# LUCID DATABASE

## Section 2 - Reference

## For tutorial help, see section 1

## Chapter five

## Introduction to Lucid data

### I. What is a Lucid Database?

**A. Two kinds of file.**

1. A Lucid Database is effectively a collection of spreadsheet files and text files. The spreadsheet files contain the data and information about relationships of the data, and the text files contain templates that specify the layout and content of input screens and reports.

2. Since these text files are easily edited using the Model 100's built in text editor, you have total control over the appearance of your input screens and forms.

**B. Data files**

The spreadsheet files are regular Lucid spreadsheets in every way, except that they are built to follow a few conventions that enable Lucid Data to use them as data files.

## C. Records and fields

1. A regular computer data file as used by most conventional data bases and file managers is composed of records and fields. A record is a collection of fields. A record might for example consist of a person's name and address.

2. So for each name and address in the file, there would be one record. For example, if a data file contained 200 names and addresses, it would have 200 records, each containing one name and address.

3. Each name and address record would be a group of fields, for example a name field, a company field, a street address field, a city field, a state field and a zip field.

4. This is at the discretion of the designer of the file (you), for example, you might want to print out your name and address file alphabetically by last name.

   a. In that case it would be neccessary to have two name fields, one for the first name, one for the last name.

   b. On the other hand, if you knew you would never be wanting to sort or select by state, you might decide to put the city and the state in the same field, and have one field for 'City,state'.

**D. Like a spreadsheet**

This two dimensional structure (records forming the vertical dimension and fields the horizontal) reflects exactly the two dimensional structure of a spreadsheet.

**E. Columns are Fields**

1. Lucid Data exploits this isomorphism in the simplest possible way. Forget for a moment that the Lucid spreadsheet offers you 126 columns for your data, and imagine for a moment that it offers you as only as many columns as there are fields in your database records. In the name and address example given above there would be 6 fields (Name, Company, Street, City, State, Zip).

2. So if we associate each field with a column, we would have the name in column A, the company in column B, the street address in column C, the city in column D, the state in column E and the zip in column F.

3. To extend the analogy, let us now identify spreadsheet rows with database records.

**F. Rows are Records**

1. This means that the first record would occupy the first spreadsheet row, the second record the second spreadsheet row and so on.

2. The name field of record 1 would be in

cell A1, the zip field of record 1 would be in cell F1. The company field of record 200 would be in cell B200 and so on.

**G. Databases are Spreadsheets**

1. This choice of representation of our data files has some fortunate theoretical and practical consequences.

   a. The major practical consequence is that any spreadsheet can be considered as a database, and more importantly any database can be considered as a spreadsheet.

   b. This means that database fields can contain spreadsheet formulas, so calculated fields can be specified in a familiar notation.

## II. Relational considerations

### A. Flat Files

A major theoretical consequence is that all the Lucid Database files are forced to be 'flat' files. This 'flat' (two dimensional) structure is one of the defining characteristics of a relational type database.

### B. Join Operations

1. A second defining characteristic of a relational database is the ability to perform a 'Join' operation on two related files, by matching the contents of two common (key) fields to access information from a second file while performing operations on a first.

2. This sounds complicated but is a very useful feature for eliminating redundancy in a database.

### C. Eliminate Redundancy

1. Redundancy is very bad indeed for a database, since the same information stored in two or more records presents sticky problems when updating or changing it.

   a. You can never be sure that you have updated all the occurances, and it is always possible that there is some old or corrupt data in your database even though you have updated it.

b. Redundancy is a further nuisance in the Model 100, simply because it is a shocking waste of memory.

2. The sales and invoicing database example given later gives exact details of how to perform join operations and get rid of redundancy, but lets peep ahead to get a clearer idea of what we mean by eliminating redundancy.

**D. Redundancy Example**

1. A small retailer may have 20 suppliers and 80 lines. He wants to keep a computer record of all his purchase orders.

a. A purchase order must contain the address of the supplier and a description and price for each item.
b. But putting the supplier name and address into each purchase order record would create considerable redundancy.

2. The supplier name and address would occur in the database as many times as an order had been placed.

a. We could eliminate this redundancy and only have the name and address once in the database by identifying each supplier with an I.D. code (for example the first three letters of their name),

b. We could put this I.D. code (rather than the entire name and address) into each purchase order record.

