PHILIPS Data Systems

PHILIPS

P 2268

LOGICALC

OPERATOR MANUAL

5103 992 26821

Preface

This manual describes the operation of the Logicalc package. It has been designed to be read and understood by anyone who wants to learn how to use Logicalc, whether he be experienced or inexperienced in the use of computers.

Logicalc is an easy-to-use package and this manual will enable you get the most out of it. The chapters are so arranged that you will be lead from the basic functions of Logicalc to the more complex arithmetic features. You will find you grasp both the theory and the operating principles if you perform the examples as you read through the manual.

In this manual you may find words or terms that are unfamiliar to you. Do not be alarmed by this; there is a glossary of terms at the back of the manual where you will find simple definitions for the most complex of terms.

Page Layout

The pages in all P2000 manuals are arranged as follows:

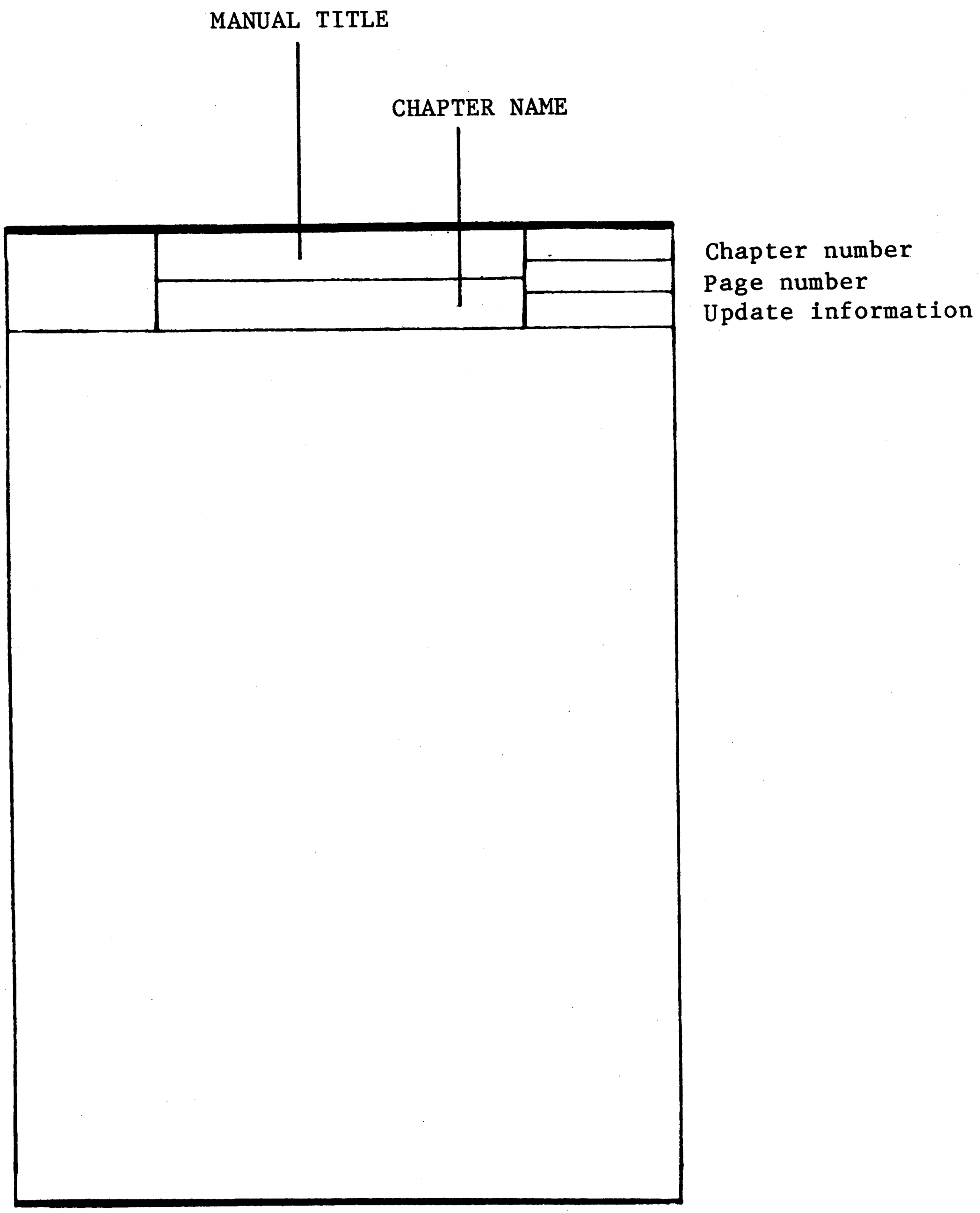


TABLE OF CONTENTS

Page

1 INTRODUCTION

1.1 INTRODUCING LOGICALC

This chapter introduces you to Logicalc, a valuable tool which will assist you in producing professional reports and solving many of the problems encountered in a modern business.

Logicalc can increase the efficiency of your organization without disrupting the existing system. Some of the possible applications for Logicalc include:

- balance statements
- cash flow analysis and forecasting
- general ledger
- inventory records
- job cost estimates
- market share analysis and planning
- patient records
- profit details, projections and statements
- salary records and tax details

1.2 HOW TO USE THIS MANUAL

Before we talk about Logicalc in more detail, let's have a quick word about using the manual. If you have never seen Logicalc before, study the first two chapters carefully. When you are familiar with their contents, continue with the subsequent chapters.

A more detailed explanation of Logicalc is given in Chapter 3 onwards. Don't worry if there is a section you find difficult, re-read it and use the examples given. All the reference material you need is in this manual, and the more you use it the more expert you will become with Logicalc.

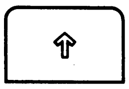
1.3 BEGINNERS NOTES

In this manual, as with other P2000 documentation, you will find that the keying instructions and messages displayed on the screen have been printed in a special manner. This section describes the keyboard, and the way in which keys are represented.

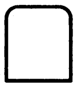
1.3.1 Key Representation

The P2000 keyboard consists of three parts. The main part is just like a normal typewriter keyboard. It is the part you will use most when you are typing in commands and data. The rest of the keyboard consists of keys that are used for special functions. These keys will be described when you need them.

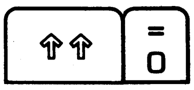
Most of the keys on your P2000 have only one symbol marked on them. However, some keys have two symbols. The lower symbol is obtained by pressing the key on its own, while the upper symbol is obtained by pressing the key and the shift key at the same time. The shift key looks like this:



When you have to press a single key, it will be represented on its own, like this:



When you need to press two keys at the same time, they will be shown together like this:



When you need to press a sequence of keys they will be represented like this:

$$\boxed{}\;,\;\boxed{}\;,\;\boxed{}$$

There is one key on the keyboard which may look unfamiliar to you. It is known as the supershift key, and looks like this:

$$\boxed{\Uparrow\Uparrow}$$

It is used to enter characters which are not represented on the keytops, for example '¼'. It is also used in conjunction with the key,

$$\boxed{\begin{array}{c}\pounds\,\#\\3\end{array}}$$

to obtain the character '#'.

1.4 RUNNING LOGICALC FOR THE FIRST TIME

Before you can use Logicalc's facilities, you must initialize the Logicalc Disk. Go through the following steps carefully. You will only need to do so once.

1. Put the SESAM Key into the side input socket.

2. Switch on the P2000 by pressing the button in the bottom right-hand corner.

3. Switch on the P2000 Monitor by pressing the button in the bottom right-hand corner, or on the stand-alone monitor, at the back.

4. If you have a separate Disk Drive Unit, switch it on.

5. Put the Logicalc System Disk into Disk Drive 1.

6. Press the RESET button.

You will now see a message on the screen requesting you to type a space, in order to commence the initializatiion procedure. When the system has performed the disk initialization, you will see a message to this effect.

Almost immediately, the P2000 Logo appears, underneath which you will see the date fields. If you wish to retain the date which is already displayed on the screen, simply press:

$\boxed{!}$

If you wish to alter the date, you must type over the date displayed. Once you have typed two digits for the day, the cursor moves on automatically to the month. Similarly, when you have entered two digits for the month, the cursor moves on to the year. If you only need to enter one digit, you can move on to the month or year by pressing:

$\boxed{\text{TAB}}$

You can move back to the month or day by pressing:

⇧⇧ TAB

Once you have entered the day and month, you can enter the year, or tab across the year field. You will then have to confirm the date by pressing:

!

The next screen will now appear - the System Menu. Of the three options given to you, select number 3. The Logicalc 'banner' will then appear momentarily on the screen, followed by the appearance of the Logicalc 'window' (this term will be explained later in the chapter). You are now ready to use Logicalc.

Note - Once your Logicalc Disk has been initialized, it will be protected. This means that you will not be able to use it with any SESAM Key other than the one used during the above procedure.

1.5 RUNNING LOGICALC ON SUBSEQUENT OCCASIONS

To run Logicalc on subsequent occasions, carry out exactly the same routine as described in the previous section. The only difference in this procedure is that, because your Logicalc System Disk has been initialized, you will first see the P2000 welcome screen containing the date fields. Again, once you have entered the date, the System Menu appears on the screen, so, select option 3 to get into Logicalc.

1.6 TRANSFERRING DISKORG TO YOUR LOGICALC DISK

Some of the operations we have described in this manual require the use of the TKS DISKORG program, for instance making a backup of your Logicalc Disk, and reorganizing the files on your data disk. For convenience, it would therefore a good idea to transfer this program onto the Logicalc System Disk. Go through the steps shown overleaf.

1. Place your TKS System Disk in Disk Drive 1, and the Logicalc System Disk in Disk Drive 2.

2. Boot the system by pressing the RESET key. The TKS welcoming message will then appear on the screen, and you must specify the date (see section 1.4). Once this has been done, the system menu will appear.

3. From the system menu select option 3 (DISKORG). The DISKORG menu will then appear on the screen.

4. From the DISKORG menu select option 2 (COPY). The following prompt will then appear:

```
1 FILE
2 DISK

0 EXIT TO MAIN MENU
```

5. Select option 1 (FILE). This will lead to a further prompt, as follows:

```
SOURCE                          DESTINATION
DRIVE: 1 FILE: ______________   DRIVE: 2 FILE: ______________   COPY FILE
```

6. Specify Drive 1, and the name of the source file:

   Type: **DISKORG.CODE** ↲

   Now specify Drive 2. The above file name will appear as the name of the default destination file. You must retain this file name as the destination file name:

   Press: TAB

   Confirm that you wish to carry out the specified transfer:

   Press: !

The above steps will result in the DISKORG program being transferred to your Logicalc System Disk.

1.7 MAKING A BACKUP

Having transferred DISKORG, the first thing you must do is to make a backup copy of your Logicalc Disk. Follow the instructions given below.

- With your P2000 switched on, and the initialized Logicalc Disk in Disk Drive 1, place an unused disk in Disk Drive 2.

- Press the RESET button. When the P2000 logo appears, specify the date, and then from the System Menu, select option 1 (EXECUTE PROGRAM).

- The following prompt will appear:

```
PROGRAM NAME: __________________                  EXECUTE PROGRAM
```

- Type: **DISKORG** ↵

- Confirm the program to be executed:

  press $\boxed{!}$

- The DISKORG program menu will now appear on the screen. It consists of 9 options, plus an option which enables you to return to the system menu.

- Select option 2 (COPY). This will lead to a further prompt:

```
                    1 FILE
                    2 DISK

                    0 EXIT TO MAIN MENU

SELECT: 1                                           COPY
```

- Select option 2 (DISK) again. A further prompt will appear:

```
DRIVE: 1  VOLUME: _______  DRIVE: 2  VOLUME: _______   COPY DISK
```

- The cursor is at the first drive field. Specify drive 1, or press the tab key to 'tab across' this field (drive 1 is the default drive). The volume name of the Logicalc disk will then appear in the prompt line, and the cursor will move on to the second drive field.

- Specify drive 2, or tab across this field (drive 2 is the default drive). The volume name of the disk in drive 2 will then appear in the prompt line, unless you are using a brand new disk to copy to.

- Type in the volume name for the Logicalc backup, followed by a return. You can use whatever name you like, but it cannot be more than seven characters.

- Confirm the copy operation by pressing: $\boxed{!}$

- You will now see a screen message, informing you that the Logicalc Disk is being copied, followed by a message informing you that the copying is finished.

- Press $\boxed{\cancel{!}}$

- You will now see the COPY menu. Select option 0 (EXIT TO MAIN MENU), and the DISKORG menu will appear on the screen. Select option 0 from this menu, and you will be back at the System Menu.

You will now have a copy of you Logicalc System Disk. Remove the copy from Drive 2, and return it to its sleeve. The name of the disk should be written on the disk label with a felt-tip pen. It is advisable to make another copy, so repeat the above procedure with a second unused disk. When the second copy has been made, remove the original disk from Drive 1, and store it in a safe place. Set one of your copies aside as the work copy, and then put the other copy away in a safe place as a backup of the original disk.

1.8 PREPARING A DATA DISK

It is recommended that you use a separate disk for your Logicalc files, so it will be necessary to prepare a disk for this purpose. Follow the instructions given below.

- Place your work copy of the Logicalc System Disk in Drive 1, and an unused disk in Drive 2.

- Execute the DISKORG program. From the DISKORG menu, select option 5 (PREPARE). The following prompt will be output:

```
DRIVE: _   VOLUME: _______   BLOCKS: ___             PREPARE DISK
```

- Specify Drive 2. The number of blocks on the disks in Drive 2 (608) will now appear in the prompt line, and the cursor will be positioned at the start of the volume field.

- Type in the name of the volume to be used for your data disk, followed by a return. The name may consist of up to seven characters.

- Confirm the prepare operation by pressing:

!

- A message will appear on the screen, informing you that the disk has been prepared.

You will now be in a position to return to the system menu, and start working with Logicalc. The result of the PREPARE operation, is to create a directory on the specified disk, thus enabling you to store files on it. If you do not prepare your data disk, you will not be able to write files to it.

2 GETTING STARTED

In this chapter you will see how Logicalc is structured to give maximum flexibility. The chapter is arranged so that you first learn about the command keys and then how to load the Logicalc system into your P2000.

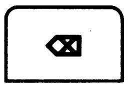
With Logicalc loaded, we will then describe the screen display and framework of Logicalc in some detail.

2.1 LOGICALC KEYBOARD

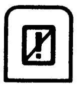
2.1.1 Function Keys

You are now familiar with the way this manual describes keys, so we will introduce you to some of the Logicalc Function Keys.

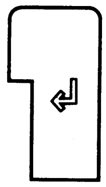
When you are typing your instructions to Logicalc you may occasionally make an error. To delete the last character you have entered, you must press the Backspace key once. To delete the last three characters, press the Backspace key three times. The Backspace key looks like this:



To delete the entire line that you have just typed, use the following key:



When you are satisfied that the characters you have typed are the correct ones, press the return key. Once you have done this you will not be able to change the characters entered with the keys mentioned above. The return key looks like this:



Throughout this manual, when you have to press return after answering a prompt, the return key will be represented like this:

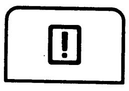
↲

The return key also allows you to move the Field Pointer. This is described in more detail in section 2.2. In addition to the function keys described above, Logicalc also uses special command keys. These are described below.

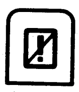
2.1.2 Logicalc Command Keys

This section describes the keys that will allow you to use the Logicalc commands. At first you may think they are complicated to use. With the example we shall be using in later chapters, however, you will have plenty of opportunity to practise using them.

The most important and most used key is the Command Key, which allows you to access the command directory. The same key is also sometimes referred to as the 'Do' key. The Command (Do) Key looks like this:



Once you have used the command key you will be in the Command line and you can then enter a command. If you enter an incorrect command or a command you no longer wish to have carried out, you can escape using the Escape Key which looks like this:



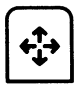
The Escape Key is also sometimes referred to as the 'Undo' key.

The Do and Undo keys are sometimes used in different circumstances. With certain Logicalc commands you enter, you will be asked to confirm the specified operation. When you wish to delete all the data on the screen, for example, Logicalc will ask you to confirm your intentions as follows:

```
Verify (Do/Undo) -
```

To go ahead with the operation, press the Do key. To abort the operation, press the Undo key.

To enable you to move around the array more easily, you can use the Goto key which looks like this:



The Goto key will give the prompt,

```
Goto > A1
```

the default value being A1. However, you can enter any location and when you press return, the Field Pointer will move to the specified location. You will see how useful this is when you have used Logicalc more.

2.2 STARTING LOGICALC

Before you can work with Logicalc, you must load the program. With the Logicalc system disk in drive one, and your zeroed disk in drive two, press the RESET key and this will activate your newly initialized disk. After you have specified the date, and selected Logicalc from the System Menu, your monitor screen should match the illustration below.

```
Col>|A          |B          |C          |D          |E          |F
Row+-----------------------------------------------------------------
  1|>          <
  2|
  3|
  4|
  5|
  6|
  7|
  8|
  9|
 10|
 11|
 12|
 13|
 14|
   +-----------------------------------------------------------------
            Field pointer:     A1    Entry loc.:     A1

CURRENT DATA    type:
            contents:
                edit:
```

2.3 THE LOGICALC STRUCTURE

As you can see, Logicalc is displayed in tabular form with column and row headings. This has the advantage of ease of use e.g. you can refer to, and work with, units of information independent of each other through a co-ordinate, which is a combination of the column and row number of the item, e.g. A6, B52 and U2 are all co-ordinate locations.

If you imagine yourself sorting mail in the post room, you may find in front of you a wall of post boxes, each row identified by a number (1,2,3..255) and each column by a letter (A,B,C.. Z,AA,BB,CC..DW). Thus, we have 255 rows and 127 columns of post boxes. To locate the address of any post box you simply refer to its co-ordinate, e.g. A6, B52, U2. With the post box located you store the mail inside that box.

In Logicalc, the memory locations are very similar to the wall of post boxes. Once we have specified the address of a memory location (the co-ordinate) we can store data inside that memory location. Logicalc has 32,385 (i.e. 255 rows by 127 columns) such memory locations and allows you over 400 real entries.

This structure will save you a great deal of time by not having to erase columns and columns of numbers when only one item of information changes. Instead, you simply change the single item and then everything can be recalculated for you by Logicalc (these operations are explained later in the manual).

2.4 LOGICALC SCREEN DISPLAY

The Logicalc display can be split into two distinct sections, the window, and the Edit and Command section. Before we discuss the window, let's have a word about the Logicalc framework.

2.4.1 The Framework

We have now seen that Logicalc has 255 rows by 127 columns yet the main menu you have displayed on your screen has only 14 rows by 6 columns. At this point we will introduce some of the new terms you will encounter on the following pages, and also explain in a little more detail the Logicalc framework (remember, if you do not understand any of the terms, consult the Glossary at the rear of the manual).

The limit of the Logicalc table of data is called the array, and this is 255 rows by 127 columns. As the monitor screen is limited in size, it is impossible to view all the array elements (the 32,385 memory locations) at one time. Instead, a small section of the array is displayed which is called the window. As you can see from your monitor, the window shows 14 rows and from 1 to 15 columns (depending on the width of each column).

2.4.2 The Logicalc Window

If you look at the Logicalc window you will notice a pointer (')   (') at the co-ordinate A1. The pointer is called the Field Pointer.

The Entry Location is the present co-ordinate to which data will be entered. When Logicalc is first loaded, both the Field Pointer and Entry Location will be at location A1. As we shall see later, the Field Pointer and Entry Location can be at different co-ordinates. Once data has been entered, the Field Pointer moves to the next Entry Location and they are both the same. This will become more obvious to you when we discuss the 'At Location' function in Chapter 5.

