10 %D Row(1)' '19 %A Row(3)' 'Row(4)+5' }}
```

Col *(In this Column)*



Type Function

Scope Inside Omnibus

Remarks Retrieves a value from the current column.

See also **Mat, Row**

Ctag? *(Column tag ?)*

4	3	2	1	→	4	3	2	1
			name					c

Type Function

Scope Inside Omnibus

Remarks Locates the tag in the current column. Only tagged objects are searched and the search key must match the tag exactly. Since the argument of the function is a global name and not a string, tags containing characters such as commas, colons, spaces, etc. cannot be searched.

Example

Current

4	3	2	1	→	4	3	2	1
								(r, c)

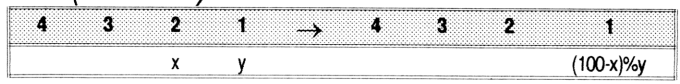
Type Function

Scope Anywhere

Remarks Returns a complex number representing active cell's coordinates.

See also **Anchor**

%D *(Discount)*



Type Binary operator

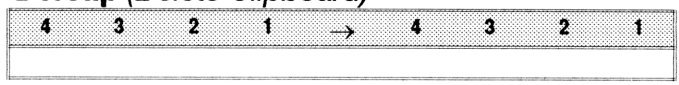
Scope Anywhere

Remarks Returns the value y discounted by x.

See also **%A, %CH, %T, %**

Example '5 %D 20' gives 19

DelClip *(Delete Clipboard)*



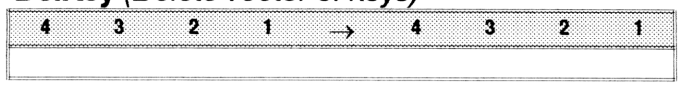
Type Command

Scope Anywhere

Remarks Erases the clipboard.

See also **Clipboard, Scratch, WipeInfo**

DelKey *(Delete vector of keys)*



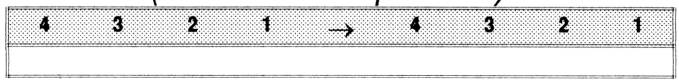
Type Command

Scope Anywhere

Remarks Deletes the current vector of keys.

See also **GetKey, Scratch, SetKey, WipeInfo**

DelOrder *(Delete vector of positions)*



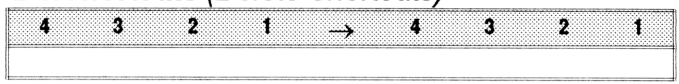
Type Command

Scope Anywhere

Remarks Deletes the current vector of positions.

See also **GetOrder, Scratch, SetOrder, WipeInfo**

DelShortcuts *(Delete Shortcuts)*



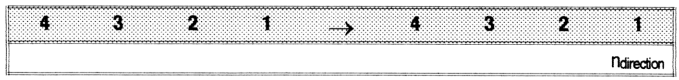
Type Command

Scope Anywhere

Remarks Deletes the shortcuts from memory.

See also **RclShortcuts, StoShortcuts, Scratch, WipeInfo**

Direction



Type Function

Scope Anywhere

Remarks Returns the current fill or search direction.

0 = horizontal (GO→)
1 = vertical (GO↓)
2 = none (GO→ and GO↓)

See also **SetDirection**

Eval *(Evaluate)*

4	3	2	1	→	4	3	2	1
			obj					

Type Command

Scope Anywhere

Remarks Evaluates the object on the stack. Unlike **EVAL** does not evaluate lists.

→**File** *(Stack to file)*

5	4	3	2	1	→	4	3	2	1
{{*worksheet*}}	{ *list* }	(r_3,c_3)	(r_2,c_2)	(r_1,c_1)					*file*

Type Command

Scope Anywhere

Remarks Returns a library data object (**TYPE** = 26) representing a worksheet with embedded configuration data.

The five arguments are as follows:

Term	Description
file	A library data object. It is automatically generated by **MATWRT** when flag 33 is set ([⍁]).
{{*worksheet*}}	A symbolic matrix representing a worksheet.
{*list*}	A list containing:

Term	Description
{ *#sys #user* }	☑ A list of two binary integers representing system and user flags; the format is the same as that returned by **RCLF**.
(*width, direction*)	☑ A complex number representing column width and fill direction.
(r_3, c_3)	The location of the *anchor* cell.
(r_2, c_2)	The location of the cell in the *upper left corner* of the display.
(r_1, c_1)	The location of the *active* cell.

File→ *(file to stack)*

4	3	2	1	→	5	4	3	2	1
			file		{{*worksheet*}}	{*list*}	(r_3,c_3)	(r_2,c_2)	(r_1,c_1)

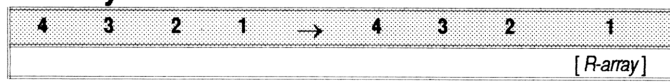
Type Command

Scope Anywhere

Remarks Splits a worksheet onto the stack.

File→ accepts only Library data objects, in Omnibus format, as argument. For a description of the parameters returned by this command, please refer to the keyword →**File**.

GetKey



Type Command

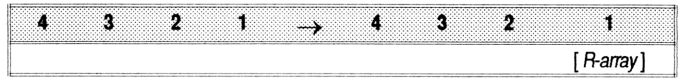
Scope Anywhere

Remarks Retrieves the current vector of keys. The command may issue the following error:

No vector available	If no vector of keys is currently defined.

See also **DelKey, Scratch, SetKey, WipeInfo**

GetOrder



Type Command

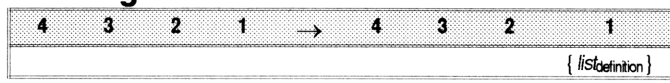
Scope Anywhere

Remarks Retrieves the current vector of positions. The command may issue the following error:

No vector available	If no vector of positions is currently defined.

See also **DelOrder, Scratch, SetOrder, WipeInfo**

IconFlag



Type Command

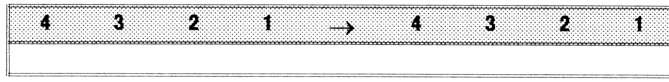
Scope Anywhere

Remarks Returns the list containing the default configuration menu (also called Iconflag) to the stack. The list is in a format suitable for **TMENU**.

When you are inside Omnibus, this menu is accessible through the menu key CONFI.

See on page 29 for a detailed description of each menu entry.

IDLE



Type Command

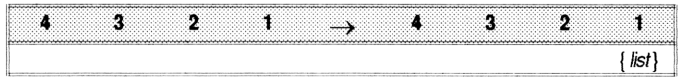
Scope Anywhere

Remarks Suspends program execution until an interrupt occurs.

The type of the interrupt determines what happens later on:

Type of interrupt	Effect
Keypress	The program resumes the execution while the key is kept in the keyboard buffer.
ON	The program resumes the execution.
Auto power off	The *HP48* is turned off and the program resumes the execution when the *HP48* is turned on again.
Alarm	The program resumes the execution and the alarm is serviced when the program ends.

Left



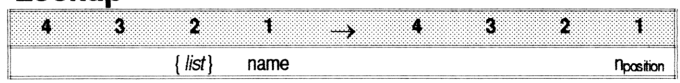
Type Function

Scope Inside Omnibus

Remarks Returns a list of all the objects on the left side of the active cell (belonging to the current row) without evaluating them.

See also **Above, AsIfCol, AsIfRow**

Lookup



Type Command

Scope Anywhere

Remarks Returns the position of the object whose tag is *name*. Since *name* cannot contain spaces or punctuation characters, tags which do not meet these requirements cannot be found. The search begins from the bottom of the list and proceeds backwards.

Example

```
{ :A:12 :B:34 :C:56 } 'B'
Lookup
2
```

MATWRT *(Matrix Writer)*

4	3	2	1	→	4	3	2	1
			worksheet				*worksheet*	1
			worksheet					0

Type Command

Scope Anywhere

Remarks This command starts Omnibus. A detailed description of the configuration and usage of Omnibus can be found in chapter titled *Working with Omnibus*.

The following table summarizes input and output conditions assuming that ENTER has been pressed. The object in level 1 is always the boolean value 1, under these circumstances.

INPUT		OUTPUT
Level 1	**Flag 33**	**Level 2**
[[*matrix*]]	[▦]	[[*matrix*]] or {{*matrix*}}
	[🖹]	*file*
{{*matrix*}}	[▦]	[[*matrix*]] or {{*matrix*}}
	[🖹]	*file*
file	[▦]	[[*matrix*]] or {{*matrix*}}
	[🖹]	*file*

The type of matrix returned when flag 33 is clear ([▦]) depends on the presence of certain object types. If the worksheet contains only real numbers, a real array is returned. If the worksheet contains a mixture of real numbers and complex numbers, a complex array is returned. In all other cases, a symbolic matrix is returned.

When the ON key is pressed, the worksheet is discarded and the boolean value 0 is returned to the stack.

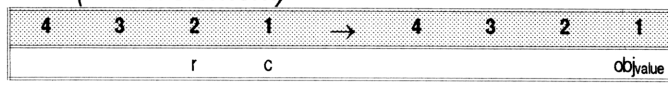
The command may issue the following error:

Can't edit null array	When any of the flags 34, 35 or 36 is set and you try to start Omnibus from scratch.

See also **File→, →File**

Example Since **MATWRT** returns a boolean value as result, it is common to find this command placed inside a IF-THEN-ELSE-END clause. See the source code of programs starting from page 173.

Mat *(Matrix element)*



Type Function

Scope Anywhere

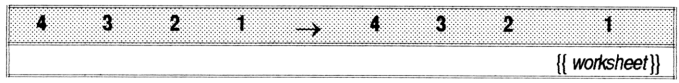
Remarks Returns the value of the cell specified. You may include the tilde ~ to specify the current row or the current column. For value we mean *any* object returned after the evaluation of the contents of the cell referenced by **Mat**, not only numbers. The function may issue the following errors:

Subscript Out Of Range	If you specify an invalid or not existent cell location.
Any other error	Occurring during the evaluation of the procedure stored in the target cell.

See also **Col, Row**

Example The expression '**Mat**(~-1,~)' is equivalent to 'Col(~-1)' and indicates the value of the cell immediately above the current cell, in the same column.

Matrix

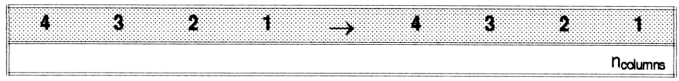


Type Function

Scope Inside Omnibus

Remarks Returns the worksheet to the stack. The worksheet is always returned in form of a symbolic matrix.

MaxCol



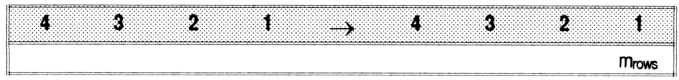
Type Function

Scope Inside Omnibus

Remarks Returns the number of columns in the worksheet.

See also **MaxRow**

MaxRow



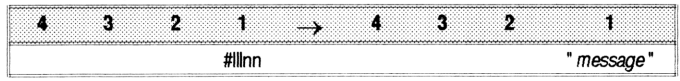
Type Function

Scope Inside Omnibus

Remarks Returns the number of rows in the worksheet.

See also **MaxCol**

→MSG$



Type Command

Scope Anywhere

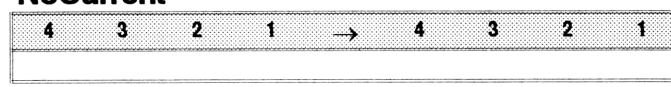
Remarks Returns the corresponding library message in form of a string. The input value must be a binary integer where the three digits lll (in hex format) represent the library number and the remaining two digits (in hex format) represent the message number.

If the library is missing or the message number is out of range, →**MSG$** returns a null string.

Example

```
#33508h
→MSG$
"Redundant subscript"
```

NoCurrent

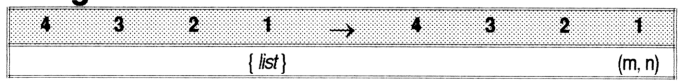


Type Command

Scope Inside Omnibus

Remarks Cancels the internal pointer to the current cell. It prevents dummy circular references in user programs launched from the command line.

Range



Type Command

Scope Anywhere

Remarks Returns a complex number representing the locations of the first (*m*) and the last object (*n*) of a homogeneous set of objects comprised in the category given below, in the input list. The search begins from the last object and proceeds backwards. When no valid object is found, **Range** returns (0,0). Typically the input list has been extracted with **Left** or **Above**.

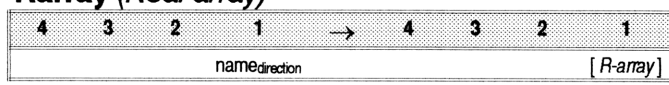
Valid objects are:

- ☑ *real numbers*
- ☑ *complex numbers*
- ☑ *global names*
- ☑ *local names*
- ☑ *expressions*
- ☑ *units*

Example