3. Then when Lucid was getting ready to print a purchase order, it could read the supplier I.D, go to the supplier name and address file, look up the I.D. and bring back the complete name and address to print on the purchase order.

**E. Real Life**

1. In real life, the purchase order example given here might not be appropriate, since the contents of a historic purchase order should not change dynamically over time.

a. That is to say, suppose a supplier moved to a new address, and you wanted to examine an old purchase order.

b. Since the supplier ID would now be associated with the new address, the duplicate purchase order you were examining would have the new address on it, and would not be identical in this respect to the purchase order you mailed to the supplier.

2. This would not present major problems in the case of an address, but you would have to be careful in the case of such information as prices. If the price was drawn from a product file by product ID, and the price was changed in the product file between printing the first PO and it's duplicate, the dollar totals on the two copies would not match.

3. Of course when you are aware of this

double edged characteristic of redundancy elimination you can easily design your database to perform accurately in all cases.

# LUCID DATABASE

## Section 2 - Reference

## For tutorial help, see section 1

## Chapter six

## Data Functions

A Tutorial manual precedes this overview. If you are adept with the computer and do not require step by step training then this chapter will provide you with enough information to use the enhancements to Lucid and to introduce you to all the features of Lucid Database.

You will want to use the preceding tutorial if you wish to be taken step-by step through the new features available in SuperROM.

## I. New functions and operators in Lucid.

### A. INT

1. By popular request we have changed the 'INT' function of Lucid to be able to deal with numbers larger than 32,768.

2. It will now truncate the decimal portion of any number up to the limit of what it can display.

**B. NOT**

1. This is a logical (bitwise boolean) function. All you really need to know is that Lucid represents the boolean value 'True' as -1 and 'False' as 0.

2. So NOT(0) gives -1 and NOT(-1) gives 0. This is useful in dealing with comparisons, such as A1=A3.

3. See 'relational operators' below for more info.

**C. & (AND)**

1. This is a logical (bitwise boolean) operator. It selects the matching bits only from it's two operands.

For example:

```
-1&-1 evaluates to -1
 0&0  evaluates to 0
-1&0  evaluates to 0
```

2. All you really need to know is that both operands must be true to yield a result of true.

For example: (A1<B1)&(C1<D1)

is only true if the contents of cell A1 is less than the contents of cell B1 and also the contents of cell C1 is less than the contents of cell D1.

**D. | (OR).**

1. This is produced on the M-100 keyboard with shift Grph - .

2. This is a logical (bitwise boolean) operator. It selects any bit set in either of it's two operands.

For example:

```
-1|-1 evaluates to -1
0|0   evaluates to 0
-1|0  evaluates to -1
```

3. All you really need to know is that if either operand is true the expression will yield a result of true.

a. For example:

(A1<B1)|(C1<D1)

Is true if the contents of cell A1 is less than the contents of cell B1 regardless of the contents of C1 and D1.

b. If however the contents of cell A1 is greater than the contents of cell B1, rendering the expression in the first set of parentheses false, the OR expression will still yield a result of true if the contents of cell C1 is less than the contents of cell D1.

c. Only if both relational expressions are false will the logical OR yield a result of false.

## II. Relational operators.

**A. Lucid now supports relational expressions.**

This is chiefly of use in reports for establishing whether to select a record for processing or not.

1. For example if we wished to print only addresses in the zip code 75229, we would put '75229 into a cell, for example cell AD8.

2. If the name and addresses in our sample database were in column F, in our selection criterion cell we would put F#=AD8

3. This would select only those records for which the contents of column F were 75229.

**B. Relational operators always yield a logical value of true (-1) or false (0).**

**C. Relational operators supported by Lucid are:**

=, <, >, =>, <=, <>

# III. New functions for Lucid Data.

These new functions yield meaningless results in Lucid's spreadsheet mode, they are useful only in Lucid Data for generating reports.

**A. JYN - Join.**

1. This function enables Lucid to grab data from more than one spreadsheet.

2. It does not require normalized keys, but will return a value from the first matching key in a sequential search. It will only return data when the key matches perfectly, unlike TBL which returns data from the first target range value greater than the key.

3. The two functions are therefore in a sense complementary. This feature, of only delivering a result on an exact match means that the keys in the target range do not need to be in sorted order.