2.4.3 The Edit and Command Section

The area below the display window is the section of the screen where you will enter your data. Throughout the manual when we show replies to prompts they will always appear in upper case, however lower case characters will be accepted by Logicalc.

The display window is divided into three lines and their specific roles within Logicalc are given as follows:

Type - indicates whether the information stored at the entry location is text or numeric. It also indicates the positioning of the data within the field, i.e. whether it is on the left side of the column (left-justified), on the right side of the column (right-justified) or centred (centre-justified). If the entry location is blank, the Type line will be blank.

Contents - displays the contents of the entry location at which the Field Pointer is currently positioned. If the entry was text, the contents line will display text. If the entry was a numerical formula, the contents line display will display the formula, with the calculated results appearing in the entry location. If the current location is empty, the contents line will be blank.

Edit - the edit line will be blank at any location until you start to type an entry. The edit line will show you exactly what you have typed until you press the return key. Once you press return, the edit line will become blank and the entry will be displayed on the Contents line. It is important to remember that the edit line can contain up to 35 characters, however if the column is only 10 characters wide, then only the first ten characters will be displayed in the array. The column width will have to be altered to accommodate the entry (this is discussed in section 4.3). The edit line is also used to access the commands in the lower section of the display.

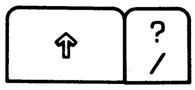
The Logicalc prompts, and your responses to the prompts, appear on the line below the edit line.

3 LOGICALC OPERATIONS

In chapters 1 and 2, we dealt with the make up of Logicalc. In this chapter you will learn how to use Logicalc. You will see how the Calculator function works, and learn how to use the Field Pointer. There is also an introduction to a sample file we have provided on your Logicalc Disk. You will be required to perform some of the basic Logicalc operations on this file.

3.1 CALCULATOR FUNCTION

The calculator function provides you with the facility to make quick calculations and have the results displayed without affecting your Logicalc array. To use the calculator, simply enter the figures on the edit line. Instead of pressing return, press:



after your entry. The calculator function uses the following symbols and expressions:

'+' for addition
'-' for subtraction
'*' for multiplication
'/' for division
'%' for percentages
'MAX(list)' to find the maximum value in the list
'MIN(list)' to find the minimum value in the list

The calculator will display figures accurate to 12 decimal places. Try some of the examples given below.

```
 45 + 12.091 ?  ,   234 * 32 ? ,  23451 - 34.32 ? ,
 (31.457 - 6.093)*26.8 ?  , 120%18.53333 ?
```

It is sometimes necessary to use parentheses to ensure that the sequence of arithmetic operations is carried out in the order you intend. The use of the MAX and MIN expressions will be discussed in a later chapter.

The answer to your calculation can be entered directly into the Logicalc array. Instead of typing '?' after the calculation in the edit line, press return. The result of your calculation will be entered at the current Entry Location and not on the Command line. As we will see later, you can also use location co-ordinates in your calculations, e.g. 110%A7 will give you the result of taking 110% of the value stored at A7.

Should you enter an expression which evaluates to something divided by zero, then the system will display '?n?', coupled in some cases with an error message. This is discussed more fully in section 4.6.

3.2 FIELD POINTER MOVEMENTS

The Field Pointer control keys are listed below together with a brief explanation of their functions.

↑	This moves the Field Pointer to the row directly above, unless the pointer is at row 1, in which case there will be no movement.
↓	This moves the Field Pointer to the row directly below, unless the pointer is at row 255, in which case there will be no movement.
→	This moves the Field Pointer one column to the right, unless the pointer is in column DW, in which case there will be no movement.
←	This moves the Field Pointer one column to the left, unless the pointer is in column A, in which case there will be no movement.
⇧⇧ ↵	This combination moves the Field Pointer to the first column of the next row, unless the pointer is in row 255, in which case there will be no movement.
↵	In Logicalc, the return key also allows you to move the Field Pointer one column to the right, providing the edit line is blank. If you are in column DW, then return will cause the pointer to move to the next line of the first column.

3.3 LOGICALC SAMPLE FILE

One of the many possible applications for Logicalc is in producing Financial Reports. On your system disk you will find a sample file which we will use to demonstrate some of the Logicalc features.

Before we can load the file, we must use one of the Command Keys you have just read about. To access the Command line press:

[!]

Below the Edit line you will see the prompt:

Command〉

In response to this prompt, press:

[⏎]

The system will now request the name of the file to be loaded. You should reply as follows:

File name : **BALSHEET** ↲

The next prompt gives a default co-ordinate of A1. In this example, a simple return will load the file with the upper left-hand corner at position A1. However, should you want the file loaded into different co-ordinates, then you can specify the location. We will discuss how you can manipulate the file within the array in a later chapter.

The window into the Logicalc array should now contain a set of figures resembling part of a balance sheet. To bring more of the figures into view, use the arrow keys to move the Field Pointer around the screen. Following the instructions below, we will demonstrate how Logicalc can recalculate a set of figures.

The figure we wish to alter is the Inventory total in the Assets column. Using the Goto key, move the Field Pointer to location B7 and type in the new entry:

5235000 ↲

We will now have to alter the figure in the Current Liabilities column, so again using the Goto key, move the Field Pointer to location B24. The entry has to be changed to take into consideration the alteration at B7, so type:

19000 ↲

We have now substituted two new values into the balance sheet. The next task is to recalculate the totals. Instead of using a pen and paper or a calculator to alter the totals, we can use Logicalc. Access the command line by pressing:

$\boxed{!}$

Answer the prompts as follows:

```
Command> R

Recalculate : A)ll E)ntry  A
```

The totals will now be altered to accommodate our new entries.

Note - We have included in Appendix A, a printout of the BALSHEET file, after the above corrections have been made. You might like to compare your screen display with this printout.

3.4 SUMMARY

You should now understand how the Logicalc system is organized and you have seen a brief demonstration of one of its applications. The next stage is to apply this knowledge and start on the Logicalc example. However, before we start the next chapter, we should clear the screen.

press: [!]

This takes you to the command line and you should reply to the prompts as follows:

Command〉 [DELETE]

Delete : A)ll R)ow C)olumn E)ntry **A**

Verify (Do/Undo) - [!]

We will discuss these commands in greater detail in a later chapter. Briefly, the verify prompt is a security check, to ensure that you delete only the data you wish to delete.

4.1 TEXT JUSTIFICATION

Now that you have seen the sample package and cleared the screen you are ready to start our inventory example. The window into Logicalc should have the CURRENT DATA lines blank and the Field Pointer at A1. If this is not the case, go back to Chapter 3.4 and clear the screen again.

The first entry for our inventory will be 'PART #', so enter:

PART # ↲

(Remember that the character '#' must be produced with the supershift key.)

You will have noticed the CURRENT DATA lines now display:

```
CURRENT DATA     type: text:left justified
             contents: 'PART #'
                 edit:
```

The entry is interpreted as text since the first character was a 'P', and as text, will automatically be left justified. In this example the data to be entered in this column will be numeric which is automatically right-justified. This would give our example an unbalanced look. In order to change our entry from left-justified to right-justified enter:

/R ↲

Our column header will now be right justified. Similarly, if we had wished to centre-justify the heading, we could have entered '/C'. A quicker way to justify text is to specify the justification on entry, as follows :

```
/LPART # - will left justify the entry
/RPART # - will right justify the entry
/CPART # - will centre justify the entry
```

It is also possible to alter the justification of numeric data, using the Data Type Key. Move the Field Pointer to location B1 and then, without pressing return, type:

/C4027

The Text Editor command options are summarized in the following table:

option	key	explanation
	←	moves the cursor one character to the left without erasing the character.
	→	moves the cursor one character to the right without erasing the character.
⟨INS⟩	INSERT	inserts one space before the character on which the cursor is currently located.
⟨s/shft INS⟩	⇧⇧ INSERT	inserts as many spaces as possible from the cursor location to the end of the edit line without erasing any characters.
⟨DEL⟩	DELETE	this deletes the character at the current cursor location.
⟨s/shft DEL⟩	⇧⇧ DELETE	deletes all blank spaces until the first character. It is often used in conjunction with ⟨s/shft⟩ INS .
⟨Do⟩	!	exits from Text Editor. A return is required to enter the text into the Logicalc array

We can now use the Text Editor to alter our entry. Using the right arrow key, move the cursor to the 'R' in 'PRA T' and type the correct letter. The cursor will now move on to the next character, so repeat the process. Our entry will now read 'PAR T #', so all that remains is to delete the incorrect space. Simply move the cursor to the blank space and press:

DELETE

To leave the Text Editor, press the Do key and you will be back at the edit line. A return will then place the entry into the Logicalc array.

We can now move to our next entry location, C1. Using the procedure we have just demonstrated, alter the width of column C to 15 spaces.

The heading at this location will be 'SUPPLIER #' with right-justification, so on the edit line type:

/RSUPPLIER # ↲

The rest of the column headings will be very similar to those we have demonstrated. You can now practise entering text using the data listed below for columns D to L. Remember, all you need do is to move to the next location; widen the column; then enter the data. If you are unsure of the procedures, re-read this section and try again.

Location D1 - column width of 18, heading '/RSUPPLIER NAME'
Location E1 - column width of 11, heading '/RCOST'
Location F1 - column width of 11, heading '/RPRICE'
Location G1 - column width of 18, heading '/RPROFIT/UNIT'
Location H1 - column width of 12, heading '/R%PROFIT'
Location I1 - column width of 24, heading '/RQUANTITY STOCKED'
Location J1 - column width of 24, heading '/RQUANTITY ON HAND'
Location K1 - column width of 24, heading '/RQUANTITY ORDERED'
Location L1 - column width of 18, heading '/RCOST/ORDER'

With the data entered we now want to return to A1, so press:

✥

As the default co-ordinate for Goto is A1, we need only press return. Your monitor screen display should be similar to the illustration overleaf.

Now you have started entering data, you may wish to save it on disk before you continue. You should therefore refer to the Save command, described in chapter 5.4.

4.5 THE COPY COMMAND

Now that we have dashes in location A2, we would like to copy them into locations B2 through to L2. We can do this using the Copy Command. First, access the command line

Press [!]

You should reply to the command prompt as follows:

```
Command> C
```

You will then be prompted for the co-ordinates of the data to be copied. Reply as follows:

```
from coord (>coord) :  A2 ↵

to coord (>coord) :  B2>L2 ↵
```

You should now have dashes throughout row 2 from column A to column L.

NOTE - If we had just entered dashes at location A2 and not used the Repeat function, the Copy Command would only have copied the number of dashes which had been entered. As the columns are of different widths, this would not give the desired effect. What we have done here is to copy the repeat function from column to column and so the number of dashes entered will vary, according to the width of the column.

The Copy Command offers you three choices. You may copy

- from an entry to an entry
- from an entry to a range of entries
- from a range of entries to a range of entries

With the last option, it is possible to specify the range of entries to which the existing range should be copied, in one of two ways. You may either specify that the range be copied into another range, e.g. A2>C2 to A3>C3, or you may specify that the range be copied into a range starting from a particular entry, e.g. A2>C2 to A3. In both these examples, the end result will be the same.

The values which are contained within the argument list of a system function may be any of the following:

- reference co-ordinate, e.g. F4

- range of reference co-ordinates, e.g. F4>F10

- other expressions, enclosed in parentheses

There are a number of possible ways to manipulate numeric data with a minimum of complexity and these will be demonstrated in the inventory example. As with the calculator function, Logicalc will display the symbol '?n?' whenever it encounters a numerical expression which evaluates to something divided by zero. You may add or subtract values from this entry and the system will ignore the undefined expression. If you try to copy such an entry, all the entries you copy will show the '?n?' symbol.

4.7 COLUMN PRECISION

In the inventory example, we wish to enter the part numbers for eight different-sized bolts. The number range will be from 213456 to 213463 in sequence. There are two ways in which we could enter this data. We could enter each number individually or we could enter one number and then use a simple formula.

On the edit line type:

213456 ↵

You will notice the entry at location A4 reads '213456.00'. This is because the system sets a default of two decimal places for numeric entries. As the entry is a part number we do not need two decimal places. In order to change the number of decimal places use the Format Command. First, access the command line:

Press $\boxed{!}$

You should reply to the command prompt as follows:

```
Command > F
```

You will notice that before entering the number of decimal places, you must enter 'E'. This indicates to the system that only this particular entry is to have a decimal precision of one. For our example, we wish the precision of the part number to be zero, so before continuing, reset the precision for this entry to zero in the same way.

4.7.2 Precision and Arithmetic Rounding

It is important to realize how decimal precision will effect your arithmetic equations. Unlike some packages, Logicalc will round your figures instead of truncating them. This, combined with Logicalc's facility to give you decimal precision if required up to 10 places, allows you to produce very accurate reports. As an example, let's multiply 3.124027 by 6.036. The answers at the various decimal precisions will be as follows:

1 place precision = 18.9
2 place precision = 18.86
3 place precision = 18.857
4 place precision = 18.8566
5 place precision = 18.85663
6 place precision = 18.856627
7 place precision = 18.8566270

If the precision is set to single entry precision as in 4.7.1, and then that entry is copied using the Copy command, all the entries copied will have the same single entry precision and will not be affected by the column precision change.

, VOORBEELD ,

4 CREATING THE EXAMPLE FILE

In this chapter we will introduce you to the Logicalc example, and put into practice the knowledge gained in the previous three chapters. The chapter is arranged so that you will first learn how to enter and amend text, and then how to use more of the Logicalc commands.

The Logicalc example we will be working with is an inventory of eight different sizes of bolts. It has been chosen because it will familiarize you with all of Logicalc's functions, yet retain a simplicity of operation.

If you are unsure of anything you have read so far, do not hesitate to go back and re-read the previous chapters.

The Type line in Current Data will indicate that the entry is text. Press:

The Type line in Current Data will now show the entry to be numeric, so press return. You will have plenty of practice using the Data Type Key in the example featured in chapter 5.

4.2 TEXT EDITOR

To demonstrate the Text Editor command, with the Field Pointer returned to location A1, enter 'PRA T #', and press return. This represents a simple typing mistake as we would like to enter 'PART #' in this location. We have already seen that we can backspace to the incorrect character and retype the entry. However, as we have already pressed return we must use the Text Editor.

To access Text Editor first access the command line. Press:

!

You should reply to the command prompt as follows:

```
        Command〉 T
```

The Text Editor options will now be displayed as follows:

```
〈INS〉,〈s/shft INS〉 insert; 〈DEL〉,〈s/shft DEL〉 delete; 〈Do〉 exit
```

Although this example is very simple, it demonstrates the Text Editor's ability to make changes without the need for re-typing the entire entry. The same principle is also used to correct data before entry. This is particularly useful should you make a mistake entering long formulae or text entries.

The second entry in our inventory example will be at location B1. This time type:

/RPART NAME ↵ (overwriting the value 4027)

4.3 COLUMN WIDTH

You will notice that the gap between the columns is only two spaces. If we had entered 'PARTS NAME' then the two columns would have run into each other. Although the default column width is 10 characters, Logicalc allows you to increase, or decrease, column width from a minimum of 3 characters to a maximum of 63. This will allow you to space your columns to fit preprinted stationery or to provide margins for comment on your reports.

Column width is varied by using the Format command. First, access the command line:

Press $\boxed{!}$

You should reply to the command prompt as follows:

```
Command> F
```

The Format options will then be displayed:

```
P)recision (2) or W)idth (10) or F)orm mode (clear)
```

Reply: **W**

The following will then be displayed:

```
Column B width (3..63)
```

Reply: **15 ↵**

The result of this action is to widen column B from 10 characters to 15 characters along all 255 rows (i.e. from location B1 through to location B255).

```
Col>|A           |B             |C             |D             |E
Row+--------------------------------------------------------------------------
  1|>        PART #<       PART NAME      SUPPLIER #   SUPPLIER NAME          COST
  2|
  3|
  4|
  5|
  6|
  7|
  8|
  9|
 10|
 11|
 12|
 13|
 14|
   +--------------------------------------------------------------------------
          Field pointer:    A1   Entry loc.:    A1

CURRENT DATA    type: text:right justified
            contents: 'PART #'
                edit:
```

4.4 THE REPEAT FUNCTION

To make our inventory report look more orderly and attractive, we will underline the column headings. This will introduce the Repeat function.

First, move the Field Pointer to location A2 and type:

```
/=- ↵
```

Location A2 will now be filled with dashes. With this function, any character following the Repeat symbol (/=) will be repeated the entire width of the column.

4.5.1 Copy Command Options

When you copy numeric values or a formula, the Copy Command will display another prompt after requesting the co-ordinates:

```
R)elative or N)o adjustment ?
```

If the value you are copying is to remain constant, then enter **N** for no adjustment. If the value is a formula and will change as it is copied from one location into another, then enter **R** . The formula will be copied and will adjust all the co-ordinates automatically, i.e. the formula will be relative to the new location. Examples of both these options are given later in the manual.

4.6 NUMERIC DATA

We have now entered our column headings, so move the Field Pointer to location A4. Before we start entering the part numbers let's have a quick word about numeric data.

4.6.1 Entering Numeric Data

A numeric expression may be entered into any location and may be a number, a co-ordinate reference or a system function. If the expression is a co-ordinate reference, the co-ordinate is replaced by the value stored at that co-ordinate.

A system function is an arithmetic operation which the system already knows how to execute without explicit directions. You must specify which function and on which list of values (argument list) the function is to operate. The system functions available are:

SUM	- sum of values in argument list
CNT	- number of items in list that are numerically valued
AVG	- mean value, i.e. SUM/CNT
MAX	- the MAX value in the list
MIN	- the MIN value in the list

The Format options will then be displayed :

```
P)recision (2) or W)idth (10) or F)orm mode (clear)
```

Reply : **P**

The following will be displayed:

```
Column A precision (0..10)
```

Reply : **0** ↲

The result of this action is to alter the decimal precision to 0 for all entries in column A. Our part number will now read '213456' and not '213456.00'.

4.7.1 Single Entry Precision

Logicalc also offers you the choice of specifying the decimal precision for a single entry, which will permit you to have values with differing decimal precision within the same column. The method for setting single entry decimal precision is very similar to the way we have just described. The procedure is listed below.

Press **[!]**

You should then reply to the command prompt as follows:

```
Command> F
```

The Format options will then be displayed:

```
P)recision (2) or W)idth (10) or F)orm mode (clear):
```

Reply : **P**

The following prompt will be displayed:

```
Column A precision(0..10)
```

Reply : **E1** ↲

5.1 USING A FORMULA

We have entered the first part number at location A4 and since the part numbers are in sequence, it will be easy to write a simple formula to calculate them. The next part number is determined by adding 1 to the previous part number, so move the Field Pointer to location A5 and type:

1 + A4 ↵

The current data line will now display:

```
CURRENT DATA    type: numeric:right justified
            contents: 1 + A4
                edit:
```

The entry is interpreted as numeric since the first character was a '1', and as a numeric, will automatically be right justified. The contents line shows that the entry is equal to 1 plus the value of location A4. The value at location A5 is therefore '213457'.

The reason for creating the formula was to be able to copy it into the remaining locations. This time when we use the copy instruction you will see the adjustment prompt, so, access the command line and reply to the prompts as follows:

```
Command> C

from coord  (>coord) :  A5 ↵

to coord (>coord) : A6>A11 ↵

R)elative or N)o adjustment ?  R
```

If N had been entered for No adjustment, then the formula at A5, (1 + A4), would have been copied to locations A6 through to A11. Since R was selected, each value was obtained by adding one to the preceding formula, i.e. the formula at A6 will be 1 + A5.