```
{ "string" 1 2 X "string" }
Range
(2,4)
```

Rarray *(Real array)*



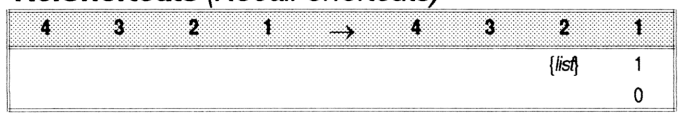
Type Function

Scope Inside Omnibus

Remarks Returns an array of real values extracted by the current row or column depending on the symbolic direction. The direction must be either an up arrow ↑ or a left arrow ←. **Rarray** dinamically detects wrong object types and collects the nearest cluster of real values into a real array. The array is always one-dimensional.

See also **Carray**

RclShortcuts *(Recall Shortcuts)*



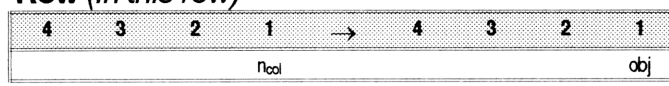
Type Command

Scope Anywhere

Remarks Recalls the keyboard shortcuts from memory. See on page 100 for further details about keyboard shortcuts.

See also **DelShortcuts, StoShortcuts, Scratch, WipeInfo**

Row *(In this row)*



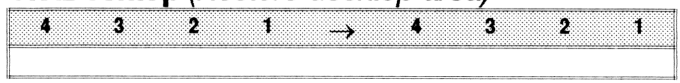
Type Function

Scope Inside Omnibus

Remarks Retrieves a value from the current row.

See also **Col, Mat**

RstDesktop *(Restore desktop area)*



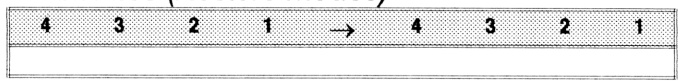
Type Command

Scope Anywhere

Remarks Restores the display grob saved with **SaveDesktop** and other parameters like the column width, the fill direction, system and user flags.

See also **SaveDesktop, Scratch, WipeInfo**

RstModes *(Restore modes)*



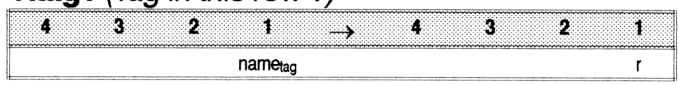
Type Command

Scope Anywhere

Remarks Restores the settings saved with **SaveModes**. Saved parameters are the column width, the fill direction, system and user flags.

See also **SaveModes, Scratch, WipeInfo**

Rtag? *(Tag in this row ?)*



Type Function

Scope Inside Omnibus

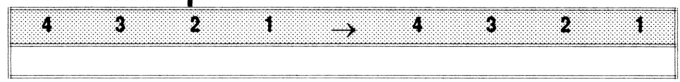
Remarks Locates the tag in the current row. Only tagged objects are searched and the search key must match the tag

exactly. Since the argument of the function is a global name and not a string, tags containing characters such as commas, colons, spaces, etc. cannot be searched.

See also **Ctag?**

Example See the example for **Ctag?**.

SaveDesktop



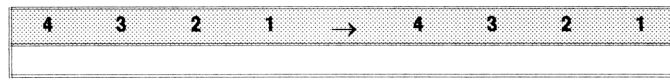
Type Command

Scope Anywhere

Remarks Saves the current display grob together with the current column width the fill direction, system and user flags.

See also **RstDesktop**

SaveModes



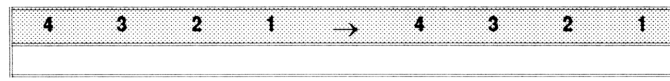
Type Command

Scope Anywhere

Remarks Saves the current Omnibus settings like the current column width the fill direction, system and user flags.

See also **RstModes**

Scratch



Type Command

Scope Outside Omnibus

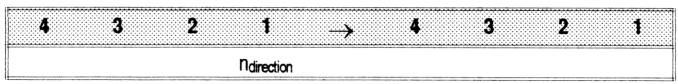
Remarks Deletes all Omnibus environmental variables like the:

- ☑ *search function*
- ☑ *shortcuts*
- ☑ *holidays*
- ☑ *current vector of keys*
- ☑ *current vector of positions*
- ☑ *clipboard*
- ☑ *saved stack items*

It must be used when Omnibus is not running to free memory.

See also **WipeInfo**

SetDirection



Type Command

Scope Anywhere

Remarks Set the fill or search direction according to the following table:

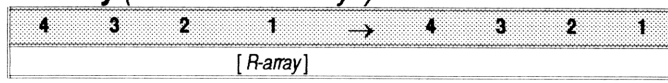
0 horizontal (GO→)
1 vertical (GO↓)
2 none (GO→ and GO↓)

The command may issue the following error:

Bad Argument Type	if the argument is not a real number.
Bad Argument Value	if the argument is not 0,1 or 2.

See also **Direction**

SetKey *(Set vector of keys)*



Type Command

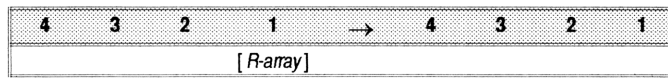
Scope Anywhere

Remarks Stores the current vector of keys. The vector must be a real-valued one-dimensional array. Each subscript must be greater than zero and must appear only once. The command may issue the following errors:

Invalid dimension	If the array is not a vector.
Real values required	If the vector contains complex values.
Redundant subscript	If two or more identical subscripts have been found.
Subscript Out Of Range	If a subscript equal to zero has been found.

See also **DelKey, GetKey**

SetOrder



Type Command

Scope Anywhere

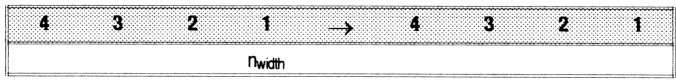
Remarks Stores the current vector of keys. The vector must be a real-valued one-dimensional array containing unique subscripts whose value must be greater than 1 and not larger than the size of the array. The command may issue the following errors:

Invalid dimension	If the array is not a vector.
Real values required	If the vector contains complex values.

Redundant subscript	If two or more identical subscripts have been found.
Subscript Out Of Range	If a subscript equal to zero or greater than the size of the array has been found.

See also **DelOrder, GetOrder**

SetWidth



Type Command

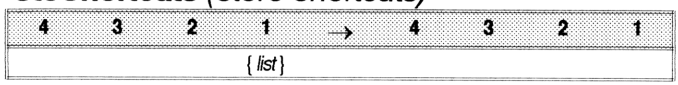
Scope Anywhere

Remarks Sets the current width of columns. Valid numbers are in the range 1-6 and 10. The command may issue the following error:

Bad Argument Type	if the argument is not a real number.
Bad Argument Value	if the argument is not 1-6 or 10.

See also **Width**

StoShortcuts *(Store Shortcuts)*



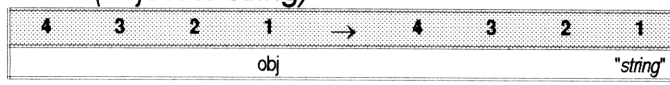
Type Command

Scope Anywhere

Remarks Stores the keyboard shortcuts into memory. See on page 100 for further details about keyboard shortcuts.

See also **DelShortcuts, RclShortcuts, Scratch, WipeInfo**

→Str *(object to string)*



Type Command

Scope Anywhere

Remarks Converts an object into a string. →**Str** converts object types that →**STR** does not recognize, plus all the other standard objects.

New object types recongized by →**Str**:

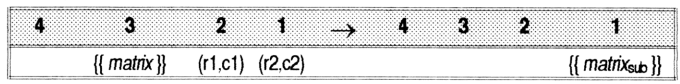
- ☑ *characters*
- ☑ *dates*
- ☑ *times*

Example

31 →Char
→**Str**
"..."

13 R→T
→**Str**
"13:00:00" (24+00) or "01:00:00P" (12 PM)

SUBMAT



Type Command

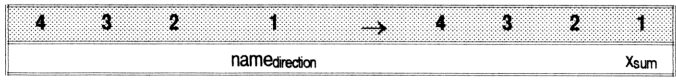
Scope Anywhere

Remarks Returns the sub-matrix delimited by the opposite corners.

Example

{{ 1 2 3 }{ X Y Z}} (1,2) (2,3)
SUBMAT
{{ 2 3 }{ Y Z }}

Sum



Type

Function

Scope

Inside Omnibus

Remarks

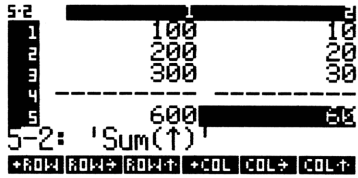
Returns the sum of the values found in the current row or current column as specified by the symbolic direction. The direction must be either an up arrow ↑ or a left arrow ←. The result can be symbolic. The function determines dynamically the range of values to add, discarding invalid objects until the first valid object is found. The list of valid objects follows:

- ☑ *real numbers*
- ☑ *complex numbers*
- ☑ *global names*
- ☑ *local names*
- ☑ *expressions*
- ☑ *units*

When no valid object is found, **Sum** returns 0.

Note that in order to get a valid result, when using unit objects, all the terms of the sum must be dimensionally consistent. This fact implies that **Sum** would return an error if meters are added to square meters or to bare numbers.

Example



%T *(Percent of total)*

4	3	2	1	→	4	3	2	1
		x	y					100y/x

Type Binary operator

Scope Anywhere

Remarks Returns the percent of the total x respect to y. Replaces the built-in function with an operator that improves legibility. When Omnibus is plugged in you can write 'X **%T** Y' instead of '**%T**(X,Y)'.

See also **%A, %CH, %D, %**

Example '5 **%T** 20' gives 400

TF *(Toggle flag)*

4	3	2	1	→	4	3	2	1
		0/1	n_{flag}					

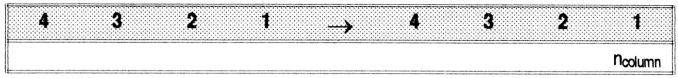
Type Command

Scope Anywhere

Remarks Toggles the flag on or off, depending on the value specified. A value of zero clears the flag, any other value sets the flag. Positive flag numbers must be in the range 1

to 64 and refer to *user flags*. Negative flag numbers must be in the range -64 to -1 and refer to *system flags*.

ThisCol



Type Function

Scope Inside Omnibus

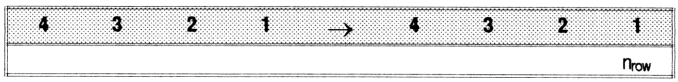
Remarks Returns a real number representing the current column. The value is computed dynamically and does not necessarily coincide with the column where the active cell is. Note that the following two expressions are equivalent:

Row(**ThisCol**-1) Row(~-1)

For this reason the function is mainly used within programs where the place holder ~ cannot be employed.

See also **ThisRow**, ↑↓, ←→

ThisRow



Type Function

Scope Inside Omnibus

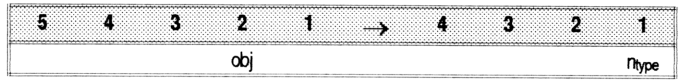
Remarks Returns a real number representing the current row. The value is computed dynamically and does not necessarily coincide with the row where the active cell is. Note that the following two expressions are equivalent:

Col(**ThisRow**-1) Col(~-1)

For this reason the function is mainly used within programs where the place holder ~ cannot be employed.

See also **ThisCol**, ↑↓, ←→

TYPE

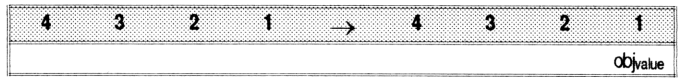


Type Command

Scope Anywhere

Remarks Returns a number representing the object type. Extends the built-in command **TYPE**, which returns numbers in the range 0-27, by adding the number 28 for date objects and 29 for time objects.

Value



Type Function

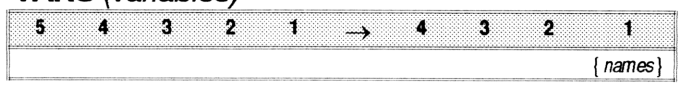
Scope Inside Omnibus

Remarks Returns the value of the current cell.

See also **Active**

Example See on page 88.

VARS *(variables)*



Type Command

Scope Anywhere

Remarks Returns a list containing the names of the variables of the current directory. It replaces the built-in command **VARS** providing a significant improvement in terms of speed for large directories.

Whatlf

Type Function

Scope Inside Omnibus

Remarks Returns a matrix where each row contains the values of the functions specified by the symbolic range, computed using the current value of the variable specified, starting from *begin* to *end*, at intervals of *step*.

The algebraic form is **Whatlf**(var=begin,end,step,direction)

the argument *direction* must be either an up arrow ↑ or a left arrow ←. The expressions returning non-numeric values will not be collected into the array.

☞ The resulting matrix is stored in ΣDAT.

The range of functions is determined dynamically. **Whatlf** looks for the nearest set of functions in the direction specified, then evaluates each function incrementing, or decrementing, the counter variable after each iteration. Of course each function should be parameterized respect to the counter variable, otherwise a constant value would be returned. The resulting matrix is always two-dimensional and has got as many columns as the functions to evaluate are.

The command may issue the following error:

Undefined name	If any of the expressions returns a non-numeric value.

Example The worksheet \EXAMPLES\WHATIF.WKS allows you to create a discount table given the following parameters:

- minimum discount
- maximum discount
- basic price

The expression in the active cell,

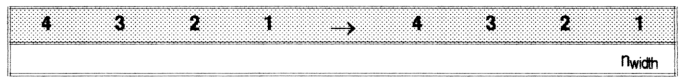
'**WhatIf**(k=Col(1),Col(2),1,↑)'

returns the matrix on the right side of the page when it is evaluated, given the values shown in the figure.

Discount	Price
10	94.5
11	93.45
12	92.4
13	91.35
14	90.3
15	89.25
16	88.2
17	87.15

Adding a column in the table is fairly easy: Just add an entry in the set of functions to evaluate.

Width



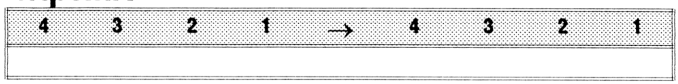
Type Function

Scope Anywhere

Remarks Returns the number of columns in the display.

See also **SetWidth**

WipeInfo



Type Command

Scope Anywhere

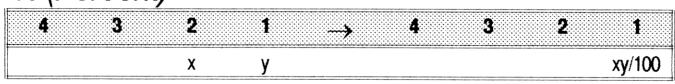
Remarks Deletes Omnibus environmental variables like:

- ☑ *search function*
- ☑ *shortcuts*
- ☑ *holidays*
- ☑ *current vector of keys*
- ☑ *current vector of positions*
- ☑ *clipboard*

It can be used when Omnibus is running to reclaim memory.

See also **Scratch**

% (Percent)



Type Operator

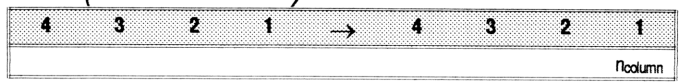
Scope Anywhere

Remarks Returns the percentage x of y. Replaces the built-in function with an operator that improves legibility. When Omnibus is plugged in, you can write '5 **%** Amount' instead of '**%**(5,Amount)'.

See also **%A, %CH, %D, %T**

Example '5 % 20' gives 1

⟷ (current column)



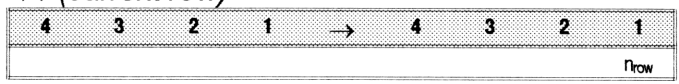
Type Function

Scope Inside Omnibus

Remarks See **ThisCol** on page 134.

This keyword cannot be employed because it is interpreted as a temporary variable. Use **ThisCol** instead.

↑↓ (current row)



Type Function

Scope Inside Omnibus

Remarks See **ThisRow** on page 134.

Time management

This chapter teaches you how to:

- ☑ *run the calendar (p. 142);*
- ☑ *read the watch (p. 145);*
- ☑ *set up an appointment from the calendar (p. 146);*
- ☑ *keep track of time events (p. 149);*
- ☑ *make calculations involving date and time objects (p. 150);*
- ☑ *use new programming commands (p. 151);*
- ☑ *assign the calendar to a hot (user) key (p. 159);*
- ☑ *change holidays plans (p. 161);*
- ☑ *build your own holidays plan or modify an existing one (p. 163);*
- ☑ *use the utilities stored on the card (p. 167).*

The software must have been properly installed in order to work as described. Return to page 89 to learn about the start-up procedures if you have not yet run WELCOME.

All the auxiliary programs stored on the card have been written using plain RPL commands, therefore they can be recalled and modified by the user if required.

Of course they cannot be erased from the card but they can be overridden by storing the modified copy in a port with higher number.

To access such programs you should reach the menu of the port where Omnibus has been installed, typically port 1 or port 2 (↰ LIBRARY ...) and scroll through the menu pages.

The calendar

The calendar of Omnibus is a very special one. You can see at a glance the situation of the month including the appointments and the holidays for a particular country. You can change the country holidays plan by means of a resident program or modify, correct, extend, reduce or delete any holiday plan stored on the card.

The calendar can work like a stopwatch if you need to control the time spent at the telephone or the duration of a travel. This function is explained on page 149.

The calendar is built during the installation of the software and is automatically maintained up to date by means of a control alarm. Should you inadvertently delete or corrupt it, you can restore the original alarm running the utility CSETUP, stored on the card. You can cancel at any time this automatic process purging the control alarm.

The calendar application (CALEND) appears in the first page of the port menu. If your *HP48* is idle, press the menu key labeled "CALE" and watch what happens. If the error Bad Argument Type is returned, it means that you missed to create the calendar first, as mentioned earlier. In this case return to page 89.

A list of months appears; if the names of the months are missing, you have no library of messages currently active. This implies one of the following facts:

- You failed to run the program WELCOME. Go to page 89 and read about it.
- You ran WELCOME but you forgot to turn the calculator off and on again. Quit the calendar by pressing ON and turn the *HP48* off and on again, then rebuild the calendar.
- You ran WELCOME but you skipped over the language selection. Go to page 89 and repeat the installation.

The current month is shown in bold typeface with the cursor on it. By moving the cursor up or down you can select the desired month. Past months are rendered with a shadowed typeface.

By default, the calendar is made out of nine months; two months backwards and six months forwards. The span of the calendar is set to (2,6) every time you run the utility CSETUP. If you wish to increase or decrease the span, you must manually edit the control alarm initialized by CSETUP. The modification will be maintained indefinitely until you purge the control alarm. Note however that this modification does not affect the current calendar, but takes place when the alarm expires.

It is worth noting that the control alarm set by CSETUP automatically updates all the information related to the calendar including the holydays plan of the current country.

After that you entered the list of months, you can recall a monthly plan by pressing EVAL. The cursor is placed on the current day if you are looking at the current month, or on the first day of the month in all other cases. Again, by pressing EVAL you can watch the name of the day in the current language. If no language library is installed, the name of the day is missing.

If you wish to watch the calendar given a certain date in the range of years 1846-2199, push on the stack the date and the span, thereafter run the utility MCAL, stored on the card. For instance, to look at the calendar of December 1999:

- push 1.121999 or 12.011999, depending on the format of the date;
- push (0,0);

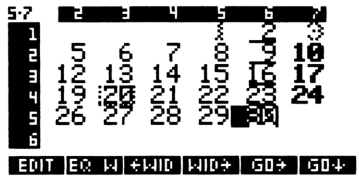
- run MCAL;
- run CALEND (may be you assigned it to a user key).

MCAL builds a list of months only if required. When the span is null, as in the example given, only the monthly plan is created. In both cases the calendar object is stored in the home directory under the name ¤DAT; if you purge it, the current calendar is lost.

The built-in utilities do not save the current calendar when you run MCAL***, therefore you must rebuild the current calendar running*** CSETUP ***or put in a safe place beforehand.***

To build the calendar given the current date, execute the command **DATE** followed by the span then start MCAL.