4. The syntax is as follows:

    a. JYN("filename",dataref,dataref, dataref)

        (1) Where "filename" is the name of a .CA file containing information you wish to access, and 'dataref' is a cell reference to a cell on the line currently being printed.

(2) This is equivalent to a field reference in the current record. Here is an example:

JYN("MPLYEE",A#,B#,D#)

b. The filename MPLYEE would be a file of employee information. The first data reference is to a field in the lucid data file we are currently in. Lucid looks at the value in this field which for the sake of example we shall say is K500, an employee number. Lucid takes this value to the specified file (MPLYEE.CA) and searches in column B for a match.

(1) Obviously in this particular case, our current file keeps the employee number in column A, and the other file (MPLYEE.CA) keeps the employee number in column B.

(2) When Lucid finds a match in column B, it looks for the corresponding value in column D, which might in this case be the color of employee K500's aura.

5. OK, lets be even more exemplary:

a. Here is a sample file which we happen to be in at the moment.
It is called INVEN.CA

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 001 | 1-11 | Brass Parrot | | PYT | 200 | 100 |
| 002 | 3-32 | Wooden Leg | | BPE | 30 | 10 |
| 003 | H-98 | Ships Bell | | BPE | 100 | 10 |
| 004 | 9-00 | Dead Parrot | | DPI | 2 | 1 |
| 005 | 0-02 | Muenster cheese | | PYT | 20 | 2 |
| 006 | M-01 | Shrubbery | | PYT | 10 | 1 |

As you can see column A has a part number, column B has a description, column C is empty, column D has a supplier I.D, column E has the reorder quantity and column F has the reorder level.

b. Here is a sample file which has supplier information corresponding to the example above. Lets call this file SPLIER.CA

| | A | B | C | D |
|---|---|---|---|---|
| 001 | PYT | Python Supplies | 100 King St | London WC2 |
| 002 | BPE | Black Patch Ent | 100 King St | Christiansted |
| 003 | DPI | Dead Pets Inc. | 100 Royal | Dallas |

(1) If we were generating a report from INVEN.CA, and we wanted to access the supplier name, we would use the following expression:

JYN("SPLIER",D#,A#,B#)

(2) This would take the supplier ID from column D in the current file (INVEN.CA), go over to SPLIER.CA and look in column A for a match. For the first record in INVEN this ID would be found in cell D1, and it would be PYT. This matches in SPLIER column A row 1. Now Lucid knows what row to get the result from.

(3) The JYN expression specifies column B as the result column. So the contents of cell B1, namely 'Python Supplies' would be printed as the result of the JYN expression. Other examples of JYN will be found in the section on report generating.

**B. NUL**

1. This function is used to specify which records are to be selected in a report.

2. For example the selection criterion

NOT(NUL(A#))

Would print every record (row) which had a non-blank column

**C. CHG**

1. This is useful for specifying subtotal breaks in a report.

2. For example suppose that we had a file of items sold, price and salesman. If column C

contained salesman ID's in sorted order, the expression

CHG(C#)

would cause a subtotal break when it got to a fresh salesman ID.

**D. TLG,TL1,TL2**

1. These functions return a grand total and two levels of subtotals for any column.

2. For example

TLG(A#)

would print the grand total to date of column A.

3. TL1(A#)

would print the subtotal of column A since the last Break 1.

4. TL2(A#)

would print the subtotal of column A since the last Break 2.

### E. LNG, LN1. LN2

These functions return a total count of lines printed so far, and a count of lines printed since the last subtotal break.

1. LNG

Gives the total number of lines selected so far.

2. LN1

Gives the total number of lines printed since the last Break 1.

3. LN2

Gives the total number of lines printed since the last Break 2.

### F. NBP,NB1,NB2

These functions return the number of pages and the number of breaks to date.

1. NBP

Gives the number of pages printed so far.

2. NB1

Gives the number of Break 1's so far

3. NB2

Gives the number of Break 2's so far.

## III. Data formulas.

These new operators and functions can be combined with cellrefs and datarefs to provide remarkable power for users of Lucid Data. There are numerous examples of such uses throughout the two sample databases in the preceding chapter. The main categories of use for these formulas are:

**A. For selection and break criteria.**

**B. For virtual fields.**

# LUCID DATABASE

## Section 2 - Reference

## For tutorial help, see section 1

## Chapter seven

## Template files

## I. Definition

### A. Two kinds

1. Lucid Data uses two kinds of template forms, View templates and Report templates. They are similar in many ways, but quite distinct.