Your screen should now display part numbers from 213456 to 213463. We can now move onto the next column and another new function.

5.3 ENTERING DATA

Your Logicalc window should now contain part numbers and names. The next task is to enter the supplier name and number. In this example we have used two suppliers: the 6,9 and 12mm bolts are from ACME Bolts, whilst the remainder come from GLOBAL Bolts. Each supplier has an individual reference number, for ACME it is 104-23-33 and for GLOBAL, 216/12. You can now enter this information at the co-ordinates given below.

Remember - you have practised all the features required to enter this data. The reference numbers are to be treated as text and not as numeric equations. If you get into difficulties there is a model answer following the data, but try at least once.

At locations C4>C6, enter **104-23-33**
At locations C7>C11, enter **216/12**
At locations D4>D6, enter **ACME Bolts**
At locations D7>D11, enter **GLOBAL Bolts**

Your monitor screen should resemble the illustration overleaf. If it does not, take a look at the model answer and see where you went wrong.

5.3.1 Model Answer for Columns 'C' and 'D'

The procedures you should have taken are as follows:

Move the Field Pointer to location C4 and on the edit line type:

```
/R104-23-33 ↲
```

Access the command line and reply to the prompts:

```
Command〉 C

from coord (〉coord) :  C4 ↲

to coord (〉coord) :  C5〉C6 ↲
```

The result of this action is to copy '104-23-33' from C4 into locations C5 and C6. With the Field Pointer at location C7, the same procedure should have been used for the other supplier number.

Move the Field Pointer to location D4 and type:

```
/RACME Bolts ↲
```

Access the command line and reply to the prompts:

```
Command〉 C

from coord (〉coord) :  D4 ↲

to coord (〉coord) :   D5〉D6 ↲
```

As with the supplier number, the result of this action is to copy the contents of D4 into locations D5 and D6. With the Field Pointer at location D7, the same procedure should have been used for the other supplier name.

Reply: **A**

This prompt is asking whether you want to save all the file or only a portion of it. We will discuss this further in section 5.4.2

5.4.2 Saving an Existing File

As the method of saving a new file differs slightly from that of a file which already exists, we will run through the Save procedure again. First, access the command line and then answer the prompts as follows:

```
      Command>

      File name : EXAMPLE ↵
```

As you can see, the name of the existing file will appear in the 'File name :' prompt, as the default file name. If the existing file is on a disk in drive 2, then the file name '#5:EXAMPLE' would have appeared as default. To save the file, keeping the same name, simply press return.

The system will then prompt:

```
      file exists. Destroy old contents ? (Do/Undo) -
```

If you press the Undo key, the Save command will be aborted.

If you press the Do key, the system will check to see if the existing file is protected by a password. If it is not, the system will prompt:

```
      Password (<CR> = none) :
```

This indicates that the old file has been removed and the system is now asking if you would like a password on your new file.

If, after pressing return, the existing file is protected by a password, the system will prompt:

```
      verify password to remove :
```

5.5 THE STORAGE COMMAND

The Logicalc system has a limit to the amount of data it can store. As we are at a halfway stage in our inventory it would be a good idea to see how much storage space is still available. To execute the storage command, first access the command line and answer the prompt as follows:

```
Command> ?
```

The system will reply:

```
Room for .... Entries
```

Numbers will replace the dots informing you of the approximate number of entries that you can still make. It is advisable to check the storage space available from time to time. If you have been entering a long report and the system displays the message,

```
Memory getting LOW
```

you must stop entering data **immediately** and save the data entered. If you continue to input data you will be unable to print or save the whole report. If the system displays the message,

```
Out of Memory
```

nothing else can be entered and you will certainly lose something from the end of your file when you try to print or save the report.

The next step is to work out the Profit per unit, so move the Field Pointer to G4. Again we can write a simple formula and copy the formula from location to location. It will be necessary to set the decimal precision to three places as the calculations for profit involve the cost entries. Refer to the outline given for column E, and alter the decimal precision for column G.

5.7 THE 'AT LOCATION' FUNCTION

To enter the formula for profit per unit, we will use the 'At Location' function which you will find a very helpful aid in entering data. With the Field Pointer at location G4, follow the instructions detailed below. At location G4,

type: +

Move the Field Pointer to F4. You will notice the Entry Location indicator will remain at location G4. With the Field Pointer at F4, press:

F1

When the above key is pressed, the value where the Field Pointer is located is transferred into the Entry Location without having to work out the co-ordinates, so the edit line should now display:

+F4

We now want to move the Field Pointer to the next location, but first enter a minus sign ("-") on the edit line. With the Field Pointer at E4, press:

F1

5.8 THE EDGE COMMAND

With all the figures entered it would be useful to have a display of the data just entered. Using the Edge Command this can easily be obtained. The Edge Command locates the Field Pointer and moves its location to the upper left hand corner of the display window, displaying the next 13 rows down and as many columns across as will fit the screen. Therefore, if you position the Field Pointer on a row other than the first row, this row will be moved to the top. To demonstrate, move the Field Pointer to D1, then access the command line and reply to the prompt as follows:

```
        Command> E
```

The display on your screen should match the following illustration:

```
Col>|D              |E              |F              |G              |
Row+-----------------------------------------------------------------
  1|> SUPPLIER NAME<          COST          PRICE    PROFIT/UNIT
  2|-----------------------------------------------------------------
  3|
  4|    ACME Bolts           0.015          0.02          0.005
  5|    ACME Bolts           0.023          0.03          0.007
  6|    ACME Bolts           0.035          0.05          0.015
  7|  GLOBAL Bolts           0.053          0.08          0.027
  8|  GLOBAL Bolts           0.080          0.12          0.040
  9|  GLOBAL Bolts           0.120          0.18          0.060
 10|  GLOBAL Bolts           0.180          0.27          0.090
 11|  GLOBAL Bolts           0.270          0.41          0.140
 12|
 13|
 14|
   +-----------------------------------------------------------------
              Field pointer:     D1   Entry loc.:     D1
 [         #5:EXAMPLE]
 CURRENT DATA    type: text:right justified
             contents: 'SUPPLIER NAME'
                 edit:
```

If you want to check the figures in column H, set your field pointer to E1, and repeat the EDGE command. The figures in column H are shown in 5.9.1.

5.9.1 Model Answer for Columns 'I', 'J' and 'K'

The first task should have been to set the decimal precision of all three columns to zero. Then, the steps you should have taken are as follows:

Move the Field Pointer to location I4 and on the edit line type:

```
        5000 ↵
```

Access the command line and reply to the prompts:

```
        Command> C

        from coord  (>coord) :  I4 ↵

        to coord (>coord)    :  I5>I11 ↵
```

The figures for column 'J' should be entered straight from the data given.

With the data entered, move the Field Pointer to location K4. Either using the 'At Location' function or entering it directly, the following formula should have been entered:

```
        +I4-J4 ↵
```

The result at location K4 should be 2880. The formula should then be copied, using relative adjustment, into locations K5 to K11.

The final column is COST/ORDER and with the Field Pointer at location L4, you can enter the following formula:

```
        +E4*K4 ↵
```

The result at location L4 should be 43.20, being the Quantity Ordered multiplied by the cost. The formula should then be copied into locations L5 to L11.

Using the Edge command, check that your screen display matches the illustrations overleaf.

5 COMPLETING THE EXAMPLE FILE

In this chapter you will see how to enter and manipulate data within the Logicalc array. We have already set up column widths and learnt how to alter decimal precision, so the framework of the inventory example is now ready for the data to be entered.

The chapter is arranged so that you will have a complete inventory stored on disk after reading through and following the examples given. From this point on, when we ask you to "access the command line", we will not show you the key required. You will have seen it illustrated enough, not to need reminding of its appearance.

5.2 ALTERING THE DATA TYPE

Move the Field Pointer to location B4. We now want to enter the part names for our bolts, which range in size from 6mm to 36mm. The part name should be a text entry. If you try and enter '6mm Bolt' into the array, however, the following error message will be output:

```
Error 7-> 6?mm Bolt  Hit <cr>  (B4)
```

When you hit return, the entry at B4 will be '?n?'. To avoid this error, you must let the system know that the entry is to be treated as text. There are two possible ways of entering this data the way we wanted it, using the Data Type key or using text justification.

To change the entry to text using the Data Type key, type:

6mm Bolt

Before entering return, press the Data Type key:

$\boxed{\sqcup\sqcap}$

The data is then switched from numeric to text. When return is pressed the data will be left-justified. In order to right-justify the entry type:

/R ↲

The alternative is to use justification, so move the Field Pointer to location B5. This time type:

/R9mm Bolt ↲

The system automatically interprets the '/' as text since only text can have its justification altered on entry.

You can now enter the remainder of the data as follows:

location B6	**/R12mm Bolt**
location B7	**/R15mm Bolt**
location B8	**/R18mm Bolt**
location B9	**/R21mm Bolt**
location B10	**/R24mm Bolt**
location B11	**/R36mm Bolt**

```
Col>|A            |B            |C            |D            |E
Row+-----------------------------------------------------------------------
  1|>       PART #<     PART NAME    SUPPLIER #   SUPPLIER NAME        COST
  2|-------------- -------------- ------------- -------------- -----------
  3|
  4|        213456     6mm Bolt     104-23-33      ACME Bolts
  5|        213457     9mm Bolt     104-23-33      ACME Bolts
  6|        213458    12mm Bolt     104-23-33      ACME Bolts
  7|        213459    15mm Bolt        216/12    GLOBAL Bolts
  8|        213460    18mm Bolt        216/12    GLOBAL Bolts
  9|        213461    21mm Bolt        216/12    GLOBAL Bolts
 10|        213462    24mm Bolt        216/12    GLOBAL Bolts
 11|        213463    36mm Bolt        216/12    GLOBAL Bolts
 12|
 13|
 14|
   +-----------------------------------------------------------------------
           Field pointer:   A1   Entry loc.:   A1

CURRENT DATA    type: text:right justified
            contents: 'PART #'
                edit:
```

NOTE - In the diagram above, the Field Pointer has been returned to location A1. However, your screen display may have a different location depending on which method you employed to enter the data.

5.4 THE SAVE COMMAND

Whenever you use Logicalc it is advisable to save at regular intervals, the data entered on the screen. This eliminates the possibility of you accidentally losing all your work.

5.4.1 Saving a New File

We are approximately half-way through our inventory example, so now is a good time to save what we have done so far. Although we have shown below how to save your file on a disk in drive 1 (#4), it is recommended to use drive 2 (#5), so that you may use separate disks for different applications, leaving your System disk free.

First access the Command line and reply to the prompts as follows:

```
Command>   [⇒]

File name :  EXAMPLE ↵
```

The file name is for you to choose, and can be up to 10 characters. If you want the file to be written to drive 2, you must reply as follows:

```
File name :  #5:EXAMPLE ↵
```

The system then prompts:

```
Password (<CR> = none) :
```

If you enter a password and then press return, the system prompts again for you to enter it a second time:

```
again :
```

The password is a file protection and security feature. It can be up to 10 alphanumeric characters, but remember the password must be re-entered exactly, so do not make it too complicated. If your entry to the 'Again' prompt differs from your first entry, the Save command will be aborted. When you have entered the password correctly and pressed return, the system will prompt:

```
P)artial or A)ll
```

If you enter an incorrect password, the Save command will be aborted. If you enter the correct password, the system prompts:

```
Password (<CR> = none) :
```

After you have typed in a password and pressed return, the system will then prompt for you to type in the password a second time:

```
again :
```

This indicates that the old file has been removed and the system is now asking if you would like a new password on the new file. Once the system has acknowledged the password, or the absence of one, it will prompt:

```
P)artial or A)ll
```

This prompt is asking whether you want to save all the file or only a portion of it. If you only need to save a portion of the array, or would like to store a portion of the array under a different name, enter 'P' for partial. The system will then prompt:

```
top left corner : A1
```

This is the upper left corner of the portion you wish to save and always defaults to A1. If you would like A1 to be the upper left hand corner press return, otherwise enter the location you require and press return. The system will then prompt:

```
bottom right corner : L11
```

The default for the bottom right corner of the portion you wish to save is the last entry in the array. If you would like this to be the bottom right corner press return, otherwise enter the location you require and press return.

If you enter 'A' for All, the system will save the present contents of the Logicalc array.

The system will acknowledge that the file has been saved and the cursor returns to the edit line. As a reminder, when saving a file, Logicalc displays the file name under the bottom left hand corner of the array.

5.6 USING A FORMULA WITH THE COPY COMMAND

We are now ready to enter the data for the cost and price, and to calculate the profit per unit. The cost of the 6mm bolt is £0.015p and each of the bolts thereafter is 50% higher than the previous bolt. This figure uses three place decimal precision, so the first task is to alter the precision. With the Field Pointer at location E1, access the command line and reply to the prompts as follows:

```
Command> F

P)recision (2) or W)idth (11) or F)orm Mode (clear)
```

Reply: **P**

```
Column E precision (0..10)
```

Reply: **3 ↵**

The decimal precision is now set to three so we can enter the data for the first entry and then use a formula with the copy command for the successive entries.

Move the Field Pointer to location E4 and type:

.015 ↵

Then, with the Field Pointer at E5, type:

150%E4 ↵

This will give an answer of 0.023 which is 150% of E4 rounded off to three decimal places. Once the formula is entered at E5, use the Copy command to copy the formula with relative adjustments into locations E6 to E11. By using the Copy command with relative adjustments, you will ensure that the system automatically adjusts the co-ordinates within the formula to reflect the correct locations.

The price of the bolt is 50% higher than the cost, so we can again use a formula to get our answer. Move the Field Pointer to location F4 and type:

150%E4 ↵

As the price is set to a decimal precision of two, the system will round our answer off. Using the Copy command in the same way as before, copy the formula into locations F5 to F11.

The formula on your edit line should now read:

+F4-E4

This is the formula to be used for calculating the profit per unit, so press return. This will give an entry of 0.005 at G4. Although you may feel using the 'At Location' function is rather complicated, when working on large reports it will save you time trying to work out the co-ordinates of values you wish to use. The next step is to copy the formula with relative adjustments into locations G5 to G11.

With column G successfully completed we can use the 'At Location' function to set up column H. However, you must first change the decimal precision to one.

With the Field Pointer at H4 type:

+

The plus sign locks the Entry Location to this co-ordinate. For a numerical entry you can enter either a plus or minus sign, whereas a space is entered for text.

The formula for %Profit per bolt sold is profit per unit divided by price multiplied by 100, or:

G4/F4*100

To obtain this formula using the 'At Location' function, move the Field Pointer to G4 and press:

F1

The edit line will now show part of our formula but before moving to the next location, type:

/

With the Field Pointer at the next location (F4), press:

F1

Without moving the Field Pointer, type:

***100 ↵**

The answer 25.0 will now be displayed in H4 and the formula will be in the contents line. You can now enter the remainder of data using the Copy command with relative adjustment.

5.9 COMPLETING ENTRY OF THE DATA

You have covered several functions and commands, so now is the time to practise what you have learnt. The next three columns are 'QUANTITY STOCKED', 'QUANTITY ON HAND' and 'QUANTITY ORDERED'. All the information you need to complete these columns is given below. If you have problems, re-read the relevant sections of this chapter. Remember, you cannot have fractions of a bolt so alter the decimal precision. There is a model answer on the next page, but try at least once before reading it.

QUANTITY STOCKED - the company stocks 5000 of each bolt size.

QUANTITY ON HAND - the number of bolts on hand is as follows:

6mm	2120
9mm	2900
12mm	3780
15mm	3110
18mm	4105
21mm	2451
24mm	3063
36mm	4350

QUANTITY ORDERED - the difference between the Quantity Stocked and the Quantity on Hand.

You may also like to try and work out the cost of the order for each bolt. The COST/ORDER is the QUANTITY ORDERED, multiplied by the COST. The technique used to work out this figure is also given on the next page, so don't worry if you get stuck!

```
Col>|H               |I                        |J                       |
Row+----------------------------------------------------------------------
  1|>         %PROFIT<         QUANTITY STOCKED      QUANTITY ON HAND
  2|-----------------  ---------------------------  ---------------------
  3|
  4|              25.0                     5000                  2120
  5|              23.3                     5000                  2900
  6|              30.0                     5000                  3780
  7|              33.8                     5000                  3110
  8|              33.3                     5000                  4105
  9|              33.3                     5000                  2451
 10|              33.3                     5000                  3063
 11|              34.1                     5000                  4350
 12|
 13|
 14|
   +-------------------------------------------------------------------
              Field pointer:     H1    Entry loc.:     H1
 [          #5:EXAMPLE]
 CURRENT DATA     type: text:right justified
              contents: '%PROFIT'
                  edit:
```

```
Col>|K                              |L                |
Row+----------------------------------------------------------------------
  1|>          QUANTITY ORDERED<      COST/ORDER
  2|-------------------------------  ---------------
  3|
  4|                         2880          43.20
  5|                         2100          48.30
  6|                         1220          42.70
  7|                         1890         100.17
  8|                          895          71.60
  9|                         2549         305.88
 10|                         1937         348.66
 11|                          650         175.50
 12|
 13|
 14|
   +-------------------------------------------------------------------
              Field pointer:     K1    Entry loc.:     K1
 [          #5:EXAMPLE]
 CURRENT DATA     type: text:right justified
              contents: 'QUANTITY ORDERED'
                  edit:
```

6.1 SYSTEM FUNCTIONS

Logicalc provides you with five different system functions:

- summation
- average
- maximum value
- minimum value
- numeric count

In order to demonstrate system functions, we will use our example and perform the following tasks: total cost of the order for all bolts, total number of bolts stocked, on hand and on order, and the number of different types of bolt.

6.1.1 The SUM Function

Instead of writing a formula to add up the values of locations, e.g. L4 + L5...+ L11, we can use a System function. Move the Field Pointer to location L13, and using the Repeat Function (/=), underline the column with dashes. At location L14, type:

+SUM (L4 > L11) ↵

The plus sign tells the system the entry is numeric. An alternative would be to use the Data Type key. SUM is the system function used to calculate the sum of the given list of values. The list of values in this instance is L4>L11, however if we had only wanted some of those values we could have separated the chosen locations by commas, e.g. (L4, L6, L10, L11). The answer at location L14 for the total from L4 to L11 should be '1136.01'.

We can now practise using the SUM function, but first use the copy function to underline columns I13 to K13. With the Field Pointer moved to I14 , using the SUM function, enter a formula to total columns I4 to I11. The answer at I14 should be '40000' for the total number of bolts stored.

You can repeat the procedure for column J to give the Quantity on Hand. The answer at location J14 should be '25879'.

6.1.3 The MAX and MIN Functions

The MAX and MIN system functions are very straightforward, as we saw briefly when we introduced the Calculator function. The functions will find the MAXimum or MINimum value of a string of locations e.g. type:

```
+MAX (E4:F6>F8:H7)?
```

The answer you should get is '33.80', being the highest value of the locations. Similarly, type:

```
+MIN (E4:F6>F8:H7)?
```

The answer you should now get is '0.015', being the lowest value of the locations.

6.1.4 The AVG Function

The AVG function will give you the mean value of a list of locations, i.e. it divides the summation of values in the argument list by the count. As an example, we will find out the average cost of our bolts, so using the calculator function again type:

```
+AVG (E4>E11)?
```

The answer for the Sum of E4 to E11, divided by the Count will be '0.097'.

Note - The answers you obtain from the calculations on this page will appear on the screen with 12 decimal places.