And now let's examine in detail how Omnibus represents calendar enhancements:



The calendar of April 1994 for Sweden

- working days are shown with plain characters;
- saturdays are underlined;
- sundays are rendered in bold typeface;
- holidays are represented with shadowed characters;
- pending appointments are surrounded by a dotted frame.

Of course any possible combination is represented with a blend of typestyles, with the exception of bold shadowed characters which are not allowed, but, after all, this does

not represent a severe limitation because the holiday has the precedence over the Sunday.

Take into account that:

☞ control alarms, that is alarms whose action object is not a string, are never considered as appointments, therefore are not displayed.

☞ Repeating appointments are shown as normal appointments.

You can turn calendar enhancements off by switching off flag 42 ($&Videono[-])[ ]e IconFlag menu. When the enhancements are disabled, the display redrawing time is slightly faster.

☞ Calendar enhancements are maintained until the display is subdivided in 4, 5 or 6 columns and the corresponding flag is set.

By turning off and on again calendar enhancements, you effectively refresh the display. This technique may return useful if an alarm comes due, while watching the calendar, because, at that point, the dispay could give a wrong representation of pending events.

Date and time formats can be adjusted by means of the toggles in the IconFlag menu.

☞ Every change to the status of the flags is local and not global; upon exit of Omnibus all flags are restored.

Reading the watch

The key [↱][TIME] is permanently assigned to the internal watch. The watch displays the current time and date in the status area until the key is hold down. The key [↱] must

be pressed only once. You can display the watch at any time, even when the command line is active. If the library contaning the messages is not attached, the name of the day is omitted.

If the current date and time are invalid, you must set the value by means of the keywords →**DATE** and →**TIME**. If you prefer to use the facilities provided by the built-in operating system, you must quit Omnibus.

Setting an appointment

You can set an appointment for a certain date by pressing the key STO while the cursor is placed on the corresponding cell. Omnibus displays a message in the status area to inform you of the date and time of the alarm being scheduled.

This function works independently by the calendar application and it is bound to the date object itself. For instance, if you create a worksheet to keep track of the pending payments, you can set an alarm for the date in which a payment expires by pressing STO with the cursor placed on the cell containing that date.

If you try to set an appointment for the current day, Omnibus automatically switches to the time tracking function. See later on for the details.

The appointment is added to the alarm list automatically and, unless you have another appointment at the same hour, it is scheduled for the 9.00 AM. When you set more than one appointment on the same day, the second is delayed of 1 hour, the third of two hours and so on. By default the appointment string is Appointment. The appointment scheduler will also skip over the hours from 12 PM to 2 PM and from 8 PM to 9 AM. If the scheduler

cannot find an empty hour, it will set the appointment to 9 AM o'clock.

Keep in mind that 9 o'clock is different from 9.15 o'clock; Omnibus does not check whether an appointment is scheduled at fractions of hour.

The appointment scheduler is designed so that you can quickly set an alarm during a conversation, while you may change it later.

If you place the cursor on a cell containing a past day, STO beeps and displays a warning message because you cannot set a past-due alarm.

STO allows you to keep track of a sequence of time events. Read the next section to know more about it.

To modify an appointment you can browse the alarms using the built-in catalog, following the procedures described in the *HP48GX* User's guide. To start the catalog just press → RCL whith the cursor positioned on a date object.

To remedy the lack of a direct link between the alarm catalog and Omnibus, users of *HP48SX* may run the ALRB program stored on the card. If you assign this application to a free key using the utility HOTASN, you can recall it inside the calendar.

The keys permanently assigned to Omnibus are reproduced on the back cover of the manual

ALRB makes nearly the same things as the built-in catalog does. Here is a table of differences between the two applicatons:

Built-in catalog	ALRB
The built-in catalog can be started only when the *HP48* is idle.	ALRB can be assigned to a hot user key or started inside another program.
The built-in catalog has got a set of custom menu functions.	ALRB has got the same menu and key functions as any other Omnibus based application.
Every change you make inside the catalog takes place immediately, no matter the key you pressed to quit the application.	ALRB updates the alarm list after that you have confirmed the changes by pressing ENTER. If you press ON the modifications are ignored.
The catalog can be empty. In this case you must leave the catalog to add an appointment.	ALRB automatically adds a dummy appointment when the alarm list is empty. You can disregard this alarm by pressing ON. You cannot erase all the appointments inside ALRB.
The structure of the catalog is fixed and you cannot change it.	ALRB lets you change the order of the columns and how data appear on the screen.
When you edit an alarm, the variable ALRMDAT is created in the current directory.	An alarm is just a row in the catalog.
Times and dates are represented through real numbers.	Times and dates are separate objects.
You must leave the catalog to change the format of dates and times.	Inside ALRB you can change the way in which dates and times are displayed by setting the date & time format flags.

When you change the list of alarms during a calendar session, either using the built-in application or ALRB, the changes are reflected as soon as you return to CALEND.

You can find the source code of ALRB on page 187.

Tracking time events

As mentioned earlier, the scheduler does not set appointments for the current day. When you press STO with the cursor placed on the current day object, Omnibus records in the alarm list the time at which you pressed the key. This kind of alarms is named Stop and you can observe that Omnibus reports this message instead of Appoinment in the alarm catalog.

***"Stop" is by nature a past-due alarm but it cannot be deleted by* ACK *or* ACKALL.**

The time tracking function may help you to calculate the time spent at the phone, the duration of a travel and so on.

To measure the time between two events, you must press STO twice, one time to set the beginning of the event and one time for stopping; then you recall the alarms from the alarm list by means of the command **RCLALARM** and calculate the difference in hours, minutes and seconds between the events with the command **DHOURS**.

Since, under some circumstances, it could be difficult to know the number of the entries in the catalog, you can recall the whole alarms list with the command **ALARMS**, *split the list on the stack with* **OBJ→** *and drop unwanted alarms.*

The stopwatch function is clearly meant for keeping track of events occurring at large intervals, where large is considered a period greater than 2-3 seconds.

Working with date and time objects

Omnibus is able to distinguish between dates and times because they are no longer represented by real numbers but by selfstanding objects. This fact implies a series of benefits and drawbacks for the user. We summarized the situation in the following table.

Real numbers	Selfstanding objects
Real numbers occupy 12.5 bytes.	Time and date objects consume less memory (7.5 bytes);
Real numbers are affected by decimal format flags.	The textual representation varies with the currently active environment: you will always get External when these objects are manipulated outside of Omnibus.
Real numbers have got a fixed textual representation. Times have got a fixed format (hh.mmss), while real numbers representing dates vary (mm.ddyyyy if flag -42 is clear, dd.mmyyyy if set). If you push a date real on the stack you cannot change its representation without affecting its format.	The textual representation is different from the internal format. It depends on the state of time and date format flags and can be changed at any time, while the internal format remains fixed;
Real numbers could represent invalid dates.	Date and time objects are always valid because they are checked upon creation;
Date real numbers have got a broader span.	Date objects have got a limited span of 256 years (1845-2200);
Real numbers cannot be used effectively for sorting if they represent dates.	Date and time objects can be used in sorting, including sort on multiple keys.

Omnibus extends some built-in commands to deal with date and time objects and adds new commands to enhance time management.

In addition to date and time objects you will encounter *events*. Events are lists where the first object is a date object, or a real number representing a date, and a time object, or a real number representing a time.

To convert an alarm into an event, recall the desired alarm from the alarm list and extract the first two elements with **SUB**.

Using programmable commands

Omnibus provides several new commands that greatly simplify programming needs. This section describes only the commands dealing with time management. The section is structured in form of reference, with the commands sorted in alphabetical order.

ALARMS

4	3	2	1	→	4	3	2	1
								$list_{alarms}$

Returns the whole alarms list. You can extract a single alarm with **GET**.

After?

4	3	2	1	→	4	3	2	1
		$date_1$	$date_2$					0/1
		date	x_{date}					0/1
		x_{date}	date					0/1
		$time_1$	$time_2$					0/1
		time	x_{time}					0/1
		x_{time}	time					0/1

Returns 1 if $date_1$ is later than $date_2$ or $time_1$ is later than $time_2$. You can match one date object against one real number or one time object against one real number. You cannot compare two real numbers. This command can be employed in search programs to locate a cell satisfying a certain condition. It is affected by flag -42*.

Before?

4	3	2	1	→	4	3	2	1
		$date_1$	$date_2$					0/1
		date	x_{date}					0/1
		x_{date}	date					0/1
		$time_1$	$time_2$					0/1
		time	x_{time}					0/1
		x_{time}	time					0/1

Returns 1 if $date_1$ is earlier than $date_2$ or $time_1$ is earlier than $time_2$. You can match one date object against one real number or one time object against one real number. You cannot compare two real numbers. This command

* Flag's status affects the command only when the date is represented by a real number.

can be employed in search programs to locate a cell satisfying a certain condition. It is affected by flag -42*.

DATE+ *(Date plus)*

4	3	2	1	→	4	3	2	1
		date	n_{days}					x_{date}
		x_{date}	n_{days}					x_{date}

Adds *n* days to the date object or real number representing a date. It always returns a real number representing a date. *n* can be negative. It is affected by flag -42*.

DATEH+ *(Date and Hour plus)*

4	3	2	1	→	4	3	2	1
		$list_{event}$	$x_{hh.mmss}$					$list_{event}$

Adds hours and seconds to an event. Hours and seconds can be negative. It is affected by flag -42*.

DAY

4	3	2	1	→	4	3	2	1
			date					n_{day}
			x_{date}					n_{day}

Extracts the day number from a date object or real number representing a date. It is affected by flag -42*.

DAY$ *(Name of Day)*

4	3	2	1	→	4	3	2	1
			date					string
			x_{date}					string

Returns the name of the day in the currently selected language given a day of week in the range 0-7; 0 is equivalent to 7 and stands for Sunday. Library 930 must have been installed and attached.

DDAYS *(Delta Days)*

4	3	2	1	→	4	3	2	1
		date1	date2					n_{days}
		date	real					n_{days}
		real	date					n_{days}
		real1	real2					n_{days}

Returns the number of days between two dates. It can be negative. It is affected by flag -42*.

DHOURS *(Delta Hours)*

4	3	2	1	→	4	3	2	1
		$list_{event\ 1}$	$list_{event\ 2}$					$x_{hh.mmss}$

Returns the hours, minutes and seconds in sexadecimal format between two events. It can be negative. You can convert the result into sexagesimal format by means of →**HMS**. The events may be alarms in the format returned by the command **RCLALARM**. It is affected by flag -42*.

DOM *(Days of a Month)*

4	3	2	1	→	4	3	2	1
			date					n_{days}
			x_{date}					n_{days}

Returns 31 for January (1), 28 or 29 for February (2) and so on. It is affected by flag -42*.

DOW *(Day Of Week)*

4	3	2	1	→	4	3	2	1
			date					n_{dow}
			x_{date}					n_{dow}

Returns 1 for Monday, 7 for Sunday. It is affected by flag -42*.

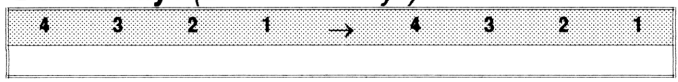
DT→R *(Date to Real)*

4	3	2	1	→	4	3	2	1
			date					x_{date}

Converts a date object into a real number. It is affected by flag -42*.

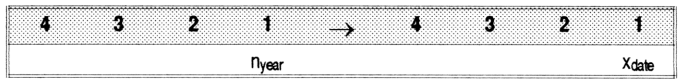
* Flag's status affects the command only when the date is represented by a real number.

DelHolidays *(Delete Holidays)*



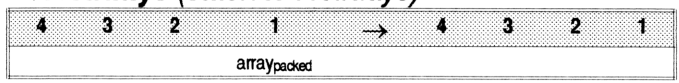
Deletes the database of moveable and fixed holidays from memory. If you want to delete only one type of holidays, you must first recall the array to be saved, perform **DelHolidays** and then restore the array.

EASTER



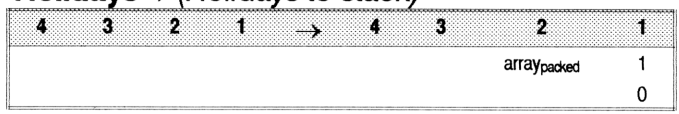
Returns the gregorian date of Easter given the year. It is affected by flag -42*. The year must be in the range 1845-2200.

→Holidays *(stack to Holidays)*



Stores the array of packed fixed holidays into memory.

Holidays→ *(Holidays to stack)*



Returns the array of fixed holidays and 1 if it exists, otherwise returns 0.

MONTH

4	3	2	1	→	4	3	2	1
			date					n_{month}
			x_{date}					n_{month}

Extracts the month number from a date object or real number representing a date. It is affected by flag -42*.

MONTH$ *(Name of Month)*

4	3	2	1	→	4	3	2	1
			date					string
			x_{date}					string

Returns the name of the month in the currently selected language. Library 930 must have been installed and attached.

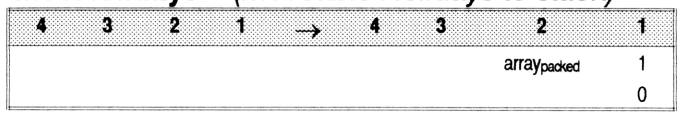
→MovHolidays *(stack to Moveable Holidays)*

4	3	2	1	→	4	3	2	1
			$array_{packed}$					

Stores the array of packed moveable holidays into memory.

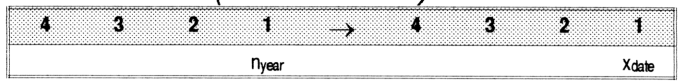
* Flag's status affects the command only when the date is represented by a real number.

MovHolidays→ *(Moveable Holidays to stack)*



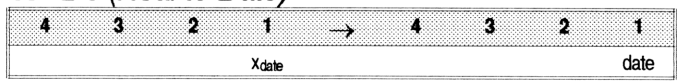
Returns the array of moveable holidays and 1 if it exists, otherwise returns 0.

ORTEASTER *(Ortodox Easter)*



Returns the gregorian date of the ortodox Easter given the year. It is affected by flag -42*. The year must be in the range 1845-2200.

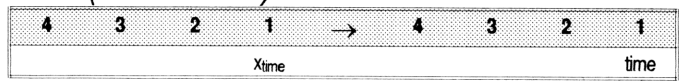
R→DT *(Real to Date)*



Converts a real number into a date object. It is affected by flag -42*.

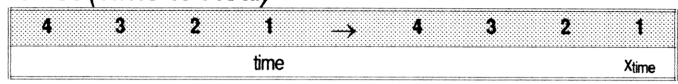
* Flag's status affects the command only when the date is represented by a real number.

R→T *(Real to Time)*



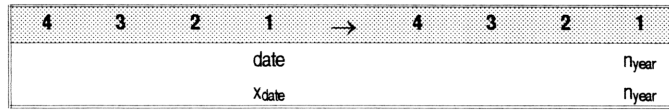
Converts a real number into a time object.

T→R *(Time to Real)*



Converts a time object into a real number.

YEAR



Extracts the year from a date object or real number representing a date.

How to assign the calendar to a key

The utility HOTASN, stored on the card, allows you to assign programs to keys of the user keyboard so that you can run them when usually you could not. Unlike **ASN**, the utility allows you to assign only keys of the first three planes, that is excluding [α] sequences.

A key whose action takes place immediately is called *Hot key*. Hot keys are of two kinds:

- Keys which start applications immediately, no matter what the current mode is.
- Keys which start applications immediately, provided that the command line is not active.

The utility first shows you a banner with a brief memorandum, then asks you to press the desired keystroke to which bind the object lying on the first level of the stack.

To create an assignment of the first type, you must begin the keystroke with the [α] key, otherwise an assignment of the second type will be created.

A keystroke like [α] [→] [MODES] assigns the object on the first level of the stack to the keystroke [→] [MODES], so that the object will be evaluated even if you are typing characters in the command line.

A keystroke like [→] [MODES] assigns the object on the first level of the stack to the keystroke [→] [MODES], so that the object will be evaluated only if you are not typing characters in the command line otherwise a beep will be issued.

The calendar application is stored on the card under the name CALEND. Since backup objects cannot be evaluated directly, we must create a procedure calling CALEND. You can do it in the following manner:

```
« :&:CALEND
EVAL »
```

The ampersand (&) means: "search the program in external ports first, if not found, search in main memory". This technique grants that the program will be executed whichever is the port where CALEND is stored.

At this point we can push this tiny program on the stack and run the utility HOTASN stored on the card.

This utility lets you assign a program to a user key, so that you can start it when the *HP48* is idle. When Omnibus is running and cell mode is active, the *HP48* is idle too.

You must assign only the keys which are not currently defined by Omnibus, if you wish to run programs associated to them in that environment. Failure to do so inhibits the access to the desired application.

- Start HOTASN;
- Press any key after the initial screen;
- Press the keystroke to be assigned (Suggested [4] or [↰][TIME]);
- Try it out soon.

Holidays plans

A holiday plan is a special object used by Omnibus to maintain in memory the country dependent database of holidays. The holidays plans of more than twenty countries have been stored directly on the card, but you can add, delete or modify each one, following the instructions given in this section.

Maintaining a system for the calculation of holidays is not trivial because hoydays can be of different kinds:

- fixed, like the New year's day;
- linked to Easter;
- linked to the ortodox Easter (following the julian calendar);
- moveable, linked to a day of week;
- either fixed or moveable, but depending on non-gregorian calendars.

The first four types listed above can be easily handled by means of the tools provided with Omnibus. With a fair amount of work you should be able to add holidays following the Islamic and Judaic conventions which are not based on the gregorian calendar.

Although we made of our best to check the plans up, we cannot guarantee that they are perfect for every region of every country.

We will be very pleased if you could report any mistake or suggest improvements.

Holidays plans stored on the card bring the name of the corresponding country. You can recall the plan of the desired country running the application named HP (Holidays Planner) at any time. This program lets you configure the internal database effortlessly and we recommend that you always resort to it when you need to change the current plan.

Its working is fairly simple:

- press the menu key corresponding to "HP" in port's menu.
- move the arrow keys [▲] or [▼] until you reach the desired country. You can also look ahead country names by typing [α] **A..Z**.
- press [ENTER] to confirm or [ON] to abort the selection.

HP will start with the cursor positioned on the currently selected country, if any. If you abort the selection the current setting will not be changed.

After the confirmation, HP updates the database and this may take a while. HP stores the selected country in the home directory under the name σDAT.

If you purge the variable σDAT, ***the control alarm set by*** CSETUP ***and the utility*** MCAL ***will not run correctly. Either purge the control alarm or restore*** σDAT ***by running*** HP.

If you assign this application to a hot key, as explained in the previous section, you will be able to change the current holidays plan while running the calendar. Just choose a key that has not been already defined by Omnibus.

Unless you need to modify a holidays plan, you can skip the rest of the chapter.

Modifying the holidays plan

There can be situations where you need to modify the holidays plan of a certain country to correct our mistakes (hopefully a few ones) or add holidays on a regional basis. You may also desire to include your vacation periods and events alike.

Depending on the type of holidays you need to include, may be you have to do minor changes, or major changes to the default country holidays plan.

If you are going to add a holidays plan for a new country (that is a country not listed in our database) or change the plan of an existing entry, you must first insert the name of the country in the existing database.

The utility FIXHP, stored on the card, will do this job for you. FIXHP stores a new database into port 0, overriding the database stored on the card. Of course, should you purge this object from memory, the database on the card will pop up again.

FIXHP shows you the current database and allows you to delete, change or add new entries. If you want to maintain an ordered database, either insert the name in the desired place or sort the names before exiting. Keep in mind that accented characters come after plain ASCII characters, thus Österreich will be listed *after* USA. In such cases you must rearrange the order of the names manually.

If you need to redefine an existing entry, you must enter the name of the country either tagged with the port number where the new plan is kept, or untagged if it is stored in main memory. For instance, if you want to add an entry for Romania, whose plan is stored in port 0, you must enter :0:Romania into a cell. Note that FIXHP always shows the name of the country untagged to improve legibility. Again, it is recommendable that you insert Romania directly in the right place (e.g. between Québec and Schweiz).

It is worth noting that:

☞ the wildcard (&) character should be reserved for internal plans.

☞ If you use the wildcard in place of a specific port number, be sure that the name of the plan is unique, otherwise a misbehavior may occur.

⛔ ***Do not include spaces or separation characters in the name of the country. Failure to do so may lead to run-time errors.***

➤ Press ENTER to confirm or ON to quit FIXHP.

This was the first step and it was longer to say than to do.

The second step consists in the preparation of country's holidays plan. Holidays plans can be of three types:

1 A vector of packed (fixed) holidays;

2 A list containing three objects, in the order:

i The Easter command;

ii

A vector containing the difference in days between each holiday and the Easter;

iii A vector of packed (fixed) holidays;

3 A program taking the current year from the stack and returning two arrays in the following order:

i a vector of packed, moveable holidays;

ii a vector of packed, fixed holidays.

The first type is suitable for countries where Easter is not holiday, e.g. USA, Japan, China, Saudi Arabia and so on. You may also apply to this kind of structure if you wish to include family's vacation periods, school holidays or any other period of rest from work.

The second type is suitable for countries where occur Easter or ortodox Easter but you have not other moveable holidays. The first object of the list must be a command or a name of a program taking a year number and returning a valid date. This could be for instance a program to calculate the judaic easter in the gregorian calendar.

The third type is suitable for all the countries where the first two cases fail, e.g. United Kingdom, Ireland or countries with non-gregorian calendar and multiple moveable holidays.

To create a vector of packed dates, you can proceed as follows:

- Quit any active application.
- Run the utility MAKEHOL, stored on the card.
- Using the command **R→DT** enter each date into the first column. When you are done, press ENTER. A vector of packed dates will be returned.

If the holidays plan you need is of the first type, you have done.

For the second type you need to create a vector of displacements, that is a real-valued 1-dimensional array consisting of integer numbers in ascending order representing the whole set of holidays linked to a given date in terms of reciprocal differences. An example will clarify this verbose definition.

Say you want to create a vector representing four holidays linked to the gregorian Easter where:

-7	the first one is the Sunday preceding Easter (the Palm Sunday)
0	the second one is the Easter itself
1	the third one is the Easter Monday
49	the last one is the Whit Sunday

As you can see each entry is defined as the difference in days between the Easter and the other dates; the vector will be then [-7 0 1 49]. Days preceding Easter give negative offsets.

If you don't know what is the difference in days between the Easter and the other holidays, the command **DDAYS** *can do this computation for you.*

At this point it is fairly easy to build the list for the second case:

- open a new list, { } ;
- type the command **EASTER** and a space,
- enter the array of displacements,
  [ ] 7 +/- SPC 0 SPC 1 SPC 4 9
- enter the array of packed date built with MAKEHOL.

If such array is on the stack, you can paste it at the cursor position by entering the interactive stack application from

the edit menu and choosing "ECHO" from there. To access the edit menu, just press EDIT.

- store the list into a backup object whose name and port number match the name and port number stored in the holidays planner.

The list must have three elements in the order specified above. You cannot swap or delete any object. If your plan does not include fixed holidays, you must specify [0] to ensure proper working.

You can try how it works right now.

Finally, let's talk about the third holydays plan structure. Since it is represented by a user program, it allows the greatest flexibility and complexity.

In order to assist you in the creation of such plan, Omnibus includes a few utilities tailored for typical situations. You will find below the list of these utilities and a brief commentary. An example of programming is also given at the tail of the section.

Using the utilities of the card

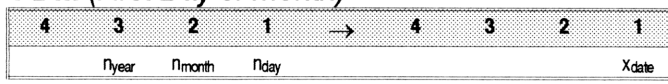
The utilities stored on the card are meant for a twofold purpose; they help you to simplify program design by providing *ready-for-use* pieces of code and give you a template for further customization.

Although they are not commands, they are organized here in the form of reference, for a quick and comfortable retrieval of information.

All the utilities do not check for the correctness of the arguments being passed on the stack. This implies that errors will be issued by the commands being called inside

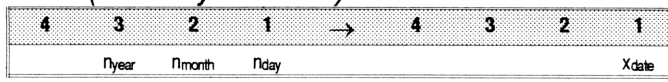
the utility, rather than by appropriate argument checking code and this may lead the neophite to a certain confusion.

FDM *(First Day of Month)*



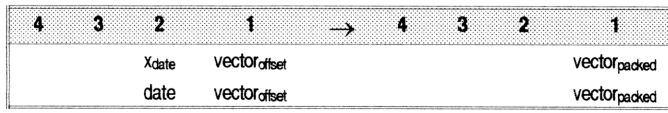
Given *year*, *month* and *day* of week, FDM returns a real number representing the date of the first occurrence of the day of week specified.

LDM *(Last Day of Month)*



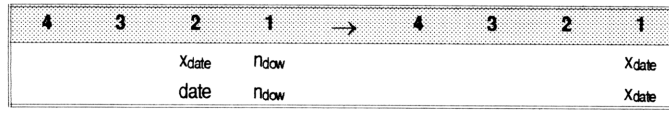
Given *year*, *month* and *day* of week, LDM returns a real number representing the date of the last occurrence of the day of week specified.

LINKDATE



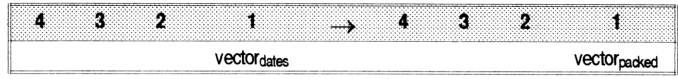
Given a *date* and an array of displacements, LINKDATE returns the corresponding packed holidays array. The arrays of packed holidays created this way is suitable for the command →**Movholidays**.

NXTDAY



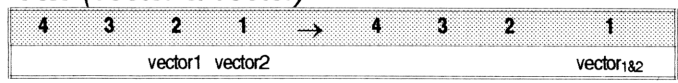
Given *date* and *dow* (day of week), NXTDAY returns a real number representing the date of the next occurrence of day of week. If the day of week of *date* is the same as *dow*, *date* is returned.

PACKDATES



Given a vector of real dates, PACKDATES converts it into a vector of packed dates suitable for →**Movholidays**.

V&V *(Vector & Vector)*



Given two vectors, V&V merges them into a single one.

We will examine how the holidays plan for Ireland has been programmed to learn how to combine these utilities.

The program is called either by the application HP or by the utility MCAL. It must take a real number from the stack (the year) and must return two vectors of real numbers (the holidays).

« DUP EASTER [-2 0 1] SWAP :&: LINKDATE EVAL BAA 6 1 :&:FDM EVAL BAA 8 1 :&:FDM EVAL SWAP 10 1 :&: LDM EVAL 3 →ARRY :&: PACKDATES EVAL :&: V&V EVAL [257 785 1804 3097 3098] »	Calculates gregorian Easter and holidays linked to it; the year is still on the stack; calculates the first Monday of June, the first and the last Monday of October. Packs these three dates and merges the two arrays on the stack. Finally appends the array of fixed holidays. The program ends returning two arrays, one for moveable holidays and one for fixed holidays.

This program is stored on the card under the name Ireland.

The need for a program object representing the plan of Ireland is due to the presence of the holidays falling on Monday. Since these holidays change year after year, we must resort to the utilities FDM and LDM to calculate the correct dates.

Once you have obtained the dates, you must convert them into the packed format through PACKDATES, remembering that all the floating holidays must be collected into a single array. In this case, the program combines the holidays linked to Easter with the other floating holidays collected into a vector. Merging two vectors is easily accomplished by V&V.

Fixed holydays are appended at the tail of the program.

If the plan you are creating does not include fixed or moveable holidays, use [0] as array.

You may also encounter situations where you need to find the date of the first Sunday after a certain date. The utility NXTDAY finds the day of week specified given a starting date. This method applies to the cases where the starting date is not linked to Easter.

If the date is linked to Easter, you had better to include the displacement of the day in the proper array and calculate it through LINKDATE.

NXTDAY is an utility provided for your comfort, it is not employed inside the holidays plans stored on the card.

Source code

Here are the listings of several programs mentioned in the manual. The are formatted as they would appear in the HP48 editor.

RPLMAN