2. View templates allow you to design custom input screens for your database just using TEXT.

3. Report templates allow you to do the same for your printouts.

### B. Like "fill in the blank" forms

1. Templates are conceptually similar to 'fill in the blanks' forms, such as IRS forms.

2. When you are in TEXT, designing the template, you can edit anything on the screen.

a. This is when you type the static, unchanging, background text on your template, the text corresponding to the pre-printed stuff on an IRS form.

b. At this time you also specify the 'blanks', which we call 'fields'.

## II. Designing the template

### A. You must imagine

1. The start of a blank is indicated by a left handed square bracket '['. The end of a blank or field is indicated by a right hand square bracket ']'.

2. While designing a template you must imagine what it is going to look like when you are in Lucid, and you have accessed the template for data input or output.

a. At that time, *only the fields will be accessible*, the cursor will go nowhere else on the screen.

b. To change the background (static) information, you will have to go back to TEXT.

### B. Unique feature

1. Incidentally, it is a unique and very user friendly feature of Lucid Data that you can change your templates whenever you like without having to alter your data files.

2. Anyone who has ever worked with a database prior to Lucid knows that designing the input screens is a very dynamic process.

a. When you work for a while with any screen you soon realize that you would

like to add something, or modify a prompt or a field.

b. You just can't do that on most data bases, even the most popular desktop programs without having to make changes in the data files.

3. It is so wonderful to be able to freely make changes in your templates without any concern as to the effect on your data files.

**C. Regular Fields**

1. Each field in a template (ie, each 'blank' delimited by square brackets) corresponds to a field in the database, ie to a column in a spreadsheet. The way in which we specify which spreadsheet column is associated with which template field is to include a field specifier (dataref) between the square brackets thus:

[A# ]

2. This specifies the contents of this field as whatever is in column A of the current spreadsheet row (ie record).

**D. Virtual Fields**

1. It is possible to override the mechanism by which the contents of the current record are substituted into the template field by Lucid simply by specifying a record (row) number in

addition to the field identifier thus:

[AA1 ]

This feature is occasionally very useful indeed. Its use is demonstrated in the invoicing database example.

2. In this case cell AA1 would have a formula in it, including one or more datarefs.

   a. If the formula had no datarefs in it, the cell would evaluate to the same result for each record in the database, and show the same value in each View or Report line.

   b. By having a dataref in the formula, a different cell reference is substituted in for each record displayed or printed, and the formula evaluates to a unique result each time.

3. As a convention to make it easier to navigate around the database, we suggest that all your virtual fields are kept in column AA, except those which are being used to accumulate totals in Report, which must each be at the head of a column, and which must be the only entry in that column. This is explained more fully in the tutorial section on the invoice application.

4. By using virtual fields, you can display information on an input screen that is associated with the record you are typing

in, but which is not actually stored in that record (hence *virtual*, since it doesn't exist in the record). The sales input example in the tutorial shows how you can display a customer name on the screen when the customer I.D. has been typed in.

5. Virtual fields are also used in report template files, to draw information from multiple data files (using JYN), and to access data functions that are meaningless in the context of a single record, such as TLG, TL1, TL2, NBP, NB1, NB2, LNG, LN1, LN2.

**E. Record Number Fields**

1. There is another symbol that can be used in the field specifier, it is the hash or pound sign, '#'. When used in a field specifier without a column designation, it causes Lucid to display the cu$^{r}$ent record number in the field. Here is an example:

[# ]

2. This is also in a sense a virtual field, since it is not stored in any record. It is most useful for View screens to keep track of where you are in the database, but it can also be used in report templates.

# LUCID DATABASE

## Section 2 - Reference

## For tutorial help, see section 1

## Chapter eight

## Input screens in Lucid Data

### I. Structure of a View file.

**A. Multiple page input form**

1. The input form or "View" file can have multiple screens or "viewpoints".

2. A View file can have as many viewpoints as you like in it. Think of each viewpoint as a different page on a form.

3. This 'paging' feature of Lucid Data frees the input form designer from the limitations of the Model 100 screen size.

**B. Viewpoint Separators**

1. The Viewpoints are separated by a paragraph symbol, gotten on the Model 100 by holding down the 'CODE' key and pressing the '0.