To find the percentage increases for the 24 and 36mm bolts, first move the Field Pointer to location U5 and repeat the process. Then move the Field Pointer to location U6 and run through the procedure again for the 36mm bolt.

The second exclamation mark represents the variable in the expression. If we were therefore trying to raise the variable to some power, then as many exclamation marks must be entered as the number of powers the variable is being raised to.

E.g. 4x**2 + 3x + 5 (four times x squared, plus three times x, plus five) would be !4*!*!+3*!+5. This says take 4 times x times x plus 3 times x plus 5.

6.2.1 Sample Applications for User Defined Functions

As the name suggest, these functions are defined by you, the user. At first, it may seem difficult to find some applications, but the more you use Logicalc, the more you will find that it is convenient to have the facility to calculate one value given another value.

Below we have listed a few sample applications to give you an idea of the diverse ways you can use the function.

- To find the approximate area of a circle.
```
      formula        : pi r**2
      logicalc entry : !3.1416*!*!   with r=!
```
- Finding 'n' raised to the fourth power.
```
      formula        : n**4
      logicalc entry : !!*!*!*!   with n = !
```
- Finding average range of n entries and variable entered.
```
      formula        : avg = (sum of range + x)/n+1
      logicalc entry : !avg(F4>F11,!)   with x = !
```
- Finding a salesman's commission.
```
      formula        : 10.25% product price
      logicalc entry : !10.25%!   with product price = !
```

6 SYSTEM AND USER DEFINED FUNCTIONS

In this chapter we will demonstrate the advantages of a computer report writer. The chapter is arranged so that after a brief introduction to the functions, you will be able to practise using them with the Logicalc example we have just built up.

Before you start this chapter, refer back to chapter 5.4.2 and follow the instructions carefully, saving the example file. You should get into the habit of making regular saves of your Logicalc files. This will prevent any accidental loss of your data.

When we get to column K, we can either use the Sum function as before, or type the following formula:

+I14 - J14 ↲

This is a simple subtraction and in this example, will provide you with the answer a little more quickly than the Sum function. The answer at Location K14 should be '14121'.

6.1.2 The CNT Function

Move the Field Pointer to location A13 and underline the column as before. We would like to know how many different types of bolt are stocked and we can do this using the CNT (count) function. This will count the number of numerically valued entries in the argument list. With the Field Pointer at location A14, type:

CNT (A4 > A11) (do not press ↲)

As the answer is to be numeric, we must switch the entry from text to numeric before hitting return, so press:

$\boxed{\sqcup\!\sqcap}$ (followed by ↲)

The result at location A14 will be '8'. To make the meaning of '8' clear, move the Field Pointer to location B14 and type:

BOLT SIZES ↲

Since numeric entries are always right justified, and text entries are always left justified, the result of columns A and B will show '8 BOLT SIZES'.

6.2 USER DEFINED FUNCTIONS

When using a User Defined function, ensure the Field Pointer is at the location you want the resulting answer to be displayed. This will be the location where you enter the expression which defines the function and the value you wish to evaluate. Now we have a general idea where to put a User Defined function, let's find out exactly what the format will be.

The expression may be any of five standard arithmetic operations: addition, subtraction, multiplication, division, or a percentage. It can also include a system function, numeric constant or reference co-ordinate.

Using the Logicalc example, imagine you wanted to know what the retail prices would be for the 21, 24 and 36mm bolts if the current prices were raised either 50%, 75% or 125% (the current prices are .18, .27 and .41 respectively). Move the Field Pointer to location U4 and set the precision to three decimal places, then type:

```
!!%.18 ↵
```

Although this expression may appear bizarre it has a simple explanation. The first exclamation mark ('!') informs the system that the following entry will be a user defined function. The second exclamation mark represents a variable, one exclamation per variable in the expression. In our example we only want to find the value of a fixed constant, the constant value being .18 which is the current price for the 21mm bolt. The second exclamation mark therefore says 'take x % of .18'. With the Field Pointer still at location U4, type:

```
150 ↵
```

The user defined function inserts '150' in place of the second exclamation mark, so the expression now reads '150%.18'. The answer, 0.270, will now appear in location U4. To get the results of a 75% rise, without moving the Field Pointer, simply type:

```
175 ↵
```

The result of a 75% price rise, 0.315, will now be displayed at location U4. For 125% rise, type:

```
225 ↵
```

The result of a 125% price rise, 0.405, is now displayed.

6.3 THE COMMENT FUNCTION

Although not strictly a system or user defined function, the Comment Function can be used to good advantage in conjunction with them.

The Comment function allows you to include a comment in a numeric entry which will not be evaluated and so act as a reminder for what the entry represents. The comment does not appear in the Logicalc window or on a printout, but will be displayed on the contents line of the relevant location.

As an example, let's put a comment with the Quantity Ordered total. With the Field Pointer at location K14, type:

+SUM (K4>K11)

Before you hit return, we have to enter our comment, so press the Comment function key:

$$\boxed{\begin{array}{c}|\\ \backslash\end{array}}$$

Anything that is typed in after pressing the key will be taken as a comment, so type:

total ordered

The Current Data line should now be as follows:

```
CURRENT DATA   type: numeric:right justified
           contents: +SUM (K4>K11)\total ordered
               edit:
```

7.1 THE RECALCULATE COMMAND

You will find the Recalculate command to be one of the most useful in Logicalc (remember, you saw the use of this command in the BALSHEET demonstration). Imagine after completing the inventory file you find the cost of the 6mm Bolt is to be increased by 10%. Since the cost of all other bolts is based on the cost of the 6mm bolt, the values for cost, price, profit per unit, %profit and cost per order are all involved.

The Recalculate command will allow you to use one command to recalculate everything automatically. Using the Inventory example, we will implement the 10% cost increase using the following simple steps.

First, move the Field Pointer to E4, and type:

110%E4 ↲

This expression will be rounded to three decimal point precision, giving you the answer of 0.017 at location E4. You can now move to the next location.

Access the command line and answer the prompts as follows:

```
Command > R

Recalculate -  A)ll E)ntry   A
```

All the values will now be recalculated to accommodate the increase. When 'A' is entered, the Recalculate command examines each row and recomputes any formulae or system functions in the row. Although in our example we recalculated all the entries (in this instance they were inter-dependent), the Recalculate command allows you the option of recalculating only the entry at the Entry Location, by entering 'E'.

Remember, any entry that evaluates to something divided by zero will result in the error symbol '?n?'.

7.3 THE MERGE COMMAND

Using the Recalculate command you can produce new reports with new figures while preserving all the text entries of your original report, saving the need to write an entire report again because of a change in numerical data. Another way to preserve the report text whilst substituting new numerical data is with the Merge Command. This is particularly useful if you want to preserve your report text, but because of complex dependencies or too many changing values, the Recalculate command would be too complex to use. The Merge command allows you to create two or more separate files, and then superimpose the files, one on top of the other. By using the Merge command you do not need to spend time re-entering and recalculating numerical data.

As we do not want to quit the EXAMPLE file we will demonstrate the merge command in a series of hypothetical examples. Later, when you have read the following sections, you may want to try the procedures out for yourself.

Based on our Example file, we could create a standard inventory file containing:

the entries in columns A, B, C, and D;
the column headings for E, F, G, H;
the entries for column I,
headings for columns J, K, and L

Note - these files can be created in several ways. Here are 2 suggestions you may adopt, after having completed and saved the example:

- From the example file, delete all the entries you do not want for the STDINV and save the STDINV file. Reload the Example file, delete all entries you do not need for the INVDATA file.

- Delete the example file from the screen, and enter the data for each file from scratch.

This file we could call INVDATA for INVentory DATA. The illustration below is the screen display for the INVDATA file at the same location of the array as the display for STDINV.

```
Col>|C              |D              |E            |F          |
Row+-----------------------------------------------------------
  1|
  2|
  3|
  4|                                        0.017 >     0.03<
  5|                                        0.026       0.04
  6|                                        0.039       0.06
  7|                                        0.059       0.09
  8|                                        0.089       0.13
  9|                                        0.134       0.20
 10|                                        0.201       0.30
 11|                                        0.302       0.45
 12|
 13|
 14|
   +-----------------------------------------------------------
            Field pointer:    F4   Entry loc.:    F4
[        #5:INVDATA]
CURRENT DATA    type: numeric:right justified
            contents: 150%E4
                edit:
```

The INVDATA file contains only the numerical information that is likely to change (variable data). Instead of writing an entirely new inventory or report each time the data changes or having to try and edit a file with complicated formulae and dependencies, we can create a new file with the new data and merge this file with our STDINV file.

The Merge command also lets you choose the location for the upper left hand corner of any of the Merge files. This allows you to adopt a modular approach to constructing inventories and reports, minimizing duplication of work.

To merge our two files, STDINV and INVDATA, we would adopt the following procedures. First, access the command line and answer the prompts as follows:

```
Command> M

File name: #5:STDINV↵

load position: A1↵
```

When you merge a file onto the array with the upper left hand corner at a different location to the one originally created, the results could be disastrous. Logicalc will alter the formulae of a merged file so that they correspond correctly to the new locations. However, any formulae which reference locations outside the module being merged will not contain the updated locations. These results will obviously be incorrect. It is therefore important to ensure that the modules being merged only contain formulae which reference locations within that module.

7.4 THE AUTO COMMAND AND FORM MODE FUNCTION

The Auto Command is used to make it easier to edit specific entries that are subject to changes. The command is used in conjunction with the Form mode specification of the Format command. It works in a similar way to the tab facility of a typewriter. Move the Field Pointer to a location where an entry is liable to change and set Form Mode, then move to the next location and continue the procedure until you have set Form Mode at all the entries you require. When you execute the Auto command, the Field Pointer will go to the first location that has Form Mode set. The Auto command will automatically delete the entry at that location so you may enter the new information immediately. When you press return, the Field Pointer moves to the next entry with Form Mode and so on until all the entries are changed. The Auto command proceeds in a row by row order through the file, one entry at a time.

Note - Once you have set Form Mode for an entry, that entry will be deleted on executing the Auto command.

Using our inventory example, the two entries likely to change most are Quantity on Hand and Price. To show how much easier it is to edit using Auto, with the Field Pointer at location F4 access the command line and then answer the prompts as follows:

```
Command>  F

P)recision (2) or W)idth (11) or F)orm mode (clear)    F
```

The system will display:

```
F)orm mode (set)    Hit <cr>
```

Press the return key, and the cursor will return to the edit line. Repeat this procedure for locations F5 to F11 and J4 to J11.

By typing 'F' the Form Mode is automatically reset back to clear. With Form Mode clear, the Auto command will no longer jump to this location.

7.5 THE INSERT COMMAND

The Insert command will allow you to insert either a new row or column into an existing file. This can be very useful if you have to add a new line or column to a report, or product to an inventory. To demonstrate the insert command, we will add new items to our inventory, the 28mm bolt and the suppliers telephone number.

7.5.1 Inserting a New Row

To amend our inventory, first move the Field Pointer to location A11, as we want to enter our new line between the 24 and 36mm bolts, then access the command line. The prompts should be answered as follows:

```
     Command〉          INSERT

     Insert : R)ow  C)olumn     R
```

The result of this action is to insert a row from A11 to DW11. The previous row 11 now becomes row 12 and the remaining rows all move one place down. Although we moved the Field Pointer to location A11, it could have been anywhere along row 11. Now you can enter the new information into the row and the data is as follows:

- Part #	213463
- Part Name	/r28mm Bolt
- Supplier #	/r104-23-33
- Supplier Name	/rACME Bolts
- Cost	150%E10
- Price	150%E11
- Profit/Unit	+F11-E11
- %Profit	+G11/F11*100
- Quantity Stocked	5000
- Quantity on Hand	0
- Quantity Ordered	+I11-J11
- Cost/Order	+E11*K11

NOTE When you have completed your new entries, it is important that you save the file again at this point. This file will be used for re-loading in 7.7 and is the basis for all remaining examples.

7.6 THE DELETE COMMAND

Using our Inventory file again, imagine the consequences of a decision to stock no longer the 36mm Bolt. Without Logicalc it would mean a considerable amount of work in rewriting the report and recalculating the values. However, we only need two commands and our inventory will be updated.

With the Field Pointer at location A12, access the command line and answer the prompts as follows:

```
        Command>              DELETE

        Delete : A)ll R)ow C)olumn E)ntry    R
```

If you followed the steps carefully you should have the following error message displayed:

```
        ERROR: would delete refs at A15, J15, K15, L15, M15
```

This is a safety feature of the Delete command to prevent you deleting any location which has other values dependent on it. It also stops you from accidentally deleting a value and causing other values to become undefined as a result. In order to delete the entries that have values dependent on them, you will have to alter the formulae of the dependent values. For our example, we need to alter the four references stated in the error message. Move the Field Pointer to location A15 and look at the contents line. This shows that the formula to be changed at A15 is a Count system function, so type:

+CNT (A4>A11) ↲

The formula is now altered to exclude the value at A12. Now change the other locations (J15, K15, L15, M15) in a similar way, taking note of the contents line in each instance.

When that simple task is completed, move the Field Pointer back to location A12 and repeat the Delete command. This time, after you have entered 'R', the system will prompt:

```
        Verify (Do/Undo) -
```

- Press the Do key

7.7 THE LOAD COMMAND

The Load command is used to reload files into the Logicalc array from disk. We have seen briefly how to use the Load command in Chapter 3, however we will now run through the procedures in more detail.

You can clear the screen and load your file (remember you were told to save it in 7.5.2). First access the command line and then answer the prompts as follows:

```
     Command〉          DELETE

     Delete : A)ll R)ow C)olumn E)ntry      A

     Verify (Do/Undo) -  Press the Do key
```

You should now have a blank array on your monitor screen. Although we have cleared the screen before loading a file, it is not a mandatory procedure. To execute the Load command, first access the command line and then answer the prompts as follows:

```
     Command〉

     File name :    #5:EXAMPLE↲

     load position : A1  ↲
```

If you used a password when saving the file, note that you will be required to type it in, after you have specified the name of the file.

The result of this action is to load the file, EXAMPLE, back into our Logicalc array with the upper left hand corner at position A1. Although the system defaults to A1, you are free to enter any location within the array but remember, it will be easier to use if loaded at A1.

This ability to alter the load position can be very useful, especially in conjunction with the Merge command.

There is one important point to remember, as we stated earlier, when using the Merge command. The file you are merging onto the Logicalc array will override the existing file if entries in both files have the same co-ordinates.

Using the Goto key, move the Field Pointer to location B2, access the command line, and call the Extended What command:

```
Command>  =
```

The following prompt will be output:

```
Lock: R)ow C)ol B)oth
```

If we type 'R' with the Field Pointer at B2, then rows 1 and 2 will be locked. If we type 'C' with the Field Pointer at B2, then columns A and B will be locked. If we type 'B' with the Field Pointer at B2, then both rows 1 and 2 and columns A and B will be locked.

By 'locked' we mean that the entries will always be displayed on the screen, no matter where the Field Pointer is located. Although the lock will take in as many rows and columns that will fit on the screen, we will obviously never be in a position where we want the whole screen to be displayed all the time.

We must emphasize again that care must be taken with the location of the Field Pointer when executing Extended What. With the Field Pointer at location B2, answer the prompt as follows:

```
Lock: R)ow C)ol B)oth   R
```

Move the Field Pointer to location M15, and the monitor display should resemble the illustration below.

```
Col>|K                      |L                   |M                 |
Row+-------------------------------------------------------------------
  1*       QUANTITY ON HAND     QUANTITY ORDERED          COST/ORDER
  2*----------------------- -------------------- -------------------
  9|                   2461                 2539              340.23
 10|                   3063                 1937              389.34
 11|                      0                 5000             1510.00
 12|                   4350                  650              294.45
 13|
 14|----------------------- -------------------- -------------------
 15|                  26149               18851 >           2865.75<
 16|
 17|
 18|
 19|
 20|
  +-------------------------------------------------------------------
                 Field pointer:    M15   Entry loc.:    M15
[          #5:EXAMPLE]
CURRENT DATA     type: numeric:right justified
             contents: '+sum(M4>M12)'
                 edit:
```

7.9 THE QUIT COMMAND

The Quit command is used to exit the Logicalc program and gain access to the System Menu. When you have finished working with Logicalc and you have saved your file, access the command line and answer the prompts as follows:

```
Command> Q

Verify (Do/Undo) -
```

If you press Undo, the Quit command will be aborted and control will be returned to the edit line. If you press Do, the System Menu will be displayed on the screen (see section 1.6).

From the system menu you will be able execute the LCDUMP program (see chapter 12) or the DISKORG program. In both cases, you will have to select option 1 (EXECUTE PROGRAM). To return to Logicalc, you must select option 3.

7 FILE MANIPULATION

You have now finished a very thorough and accurate inventory record. In this chapter we will demonstrate some of the commands which will enable you to edit, merge and reload the file back into Logicalc. The chapter is arranged so that each command will follow logically from the next as opposed to an alphabetical order.

7.2 THE ORDER COMMAND

The Order command is used in conjunction with the Recalculate command, to determine in which direction the Recalculate command will move. Normally, the entries in the array are evaluated from left to right through each column and then through the next row in the same way until all the rows have been evaluated. The Order command switches this order of evaluation so that the entries are computed from top to bottom in each column and then through the next column in the same way until all the columns have been evaluated. The diagram below illustrates how the two sequences of evaluation available with the Order command would recalculate through 20 entries.

```
Left to Right, Top to Bottom          Top to Bottom, Left to Right
      1   2   3   4   5                       1   5   9  13  17
      6   7   8   9  10                       2   6  10  14  18
     11  12  13  14  15                       3   7  11  15  19
     16  17  18  19  20                       4   8  12  16  20
```

The use of the Order command is sometimes made necessary because the Recalculate command, in moving in its normal manner, may encounter a formula that depends on a value at a co-ordinate that has not been yet been evaluated. This would give an incorrect answer at our original formula, e.g if there was a formula at D2 and the result of this formula depended on the value at B6, should the value at B6 change then when the Recalculate command was executed, the formula at D2 would be evaluated before the expression at B6. The result at D2 would therefore be incorrect. The Order command allows you to recalculate in the order which would give you the correct answer. To execute the Order command, access the command line and enter:

```
Command> O
```

Each time the Order command is accessed it will switch the evaluation order, so if this time the system replies:

```
Order is Top to Bottom, Left to Right
```

execute the Order command again and it will switch the evaluation order to its normal setting:

```
Order is Left to Right, Top to Bottom
```

We could give this file the title of 'STDINV', for STanDard INVentory. Illustrated below is the screen display that would result if we were to isolate columns C to F in STDINV.

```
Col>|C              |D                  |E                |F
Row+-------------------------------------------------------------------
  1|>   SUPPLIER #<  SUPPLIER NAME            COST             PRICE
  2|--------------- ------------------ ---------------- ----------------
  3|
  4|     104-23-33      ACME Bolts
  5|     104-23-33      ACME Bolts
  6|     104-23-33      ACME Bolts
  7|        216/12    GLOBAL Bolts
  8|        216/12    GLOBAL Bolts
  9|        216/12    GLOBAL Bolts
 10|        216/12    GLOBAL Bolts
 11|        216/12    GLOBAL Bolts
 12|
 13|
 14|
   +-------------------------------------------------------------------
            Field pointer:    C1   Entry loc.:    C1
 [         #5:STDINV]
 CURRENT DATA    type: text:right justified
             contents: 'SUPPLIER #'
                 edit:
```

The information contained on STDINV is subject to very little change and therefore can be stored in a separate file to be used when needed. All the information that is subject to change can be stored in a separate file or files. We could therefore create another file containing the changeable values:

PRICE
COST
PROFIT/UNIT
% PROFIT
QUANTITY ON HAND
QUANTITY ORDERED
COST/ORDER

The INVDATA file is currently displayed, so our replies to the prompts tell the system we wish to merge a file called STDINV with our existing file, INVDATA, with the upper left corner at A1. The results of merging these two hypothetical files is shown below.

```
Col>|C              |D                |E              |F              |
Row+------------------------------------------------------------------
  1|>   SUPPLIER #<   SUPPLIER NAME             COST           PRICE
  2|---------------  ----------------  ---------------  ---------------
  3|
  4|     104-23-33        ACME Bolts           0.017            0.03
  5|     104-23-33        ACME Bolts           0.026            0.04
  6|     104-23-33        ACME Bolts           0.039            0.06
  7|        216/12      GLOBAL Bolts           0.059            0.09
  8|        216/12      GLOBAL Bolts           0.089            0.13
  9|        216/12      GLOBAL Bolts           0.134            0.20
 10|        216/12      GLOBAL Bolts           0.201            0.30
 11|        216/12      GLOBAL Bolts           0.302            0.45
 12|
 13|
 14|
   +------------------------------------------------------------------
             Field pointer:    C1   Entry loc.:    C1
[        #5:STDINV]
CURRENT DATA     type: text:right justified
             contents: 'SUPPLIER #'
                 edit:
```

7.3.1 Points to Note When Using the Merge Command.

It is relatively simple and straight forward to use the Merge command. However, if care is not taken, you could get your files into disarray.