```
« CLLCD
  IFERR DIMS
  THEN
"Sorry,
you must first extract
a cluster of cells."
3 DISP 60 .2 BEEP
40 .3 BEEP IDLE
  ELSE DROP
    IF Anchor (0,0)
==
    THEN
"Sorry,
the anchor cell is
missing."
3 DISP 60 .2 BEEP
40 .3 BEEP IDLE
    ELSE { EXPRPL
```

```
STRRPL OBJRPL
USRRPL } {
"Algebraic expressions"
"Strings"
"Generic objects"
"User defined" }
OBJ→ 0 1 4 3
"What are you going to
replace ?"
1 DISP 1 FREEZE
      IF PKMETA
      THEN DROP +
OVER 2 + ROLLD
DROPN GET '&' →TAG
EVAL
      ELSE DROP2
DROPN DROP
      END
    END
  END
»
```

EXPRPL

```
« 0 → k
  «
    IFERR DIMS
    THEN
"Sorry,
you must first extract
a cluster of cells."
3 DISP 60 .2 BEEP
40 .3 BEEP IDLE
    ELSE DROP
      IF Anchor
(0,0) ==
      THEN
```

```
"Sorry,
the anchor cell is
missing."
3 DISP 60 .2 BEEP
40 .3 BEEP IDLE
        ELSE
          IFERR MARK
"Enter the expression
to replace:"
{ "'" { 1 2 } ALG
V } INPUT STR→
"Replace with ?" {
"'" { 1 2 } ALG V
} INPUT STR→
"Any restriction ?"
{ "'" { 1 2 } ALG
V } INPUT STR→
          THEN L2M
DROP "Aborting…" 7
DISP
          ELSE L2M
            « ↑MATCH
'k' STO+ COLCT
            » 2 →PRG
# 3360Dh →MSG$ 7
DISP Apply
            IF k
            THEN
"The cursor has been
moved to the upper
left corner of the
cluster. Hit [→][EEX]
to replace it."
3 DISP Current C→R
```

```
Anchor C→R ROT MIN
CAB MIN SWAP R→C
            ELSE
"There were no items
like that you are
looking for. Try with
another expression or
drop the cluster."
3 DISP
            END IDLE
          END
        END
      END
    »
»
```

STRRPL

```
« 0 → k
  «
    IFERR DIMS
    THEN
"Sorry,
you must first extract
a cluster of cells."
3 DISP 60 .2 BEEP
40 .3 BEEP IDLE
    ELSE DROP
      IF Anchor
(0,0) ==
      THEN
"Sorry,
the anchor cell is
missing."
3 DISP 60 .2 BEEP
40 .3 BEEP IDLE
      ELSE
```

```
            IFERR MARK
"Enter the string
to replace:"
{ α } INPUT
"Replace with ?" {
α } INPUT
            THEN L2M
DROP "Aborting..." 7
DISP
            ELSE L2M
OBJ→ DROP
               « REPLACE
'k' STO+
               » 3 →PRG
# 3360Dh →MSG$ 7
DISP Apply
               IF k
               THEN
"The cursor has been
moved to the upper
left corner of the
cluster. Hit [→][EEX]
to replace it."
3 DISP Current C→R
Anchor C→R ROT MIN
CAB MIN SWAP R→C
               ELSE
"There were no items
like that you are
looking for. Try with
another string or
drop the cluster."
3 DISP
               END IDLE
```

```
            END
          END
        END
      »
    »
OBJRPL  « 0 → k
      «
        IFERR DIMS
        THEN
    "Sorry,
    you must first extract
    a cluster of cells."
    3 DISP 60 .2 BEEP
    40 .3 BEEP IDLE
        ELSE DROP
          IF Anchor
    (0,0) ==
          THEN
    "Sorry,
    the anchor cell is
    missing."
    3 DISP 60 .2 BEEP
    40 .3 BEEP IDLE
          ELSE
            IFERR MARK
    "Enter the object
    to replace:"
    { V } INPUT STR→
    "Replace with ?" {
    V } INPUT STR→
            THEN L2M
    DROP "Aborting..." 7
    DISP
            ELSE L2M
```

```
OBJ→ 2
            IF ≠
            THEN
"Wrong argument number"
DUP 7 DISP DOERR
            END
            « ROT
BCAC
              IF SAME
              THEN
DROP 'k' 1 STO+
              ELSE
NIP
              END
            » 3 →PRG
# 3360Dh →MSG$ 7
DISP Apply
            IF k
            THEN
"The cursor has been
moved to the upper
left corner of the
cluster. Hit [→][EEX]
to replace it."
3 DISP Current C→R
Anchor C→R ROT MIN
CAB MIN SWAP R→C
            ELSE
"There were no items
like that you are
looking for. Try with
another string or
drop the cluster."
3 DISP
```

```
                END IDLE
              END
            END
          END
        »
      »
USRRPL  «
        IFERR DIMS
        THEN
      "Sorry,
      you must first extract
      a cluster of cells."
      3 DISP 60 .2 BEEP
      40 .3 BEEP IDLE
        ELSE DROP
          IF Anchor (0,0)
      ==
          THEN
      "Sorry,
      the anchor cell is
      missing."
      3 DISP 60 .2 BEEP
      40 .3 BEEP IDLE
          ELSE
            IFERR MARK
      "Enter the program to
      be iterated:"
      { "« » " { 1 3 } V
      } INPUT STR→ DUP
      TYPE 8
              IF ≠
              THEN
      "Not a program" DUP
      7 DISP DOERR
```

```
        END
      THEN L2M DROP
"Aborting..." 7 DISP
      ELSE L2M OBJ→
1
        IF ≠
        THEN
"Wrong argument number"
DUP 7 DISP DOERR
        END FALSE 0
→ p o k
        « # 3360Dh
→MSG$ 7 DISP
          « DUP 'o'
STO p EVAL DUP o
SAME
            IF NOT
            THEN
'k' 1 STO+
            END
          » Apply
          IF k
          THEN
"The cursor has been
moved to the upper
left corner of the
cluster. Hit [→][EEX]
to replace it."
3 DISP Current C→R
Anchor C→R ROT MIN
CAB MIN SWAP R→C
          ELSE
" The program did not
change the contents
```

```
of the cells therefore
you can discard it or
retry the operation."
3 DISP
            END IDLE
        »
      END
    END
  END
»
```

Product

```
«
  → direction
  «
    CASE direction '↓'
      SAME
    THEN 1 Above
      Range C→R DUP 0
      IF >
      THEN
        FOR r r
          Col *
        NEXT
      ELSE 3 DROPN
        0
      END
    END direction '↑'
      SAME
    THEN 1 Left
      Range C→R DUP 0
      IF >
      THEN
        FOR c c
          Row *
        NEXT
```

```
        ELSE 3 DROPN
          0
        END
      END # 202h
      DOERR
    END
    »
  »
```

ORDMNGR

```
« "Loading..." 3 DISP
VARS DUP
  IF SIZE
  THEN DUP { FALSE
2 →LIST } LOP1
SaveModes 2
SetWidth 32 CF 33
CF 36 CF 34 SF 35
SF 37 SF 38 SF
    IF MATWRT
RstModes
    THEN # 33613h
→MSG$ 3 DISP DUP
TYPE 26
      IF ==
      THEN File→ 4
DROPN
      END OBJ→ { }
SWAP 1 SWAP
      START BAA
FALSE
        IF POS
        THEN 1 GET
SWAP +
        ELSE OBJ→
DROP BAA RCL BAA
```

```
            IF VTYPE
15 ==
            THEN
PGDIR
            ELSE
PURGE
            END OVER
STO SWAP +
          END
        NEXT REV SWAP
REV AAB { SAME }
LVOP 0 POS
EndOfString SUB REV
ORDER
    ELSE DROP
    END
  ELSE DROP
  END
»
```

SENDMNGR

```
« "Loading…" 3 DISP
VARS DUP
  IF SIZE
  THEN { FALSE 2
→LIST } LOP1
SaveModes 2
SetWidth 32 CF 36
CF 33 CF 34 SF 35
SF 37 SF 38 SF
    IF MATWRT
RstModes
    THEN DUP TYPE
26
       IF ==
```

```
        THEN File→ 4
DROPN
        END TRNSP
OBJ→ DROP
        WHILE DUP
FALSE POS DUP
        REPEAT BAB
SPLIT NIP + CAB
SPLIT NIP + SWAP
        END DROP { 2
→LIST } LVOP DUP
        IF SIZE
        THEN SEND
        ELSE DROP
        END
      END
    ELSE DROP
    END
»
```

PORTMNGR

```
«
" HP48 Port Manager
    © 1993 ForCALC

     initializing...
"
3 DISP → port
  « port PVARS TYPE
2
    IF SAME
    THEN DROP # Ah
DOERR
    ELSE DUP
      IF SIZE
      THEN DUP OBJ→
```

```
1 2 →LIST →SYMBMAT
SaveModes 1
SetWidth 32 CF 36
CF 33 SF 34 SF 35
SF 37 SF 38 SF 44
SF 45 SF -59 CF 1
SetDirection
        IF MATWRT
RstModes
        THEN
# 33613h →MSG$ 3
DISP DUP TYPE 26
          IF ==
          THEN
File→ 4 DROPN
          END
SYMBMAT→ CAR →LIST
BAB SAME
          IF NOT
          THEN
"checking the
configuration..."
3 DISP port 0
            IF SAME
            THEN
              IFERR
1 FREE
              THEN
DROP
IFERR 2 FREE
THEN # Bh DOERR
ELSE "rearranging..."
5 DISP 2 MERGE
END
```