2. Lucid will only read as much of a Viewpoint template as will fit on a screen, ie the first 7 lines (the eighth line is reserved for labels).

a. Therefore, after the seventh line prior to the paragraph symbol denoting the next viewpoint, you can include as much explanatory text or notes as you wish to yourself, the designer.

b. The person who is doing data entry using this View will never be able to see these notes.

3. The person doing data entry using your view file can page between Viewpoints by pressing F5 (Vwpt).

a. After displaying the last viewpoint in the file, pressing F5 (Vwpt) again causes the first Viewpoint to be redisplayed.

b. The maximum number of Viewpoints you can have in a single View file is 255. This provides great versatility since you can implement such things as help screens and detailed instructions without any penalty.

## II. Special Lucid Data file features

### A. Input formatting

1. Lucid automatically assumes that anything beginning with an arithmetic sign, a digit or a period (+-0123456789.) is a number or a formula, and anything else is a label.

2. View changes this assumption, since database applications frequently use numbers as identifiers rather than arithmetic quantities. For example you would not want a zip to default to the numeric format

75229.00

3. View assumes everything is a label unless you specify it as a number by starting it with a plus sign ('+'). View will not let you enter a formula. To do this you must first return to the spreadsheet by pressing F7 (Lucd).

4. The problem with defaulting to label format is that it is a drag to have to type in a plus sign whenever you want to enter a number. Our invoice example has a quantity input which not only should default to numeric type, but it should not show decimal places.

**B. Specifying default types**

1. Lucid Data allows you to override the standard defaults and specify default types and formats for any of your fields.

   a. This is simply done by going to row 250 of the spreadsheet and entering a fake value in the format you want.

   b. For example in order to define the default format of field D as numeric with no decimal places, you would goto cell D250 and type in a '0'. It shows on the screen like this

   0.00

   With the cursor on cell D250 press F7-Sel, then ENTER, then F1-Disp, then F2-#.##. The screen says:

   Decimal places:2

   Backspace out the 2 and type 0, then press ENTER, and F8-Exit.

2. Now whenever you type anything into field D while in View, it will automatically default to numeric with no decimal places.

## III. Designing a View file

### A. Example

1. The following viewpoint is from the mailing list example.

2. It illustrates the main points of view file construction:

```
   Mailing list input form   [# ]
Name:   [a#                                ]
Company:[b#                                ]
Street: [c#                                ]
City:[d#           ] St:[e#] Zip:[f#   ]
Phone H:[g#           ] W:[h#              ]
  Press F5 for printout.
```

### B. Description

1. This input screen was designed using TEXT.

   a. The first line is simply a title, with one field at the end. The field is delimited by square brackets, and the contents of the field is a '#' sign.

   b. This symbol determines that when the input screen is displayed by Lucid for data input, this field contains the record number (the same as the row number in the spreadsheet).

2. The second line again has only one field, whose contents are specified as a#. This means that when Lucid displays this

screen for data input, this field will display the contents of field A, and allow input to it.

3. All the other fields work in a similar way.

4. Note that the fifth line has three fields on it.

# IV. Using View.

## A. Accessing

1. After designing your View file and typing it in, and after defining the default types and formats for your input fields on row 250, you are ready to try out the new input screen. In Lucid, press F6 (Data). The screen says

```
 View Rprt                    Text       Exit
  1    2     3     4     5     6     7     8
```

These keys are only effective if the cursor is resting on a cell with a .DO filename in it.

a. Pressing F6 (Text) puts you into the Model 100 built in text editor.

b. Pressing F2 (Rprt) causes a report to be printed, provided the cell is part of a valid report specifier range (see below).

c. Pressing F1 (View) invokes the data input function of Lucid Data, again provided the file is in a valid format.

2. Press F1 (View). The bottom line says:

```
View file name:
```

3. Type in the name of your view file, for example 'TEST', and if you have successfully followed our instructions, the first viewpoint of your view file

will appear on the screen and you can start entering data.

*HINT: If you want to make a quick change to your View file, you can get into it very rapidly by typing it's name (eg TEST.DO) into any empty field then pressing F6-Text. After exiting from TEXT having made your changes, you can just delete the file name you had put in the empty field.*

**B. Special Report Viewpoints**

1. You will notice that in all the examples in the earlier tutorial section, the last viewpoint in the view file is the Report View.

   a. Chapter nine on Report tells how there are two ways to invoke Report.

   b. The most user friendly technique is for you to design a view screen which displays all the report templates available for the current data file.