When a file is being merged, the file that is to be merged onto the array will replace the original entry, e.g. if the column widths on INVDATA were narrower than on STDINV, then the system would overwrite the original column widths with those from STDINV. Similarly, and more importantly, the system will replace any data in the original entry with data from STDINV.

Once all the form modes have been set, move the Field Pointer to location A1. To update the Price and Quantity on Hand, all we need do now is use the Auto command. First, access the command line and reply to the prompt as follows:

```
Command⟩        F2
```

The system will then prompt:

```
Automatic Form Control Mode ----⟩ Verify (Do/Undo) -
```

If you press the Do key the Field Pointer will move automatically to location F4, the location where our first Form mode was set. The entry at F4 (and all the other entries with FORM mode) will have already been deleted, so we can enter our new value, 0.04.

When you press return, the Field Pointer will move on to the next location in the row where Form Mode has been set, J4.

At J4 you can enter the new data for the Quantity on Hand, type:

2120 ↲

This time when you press return, the Field Pointer will move to the next row and the next location, F5. Continue to use the Auto command in this way to enter the remaining data which is as follows:

- location F5,	**.05**	location J5,	**2920**
- location F6,	**.08**	location J6,	**4000**
- location F7,	**.11**	location J7,	**3120**
- location F8,	**.15**	location J8,	**4115**
- location F9,	**.22**	location J9,	**2461**
- location F10,	**.32**	location J10,	**3063**
- location F11,	**.48**	location J11,	**4350**

If you want to abort the Auto command before you have finished entering data, press the escape key:

If you decide to restore some of the Form Modes back to the original status (clear), access the command line and answer the prompts as follows:

```
Command⟩    F

P)recision (2) or W)idth (10) or F)orm Mode (set)    F
```

Before you can use the Recalculate command to recompute the totals to include the information you have just entered, you must alter the formula for cost, and the part number on the row following the new row (i.e. A12 and E12). The other formulae will take into account the line change, but Part number, and Cost will not change as they are dependent on the values in the row above, and they had not been entered when the row was inserted.

It is important to note that if there are dependencies in the vertical direction (e.g. dependent on A10) at or below the newly inserted row, then the formulae for calculating the values in those rows will remain the same even though the rows have been moved to a new location. If your new row of information contains data or formulae that will alter the value of an entry in one of the moved rows, then all the formulae below the newly inserted row will have to be altered. This is carried out by entering a new formula at the row below the newly inserted one, then copying it with relative adjustment.

After altering the formulae use the recalculate command to get correct values.

7.5.2 Inserting a New Column

In the same way you inserted the new row, you can insert a column. In our example we require the addition of the supplier's telephone number at column E. First, move the Field Pointer to column E and access the command line. The prompts should be answered as follows:

Command 〉 INSERT

Insert: R)ow C)olumn **C**

The effect of this is to leave column E blank and to move columns E through to L one column to the right. The inserted column will take the width of the column it replaces. Although this is a sufficent width for this example you may need to change the width before entering data. You can make up your own telephone numbers and enter them, but remember as it is entered as a text entry there is no need to alter the precision first. Don't forget to include a column heading for the column you have just inserted.

This prompt ensures against accidentally deleting something you would like to save. If you press the Undo key, the Delete command will be aborted and control will return to the edit line. If you press the Do key, the deletion will be carried out and then any following rows or columns will be moved to fill the gap.

The other options for the Delete command are:

- Entry (E)
  If you enter 'E' the system will delete the entry at the current entry location. The system will **not** ask for verification, and will delete the entry even if other values are dependent on it.

- Column (C)
  If you enter 'C' and there are no other values dependent on any of the values in the column, then the system will ask for verification before deleting the column. If there are any dependent entries the error message will occur, and the deletion will be aborted.

- All (A)
  If you enter 'A' and then press the Do key, all the information on the screen will be deleted. The file on the disk will be safe and available for you to load back into Logicalc, but any changes made since the last Save will be lost.

When you delete an entry within Logicalc, the memory that entry previously occupied will be released for future use.

Notice that the line you have just deleted has been returned to the array. This is because the save was taken before you deleted the line and you have not made one since. It is advisable to save the existing file before merging onto a new file, particularly if changes have been made since loading.

7.8 THE WHAT AND EXTENDED WHAT COMMANDS

The What and Extended What commands are two very useful commands, which enable you to determine immediately what a particular entry or range of entries represents. They are described below.

7.8.1 The What Command

The What command enables you to find out the Row or Column headings without the necessity of moving the Field Pointer to a location where you can read it. Imagine our example to be twice the size it is now, and move the Field Pointer to location H21. In our double size example, we would have an entry at this location, but no indication as to what it represents. Access the command line and answer the prompt as follows:

```
        Command>   W
```

The system will display the following message:

```
        row, column = ---, PROFIT/UNIT
```

This tells us row 21 has no title, but column 'H' has the title 'PROFIT/UNIT'. This facility will prove useful when dealing with a long inventory or report.

7.8.2 The Extended What Command

The Extended What command takes the What command a stage further. It offers you the facility to 'lock-in' multiple rows or columns enabling you to have them constantly displayed on the screen. If you move the Field Pointer to L15, you will notice the column titles are not in view. We will now use the Extended What command to demonstrate several options.

Before executing the Extended What command, you must ensure that the Field Pointer is at the correct location, that is, at the location you wish to 'lock-in'.

The diagrams below demonstrate the displays if we had entered 'C' and 'B' respectively with the Field Pointer at B2 and then moved to location M15. To delete a Lock function enter the same lock prompt reply again and the row, column or both row and column will be unlocked.

```
Col>*A           *B               |L               |M              |
Row+--------------------------------------------------------------
  8|     213460       18mm Bolt             885            78.77
  9|     213461       21mm Bolt            2549           340.23
 10|     213462       24mm Bolt            1937           389.34
 11|     213463       28mm Bolt            5000          1510.00
 12|     213464       36mm Bolt             650           294.45
 13|
 14|-----------                 --------------- ---------------
 15|         9 BOLT SIZES                 18851 >        2865.75<
 16|
 17|
 18|
 19|
 20|
 21|
   +--------------------------------------------------------------
             Field pointer:    M15   Entry loc.:    M15
 [        #5:EXAMPLE]
 CURRENT DATA    type: numeric:right justified
             contents: '+sum(M4>M12)'
                 edit:
```

```
Col>*A          *B               |L                |M              |
Row+--------------------------------------------------------------
 1*      PART #         PART NAME   QUANTITY ORDERED      COST/ORDER
 2*----------  --------------  ------------------  ---------------
  9|    213461       21mm Bolt                2539 .         340.23
 10|    213462       24mm Bolt                1937           389.34
 11|    213463       28mm Bolt                5000          1510.00
 12|    213464       36mm Bolt                 650           294.45
 13|
 14|----------                 ---------------- ---------------
 15|        9 BOLT SIZES                  18851 >        2865.75<
 16|
 17|
 18|
 19|
 20|
   +--------------------------------------------------------------
             Field pointer:    M15   Entry loc.:    M15
 [        #5:EXAMPLE]
 CURRENT DATA    type: numeric:right justified
             contents: '+sum(M4>M12)'
                 edit:
```

For our first example, imagine we have more 6mm bolts on hand than are usually stocked. In our inventory, the amount on hand is subtracted from the amount stocked, the answer is quantity ordered, which is multiplied by the cost. This would give us a negative answer for quantity ordered and the Cost/Order column, indicating the supplier would have to pay us for ordering bolts from them. Obviously, this is an unlikly situation. The solution is to set up a Conditional expression to determine whether the amount on hand is greater than the amount stocked. If it is, the amount ordered will be set to zero, otherwise the subtraction will be carried out as before.

Move the Field Pointer to location L4 and then follow the instructions given below. On the edit line type:

```
        +K4<J4;J4-K4 ↲
```

The formula states that if the amount on hand (K4) is less than the amount stocked (J4), then subtract K4 from J4 and enter into L4, else the value at L4 will be zero, because a false conditional statement is assigned the value of zero.

Now, move the Field Pointer to location K4 and type:

```
        6000 ↲
```

This has altered the number of bolts to 6000. Move the Field Pointer to location L4, access the command line and respond to the prompts as follows:

```
        Command > R

        Recalculate : A)ll E)ntry  E
```

The result in L4 will now show the quantity ordered to be zero. This is a helpful way of keeping our records correct and avoiding negative amounts in the order column. Before we go on to define the conditional statement, check if the formula works the other way, by entering the original amount of 2120 at K4. Then recalculate at L4, and the result should be '2880'.

8.2.2 True Condition

After the first semicolon (';') you may enter the statement to be executed should the statement be true. The statement may be a number, co-ordinate, a formula involving any combination of both or a string of five or less characters enclosed in double quotes (e.g. "yes"). If you enter a condition followed by two semicolons, the value assigned will be zero if the condition is true.

8.2.3 False Condition

After the second semicolon you may enter the statement to be executed should the statement be false. This statement is optional and if a statement is not entered, zero will be assigned if the condition is false. This was the case with our first example. The statement may be a number, a co-ordinate, a formula involving any combination of both or a string of five or less characters enclosed in double qoutes (e.g. "no").

8.3 CONDITIONAL STATEMENT EXAMPLES

Having explained the theory, we will now demonstrate a few examples involving conditional statements. For our second example we will solve the same problem we encountered in the first example only using a different method. Move the Field Pointer to location L5 and type:

```
        (K5<J5) * (J5-K5) ↲
```

The effect of this is to evaluate the first expression (K5 < J5). If the condition is true, the expression is assigned a value of one. The result of the first expression, one, is multiplied by the result of the second expression (J5-K5) to obtain the figure for L5. If the first expression is false, then the result will be zero, which, when multiplied by the second expression, will also be zero. This will give us a zero figure at L5 (Quantity Ordered). To verify this, change K5 to 7000 and then recalculate for L5. The value at L5 will be zero. To ensure that the 'true' value of the conditional statement evaluates properly, enter the original value, '2920', at location K5 and recalculate for L5. The answer you get should correspond with your original answer of 2080.

Our last example shows how to scale the success of your bolts in another way. For this example we will assign a value of 100 to those bolts which satisfy both of the conditions and a value of 50 to any part which does not satisfy both of those conditions. Move the Field Pointer to location P4 and type:

```
        (M4>25)*(L4>20%J4);100;50 ↲
```

As a result of this formula, if either expression is false the numerical value becomes zero, and zero multiplied by anything results in zero. Both expressions must therefore be true for a bolt to be valued at 100, otherwise it is valued at 50. Using relative adjustment, copy the formula into locations P5 to P12. If you Edge on column N, and then position to P4, the display on your monitor screen should be as follows:

```
Col>|N            |O             |P              |Q           |
Row+-------------------------------------------------------------
  1|
  2|
  3|
  4|                          1.00>        100.00<
  5|                          1.00         100.00
  6|                          1.00          50.00
  7|                          1.00         100.00
  8|                          2.00          50.00
  9|                          1.00         100.00
 10|                          1.00         100.00
 11|                          1.00         100.00
 12|                          2.00          50.00
 13|
 14|
   +-------------------------------------------------------------
            Field pointer:    P4   Entry loc.:     P4
 [        #5:EXAMPLE]
 CURRENT DATA    type: numeric
           contents: (M4>25)*(L4>20%J4);100;50
               edit:
```

8 . CONDITIONAL EXPRESSIONS

In this chapter we will introduce you to one of the most sophisticated features of Logicalc, the Conditional expression. The chapter is arranged so that you will be introduced to a simple example before we discuss the format of a conditional expression in more detail. Then, we will demonstrate more complex examples. You may find some of the definitions a little bewildering at first, but once you have gone through the examples they will become clear. If you are unsure of anything, do not hesitate to re-read any section.

8.1 USING CONDITIONAL EXPRESSIONS

One of Logicalc's most useful functions is its ability to incorporate conditional statements into a financial reporting system. A conditional statement will evaluate the data entered and execute one of two statements defined by you, dependent on whether conditions are met (true) or are not (false). We will use our Inventory example again to demonstrate conditional expressions. If you have not already done so, reload the file EXAMPLE before reading the next section. If you are unsure of the loading procedure, refer back to chapter 7.7.

8.2 DEFINING A CONDITIONAL STATEMENT

The format for a conditional statement is:

```
Condition : executable statement if condition is true
          : executable statement if condition is false
```

The format can be split into three separate sections; the condition, the true statement and the false statement.

8.2.1 The Condition

The condition may include numerical values and co-ordinate references, but not a range of co-ordinates. This is because the sign used to signify a range of co-ordinates ('>') has a different meaning within the Condition. To incorporate a range of co-ordinates a system function must be used, e.g. SUM or AVG. The values may be operated on by arithmetic operators, logical operators, relational operators or any combination of the three. The operators included are as follows:

- arithmetic operations
  these are the standard arithmetical expressions we have been using with Logicalc (e.g. addition,subtraction etc). Refer back to chapter 3.1 for a full list.

- logical operators

*	logical AND, is the intersection of two values, i.e. both values must be true for the expression to be true
+	logical OR, the union of two values, i.e. only one of the two will need to be true for the expression to be true

- relational operators

<	(less than)
<=	(less than or equal to)
=	(equal to)
<>	(not equal to)
>=	(greater than or equal to)
>	(greater than)

In our third example, we will use the OR logical operator. The object of the exercise is to compose a list of all bolts in which the cost of the order is greater than 25.00 or the quantity on order is more than 20% of the quantity usually stocked. This would give you a list of your best selling bolts or the bolts which give the most profit. A bolt may meet either of these conditions for inclusion on the list. As this is not part of our inventory, we will write it at a different location yet still keeping it in the same file so it can be printed later.
Move the Field Pointer to location M8 and alter the value to 18.77.
Move the field pointer to location M12 and alter the value to 10.00.
Move the Field Pointer to location O4, and type:

```
(L4>20%J4)+(M4>25);1;2 ↲
```

Then using relative adjustment, copy the formula into locations O5 to O12. The display on your monitor should be as follows:

```
Col>|N          |O          |P          |Q          |
Row+----------------------------------------------------
  1|
  2|
  3|
  4|            >       1.00<
  5|                    1.00
  6|                    1.00
  7|                    1.00
  8|                    2.00
  9|                    1.00
 10|                    1.00
 11|                    1.00
 12|                    2.00
 13|
 14|
   +----------------------------------------------------
            Field pointer:   O4   Entry loc.:   O4
[        #5:EXAMPLE]
CURRENT DATA   type: numeric
           contents: (L4>20%J4)+(M4>25);1;2
               edit:
```

Before moving on to our last example, reset the values at locations M8 and M12 to +F8*L8 (=78.77) and +F12*L12 (=294.45) respectively.

NOTE all items with a value of 1 have met either condition, all items with a value of 2 have met none.

8.4 SUMMARY

When you are working with complex expressions it is important to keep a note of the order in which they will be evaluated. Without the use of parentheses, all the multiplication and division operations will be executed before any of the addition or subtraction operations. Any operation enclosed in parentheses will have a greater priority than those not enclosed. The evaluation inside the parentheses will follow the normal order of multiplication and division, then addition and subtraction. Parentheses must therefore be used carefully in order for your expressions to yield the results you intended.

The system will then prompt:

```
        top left corner : A1 ↲
```

At this point you may specify which location in the array you would like the top left-hand corner to be. As we want the entire report printed, we will accept the default value, A1, and press return. The system will then prompt:

```
        bottom right corner : V15
```

V15 is the default value as it is the bottom right corner of the array (**remember** we had an entry at U2 when discussing user defined functions. This became V2 when we inserted a column).

Care should be taken at this point when printing an entire file. The last entry on our inventory was at location M15, the entries in columns O, P and V being the results of our Conditional Statement and User defined function examples, which we do not want printed. In this example instead of accepting the default, answer the prompt as follows:

```
        bottom right corner : M15 ↲
```

This will give the true bottom right corner of the inventory, as opposed to the bottom right corner of the entries in the array.

The system will then prompt:

```
        Form Length : CONTINUOUS
```

This prompt allows you to decide how many lines per page you would like printed. If you want the lines to be continuous from page to page press return. This will set a standard 66 lines per page. Should you want fewer lines per page, then enter any number between 1 and 66. For our example, answer the prompt as follows:

```
        Form Length :  60 ↲
```

As we did not accept the default, the system will prompt:

```
        Stop on each page ? (Do/Undo) -
```

Press:

As this is the first printout of our inventory, we will give the printout a two line title as follows:

```
Title> INVENTORY FILE EXAMPLE ↲
Title> PRINTOUT #1 ↲
Title> ↲
```

After your third return, the print will start. If you specified for the print to stop after each page, you will need to type return after the first page has been printed. This will continue until the end of the report. When the printing is complete, the following message will appear on the screen:

```
..End of report
  Hit <cr> to continue
```

Press the return key, and the Logicalc array display will return to the monitor screen.