```
                    ELSE
"rearranging..." 5
DISP 1 MERGE
                    END
                  ELSE
port MERGE { DTAG
port →TAG } LOP1
"rearranging..." 5
DISP port FREE
"Turning power off
to ensure data
integrity"
3 DISP 1 WAIT OFF
                  END
                ELSE DROP
                END
              ELSE DROP
              END
            ELSE DROP
            END
          END
  »
»
```

ALRB

```
« -58
  IF FC?
  THEN
"
     Alarm browser

     © 1994 ForCALC


converting..."
1 DISP
```

```
  END ALARMS
  « OBJ→ DROP 8192
/ '1_s' →UNIT
    CASE DUP '0_s'
==
      THEN DROP
FALSE
      END '1_yr'
CONVERT DUP FP UVAL
NOT
      THEN
      END '1_d'
CONVERT DUP FP UVAL
NOT
      THEN
      END '1_h'
CONVERT DUP FP UVAL
NOT
      THEN
      END '1_min'
CONVERT DUP FP UVAL
NOT
      THEN
      END UBASE
    END BCDA R→DT
BCDA R→T 4 2 RUP 4
→LIST
  » METOP DUP 0
  IF ==
  THEN DROP DATE
R→DT TIME R→T
"Dummy" FALSE 4
→LIST 1
  END →LIST
```

```
SaveModes 32 CF 33
SF 37 SF 36 CF 35
SF 42 CF 43 CF 44
CF 34 CF -55 CF 3
SetWidth 2
SetDirection
[ 1 2 ]
SetKey
  IF MATWRT
  THEN DUP TYPE 26
    IF ==
    THEN File→ 4
DROPN
    END DIMS OBJ→
DROP 4
    IF ≠
    THEN DROPN
"Invalid Format"
DOERR
    ELSE # 33613h
→MSG$ 7 DISP 0
DELALARM DROP OBJ→
FALSE FALSE FALSE
FALSE → d t s r
      « DUP
        IF 0
        THEN 1 SWAP
          START
OBJ→ 1 SWAP
            START
DUP TYPE
              CASE
DUP 28 ==
THEN DROP DT→R 'd'
```

```
STO
END DUP 29 ==
THEN DROP T→R 't'
STO
END DUP 13 ==
THEN DROP DUP UBASE
CONVERT UVAL 8192 *
'r' STO
END DUP 2 ==
THEN DROP 's' STO
END DUP 8 ==
THEN DROP 's' STO
END DROP2
                  END
                NEXT d
t s FALSE
                IF SAME
                THEN ""
                ELSE s
                END r
FALSE SAME 0 r IFTE
4 →LIST
                IFERR
STOALARM DROP
                THEN
DROP
                END
FALSE 'd' STO FALSE
't' STO FALSE 's'
STO FALSE 'r' STO
              NEXT
            ELSE # 60Dh
→MSG$ 7 DISP 2
FREEZE
```

```
            END
        »
    END
    END RstModes
»
```

Printing the worksheet

The utility PRINT, stored on the card, gives the possibility to print the worksheet or a portion of it effortlessly.

If you find PRINT worth using, we suggest you to declare it as a shortcut, following the procedure described on page 100. In this fashion you won't have to type the name of the utility in the command line.

PRINT has been conceived to work in conjuction with the *IR printer*. Although you could easily reroute printer's output to the serial interface and then to a parallel printer by means of a converter and appropriate drivers[1], PRINT is optimized to work with the IR printer, in virtue of its compatibility with the HP48.

The IR printer is capable of 160 dots in width (slightly more than a standard HP48 screen), thus PRINT will rotate the output if the picture being printed contains less than 20 rows, so that you can get a strip chart with an unlimited number of columns *in landscape orientation.*

If there are more than 20 rows, PRINT will dump the worksheet column by column *in portrait orientation.*

PRINT adjusts dynamically the width of the column, depending on the largest element found.

1 The printer driver utility is a stand-alone library provided with the serial to parallel converter made by ***ForCALC*** or purchasable separately at minimal cost.

PRINT relies on the status of flag 32 to determine whether the cells must be recalculated or not.

To select a portion of the worksheet, just set an anchor cell and move the cursor away to bound the opposite corner, then run PRINT. If the anchor cell is undefined, PRINT will dump the entire worksheet.

To take a snapshot of the screen, follow the instructions below.

Press and hold [ON], press and release [MTH] then release [ON].

Press and hold [ON], press and release [1] then release [ON].

The procedures described above do not work with serial printers.

If you need to capture the screen in form of a graphic object, there is a technique described on page 102.

To print the *value* of a cell just press the following key:

[→][PRINT]

[→][I/O]

In both cases, the object is always evaluated, regardless of flag 32. Graphic objects are printed graphically.

Exporting text

The utility EXPORT allows you to convert a cluster of cells into a string. Once you have a string on the stack, you can store it into a directory and then upload it to a host computer. In this fashion you can paste the text representing a worksheet (or a portion of it) into a word processor or DTP program.

Every time you run EXPORT you will be asked to enter the column and row separators in form of strings, because the import filters of DTP programs require special formatting characters.

By default EXPORT proposes a double space for separating columns and a linefeed for rows[1].

If the formatting characters are ASCII codes in the range 1-30, you can easily insert these characters into the text through the special keytrap **#CHR**.

#CHR is the first label in the second page of the SmartROM's custom menu (CST NXT).

Say you want to enter the *Tab* character (ASCII code 009):

► **#CHR** 0 0 9

You cannot enter character 000.

1 These are the preferences of Ventura Publisher®

EXPORT relies on the status of flag 32 to determine if cells must be recalculated or not.

Keep in mind that many special characters displayed by the HP48 are not directly available on PCs. You should refer to the character tables

Unlike PRINT, EXPORT does not format the string in any way.

Importing text

Omnibus implements the same syntax for objects and commands as the standard language does, therefore the problem of importing text into the program is the same as importing text into the HP48.

The suggestions given in this section are meant for solving the general problem of the transfer from the host computer to the HP48. Once you have successfully downloaded the text file to the calculator, the problem of importing the resulting object into Omnibus becomes trivial.

☞ Keep in mind that we are speaking of *ASCII* files (text files). Proprietary file formats such as .XLS files of Excel®, .WKS files of Lotus 1-2-3® and so on, are not covered.

Once you have obtained an ASCII file, you can work on it with a text editor.

It is recommendable that you insert the following header at the top of the file:

```
%%HP: T(3)A(D)F(.);
```

This header string tells the HP48 that :

- ☑ *the file is an HP48 text file %%HP:*
- ☑ *the file may contain special characters that require translation T(3)*
- ☑ *the current base number format is decimal A(D)*
- ☑ *the decimal point character is the period F(.);*

In general, as long as data are made out of numbers and strings, you will likely encounter little problems or no problem at all to convert the file in a *valid RPL object.*

In particular, if you want to import text in form of a table, you should adapt the text to follow the syntax of a numeric array, if you have raw numbers, or of a symbolic matrix if you have different data types. This may require the insertion of characters such as the square brackets [] or the braces {} at critical positions.

For instance, say you want to import the following ASCII file in form of a matrix maintaining the same dimensions:

```
1.2 3.4 4.5
5.6 6.7 8.9
```

To be sure that Omnibus will recognize and load the file as an array, you need to insert the square brackets as shown below:

```
%%HP:  T(3)A(D)F(.);
[[ 1.2 3.4 4.5 ]
 [ 5.6 6.7 8.9 ]]
```

In practice you must insert a pair of brackets for each row of data, plus an extra pair of brackets to bound the object.

Each row must contain the same fixed number of objects.

In the case you have a file with 'holes', you must either insert zeros, if you prefer a numeric array, or a place-holder like the null string.

If the decimal point is represented by a comma rather than a period

```
1,2 3,4 4,5
5,6 6,7 8,9
```

you must change the header accordingly

```
%%HP:  T(3)A(D)F(,);
[[ 1,2 3,4 4,5 ]
 [ 5,6 6,7 8,9 ]]
```

If the file contains data such as strings and numbers, in free format

```
ABC 12.3 HELLO 4.5
CDE 34.56 OK 3.2
```

you *must* use the symbolic data structure:

```
%%HP:  T(3)A(D)F(.);
{{ "ABC" 12.3 "HELLO" 4.5 }
 { "CDE" 34.56 "OK" 3.2 }}
```

☞ Remember that strings must be embedded between string delimiters. Failure to do so may cause syntax errors or misinterpretation of the data.

If data are streamlined, it is up to you to decide which is the best format to adopt.

Importing formulae and macros

Until you deal with common items like numbers and strings, there are no severe constraints. Omnibus is more tolerant than other spreadsheets regarding the length of the object stored in a cell.

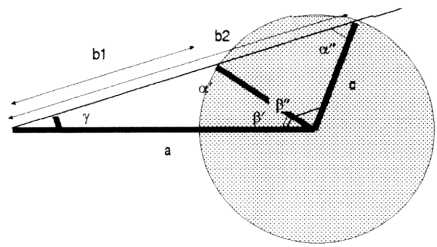
Practical examples

This section contains several examples of problems solved with Omnibus ranging from geometry to finance. They are presented on the assumption that the reader has read and understood the tutorial section.

Calculating the unknown angles and sides of a triangle given any three known parameters (with at least one side known).

Solving triangles

The HP48 directory **EXAMPLES\TRIANG.DIR** contains all the definitions necessary to solve this problem. The worksheet TRI.WKS acts as input/output area while the calculation is carried out automatically by some routines residing in the directory. As soon as the user changes an input value, the output area is updated consequently. Of course the recalculation flag must be set ([▭]). The unknown parameters are checked with a questionmark in



The case of a twofold solution, figure 1

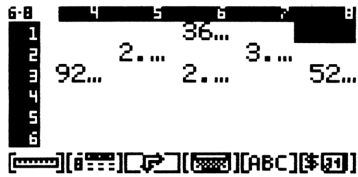
the near cell while known values are represented by real numbers (sides) or units (angles). All values (calculated or known) are then displayed in a triangular area of the

worksheet according with the logical order of the parameters (α lies in front of a, β in front of b, γ in front of c). When two solutions are available, a list with both values appears. The graphic representation of this problem is given in figure 1. Known parameters are rendered with thicker lines.

As mentioned earlier units allows greater flexibility than raw numbers at speed expense. In this case, the time is

the input area of TRI.WKS, figure 2

worth spending because the calculations are carried out independently of the current angle mode. On the other hand the user may specify the angle unit he prefers getting thereby a clear result. This is especially desirable when dealing with grads which could be confused with degrees in lack of specifications.



the output area of TRI.WKS, figure 3

Note also you are allowed to use length units for sides. This feature can be useful for on-the-field calculations.

The worksheet is named TRI.WKS. Load it into the spreadsheet. Figure 2 shows the input area (delimited by a rectangle). Each quantity is represented with a conventional name (a, β and so on); each name acts like a fixed cell pointer into the input area. Every time you change a value in that area of the spreadsheet, the value of the global name changes as well.

By pressing [→] [▶] you can visualize the output area which spans the whole display, when the display is 5 columns wide.

Since the recalculation takes place every time you press [ENTER], you had better to set [SOLVE] and change the input values all at once. For instance, say you want to know b, α and β taking into account that the current value of c is good. The fastest way to do it is to move the cell pointer on α and type in:

▶ ?32_°5?[ENTER]

[EVAL] lets you display the current item with higher accuracy which results indispensable when the solution is represented by a list containing two values. A fixed point display mode may also help you to trim exceeding digits.

The program TRG is in charge of computing the unknown values. TRG requires the name of the quantity as argument and returns the computed value, if any. Note however that the name of the quantity must be passed symbolically to prevent its evaluation and this is accomplished by the function **QUOTE** .

TRG first determines whether the name is a known value; if the value has been supplied by the user, just returns it, otherwise calls the appropriate procedures to calculate the quantity.

Teaching mathematics with the spreadsheet

The spreadsheet is an excellent tool for teaching mathematics. It is especially nice to use for showing how certain algorithms work.

In the directory EXAMPLES, on the auxiliary disk, there is a file named NEWTON.WKS; load it.

The worksheet NEWTON.WKS contains the formulae described in this example for your comfort. If you cannot down load it, the formulae are listed in the table on the next page.

In this example you will be told how to build a spreadsheet that, given certain values, returns the coefficients of a polynomial using Newton's finite differences algorithm, then you will be led step by step to the expansion of the worksheet up to yield a graphic proof of the solution found.

The example shows you how to implement a worksheet for finding the coefficients of polynomials up to the third degree, starting from the raw theory, up to the practical use of the application.

Newton's basic formula tells that: given n+1 known values of the polynomial calculated at regular intervals in 0, 1, 2, ..., n, you get the value $u^{[n]}$ for x by applying the following formula:

$$u^{[n]}(x)=\alpha_n(x-x_0)(x-x_1)\ldots(x-x_{n-1})+\ldots+\alpha_2(x-x_0)(x-x_1)+\alpha_1(x-x_0)+\alpha_0$$

At this point, supposing to work with n=3, we can reconstruct polynomial's coefficients a_0, a_1, ..., a_3 using the following formulae:

$a_0=\alpha_0$
$a_1=\alpha_1-\alpha_2/2+\alpha_3/3$
$a_2=\alpha_2/2-\alpha_3/2$
$a_3=\alpha_3/6$

where α_0, α_1, α_2, α_3 are the values taken from the first column of the table of differences given below:

$\alpha_0=x_0$	x_1	x_2	x_3	x_4
$\alpha_1=x_1-x_0$	x_2-x_1	x_3-x_2	x_4-x_3	·
$\alpha_2=\alpha_{11}-\alpha_1$	$\alpha_{12}-\alpha_{11}$	$\alpha_{13}-\alpha_{12}$	·	·
$\alpha_3=\alpha_{21}-\alpha_2$	$\alpha_{22}-\alpha_{21}$	·	·	·
$\alpha_4=\alpha_{31}-\alpha_3$	·	·	·	·

a_4 should be 0 indicating the convergence of the interpolation. A value not equal to 0 means that a third degree polynomial does not fit the points.

The table below contains the formulae you must type in the worksheet.

- Set flag 33 ([ƒ33])
- Clear flag 32 ([ƒ32])

	Row(~+1)					
X	(Row(~+1)-Mat(~+1,~+1)/2+Mat(~+2,~+1)/3)*X	Mat(~-1,~+1) -Col(~-1)	Mat(~-1,~+1) -Col(~-1)	Mat(~-1,~+1) -Col(~-1)	Mat(~-1,~+1) -Col(~-1)	
X^2	(Row(~+1)-Mat(~+1,~+1)) /2*X^2	Mat(~-1,~+1) -Col(~-1)	Mat(~-1,~+1) -Col(~-1)	Mat(~-1,~+1) -Col(~-1)		
X^3	Row(~+1)/ 6*X^3	Mat(~-1,~+1) -Col(~-1)	Mat(~-1,~+1) -Col(~-1)			
Converges	if = 0 →	Mat(~-1,~+1) -Col(~-1)				

Known values of the polynomial must be entered in the first row, starting with the value for X=0 in the third column. The coefficients are returned in the second column, in ascending order. Set the horizontal fill direction ([SHIFT]) before starting.

Try first with the following values:

▶ 5 3 1 5 21

The worksheet returns the following results:

You can verify that cell (5,3) contains a zero, indicating the convergence of the interpolation.

Now you can apply some change to the worksheet in order to get a a true symbolic expression representing the polynomial. To do so you may add each other the terms computed in column 2, by inserting the function **Sum** in cell (6,2).

What you get is:

5-3*X^2+X^3

At this point you can calculate the value of the polynomial in a certain point, e.g. X=1.25, by storing 1.25 into X.

▶ 1.25 'X' STO ← =

2.265625

You could even apply the *What If analysis* to get an array of tabulated values of the polynomial at fixed intervals. If the interval chosen and the beginning/ending values are equal to known values, what you get is a proof of the validity of the interpolation.

▶ Switch the recalculation off (□ ▦ □).
▶ Move the cursor to (6,3)
▶ write '**WhatIf**(X=0,4,1,←)' ENTER
▶ Press EVAL

The array being shown is stored in ΣDAT. This fact turns out to be quite useful because you can easily make a scatter plot of it.

You could also draw the polynomial, given its symbolic form. Why don't plot both ?

▶ move the cursor to (7,2)