The printout shown in the following pages is from our example and should compare with the one you will print. Notice from the second page onwards that row numbers are included. This is to help make it easier to match up the segmented report afterwards.

```
 1:              COST       PRICE        PROFIT/UNIT     %PROFIT
 2:       ----------- ----------- ------------------ -----------
 3:
 4:             0.017        0.04              0.023        57.5
 5:             0.026        0.05              0.024        48.0
 6:             0.039        0.08              0.041        51.3
 7:             0.059        0.11              0.051        46.4
 8:             0.089        0.15              0.061        40.7
 9:             0.134        0.22              0.086        39.1
10:             0.201        0.32              0.119        37.2
11:             0.302        0.45              0.148        32.9
12:             0.453        0.68              0.227        33.4
13:
14:
15:
 1:                 QUANTITY ORDERED        COST/ORDER
 2:          ----------------------- -----------------
 3:
 4:                             2880             48.96
 5:                             2080             54.08
 6:                             1000             39.00
 7:                             1880            110.92
 8:                              885             78.77
 9:                             2539            340.23
10:                             1937            389.34
11:                             5000           1510.00
12:                              650            294.45
13:
14:          ----------------------- -----------------
15:                            18851           2865.75
```

The next prompt will be:

```
        Printer width : 132
```

As in the previous example, we will specify the printer width as 80 to ensure that the print fits your paper. Answer the prompt as follows:

```
        Printer width : 80 ↲
```

The screen will now be cleared of its present display, and the following message will be displayed:

```
        Report printing...
        Make sure printer & paper are ready
```

If the portion of the report you are printing is larger than the printing width entered above, the print will be divided into segments as we saw in chapter 9.1. As we are printing a subsection of a larger report, the system will output below the above message, the prompt:

```
        Do you want headings ? (Do/Undo) -
```

This prompt is asking whether you would like to include the column and row headings on your printout. This is helpful in this example as we are only printing out a subsection of the file and will not necessarily know what some of the text or numbers pertain to. If you press the Undo key, printout will appear as follows:

```
              15mm Bolt      216/12     GLOBAL Bolts
              18mm Bolt      216/12     GLOBAL Bolts
              21mm Bolt      216/12     GLOBAL Bolts
```

If you press the Do key, the printout will appear as follows:

```
    PART #    PART NAME   SUPPLIER #   SUPPLIER NAME
    213459    15mm Bolt       216/12    GLOBAL Bolts
    213460    18mm Bolt       216/12    GLOBAL Bolts
    213461    21mm Bolt       216/12    GLOBAL Bolts
```

In order to give our print clarity, press the Do key, so that the Headings are included.

NOTE As we saw in the Balsheet example, column A will often be used as part of the heading information. Therefore, when you request headings, the system will print the contents of column A as well as row 1. This should be borne in mind when printing sections of reports.

9.3 OUTPUT TO MONITOR SCREEN

The Print command also gives you the option to print to your monitor screen. To output to monitor you should answer the file prompt as follows:

To which file : **CONSOLE:** ↲

It is important to place the colon (:) after typing CONSOLE otherwise the file will be written to disk. The remaining prompts are the same format as we have seen in the previous sections. You should of course remember that your monitor screen is only 80 characters wide when answering the 'printer width' prompt. If the print is too wide for the screen, then the display will be split into segments in exactly the same manner as we saw earlier. To ensure that the page displayed on your screen remains there long enough to be read, remember to respond with the Do key to the "Stop on each page" prompt.

9.4 THE / PAGE FUNCTION

The '/ Page' function is used in conjunction wih the Print command, serving as a form feed instruction to the system during the print operation. When the system encounters a row with the '/ Page' entry, the printer will skip the rest of the present page and will resume printing at the top of the next page.

The '/ Page' entry must be placed in a new row in column A. To demonstrate, we will print rows 1 to 6 on one page, and rows 7 to 15 on another page. First, move the Field Pointer to location A7, and insert a line. Then, access the Command Line.

- type: **/p** ↲
- proceed through the print command as demonstrated in section 9.2

The effect of these actions is to print rows 1 to 6 on one page and rows 7 to 15 (which have now become rows 8 to 16) on the second page.

9 PRINTING A FILE

In this chapter you will see the fruits of your work when we show you how to print out your inventory file, Example. The chapter is arranged so that you first see how to print the entire file, then you will be shown the flexibility of the Print command.

9.1 PRINTING AN ENTIRE FILE

You are now ready to print the results of your hard work. Although the print command is relatively simple to use, care should be taken before starting. If your report is wider than the maximum width of your printer, you will need to have your report printed in segments and patched together afterwards. As this is the course taken in this section, we can start straightaway. First, access the command line and reply to the prompts as follows:

```
Command〉 P

To which file : PRINTER: ↲
```

The printer is the default output file, so just press return. If you would like to print the report to disk, then enter the file name you would like the report to be stored under. In the next section we will see how you can use a file stored on disk. If you would like a hard copy of the file, then press return and by default, your file will be output to printer, as we have done in our example above.

If you decide you would like the printing to stop after each page is printed, press the Do key. This facility is especially useful when printing single sheets of paper. You will be able to resume printing by pressing return. If you decide you don't need the print to stop after each page, press the Undo key, as we have above. The system will then continue printing until the end of the section.

The system will then prompt:

```
        Printer width : 132
```

This allows you to choose the number of spaces across the page you would like the file to be printed in. This will depend on the size of paper you are using, the default being 132 for eleven inch paper. If you have narrower paper, or would like less than 132 spaces, enter a number between 1 and 132. For our example we will use 80 as our printer width, corresponding to eight and a half inch paper, so answer the prompt as follows:

```
        Printer width : 80↲
```

When this prompt has been answered, your Logicalc window will be cleared from the monitor screen before presenting the next prompts. If, as in our example, the report you are printing is wider than the printer width, the system will automatically divide your report into segments, fitting as many columns across the width of the paper as it can within the range you set in answering the 'printer width' prompt. As you can see from the illustration below, the system displays a message that it will be printing the report in segments.

```
        Report printing...
        Make sure printer & paper are ready
        Printing in segments
```

followed by the prompt:

```
        Title >
```

The 'Title' prompt allows you to enter a title for the inventory. Once you have entered the first line of the title and pressed return, the prompt will repeat itself so that you may include as many lines of title as you want. To exit the prompt, or if you prefer not to have a title, answer the prompt by pressing return.

```
                         INVENTORY FILE EXAMPLE
                              PRINTOUT #1

    PART #       PART NAME      SUPPLIER #      SUPPLIER NAME      TEL NO.
---------- --------------- --------------- ------------------ -----------

    213456        6mm Bolt       104-23-33         ACME Bolts    7171 551
    213457        9mm Bolt       104-23-33         ACME Bolts    7171 551
    213458       12mm Bolt       104-23-33         ACME Bolts    7171 551
    213459       15mm Bolt          216/12       GLOBAL Bolts    4580 737
    213460       18mm Bolt          216/12       GLOBAL Bolts    4580 737
    213461       21mm Bolt          216/12       GLOBAL Bolts    4580 737
    213462       24mm Bolt          216/12       GLOBAL Bolts    4580 737
    213463       28mm Bolt       104-23-33         ACME Bolts    7171 551
    213464       36mm Bolt          216/12       GLOBAL Bolts    4580 737

----------
         9 BOLT SIZES
```

```
 1:            QUANTITY STOCKED        QUANTITY ON HAND
 2:     ----------------------- -----------------------
 3:
 4:                        5000                    2120
 5:                        5000                    2920
 6:                        5000                    4000
 7:                        5000                    3120
 8:                        5000                    4115
 9:                        5000                    2461
10:                        5000                    3063
11:                        5000                       0
12:                        5000                    4350
13:
14:     ----------------------- -----------------------
15:                       45000                   26149
```

9.2 PRINTING PART OF A FILE

In this section we will demonstrate how to print part of your file: the part names, supplier's name and number for three types of bolt, the 15, 18 and 21mm bolts. The procedure is similar to printing a whole file, so first, access the command line and answer the prompts as follows:

```
Command > P

To which file : PRINTER:↲
```

The next prompt will be:

```
top left corner : A1
```

As this is not the location of the first required part name, we will not accept the default but instead answer B7, for the 15mm Bolt. This will ensure that column A and the first six rows of column B are excluded, so answer the prompt as follows:

```
top left corner : B7 ↲
```

The next system prompt will be:

```
bottom right corner : V15
```

Again, we do not want the default as this is the co-ordinate of the last entry into our array. As we are only interested in the information contained in three columns, (part name, supplier number and supplier name) and three bolt sizes (15mm, 18mm and 21mm), our reply to the prompt would be as follows:

```
bottom right corner : D9 ↲
```

The next system prompt will be:

```
Form Length : CONTINUOUS↲
```

As we are only printing a small section of the array, we can accept the default by pressing return.

The system will then output the following prompt:

```
        Title〉
```

This prompt allows you to enter a title for the Inventory subsection. When you have entered the title and pressed return the prompt will repeat itself so that you may enter as many lines of title as you would like. To exit the prompt, or if you prefer not to have a title, answer the prompt by pressing return. For our Inventory subsection, we will give the printout a two line title as follows:

```
        Title〉 INVENTORY FILE EXAMPLE : SUBSECTION ↵
        Title〉 PRINTOUT #2 ↵
        Title〉 ↵
```

When the printing is finished, press the return key, and the Logicalc array display will return to the screen. The resulting report will look as follows:

```
             INVENTORY FILE EXAMPLE : SUBSECTION
                         PRINTOUT #2

PART #        PART NAME       SUPPLIER #        SUPPLIER NAME
213459        15mm Bolt           216/12         GLOBAL Bolts
213460        18mm Bolt           216/12         GLOBAL Bolts
213461        21mm Bolt           216/12         GLOBAL Bolts
```

9.5 SUMMARY OF PRINT COMMAND

This section serves as a reminder of some points to note when using the Print Command.

- If you **do not** type a colon at the end of the device name, the file will be written to disk, and not to the Printer or Console. This file will then be of the Standard Text type as opposed to the Field Oriented text type (see section 10.1.1).

- Make sure you create the correct type of text file to suit the relevant package. The next chapter deals with the creation of text files.

As we saw earlier, the default co-ordinate for this prompt is the last co-ordinate in the array with an entry, which we are not interested in. The reply to the prompt will be as follows,

```
        bottom right corner :  M15 ↲
```

this being the last entry in the example file. The file will then be written away as a field oriented text file to disk. This file can then be read by a Logicalc compatible package, such as ELFIE.

10.1.2 Creating a Standard Text File

The second method of creating a text file is identical to the above, except the command in this case is 'P' (the Print command). Instead of accepting the printer as the default device, or specifying the console, you must type in the name of the file to be created.

The system prompts are then as we saw previously with the 'P' command. When this text file has been written to disk, it can be used with Logicalc compatible packages, such as Correspondence Management.

Because the text file being read into the array may not meet the same requirements as Logicalc, the following points must be noted.

- When the file is being loaded, you may get some error conditions, indicated by the standard error sign, '?n?', together with an error message. This may occur if a field is alphanumeric, starting with a numeric character e.g. 35mm FILM would give an error, because Logicalc would treat the field as being purely numeric. To overcome this error, simply re-enter the item, but before pressing return use the Data Type key, as described in chapter 4.1.

- When a text file is loaded into the Logicalc array, the column widths are not altered, i.e. they remain at the default width of ten characters. If the entry is greater than ten characters, then the first ten characters will be put into the array whilst the complete entry will be held in memory and displayed on the contents line. Once the file has been fully loaded, you can alter the column widths to accomodate the entry. The full entry will then appear automatically in the array.

- As with column width, you may have to reset the decimal precision for each column when the file is loaded. This is because the default precision of two decimal places will be used.

- Any field which contains the Repeat function (/=) will give rise to an error, and when the return key has been pressed in response to the error message, the standard error sign, ?n?, will appear on the screen. You will then have to edit the field, after the file has been loaded.

10 LOGICALC TEXT FILES

In this chapter we will demonstrate Logicalc's compatibility with other Philips packages using text files.

10.1 OUTPUT TO DISK FILE

Logicalc will allow you to output two types of text files, Field oriented text files and Standard text files. These files can then be read by P2000 packages which are compatible with Logicalc. Note that you cannot create a text file if you have less than 65 available (i.e empty) entries. Before attempting to create one, it would therefore be a good idea to use the Storage command (see section 5.5), to find out how many entries you have available.

10.1.1 Creating a Field Oriented Text File

Logicalc contains a facility which enables you to create a field oriented text file from a Logicalc file. To create such a file, you must have your Logicalc file loaded into the array. In this example we will assume you have the Inventory example loaded. Access the command line and reply to the prompts as follows:

Command> ⇧⇧ ⇨

To which file : **#5:DISKFIL** ↵

You must note that the system will only accept a total of 10 characters for a file name. In the example above our file name is constricted to 7 characters as we require 3 for the disk identity. The system then prompts:

Top left corner : A1 ↵

This, as we saw previously with the print command, is the default co-ordinate.

bottom right corner : V15

10.2 LOADING A FIELD ORIENTED TEXT FILE

The command described below allows you to load a field oriented text file from a P2000 compatible package such as ELFIE, into a Logicalc array.

- Ensure that the Logicalc array displayed on your monitor screen is clear.

- Place the disk with the text file on in drive 2.

- Access the Command line

- Press: ⇧⇧ ⇦

- The following prompt will be displayed:

  ```
  File name :
  ```

- You should now enter the text file name and press return. Remember that the text file disk is in drive two, so prefix the file name with the drive identifier, i.e. #5:

- The system will then prompt:

  ```
  load position : A1 ↵
  ```

- The default loading co-ordinate is A1, and, as we have previously mentioned, it is advisable to accept it.

The text file will then be read into the Logicalc array.

We will use the Regression function in three differing locations to demonstrate the various aspects of the function. Move the Field Pointer to location H6 and type:

```
+regr(B2>G2:B6) ↲
```

The plus sign informs the system that the entry is numeric, and 'regr' signifies the Regression function. B2 to G2 is the range of entries which contains the values for the independent variable. B6 is the start of the range for the dependent variable. The system will automatically know to include entries to G6 since that will result in the same number of entries as the range for the independent variable (remember, we saw this principle in chapter 4.5 when we used the Copy command). The first equation is regressing the amount of sales on time periods allowing us to see how sales fare each month. This will give us a basis for a forecast on future sales using the trend established with the equation.

The value at H6 is 816.67 and represents the mean value for the range of dependent values given in the the regression function.

If you do not want the display from the regression function to appear on the screen, as is the case here, you can enter it as follows:

```
+regr(B2>G2:B6);'';'' ↲
```

Note that single inverted commas are used.

The Regression function, which computes a linear equation from the two ranges you entered, is the first step in forecasting with Logicalc.

In the first example, we saw the positive relationship between sales and time i.e. as time increases, sales increase. In this next example we will show how, as time increases, and sales decrease, regression can show a negative as well as positive trend. Move the Field Pointer to location H7 and type:

+regr(B2>G2:B7) ↲

Now move the Field Pointer to location I7 and type:

+proj(10) ↲

The result of 0.18 tells you that by October, sales of this product will have declined to almost next to nothing. If our projected figure was (11) or (12) we would be building up an inventory of unwanted goods.

For the last example, move the Field Pointer to location H9 and type:

+regr(B9>G9:B6) ↲

In this example, the amount of money spent on advertising is the independent variable and sales of the first product the dependent variable. The mean for the dependent range is the same as the first example because it has the same range of entries, so your answer will be £816.67.

11.1.2 The Dependent Function

In this example we will be asking for the best estimate for the amount of money you need to spend on advertising to result in a sales figure of £2000. First move the Field Pointer to location I9 and type:

+depd(2000) ↲

The result at location I9 indicates that we will need to spend £1750.89 on advertising to reach a sales level of £2000 for product 1. This differs from the project function which would have given you a figure for the amount of sales with advertising expenses of £2000. If you type:

+proj(1750.89) ↲

you will be asking for the best estimate of sales generated by an advertising expenditure of £1750.89, your result will then be £2000.02.

The mean for the dependent range for the first regression function we entered was replaced by a blank display on the monitor, although the computer could still use the regression function stored there. We now have the slopes for the three regression functions and you can get a rough estimate of the correlation between the values regressed between each other, e.g. the second regression function results in a slope of -11.23, which means that sales are decreasing by approximately 11 units per month.

If seasonal fluctuations affect your sales, instead of performing a regression on all the months of one year, you could use the data for the same month over several years.

11.2 SUMMARY

There are several points which must be remembered to ensure the accuracy of Linear Regression and they are listed in this section.

Unless you enter the regression function immediately prior to the project, dependent or slope function, it is advisable to use the Recalculate command to ensure the correct regression function is used.

An important rule to remember is the placing of your project, dependent or slope function in relation to the regression function. The project, dependent or slope function must follow the regression function it uses and must be before the next regression function, e.g.

- location A1, regression function
- location B1, project function
- location C1, regression function
- location C2, dependent function
- location D2, regression function
- location D3, slope function

It is advisable to place your project, dependent or slope function to the right of the regression function whenever possible. This is to accommodate the Recalculate command which proceeds in a left to right, top to bottom order (chapter 7.2 discusses this in greater detail).

The Linear regression can be a useful aid in assisting you to recognize trends based on past data and so help you make predictions for the future.

11 LINEAR REGRESSION

In this chapter you will see how you can use Logicalc as a tool for financial forecasting. The chapter uses a sample file, called Sales, as the basis of the demonstration and examples. We have not included this file on the Logicalc Disk; you must create it yourself. The data for you to create the file is found in Appendix C, together with an illustration of what should be displayed on your monitor screen when it is completed.

We have now finished with the Inventory example for the time being, so before turning to appendix C, save the file and then clear the screen.

11.1 USING LINEAR REGRESSION

In this example we will be using the Regression command to compute a linear equation which may then be used to compute the best estimates of the values we wish to forecast. The regression function allows you to enter one range of entry locations, followed by a comma, followed by the first co-ordinate of the second range of entries. The number of entries within each range must be the same, so you need only enter the first co-ordinate of the second range.

The values contained within the first range of entries are the values for your independent variable. Independent variable is a statistical expression which means that the value of the variable is determined independently from the values of other variables. The second range of entries consists of the values for your dependent variable. The dependent variable is assumed to be influenced by factors other than itself.

Using our Sales example file we will assume the amount of sales to be dependent on the amount of money spent on advertising. The advertising money will be our independent variable and the amount of sales our dependent variable. This is an unlikely situation to occur in real life, however it will give a good indication of how sales correlate with advertising expenditure.

There are three other functions you can use in conjunction with regression, and they are as follows:

- proj   allows you to enter a value for an independent variable. Then, the best estimate for the dependent variable will be calculated and entered into the current Field Pointer location.

- depd   allows you to enter a value for the dependent variable and the best estimate for the independent variable is calculated and entered into the current Field Pointer location.

- slope   allows you to have the system enter the slope or gradient of the linear equation computed from the regression function, into the current Field Pointer location. The slope will give you an approximate estimate of the correlation between independent and dependent variable. If the slope is 30, a change in the independent variable of one will give a change 30 times that size in the dependent variable. If the slope is -20, it means that for an increase of one in the independent variable you will get a decrease of 20 in the dependent variable.

We will now go on to demonstrate these functions using our Sales file.

11.1.1 The Project Function

First, let us estimate how many of product 1 we will sell in October. To calculate our best estimate, move the Field Pointer to location I6 and type:

+proj(10) ↲

We know from our regression function that the month number is the independent variable and the sales amount the dependent variable. With the project function we enter a known amount for the independent variable and the system returns the best estimate for the dependent variable. At location I6 you should have the figure '2343.24' as the expected sales figure, based on the first six months trade.

11.1.3 The Slope Function

To demonstrate the slope function, move the Field Pointer to location J6 and type:

+slope() ↲

Repeat the action at locations J7 and J9. In all three locations you should have the result '3.46'. Logicalc has been designed so that all the functions that work with the Regression function, work with the last regression entered. Since the last regression function entered was at location H9, the system will work out the slope of the regression equation entered at that location. To enter a function to be used with a regression function other than the last entered, you must use the Recalculate command. Once Recalculate has been executed, and column I given its appropriate heading the array from location G1 will appear as follows:

```
Col>|G                  |H            |I              |J            |
Row+-----------------------------------------------------------------
  1 | >         june<                     october
  2 |              6                         10
  3 |    ==========                  ==========
  4 |                                 projected
  5 |
  6 |           1440                    2343.24       234.86
  7 |             47        73.17          0.18       -11.23
  8 |
  9 |           1530       816.67       2000.02         3.46
 10 |
 11 |
 12 |
 13 |
 14 |
    +----------------------------------------------------------------
               Field pointer:     G1   Entry loc.:     G1

 CURRENT DATA    type: text:right justified
             contents: 'june'
                 edit:
```

```
Disk volume LCBPHI1:                    Hit <cr> to continue
-------------------------------------------------------------------
| SYSTEM.PASCAL    |                |                |                |
| SYSTEM.STARTUP   |                |                |                |
| SYSTEM.PBIOS     |                |                |                |
| SLAVE.PBIOS      |                |                |                |
| 8INCH.PBIOS      |                |                |                |
| SYSTEM.SBIOS     |                |                |                |
| SYSTEM.MISCINFO  |                |                |                |
| SYSTEM.INTERP    |                |                |                |
| SYSTEM.LIBRARY   |                |                |                |
| SYSTEM.PRINTER   |                |                |                |
| LCDUMP.CODE      |                |                |                |
| LOGICALC.CODE    |                |                |                |
| LCMASK           |                |                |                |
|*BALSHEET.LCAR    |                |                |                |
| DISKORG.CODE     |                |                |                |
|                  |                |                |                |
|                  |                |                |                |
|                  |                |                |                |
|                  |                |                |                |
|                  |                |                |                |
-------------------------------------------------------------------
15 listed out of 15 in directory 376 used 232 free 232 in largest
```

The line at the top of the display provides a reminder of the disk file name. The line at the bottom of the display informs us that we have 15 files in our directory, we have used 376 blocks and 232 blocks are in the the largest area of memory remaining. You will notice that the Logicalc files are marked with an asterisk. Note also that the DISKORG.CODE file will only appear if you have transferred the file from the TKS Disk (see section 1.6).

12.2 THE HELP LIST

One of the most convenient and useful commands is the Help command. This allows you to have a directory of all the commands and functions displayed on the screen without leaving Logicalc. To execute the Help command, access the command line and answer the prompt as follows:

Command⟩ [?]

A display similar to the following will then be shown:

```
   ---- Help page 1 ----              ***** Hit <CR> to continue *****
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _Main entry mode_ _ _ _ _ _ _ _ _ _ _ _ _ _
  Move the field pointer using the arrows, <CR> (if nothing has been typed),
  and <s/shft CR>. For details see the manual.
  Enter data by typing the information in the location. If the 'type'
  indicator is wrong, the 'toggle' key will switch it, then hit <CR>.

 SPECIAL FUNCTIONS:
  <Cursor jump> = GOTO.  '?' = evaluate input as equation.
  <Undo> is delete line/getout.

 TEXT ENTRY:  in the 1st 2 chars of line use:
  '/r' to right justify entry in line, '/l' to left justify and '/c' to centre.
  '/=' duplicates the string across the whole column. '/p' ='/PAGE'(form feed).

 DATA ENTRY:  once at least 1 char has been typed, the field pointer may be
  moved to other positions; the <F1> key enters the field pointer position
  into the current entry. A data item may be entered as an equation; if
  there is a position entry (e.g. AZ12), the equation may be re-evaluated
  to reflect changes in the addressed data.

 FUNCTION FORMAT: name(position,position position,expression,etc.)
  Functions available for math:  sum,avg,cnt,min,max,regr,proj,slope,depd
```

12.3 PROGRAM LCDUMP

With Logicalc we have included a separate and very helpful support program called 'LCDUMP', which will allow you to print a list of the specifications and formulae of your Logicalc file. LCDUMP will print each entry within your Logicalc array giving details of column width, type of entry, justification (if the entry was text), and the contents of the entry. This will allow you to keep a hard copy of all your Logicalc files safely stored. To use LCDUMP you must leave Logicalc using the Quit command (see section 7.9). From the System Menu, select option 1 (EXECUTE PROGRAM). This will lead to the following prompt:

```
PROGRAM NAME: ______________________          EXECUTE PROGRAM
```

Type: **LCDUMP** ↲

The system will then ask you to confirm the program name.

Press [!]

At the bottom of the screen, the system will then display the following prompt:

```
Output file:
```

The printer is the default, so if you type a return, the listing will be output to the printer. If you want to write the listing to a disk file, you must specify a file name, for example

#5:TEST

For our example, we will accept the default simply by pressing return. It is advisable to check that the Printer is connected and ready for use. The next prompt asks for the name of the Logicalc file to be printed:

```
Logicalc file name:
```

Reply: **BALSHEET** ↲

For our example we have used the file originally seen earlier in the book, BALSHEET. If the file is not found the system will reprompt, giving you another chance to enter a file name.

- The disk will now be reorganized, and a message will appear on the screen to this effect.

- Press the Undo key, and the DISKORG menu will reappear on the screen. Now select option 0 (EXIT TO SYSTEM), and the system menu will be redisplayed. Select option 3, and you will be able to start working with Logicalc again.

12.4.1 Removing Files

There will be a time when you do not want to keep some of the files on your Logicalc disk. The DISKORG program contains an option which enables you to delete files from the disk. Before you start to delete files it is always best to make a work copy of the disk. If you then accidentally remove files you wanted to keep, you will have them secure on another disk. To delete files, follow the instructions below.

- From the DISKORG menu, select option 3 (DELETE). The following prompt will then be displayed beneath the DISKORG DELETE header.

```
                    1 FIFE
                    2 DISK

                    0  EXIT TO MAIN MENU

SELECT: 1                                        DELETE
```

- Select the FILE option by typing in 1 or press the tab key (option 1 is assumed by default). This will lead to a further prompt as follows:

```
DRIVE: _    FILE: _______________                DELETE FILE
```

- Specify the drive containing the disk with the file to be removed. (This will normally be drive 2.) Then specify the name of the file to be deleted, and confirm.

- When the file has been deleted, you will see a message to this effect. Press the Undo key, and the DELETE menu will return to the screen. Select option 0, and the DISKORG menu will return. The last step is to select option 0 again. This will return you to the system menu, and you can select Logicalc.

12 LOGICALC FILE MANAGEMENT

You have now seen most of the Logicalc commands and functions. In this chapter we will show you how to maintain your disk files for maximum efficiency. You will also be introduced to some new commands which will assist you with the knowledge and experience gained in the previous chapters.

12.1 DIRECTORY COMMAND

The Directory command of Logicalc allows you to display a listing of all files or all Logical files on a disk. This provides you with a convenient way of checking what files are on your disk. In addition, the Directory command allows you to see how many blocks of memory space on the disk are occupied with files, how many blocks you have left and the number of blocks in the largest area. To use the Directory command, first access the command line and then answer the prompts as follows:

Command〉 SEARCH

UCSD volume name : **#4** ↲

The UCSD volume name prompt is asking you for the volume number of the the disk you want a listing of. Although we have entered '#4', that is the default value and so we could have just pressed return. If the disk was in drive 2, we would have entered '#5'. The next prompt is:

A)ll or L)ogicalc files only ?

This prompt gives you the choice to list all the files on disk (A) or Logicalc files only (L). If you enter 'A', your monitor screen will resemble the illustration overleaf.

If you enter 'L', you will only get Logicalc files displayed, and the format will be as follows:

```
Disk volume LCBPHI1:                 Hit <cr> to continue
--------------------------------------------------------------------------
| BALSHEET.LCAR  |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
|                |                 |                 |                 |
--------------------------------------------------------------------------
1 listed out of 15 in directory 376 used 232 free 232 in largest
```

This is very similar to the previous display, except the information at the bottom of the list shows the number of Logicalc files out of the total number of files on the volume.

The second page, obtained by pressing return, is as follows:

```
      —— Help page 2 ——               ***** Hit <CR> to continue *****
                            Extended commands
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

 <F2>               enter automatic entry mode. <Undo> will abort.
 C(opy              copy a (range) entry to another (range) entry
 <DEL>              delete a row, column, entry or the whole array
 E(dge              set the window top left corner to the field pointer position
 F(ormat            change column size or precision under the field pointer
 <INS>              insert a row or column into the array
 <SEARCH>           display either full disk directory or Logicalc catalogue only
 <Transfer in>      load a file into the array
 M(erge             overlay file onto array
 O(rder             change the evaluation (column/row) order of the array
 P(rint             print a report
 Q(uit              exit the report generator
 R(ecalculate       recompute entry at cursor or whole array in current order
 <Transfer out>     save the array to a file
 W(hat              print 1st column & row (headings)
 T(ext              edit entry or input data
 <s/shft T/out>     create a field oriented text file
 <s/shft T/in>      read a field oriented text file
 ?                  space available
 =                  set/clear locked screen
```

After you press return a second time the help directory will be replaced by the main Logicalc window as you left it with control at the edit line.

Note - the help lists refer to return as CR.

If our file Balsheet had a password, the next prompt would be:

```
      Password:
```

If an incorrect password is entered the system will display the message, 'not ok' and abort the LCDUMP program returning to our original program prompt. As our example has no password, the next prompt will ask:

```
      Comments ?
```

This gives you the opportunity to include a title with the printout. If you do not need a title, as in this example, press return.

When LCDUMP has finished printing the file it will prompt:

```
      Print another file ?
```

If you answer "Y", the LCDUMP prompts are repeated. If you enter "N", the System Menu will return to the screen. To get back into Logicalc, simply select option 3.

12.4 TIDYING YOUR DISK

In our final section of the Logicalc manual we will remind you of the DISKORG 'REORGANIZE' command. This command increases the available space on your disk by removing any blank spaces between the files. It is advisable to reorganize your disk if you find that the ratio between space available and space per largest area differs greatly. To do this, you must exit from Logicalc using the Quit command, and then select option 1 from the system menu. Type 'DISKORG' as the name of the progam to be executed, and confirm. This will lead to the DISKORG menu. Carry out the following steps:

- Select option 7 (REORGANIZE) from the menu. This will lead to the prompt:

```
DRIVE: _   VOLUME: ________                        REORGANIZE
```

- Enter 2 for the drive number. The volume name of the disk in drive 2 will appear automatically, so confirm by pressing the Do key.

A LCDUMP LISTINGS

In this Appendix you will find LCDUMP listings of the completed inventory file, #5:EXAMPLE, and the BALSHEET file. The listings have been reduced slightly to enable us to fit them easily onto the pages.

As a reminder, we have described below, the sequence which must be followed in order to produce an LCDUMP listing. For a more detailed explanation of the LCDUMP program, refer to section 12.3

- Access the command line:

  Press [!]

- Use the Quit command:

  Command〉 Q

  Verify (Do/Undo) - [!]

- From the System Menu, select option 1 (EXECUTE PROGRAM). The following prompt will be output:

```
PROGRAM NAME: ____________________          EXECUTE PROGRAM
```

  Type: **LCDUMP** ↲ then press [!]

- LCDUMP will now request the name of the file to which the listing should be written (press return for the printer), the name of the file to be listed plus password if present, and the title of the listing (press return for no title).

NOTE LCDUMP does not provide information about precision settings. For examples in this appendix, take the precision settings from the array printouts.

```
Position J6 Width 24 Type:  Numeric :5000 = 5000.000000000000
Position K6 Width 24 Type:F Numeric :4000 = 4000.000000000000
Position L6 Width 24 Type:  Numeric :+J6-K6 = 1000.000000000000
Position M6 Width 18 Type:  Numeric :+F6*L6 = 39.000000000000
Position A7 Width 10 Type:  Numeric :1+A6 = 213459.000000000000
Position B7 Width 15 Type:  Text (Right justified):15mm Bolt
Position C7 Width 15 Type:  Text (Right justified):216/12
Position D7 Width 18 Type:  Text (Right justified):GLOBAL Bolts
Position E7 Width 11 Type:  Text (Right justified):4580 737
Position F7 Width 11 Type:  Numeric :150%F6 = 0.059000000000
Position G7 Width 11 Type:F Numeric :.11 = 0.110000000000
Position H7 Width 18 Type:  Numeric :+G7-F7 = 0.051000000000
Position I7 Width 12 Type:  Numeric :+H7/G7*100 = 46.400000000000
Position J7 Width 24 Type:  Numeric :5000 = 5000.000000000000
Position K7 Width 24 Type:F Numeric :3120 = 3120.000000000000
Position L7 Width 24 Type:  Numeric :+J7-K7 = 1880.000000000000
Position M7 Width 18 Type:  Numeric :+F7*L7 = 110.920000000000
Position A8 Width 10 Type:  Numeric :1+A7 = 213460.000000000000
Position B8 Width 15 Type:  Text (Right justified):18mm Bolt
Position C8 Width 15 Type:  Text (Right justified):216/12
Position D8 Width 18 Type:  Text (Right justified):GLOBAL Bolts
Position E8 Width 11 Type:  Text (Right justified):4580 737
Position F8 Width 11 Type:  Numeric :150%F7 = 0.089000000000
Position G8 Width 11 Type:F Numeric :.15 = 0.150000000000
Position H8 Width 18 Type:  Numeric :+G8-F8 = 0.061000000000
Position I8 Width 12 Type:  Numeric :+H8/G8*100 = 40.700000000000
Position J8 Width 24 Type:  Numeric :5000 = 5000.000000000000
Position K8 Width 24 Type:F Numeric :4115 = 4115.000000000000
Position L8 Width 24 Type:  Numeric :+J8-K8 = 885.000000000000
Position M8 Width 18 Type:  Numeric :+F8*L8 = 78.770000000000
Position A9 Width 10 Type:  Numeric :1+A8 = 213461.000000000000
Position B9 Width 15 Type:  Text (Right justified):21mm Bolt
Position C9 Width 15 Type:  Text (Right justified):216/12
Position D9 Width 18 Type:  Text (Right justified):GLOBAL Bolts
Position E9 Width 11 Type:  Text (Right justified):4580 737
Position F9 Width 11 Type:  Numeric :150%F8 = 0.134000000000
Position G9 Width 11 Type:F Numeric :.22 = 0.220000000000
Position H9 Width 18 Type:  Numeric :+G9-F9 = 0.086000000000
Position I9 Width 12 Type:  Numeric :+H9/G9*100 = 39.100000000000
Position J9 Width 24 Type:  Numeric :5000 = 5000.000000000000
Position K9 Width 24 Type:F Numeric :2461 = 2461.000000000000
Position L9 Width 24 Type:  Numeric :+J9-K9 = 2539.000000000000
Position M9 Width 18 Type:  Numeric :+F9*L9 = 340.230000000000
Position A10 Width 10 Type:  Numeric :1+A9 = 213462.000000000000
Position B10 Width 15 Type:  Text (Right justified):24mm Bolt
Position C10 Width 15 Type:  Text (Right justified):216/12
Position D10 Width 18 Type:  Text (Right justified):GLOBAL Bolts
Position E10 Width 11 Type:  Text (Right justified):4580 737
Position F10 Width 11 Type:  Numeric :150%F9 = 0.201000000000
Position G10 Width 11 Type:F Numeric :.32 = 0.320000000000
Position H10 Width 18 Type:  Numeric :+G10-F10 = 0.119000000000
Position I10 Width 12 Type:  Numeric :+H10/G10*100 = 37.200000000000
Position J10 Width 24 Type:  Numeric :5000 = 5000.000000000000
Position K10 Width 24 Type:F Numeric :3063 = 3063.000000000000
Position L10 Width 24 Type:  Numeric :+J10-K10 = 1937.000000000000
Position M10 Width 18 Type:  Numeric :+F10*L10 = 389.340000000000
Position A11 Width 10 Type:  Numeric :213463 = 213463.000000000000
Position B11 Width 15 Type:  Text (Right justified):28mm Bolt
Position C11 Width 15 Type:  Text (Right justified):104-23-33
Position D11 Width 18 Type:  Text (Right justified):ACME Bolts
Position E11 Width 11 Type:  Text (Right justified):7171 551
Position F11 Width 11 Type:  Numeric :150%F10 = 0.302000000000
Position G11 Width 11 Type:  Numeric :150%F11 = 0.450000000000
```

```
RG File #5:example   LCDUMP OF #5:EXAMPLE

Position A1 Width 10 Type:  Text (Right justified):PART #
Position B1 Width 15 Type:  Text (Right justified):PART NAME
Position C1 Width 15 Type:  Text (Right justified):SUPPLIER #
Position D1 Width 18 Type:  Text (Right justified):SUPPLIER NAME
Position E1 Width 11 Type:  Text (Right justified):TEL NO.
Position F1 Width 11 Type:  Text (Right justified):COST
Position G1 Width 11 Type:  Text (Right justified):PRICE
Position H1 Width 18 Type:  Text (Right justified):PROFIT/UNIT
Position I1 Width 12 Type:  Text (Right justified):%PROFIT
Position J1 Width 24 Type:  Text (Right justified):QUANTITY STOCKED
Position K1 Width 24 Type:  Text (Right justified):QUANTITY ON HAND
Position L1 Width 24 Type:  Text (Right justified):QUANTITY ORDERED
Position M1 Width 18 Type:  Text (Right justified):COST/ORDER
Position A2 Width 10 Type:  Text (Repeating)      :=
Position B2 Width 15 Type:  Text (Repeating)      :=
Position C2 Width 15 Type:  Text (Repeating)      :=
Position D2 Width 18 Type:  Text (Repeating)      :=
Position E2 Width 11 Type:  Text (Repeating)      :=
Position F2 Width 11 Type:  Text (Repeating)      :=
Position G2 Width 11 Type:  Text (Repeating)      :=
Position H2 Width 18 Type:  Text (Repeating)      :=
Position I2 Width 12 Type:  Text (Repeating)      :=
Position J2 Width 24 Type:  Text (Repeating)      :=
Position K2 Width 24 Type:  Text (Repeating)      :=
Position L2 Width 24 Type:  Text (Repeating)      :=
Position M2 Width 18 Type:  Text (Repeating)      :=
Position A4 Width 10 Type:  Numeric :213456 = 213456.000000000000
Position B4 Width 15 Type:  Text (Right justified):6mm Bolt
Position C4 Width 15 Type:  Text (Right justified):104-23-33
Position D4 Width 18 Type:  Text (Right justified):ACME Bolts
Position E4 Width 11 Type:  Text (Right justified):7171 551
Position F4 Width 11 Type:  Numeric :.0165 = 0.017000000000
Position G4 Width 11 Type:F Numeric :.04 = 0.040000000000
Position H4 Width 18 Type:  Numeric :+G4-F4 = 0.023000000000
Position I4 Width 12 Type:  Numeric :+H4/G4*100 = 57.500000000000
Position J4 Width 24 Type:  Numeric :5000 = 5000.000000000000
Position K4 Width 24 Type:F Numeric :2120 = 2120.000000000000
Position L4 Width 24 Type:  Numeric :+K4<J4;J4-K4 = 2880.000000000000
Position M4 Width 18 Type:  Numeric :+F4*L4 = 48.960000000000
Position A5 Width 10 Type:  Numeric :1+A4 = 213457.000000000000
Position B5 Width 15 Type:  Text (Right justified):9mm Bolt
Position C5 Width 15 Type:  Text (Right justified):104-23-33
Position D5 Width 18 Type:  Text (Right justified):ACME Bolts
Position E5 Width 11 Type:  Text (Right justified):7171 551
Position F5 Width 11 Type:  Numeric :150%F4 = 0.026000000000
Position G5 Width 11 Type:F Numeric :.05 = 0.050000000000
Position H5 Width 18 Type:  Numeric :+G5-F5 = 0.024000000000
Position I5 Width 12 Type:  Numeric :+H5/G5*100 = 48.000000000000
Position J5 Width 24 Type:  Numeric :5000 = 5000.000000000000
Position K5 Width 24 Type:F Numeric :2920 = 2920.000000000000
Position L5 Width 24 Type:  Numeric :(K5<J5)*(J5-K5) = 2080.000000000000
Position M5 Width 18 Type:  Numeric :+F5*L5 = 54.080000000000
Position A6 Width 10 Type:  Numeric :1+A5 = 213458.000000000000
Position B6 Width 15 Type:  Text (Right justified):12mm Bolt
Position C6 Width 15 Type:  Text (Right justified):104-23-33
Position D6 Width 18 Type:  Text (Right justified):ACME Bolts
Position E6 Width 11 Type:  Text (Right justified):7171 551
Position F6 Width 11 Type:  Numeric :150%F5 = 0.039000000000
Position G6 Width 11 Type:F Numeric :.08 = 0.080000000000
Position H6 Width 18 Type:  Numeric :+G6-F6 = 0.041000000000
Position I6 Width 12 Type:  Numeric :+H6/G6*100 = 51.300000000000
```