```
« 'ΣPAR' PURGE
ERASE SCATTER
AUTO DRAW
ThisRow 1 -
Col STEQ
FUNCTION 'X' INDEP
DRAW PICT
»
```

▶ Press ENTER

If you press EVAL, you will see drawing the points first, then superimposing the polynomial to them.

If you wish that Omnibus automatically redraws the picture as soon as you change a value, switch the recalculation on.

TVM calculations

Omnibus is an ideal workbench for financial calculations. The directory **FINANCE.DIR** contains several common functions for solving typical problems. The function have been written in plain RPL language therefore you should be able to understand how they work. In particular we suggest you to look at **NPV**, **IRR** and **PMT→PV** to learn how to create non-trivial functions.

Wherever appears an interest rate as argument, you must divide the value by 100 in advance. This means that an interest of 12% per year is represented by the decimal value 0.12. When dealing with percentage functions like **%** or **%CH**, you must adjust the magnitude of the interest accordingly.

Here is the list of all the functions provided:

CIFV(PA,interest,periods)

Returns the *future value* of a **P**resent **A**mount given a compound **interest** relatively to **periods**.

CIFVPV(FV,PV,periods)

Returns the *compound interest* rate necessary to **P**resent **V**alue to accumulate to a **F**uture **V**alue over the **periods** given.

CIPV(FA,interest,periods)

Returns the *present value* of a **F**uture **A**mount given a compound **interest** rate relatively to **periods**.

CI→SI(Interest,periods)

Converts the compound **interest** rate into a *simple interest* rate over the same number of **periods**.

EJ→SI(interest,periods)

Converts an *effective interest rate* into a *nominal interest rate over a given number of* **periods**.

FV→PMT(FV,interest,periods)

Returns the value of each payment in a uniform series of payments given the **F**uture **V**alue, the **interest** rate and the **periods**.

IRR([cash-flow])

Returns the *internal rate of return* given a **cash-flow**.

NFV([cash-flow], interest)

Returns the **N**et **F**uture **V**alue of the a series of cash flows discounted at the given fixed **interest** rate.

NPV([cash-flow], interest)

Returns the **N**et **P**resent **V**alue of the a series of cash flows discounted at the given fixed **interest** rate.

PV→PMT(PV,interest,periods)

Returns the value of each payment in a uniform series of payments given the **P**resent **V**alue, the **interest** rate and the **periods**.

SI→CI(interest,periods)

Converts a *simple* **interest** rate into a *compound interest* rate over the same number of **periods**.

SI→EJ(interest,periods)

Converts a nominal **interest** rate into an *effective interest* rate over a given number of **periods**.

SIFV(PV,interest,periods)

Returns the *future value* of **P**resent **V**alue, the simple **interest** and the **periods**.

SIFVPV(FV,PV,periods)

Returns the simple interest given the **F**uture **V**alue, the **P**resent **V**alue and the **periods**.

SIPV(FV,periods)

Returns the *present Value of* **F**uture **V**alue, the simple **interest** and the **periods**.

→T(FV,PV,interest)

Returns the total number of periods required to **P**resent **V**alue to accumulate to a **F**uture **V**alue,compounding at the **interest** given.

USFV(payment,interest,periods)

Returns the **F**uture **V**alue of a uniform series of **payments** at the given **interest** over the total number of **periods**.

USPV(payment,interest,periods)

Returns the **P**resent **V**alue of a uniform series of **payments** at the given **interest** over the total number of **periods**.

The financial library made by HP uses flag 62 as indicator for the payment mode occurring at the beginning/end of a period. In order to alleviate the user, Omnibus uses flag 62 in the same fashion and lets you change its status from ; the icon representing the annuity mode is the last of the first page and it can be either [$] or [$]; the former indicates a payment occurring at the

beginning of a period, also called *annuity due*, while the latter represents a payment at the end of a period, that is an *ordinary annuity*.

The loan

Say you received a loan of 8,750.00 dollars from your bank at the annual interest rate of 10.5% to be returned in 12 payments per year over the next 3 years. You want to know the amount of each annuity payment. Each payment occurs at the *end* of each period.

Open a new worksheet and set the fill direction to downwards (GOTO) then enter the following items:

- write "Loan", "Rate", "P/Year", "Periods", "Annuity" ENTER
- move the cursor to cell (1,2)
- write 8750, 10.5, 12, 36
  'PV→PMT(Col(1),Col(2)/100/Col(3),Col(4))' ENTER

Now check if flag -14 is clear. To do so, go in the IconFlag menu and check the rightmost label of the first page; it must be set to [$21], that is payments must occur at the end of each period (annuity due).

To prevent errors in the analysis of annuity payments, you must be aware of the type of payment requested. Long term payments are significantly affected by a wrong payment setting which turns out into a misleading analisys of the problem.

Loan amortization

To see this technique at work load the worksheet AMORT.WKS. As you can see the numerical values stored herein are the same of the example given above.

The first cell contains the macro discussed earlier and it looks like a comment when the recalculation is enabled. If you depress EVAL while the cursor is on this cell, a page of comments will appear.

The remaining cells in the first column contain string descriptors for the values or formulas stored in the neighboring column.

The row titled "PMT" returns the amount of the annuity and its value is the same as that found earlier, of course.

The last four rows allows you to find the components of the amortization given the number of the period being referred.

If you prefer to build a table with all the components (principal, interest and balance) for each period, showing them on the same line, you can iterate the computation over the total number of periods and collect the values into an array. This capability of creating arrays of values iterating a given procedure a defined number of times is usually referred to as *What If* analysis.

This kind of operation can be carried out by executing the function **Whatlf**. This function is explained on page 136.

Whatlf often requires several seconds to complete, thus it is best to use when the recalculation is off. Since it stores the result in the system variable ΣDAT, every time you run it this variable is erased.

In the example being treated, we want to spawn all the components of the amortization table given a certain variable period. Hence we must replace the numeric value in the row "Period" with a symbolic name, say 'p', that will assume different values as the iteration proceeds. We will vary 'p' from 1 to 10 at integer steps.

- Set the fill direction to *null* (GO+ and GO↓);
- turn off the recalculation;
- move the pointer to cell (8,2);
- write 'p';

- move the pointer to cell (12,2)
- enter the following expression '**WhatIf**(p=1,10,1,↑)'.
- depress EVAL.

After some seconds Omnibus shows you the result inside a new worksheet. As soon as you press EVAL, you are returned to the previous worksheet while the array of values is retained by the system variable ΣDAT.

The expression '**WhatIf**(p=1,10,1,↑)' stands for the following statement:

Show me how the expressions listed above change by assigning to p the values from 1 to 10 at intervals of 1.

Leasing

You need to buy a van whose cost is 30,000$. You asked the dealer for an estimate of a lease purchase and you got the following conditions:

down payment of $2,500, 48 monthly installments of $850 plus a balloon payment of $1,500 to take title of the truck. The best interest rate you can get on a loan is 14%.

- Set to end payment (▶ 21).

The leftmost picture on this page shows you the result of

the analysis. If any other factor isn't influential it is slightly better to purchase the truck than to lease it, given the terms and conditions of the lease. This example is stored on the diskette under the name LEASING.WKS in the HP48 directory FINANCE.DIR.

Watch the state of flag -14 when performing financial calculations involving annuities because the result may vary with different settings.

The rightmost picture on the previous page exemplifies this problem given the same input values.

Capital budgeting

You must choose the best capital expenditure proposal among the following threes.

	CAPITAL.WKS (abstract)			
2	Periodic interest rate	0.1		
4	Initial investment	-100000	-100000	-100000
5	Cash flows	15000	60000	30000
6	period 2	30000	50000	30000
7	period 3	40000	40000	30000
8	period 4	40000	40000	30000
9	period 5	50000	30000	30000
10	period 6	60000	15000	30000
12	Net Present Value	NPV(Rarray(↑), Mat(3,2))	NPV(Rarray(↑), Mat(3,2))	NPV(Rarray(↑), Mat(3,2))

The formula in the last row is the same for all the columns. The higher is the present value, the better is the expenditure proposal. When the initial investements differ from each other, present values are not directly comparable; in this case you had better to analyze the cash flows using the function IRR.The higher will result the internal rate of return, the better will be the investment plan. Note also that the result of the IRR should be greater than the periodic interest rate in cell (2,2);should it be

lesser, the corresponding NPV will be negative indicating a bad investment plan.

Investment comparison

You want to invest a certain amount of money and you must choose between two options:

- put your money into a savings account with a net annual rate of 9.5%
- purchase treasury bonds at the price of 91.2$ against a nominal price of $100.

Of course you want to know which is the most profitable of the twos, but the two options are not directly comparable because in one case you have an interest rate and in the other case you have a cost, thus you must proceed to a conversion into a common unit of measure.

We will assume the final amount of $100 as reference.

In the case of the savings account you must consider the amount of $100 as future value at the rate of 9.5% per year.

Open a worksheet from scratch. Set the fill direction to horizontal, .

Type from the keyboard the following commands:

- Write "Savings account" 100 ENTER
- move the cursor to the beginning of the next row, → ◄ ▼.
- write "Annual rate" 9.5 ENTER
- write "Total years" 1 ENTER
- write "PV of investment" 'CIPV(Col(1),Col(2)/100,Col(3))' ENTER

As you can see the treasury bonds are preferrable given these conditions.

In this example we assumed that the nominal price of the bond and the interest rate of the savings account were easily comparable each other, bypassing any

consideration about the different costs of the operations. The examples given in the next pages take into account many aspects that are often forgotten by the 'normal' user and give you a chance to verify how much it costs to purchase bonds through a bank. The examples are tailored on the italian conventions that, as usual, are the most complicated of the world.

Treasury bonds

Determining the yield to maturity of a treasury bond.

Load the worksheet called BOT.WKS from the directory FINANCE.DIR. The meaning of each entry is explained below. The worksheet is structured in four columns. The first column contains the comments. The last three columns lets you estimate the *effective interest per year* for bills of different duration, respectively 1 year, six months, three months. In the first two cases each period in the table of the cash flows represents a *half year* while the in third case it represents a *quarter*.

All the values are expressed as percentages.

The values proposed in the worksheet are the following:

	BOT.WKS (abstract)			
2	Type	1 Year	1/2 Year	1/4 Year
3	Nominal cost	89.9	94.16	97.2825
4	Transfer duty	0.009	0.009	0.009
5	Partecipation fee	0.05	0.05	0.05
6	Negotiation fee	0.5	0.5	0.5
8	Approximate interest	Sum(↑) %CH 100	Sum(↑) %CH 100	Sum(↑) %CH 100
9	Semiannual expenses	0.1	0.1	0.1

	BOT.WKS (abstract)			
11	At subscription time	-100	-100	-100
12	1 period	-Col(9)	SIFV(100,Col(8)/100,1) -Col(9)	SIFV(100,Col(8)/100,1)
13	2 period	SIFV(100,Col(8)/100,1) -Col(9)		-Col(9)
15	Periodic interest	IRR(Rarray(↑))	IRR(Rarray(↑))	IRR(Rarray(↑))
16	Effective interest	SI→EJ(Col(~-1)*2,2) *100	SI→EJ(Col(~-1)*2,2) *100	SI→EJ(Col(~-1)*4,4) *100