The second listing is for the file BALSHEET. The printout for this file can be found after the LCDUMP listing.

```
RG File balsheet   LCDUMP OF BALSHEET

Position B1 Width 20 Type:  Text (Centered)       :ASSETS
Position A4 Width 28 Type:  Text (Centered)       :CURRENT ASSETS
Position A5 Width 28 Type:  Text (Left justified) :Cash
Position B5 Width 20 Type:  Numeric :8000000 = 8000000.000000000000
Position A6 Width 28 Type:  Text (Left justified) :Accounts Receivable
Position B6 Width 20 Type:  Numeric :100000 = 100000.000000000000
Position A7 Width 28 Type:  Text (Left justified) :Inventory
Position B7 Width 20 Type:  Numeric :5235000 = 5235000.000000000000
Position B8 Width 20 Type:  Text (Repeating)      :-
Position A9 Width 28 Type:  Text (Centered)       :TOTAL CURRENT ASSETS
Position C9 Width 13 Type:  Numeric :sum(B5>B7) = 13335000.000000000000
Position C11 Width 13 Type:  Text (Repeating)      :-
Position A12 Width 28 Type:  Text (Centered)       :TOTAL ASSETS
Position C12 Width 13 Type:  Numeric :+sum(B5>B7) = 13335000.000000000000
Position C13 Width 13 Type:  Text (Repeating)      :-
Position A14 Width 28 Type:  Text (Centered)       :TOTAL ASSETS
Position D14 Width 13 Type:  Numeric :+sum(B5>B7) = 13335000.000000000000
Position D15 Width 13 Type:  Text (Repeating)      :=
Position B20 Width 20 Type:  Text (Left justified) :LIABILITIES & EQUITY
Position A23 Width 28 Type:  Text (Centered)       :CURRENT LIABILITIES
Position A24 Width 28 Type:  Text (Left justified) :Accounts Payable
Position B24 Width 20 Type:  Numeric :19000 = 19000.000000000000
Position A25 Width 28 Type:  Text (Left justified) :Notes Payable
Position B25 Width 20 Type:  Numeric :4810000 = 4810000.000000000000
Position B26 Width 20 Type:  Text (Repeating)      :-
Position A27 Width 28 Type:  Text (Centered)       :TOTAL CURRENT LIABILITIES
Position A30 Width 28 Type:  Text (Centered)       :TOTAL LIABILITIES
Position C30 Width 13 Type:  Numeric :+sum(B24>B25) = 4829000.000000000000
Position A33 Width 28 Type:  Text (Centered)       :OWNERSHIP & EQUITY
Position A34 Width 28 Type:  Text (Left justified) :Capitol
Position B34 Width 20 Type:  Numeric :8500000 = 8500000.000000000000
Position A35 Width 28 Type:  Text (Left justified) :Net Income
Position B35 Width 20 Type:  Numeric :6000 = 6000.000000000000
Position B36 Width 20 Type:  Text (Repeating)      :-
Position A37 Width 28 Type:  Text (Centered)       :TOTAL WORTH
Position C37 Width 13 Type:  Numeric :+sum(B34>B35) = 8506000.000000000000
Position C39 Width 13 Type:  Text (Repeating)      :-
Position A40 Width 28 Type:  Text (Centered)       :TOTAL OWNERSHIP & EQUITY
Position C40 Width 13 Type:  Numeric :+SUM(B34>B35) = 8506000.000000000000
Position A42 Width 28 Type:  Text (Centered)       :TOTAL LIABILITIES & EQUITY
Position D42 Width 13 Type:  Numeric :+C30+C40 = 13335000.000000000000
Position D43 Width 13 Type:  Text (Repeating)      :=
Position H11 Width 18 Type:  Numeric :+G11-F11 = 0.148000000000
Position I11 Width 12 Type:  Numeric :+H11/G11*100 = 32.900000000000
Position J11 Width 24 Type:  Numeric :5000 = 5000.000000000000
Position K11 Width 24 Type:  Numeric :0 = 0.000000000000
Position L11 Width 24 Type:  Numeric :+J11-K11 = 5000.000000000000
Position M11 Width 18 Type:  Numeric :+F11*L11 = 1510.000000000000
Position A12 Width 10 Type:  Numeric :1+A11 = 213464.000000000000
Position B12 Width 15 Type:  Text (Right justified):36mm Bolt
Position C12 Width 15 Type:  Text (Right justified):216/12
Position D12 Width 18 Type:  Text (Right justified):GLOBAL Bolts
Position E12 Width 11 Type:  Text (Right justified):4580 737
Position F12 Width 11 Type:  Numeric :150%F11 = 0.453000000000
Position G12 Width 11 Type:F Numeric :.48 = 0.480000000000
Position H12 Width 18 Type:  Numeric :+G12-F12 = 0.178000000000
Position I12 Width 12 Type:  Numeric :+H12/G12*100 = 37.100000000000
Position J12 Width 24 Type:  Numeric :5000 = 5000.000000000000
Position K12 Width 24 Type:F Numeric :4350 = 4350.000000000000
Position L12 Width 24 Type:  Numeric :+J12-K12 = 650.000000000000
Position M12 Width 18 Type:  Numeric :+F12*L12 = 294.450000000000
Position A14 Width 10 Type:  Text (Repeating)      :-
Position J14 Width 24 Type:  Text (Repeating)      :-
Position K14 Width 24 Type:  Text (Repeating)      :-
Position L14 Width 24 Type:  Text (Repeating)      :-
Position M14 Width 18 Type:  Text (Repeating)      :-
Position A15 Width 10 Type:  Numeric :cnt(A4>A12) = 8.000000000000
Position B15 Width 15 Type:  Text (Left justified) :BOLT SIZES
Position J15 Width 24 Type:  Numeric :+sum(J4>J12) = 40000.000000000000
Position K15 Width 24 Type:  Numeric :+sum(K4>K12) = 26149.000000000000
Position L15 Width 24 Type:  Numeric :+sum(L4>L12)\total ordered = 13851.000000000000
Position M15 Width 18 Type:  Numeric :+sum(M4>M12) = 1257.600000000000
```

Note - The printout for the file #5:EXAMPLE, can be found in Chapter 9.

```
                                BALSHEET FILE EXAMPLE

                                       ASSETS


             CURRENT ASSETS
Cash                                           8000000.00
Accounts Receivable                             100000.00
Inventory                                      5235000.00
                                         --------------------
    TOTAL CURRENT ASSETS                                        13335000.00

                                                              -------------
        TOTAL ASSETS                                            13335000.00
                                                              -------------
        TOTAL ASSETS                                                          13335000.00
                                                                             =============



                               LIABILITIES & EQUITY


    CURRENT LIABILITIES
Accounts Payable                                 19000.00
Notes Payable                                  4810000.00
                                         --------------------
 TOTAL CURRENT LIABILITIES


      TOTAL LIABILITIES                                          4829000.00


      OWNERSHIP & EQUITY
Capitol                                        8500000.00
Net Income                                        6000.00
                                         --------------------
        TOTAL WORTH                                              8506000.00

                                                              -------------
  TOTAL OWNERSHIP & EQUITY                                       8506000.00

 TOTAL LIABILITIES & EQUITY                                                   13335000.00
                                                                             =============
```

B SUMMARY OF LOGICALC COMMANDS

This appendix provides you with a summary of the commands and functions available in Logicalc.

B.1 Logicalc Commands

The following is a list of Logicalc commands. They can only be entered after having accessed the command line.

F2	Automatic form mode for entering data in specific entries.
DELETE	Delete an entry, column, row or entire array.
?	Displays the Help listing.
INSERT	Insert either row or column.
SEARCH	Displays a directory listing of an entire disk or just Logicalc files on the disk.
⇦	Load a file that has been saved into the Logicalc array.
⇨	Saves the contents of the array onto disk.
⇧⇧ ⇨	Allows you to write the Logicalc file to disk as a text file.

⇧⇧ ⇤	Allows you to load into the Logicalc array a text file created by Logicalc or from another compatible package.
⇧ ?/	Informs you of the number of entries still available to you.
C	Copy entry or range into an entry or range.
E	Activates Edge command and moves the array so the Field Pointer is in upper left corner.
F	Gives the option to change column width, precision or form mode.
M	Merge a saved file with the current contents of the array.
O	Toggle key for the order of evaluation for the Recalculate command.
P	Prints a complete or specified section of the array.
Q	Quits Logicalc and returns to TKS prompt.
R	Recalculates an entry or all formulae in the array.
T	Activates Text Editor enabling correction of data.
W	Shows the row and column heading for the entry where the Field Pointer is currently located.
⇧ =0	Allows you to set/clear locked screen.

B.2 System Functions

!f(!) =allows you to enter a user defined function in one variable. The expression to be evaluated replaces f(!) and the variable data you will enter replaces the '!'.

+SUM(range of entries) = sum of the values contained within the given range.

+AVG(range of entries) = computes the average of the values contained within the given range.

+CNT(range of entries) = returns the number of numeric entries contained within the given range.

+MAX(range of entries) = returns the maximum value contained within the given range.

+MIN(range of entries) = returns the minimum value contained within the given range.

+REGR(range of entries, first coordinate of another range) = computes a linear regression and returns the mean of the second range (dependent variable).

+PROJ(value) = inserts a value for an independent variable into the regression equation and returns the predicted value (the dependent variable).

+DEPD(value) = inserts a value for the dependent variable into the regression equation, solves the formula and returns the best estimate for the independent variable.

+slope() = returns the slope or gradient of the regression equation which may be used to evaluate the degree of correlation.

B.3 Editing Functions

/C	=	centre justify a text entry
/L	=	left justify a text entry
/R	=	right justify a text entry
/=	=	repeat the characters following the equals sign throughout the entry.
/P	=	printing will execute a form feed. This must be located in column A.

| \	=	allows insertion of comment into numeric entry location.
F1	=	enters the entry at the Field Pointer location into the current equation.
	=	toggles type between text and numeric.
↑	=	moves the Field Pointer up one location.
↓	=	moves the Field Pointer down one location.
←	=	moves the Field Pointer to the left one location.
→	=	moves the Field Pointer to the right one location.
⇧⇧ ⏎	=	moves the Field Pointer to the first column of the next row.

C THE SALES FILE

In this appendix we have provided you with the information necessary to create a file called Sales. This is used in chapter 10 as an example file demonstrating the Regression functions.

We have deliberately kept the instructions to a minimum as you should at this stage be fairly competent with Logicalc. The data to be entered is shown in the illustrations overleaf. Details about column precision and width are as follows:

- Column A, width 20.
- Columns B, C, D, E, F, G, width 10, precision 0.
- Columns H, I, J, width 10, precision 2.

```
Col>|A                   |B         |C        |D
Row+-------------------------------------------------------
  1|                      january  february      march
  2|                            1         2          3
  3|                      ======== ======== ========
  4|
  5|District  A
  6|         product 1         300       435        650
  7|         product 2         100        92         81
  8|
  9|Advertising £             1230      1300 >    1435<
 10|
 11|
 12|
 13|
 14|
  +-------------------------------------------------------
              Field pointer:    D9  Entry loc.:    D9

CURRENT DATA     type: numeric
             contents: '1435'
                 edit:
```

```
Col>|E         |F         |G          |H           |I          |
Row+-------------------------------------------------------
  1|     april        may       june
  2|         4          5          6
  3|======== ======== ========
  4|
  5|
  6|       875       1200       1440
  7|        64         55         47
  8|
  9|      1450       1510 >     1530<
 10|
 11|
 12|
 13|
 14|
  +-------------------------------------------------------
              Field pointer:    G9  Entry loc.:    G9

CURRENT DATA     type: numeric
             contents: '1530'
                 edit:
```

D LOGICALC ERROR MESSAGES

This appendix lists the error messages which may occur during the running of Logicalc.

bad coord	- coordinate entered cannot be used for intended purpose.
bad form length	- specified form length is not within the required range (1..66)
bad range coord	- the range of coordinates cannot be used for intended purpose.
can't create	- not enough memory to set format precision.
can't open	- not enough space on disk, or disk needs reorganizing.
can't open file	- not enough room to open file on disk.
couldn't read SYSTEM.MISCINFO	-system could not read a necessary file. Should use a back-up copy.
DATA IS PROBABLY DAMAGED	-there was insufficient memory for the size of report entered.
data too wide	- specified printing section is too wide for the specified printing width.
ERROR - expression ?	- the system was unable to interpret the entry correctly.
ERROR would delete ref(s) at coord	- specified deletion would eliminate data on which other formulae are dependent, so the dependent formulae must be altered first.
FATAL ERROR:not on disk	- the disk is missing an important Logicalc file and cannot operate.
FILEWRITE ERROR	- system had trouble writing file to disk.
math op error numeric overflow	- insufficient space for the intended operation.
math op error divide by zero	- expression would lead to a value being divided by zero, which is undefined.
MEMORY IS TOO LOW	- insufficient room for intended insertion.

!n!	- column is not wide enough.
no form flags	- no automatic form modes have been set.
not along row/column	- specified copy is not in a straight vertical or straight horizontal line.
not ok	- password is incorrect.
OUT OF MEMORY	- no memory left; an entry or two must be deleted before the array can be saved.
Prec. must be 0..10	- precision entry is not within required range.
READ ERROR	- there are bad blocks where the data is stored. You should consult your TKS manual for advice.
sizes don't agree	- number of coordinates you are copying from are different to the number you are copying to.
Width must be 3..63	- column width entry is not within the required range.
WRITE ERROR	- disk gets full while writing a file.
write error,not the same	- second password is not the same as the first password.
write error file not found	- file does not exist.
write error Close	- disk need reorganizing.
write error end start	- in saving a partial file, the top left coordinate must come before the bottom right coordinate.
write error bad coord	- invalid coordinate entered.
WRITE CLOSE ERROR	- diskette is wrong or insufficient room.

Evaluation Errors

0	- value range error: math overflow divide by zero function cannot evaluate value
1	- illegal coordinate or wrong coordinate format
2	- range is not a row or column
3	- missing a '(' or ')'
4	- unknown function, function typed incorrectly
6	- terminal expression is illegal
7	- illegal characters at the end of the line
8	- number is not in correct form
9	- error in regression values
12	- too many parenthetical values
255	- illegal value in REGR function

aA	ACK	ACKnowledge.
	Acronym	A word derived from the important letters of a phrase or expression.
	Application	Any task that a computer is given to perform.
	Application Program	A program written for a particular applicatio (q.v.).
	Array	The array is the entire matrix (table) of data. The size limits are 127 horizontally by 25 vertically although the actual amount of data i limited to 440 entries.
	Assembler	A program that translates an assembly languag into machine code (q.v.).
	Assembly Language	See Machine Code.
bB	Backup Disk	A copy that is made of another disk to prevent valuable data being lost. If data on the original disk is lost it will still be available on the backup.
	Binary	A system of numbers which uses two digits only, '0' and '1'.
	Bit	Short for Binary Digit, i.e. '0' and '1'.
	Block	A group of consecutive characters, words or bytes.
	BS	Back Space.
	Byte	Consists of 8 bits, they are the P2000's internal method of storing both data and instructions, e.g. one alphanumeric character is held in one byte.
cC	Character	A combination of bits which represent, for example, a decimal digit or a letter of the alphabet, in accordance with a particular code
	Column	The lines of data in the vertical direction are called columns within Logicalc.
	Coordinate	A position specification in the form Column/ Row within Logicalc. Also referred to as a location.
	CR	Carriage Return.

dD	Default value	Suggested as the most commonly used value in answer to a menu list. If no other value is specified the default value will be used.
	Disk	A device that is used for storing information. Disks form part of the computer's external store. Disks are coated with a material similar to that used on the tapes for making sound recordings and use a similar method for recording data.
eE	ENTRY	A single data item in the array, an entry is located by its coordinate.
kK	Keyboard	A set of keys arranged in a similar way to the keys on a typewriter. It is the main means of communication between the user and the computer.
mM	Machine Code	A program consists of a list of intructions that the computer obeys. Each of these instructions is represented by a number. The set of these numbers is the computer's machine code. It is very difficult to write programs in machine code because the prgrammer has to know what the code number is for each instruction. For this reason each instruction is given a name, each name gives some indication of the function of the instruction, e.g. the instruction to add two numbers together might be called 'ADD'. The set of instruction names is called the computer's assembly language. The programmer first writes each program in assembly language and then uses an assembler to convert it into machine code. The term 'Assembly Language' is often erroneously used instead of 'Machine Code'.
	Microcomputer	Any computer that uses a microporcessor (q.v.) as the main part of the CPU.
	Microprocessor	A special type of elctronic circuit made of a single piece of silicon and enclosed in a plastic or ceramic case with connecting pins along each side. Such circuits are called integrated circuits. Microprocessors perform most of the functions of the CPU in a microcomputer although they only occupy a fraction of the space.
	Monitor	A device that looks like a TV screen. The computer can use it for displaying information in the form of ordinary text characters. It is the main means of communication between the computer and the user.

rR	RAM	Random Access Memory. The type of memory used b computers for their internal store. At one tim RAM consisted of minute rings of ferrite. Thes rings were known as cores hence the term 'cor memory' was used to refer to the computer's RAM Modern computers use RAM that is made fro silicon. Although reading or writing from or t RAM is much faster than it is for disks th contents of RAM are usually lost when the power i turned off. For this reason most computers use RA for storing temporary results which are the transfered to disk when the calculations ar complete. The quantity of RAM in a system i measured in Kilobits (abbreviated to K). 1k 1000. Sometimes you will see figures quoted i kilobytes, 1k bytes = 8k bits. A typical amount o RAM for a computer system would be anything fro 64k bytes upwards.
	RES	REStore.
	Reset Button	The reset button is located on the front of th CPU cabinet. When this button has been pressed th contents of internal memory (q.v.) will be cleare and the CPU will attempt to load a program fro the disk in drive 1.
	ROM-key	A cartridge that is plugged into the front of th CPU cabinet for identification purposes. A ROM-Ke can also contain a program that is loaded when th system is reset.
	Row	Lines of data in the horizontal direction withi Logicalc.
wW	Window	Within Logicalc, the Window is the section of th array that is displayed on the screen.

INDEX

LOGICALC OPERATOR MANUAL

12NC: 5103 992 26822

This issue comprises the following updates:

no updates

LOGICALC OPERATOR MANUAL

12NC: 5103 992 26822

Including update(s) __ __ __

Originator:
Name ....
Address ....
....
....
....

Comment (if possible, add a copy of the page(s) affected by the comment, marked with the proposed changes):

Please return this form to: Osterr. Philips Industrie Ges.m.b.H.
MAG - PERCO / Marketing Support
A-1100 Wien, Triesterstr. 64
A U S T R I A