Row 8, titled "Approximate interest", gives you a first estimate of the real interest taking into account of the additional fees you must pay to purchase the bill. It does not take into consideration the costs for the custody by the bank.

The next three rows exemplifies the typical cash-flows of the operation leading to the computation of the internal rate of return on a semiannual basis.

The last row gives us the final result converting the semiannual interest into an effective annual interest.

Pluriannual treasury bonds

Load the worksheet called BPT.WKS from the directory FINANCE.DIR. The meaning of each entry is explained below. The worksheet is structured in three columns. The first column contains the comments; the second column exemplifies the case of a bill whose purchase cost is the same of its nominal price; the last column shows you the case of a bill with a nominal price that is lower than the purchase cost. In the case presented each period in the table of the cash flows represents a *half year* for a total duration is five years.

All the values are expressed as percentages.

The values proposed in the worksheet are the following:

	BPT.WKS (abstract)		
1	Nominal cost of bpt	96	96
2	Purchase cost	96	100.35
3	Interest rate %	6	6
4	Taxes on interest %	12.5	12.5
5	Transfer duty	0.009	0.009
6	Partecipation fee %	0.05	0.05
7	Negotiation fee %	0.5	0.5
8	Semiannual expenses	0.1	0.1
10	Approximate interest	Col(4) %D Col(3)-Col(8)	Col(4) %D Col(3)-Col(8)
12	At subscription time	-(Col(2)+Col(5)+Col(6)+Col(7))	-(Col(2)+Col(5)+Col(6)+Col(7))
13	1 period	Col(10)	Col(10)
14	2 period	Col(10)	Col(10)
15	3 period	Col(10)	Col(10)
16	4 period	Col(10)	Col(10)
17	5 period	Col(10)	Col(10)
18	6 period	Col(10)	Col(10)
19	7 period	Col(10)	Col(10)
20	8 period	Col(10)	Col(10)
21	9 period	Col(10)	Col(10)
22	10 period	100+Col(10)-Col(4) % (100-Col(1))	100+Col(10)-Col(4) % (100-Col(1))
24	Semiannual interest	IRR(Rarray(↑))	IRR(Rarray(↑))
25	Effective interest	SI→EJ(Col(~-1)*2,2)*100	SI→EJ(Col(~-1)*2,2)*100

The row titled "Approximate interest" gives you a first estimate of the real interest taking into account of the additional fees you must pay to purchase the bill. It does not take into consideration the costs for the custody by the bank.

The next rows exemplifies the typical cash-flows of the operation leading to the computation of the internal rate of return on a semiannual basis.

The last row gives us the final result converting the semiannual interest into the effective annual interest.

If you want to estimate a bill with a different duration, with the same seminannual compounding system, just add or delete the rows from the cash-flows table.

If the interest is computed quarterly you must add or delete the rows to compensate the total duration and finally you must change the formula on the last row to work with quarters rather than with half-years. You can accomplish this by replacing each 2 with a 4.

Finding out the best value

Estimating treasury bonds convenience.

The methods explained so far give you the necessary tools for estimating accurately the yield to maturity of fixed interest securities given certain conditions.

When you must choose the convenience of an investment among different options, given the same conditions, you need to find out a sort criterion. If you are able to find such criterion, the rest of the matter is child's play.

The following example shows you a typical situation where you have a bunch of securities giving comparable yields with slight differences between each other. Of course you want to discover the most profitable one.

Currently the securities are sorted by their nominal price.

CCTWKS					
Compound periods	Nominal cost	Tax on the interest	Current interest rate	Net nominal interest	Effective interest
1	96.85	0	11.1	SI→EJ(Row(3)%D Row(4)/100*Row(1), Row(1))*100	(Row(2)%CH 100+100)% Row(5)
1	97.5	0	11.55	Asln(1,5)	Asln(1,6)
1	97.65	0	11	Asln(1,5)	Asln(1,6)
1	97.75	0	11.95	Asln(1,5)	Asln(1,6)
1	97.95	0	12.2	Asln(1,5)	Asln(1,6)
1	98.15	0	12.8	Asln(1,5)	Asln(1,6)
1	98.7	0	13.2	Asln(1,5)	Asln(1,6)
1	99.35	0	11.3	Asln(1,5)	Asln(1,6)
2	99.4	12.5	8.05	Asln(1,5)	Asln(1,6)
2	99.4	12.5	7.5	Asln(1,5)	Asln(1,6)
1	99.5	0	14.15	Asln(1,5)	Asln(1,6)
2	99.8	12.5	7.3	Asln(1,5)	Asln(1,6)
2	99.9	12.5	8.2	Asln(1,5)	Asln(1,6)
2	99.9	12.5	8.5	Asln(1,5)	Asln(1,6)
2	99.95	12.5	7.3	Asln(1,5)	Asln(1,6)
1	100.45	0	15.05	Asln(1,5)	Asln(1,6)

Now, we just need to put the elements into ascending order and we will get the winner on the last row.

- Go to cell (2,6) and set the anchor;
- go to the bottom element of the same column;
- enter the sort menu, SORT;
- depress ← SORT;

the sorting process will take just a moment, thereafter the current row will contain the most convenient security. To reverse the order of the rows you can press REV soon.

Statistical functions

The HP48 directory STAT.DIR stored in the directory EXAMPLES contains a few selected statistical functions.

Note that the RPL built-in statistical functions like UTPC, UTPF, UTPN and UTPT are still usable inside Omnibus.

All the functions presented require a one-dimensional real-valued array as argument. Of course you cannot enter an array into an algebraic expression but you can pass a function returning it like Rarray.

If we take for instance the example of the Treasury bonds given on page 219, we could place below the lower right corner a function returning the largest value of the last column in this way:

'Max(Rarray(↑))'

Of course the function Max should reside somewhere on the current path, that is you should either merge STAT.DIR with FINANCE.DIR or put one inside the other.

Adev([array])

Function. Returns the *average deviation* of a set of data, defined as follows:

$$Adev(x_1,x_2,...,x_N) = \frac{1}{N}\sum_{i=1}^{N}|x_i - \overline{x}|$$

Avg([array])

Function. Returns the mean (the average value) of a set of data, defined as follows:

$$Avg(x_1,x_2,...,x_N) = \frac{1}{N}\sum_{i=1}^{N}x_i$$

Max([array])

Function. Returns the largest value of a set of data.

Median([array])

Function. Returns the median value of a set of data. In order to find the median value, data need to be sorted out; this may take a while with many entries (n>100). The median value is defined as follows:

$x_{med} = x_{(N+1)/2}$ for N odd

$x_{med} = \frac{1}{2}(x_{N/2} + x_{(N/2+1)})$ for N even

Min([array])

Function. Returns the smallest value of a set of data.

Sdev([array])

Function. Returns the standard deviation of a set of data (distribution), defined as follows:

$$Sdev(x_1,x_2,\ldots,x_N) = \sqrt{Var(x_1,x_2,\ldots,x_N)}$$

Sumx^2([array])

Function. Returns the sum of the squares of the values.

Var([array])

Function. Returns the variance of a set of data (distribution), defined as follows:

$$Var(x_1,x_2,\ldots,x_N) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Send manager

This little application allows you to upload HP48 named objects to the PC or to the Macintosh host computer, eventually changing the remote file name. To start it, just run SENDMNGR while you are in the directory where the files are located. SENDMNGR shows you a list of all the directory entries (if any) on the first column of the

worksheet. The dimensions of the worksheet are locked horizontally and vertically. In second column you can place the names of the files you want to transfer. To rename the remote file, just place here the new name. Only the objects having a name listed in the second column are transferred. To cancel a name from the second column you have two options: type **FALSE** then ENTER or depress . and the → DEF. If almost all files must be uploaded, you had better to copy the entire column onto the second one and then delete the undesired names. To do so move the pointer to the home cell (1,1) and depress ., then depress → and finally EEX. In this way you have pushed a copy of the first column onto the stack. Now move back to the cell (1,2), that is the cell beside the home cell, and depress → EEX. After a while the second column will appear; now you can delete single files as explained above. When you are done, press ENTER and the transfer will begin. If you want to quit from the send manager without transferring files, just press ON.

You can find the source code of SENDMNGR on page 184.

Order manager

ORDMNGR allows you to rearrange and/or rename HP48 directory entries. Run it from the directory where the files to rearrange are located. ORDMNGR shows you a list of all the entries (if any) on the first column of the worksheet. The dimension of the worksheet is locked both horizontally and vertically. The second column is the place where you can put the new names. To enter a new name press ' and then the name, followed by ENTER. To rearrange the order of the objects, use the menu keys ROW+ and ← ROW-. You can even sort the names into alphabetical order, either ascending or descending. To do so, enter the SORT menu and select the range of names to

which apply the new order. Then you can reverse the order by depressing REW.

To cancel a new name from the second column you have two options: type **FALSE** then ENTER or depress . and ▶ ◀.

When you are done, press ENTER and the reordering process will begin. If you want to quit from ORDMNGR, just press ON.

ORDMNGR may take a considerable time to complete when there are hundreds of variables in the directory. A low memory condition may also significantly affect the performance of the program. Renaming the last objects in the directory will take much more time than renaming the first ones.

The source code of ORDMNGR is on page 183.

Appendix A

Warranty, service and support

Care of the card

The card does not require maintenance. Do not open the shutter or touch the edge connector with fingers or tools.

Limited One Year Warranty

The card is warranted by ForCALC against defects in materials and workmanship for one year from the date of original purchase. Warranty is automatically transferred to new owner if you sell the product or give it as a gift and remains in effect for the original one-year period. During the warranty period, we will repair or, at our option, replace at no charge a product that proves to be defective, provided you return the product, shipping prepaid, to ForCALC.

The warranty does not apply if the product has been damaged by accident or misuse or as the result of service or modification by other than ForCALC.

No other express warranty is given.

ForCALC makes no express or implied warranty with regard to the software furnished. Programs are made available solely on an 'as is' basis and the entire risk as to its quality and performance is with the user. Should documentation and programs prove to be defective, the user (and not ForCALC or any other party) shall bear the entire cost of all necessary correction and all incidental or consequential damages. ForCALC shall not be liable for any incidental or consequential damages in connection

with or arising out of the furnishing, use or performance of the documentation and programs.

Service Center

Whether your unit is under warranty or not you can ship it for repair to our Service Center. If your warranty has expired, there will be a charge for the repair and for shipping costs.

The Service Center is located in Modena, ITALY.

ForCALC
Via Varese 67
41100 Modena, ITALY
phone 059-440404
fax 059-304490

Your unit will be repaired within five (5) working days of receipt.

Service Repair Charge

There is a standard repair price for out-of-warranty repairs. Out-of-Warranty units returned after repair are warranted for a limited 90 days period against defects in materials or workmanship.

Shipping Instructions

If your unit requires service, please follow these shipping instructions:

1 Include a description of the problem detected.

2 If under warranty, include documentation proving the date of purchase or repair.

3 Ship the unit in a protective packaging to prevent additional damages.

Shipping to ForCALC is at your charge. Shipping costs to return the unit are paid by ForCALC. On out-of-warranty repairs, the unit will be returned C.O.D.

Technical Assistance

ForCALC can assist as consultant those companies who need extensive support on particular projects. If you need specific information on ForCALC products or technical help on HP Calculators, you can call the aforementioned telphone number.

Index

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

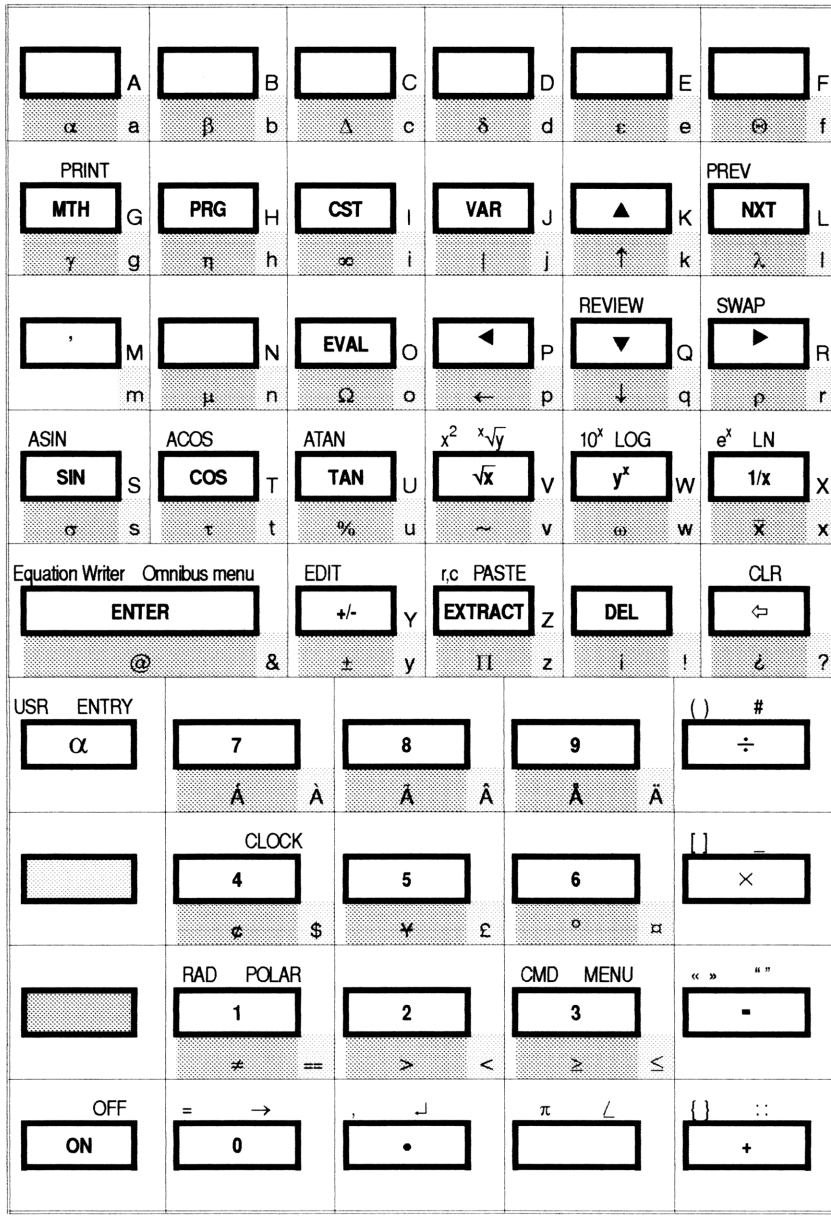
W

X

Y

Z

Omnibus keyboard layout for the *HP48SX*



Omnibus keyboard layout for the *HP48GX*