2. This feature provides an example of how you can exploit the versatility of Lucid Data to make a completely customised database.

## V. Function keys in View

If you can't see the following function key labels, press LABEL to display them:

```
Edit Dupe Prev Next Vwpt Text Lucd Exit
 1    2    3    4    5    6    7    8
```

**A. F1 (Edit)**

Works just the same in View as it does in Lucid, except that it does not allow you to edit formulas. Formulas can only be changed by Lucid.

**B. F2 (Dupe)**

1. This handy key enables you to repeat the contents of this field in the previous record.

   a. It is used for such things as selling multiple items to the same customer (press Dupe for the customer ID).

   b. This will not work for the first record since there is no previous value to duplicate.

2. For example suppose that one of the fields in your view looked like this:

   [A#  ]

   a. Then entering data into it in the first record would put the data into cell A1.

   b. After pressing F4 (Next), you would be inputting data to record 2, so if you placed the cursor on

this field and pressed F2 (Dupe), the contents of cell A1 would be copied into cell A2 and appear under the cursor.

**C. F3 (Prev)**

1. This moves back one record in the database, for example if you were viewing record 2 (row 2), it would cause record 1 to be displayed.

2. It could be thought of as performing the same function as the up arrow key in Lucid.

**D. F4 (Next)**

1. This causes the next record to be displayed, for example if record 4 was on the screen, pressing F4-Next would cause record 5 to be displayed.
2. It could be thought of as performing the same function as the down arrow key in Lucid.

**E. F5 (Vwpt)**

This advances to the next viewpoint in the View file. After the last viewpoint it wraps to the first again.

**F. F6 (Text)**

1. This works exactly the same way as the sequence F6 (Data), F6 (Text) works from the Lucid spreadsheet.

2. When the cursor is resting on a field which contains a valid RAM file name (eg TEST.DO - no need for any kind of quotation marks, but it must have the .DO extension), press F6 (Text).

   a. This will invoke the built in Model 100 editor, TEXT, just as though you were accessing the file from the main menu, except that when you exit from TEXT with F8 (Exit) you return to Lucid Data.

   b. You return to exactly the screen you were at when you invoked TEXT!

3. If the file named in the field does not already exist, Lucid will create a new, empty one.

4. This feature has the intriguing aspect that you can invoke TEXT from View to edit the Viewpoint you are currently in, and when you return to View from TEXT the change will appear on the input screen. Lucid Data doesn't miss a beat!

**G. F7 (Lucd)**

1. This accesses the spreadsheet features of Lucid. This is neccessary for such activities as setting the default formats at row 250 and entering formulas and data formulas needed by your reports and View files.

2. If you then exit from Lucid to the Main Menu, accessing the data file from the main menu will put you into Lucid Spreadsheet again, and View can be

accessed in the normal way by pressing F6 (Data), F1 (View).

### H. F8 (Exit)

This exits from View directly to the Main Menu. When the data file is subsequently accessed from the main menu you find yourself immediately back in View, in the same viewpoint, record and field that you were in when you pressed F8-Exit.

### I. LABEL

1. The LABEL key works the same in View as it does in Lucid, toggling the label line on or off. When on it diisplays the function key labels on the bottom line of the screen.

2. When off it displays the same info as is displayed at the wide bar cursor. This is useful in the case of a label that is longer than the field defined for it in the view template. Although the contents of the field on the view screen is truncated, the bottom line of the display shows the entire contents.

3. Unlike Lucid Spreadsheet, the bottom line will not show formulas. To view formulas you must first use F7-Lucd to access the Lucid spreadsheet.

### J. PRINT

1. Like F6-Text, this key does nothing unless the wide bar cursor is resting on a RAM file name. Unlike with F6-Text, this filename must additionally be at the head of a column of special information, the report specifier range.

2. The report specifier range is fully explained in the section on Report.

**K. '>' Goto**

1. This works in a similar way to Goto in Lucid Spreadsheet. It is invoked in the same way, using the Goto key, '>' (gotten by pressing the period key above CODE in conjunction with the SHIFT key).

The screen says:

Goto record #:

2. Just type in a number, which will take you to that record (row) number. Just like with Lucid Spreadsheet, View remembers where you came from, and offers that as a default next time you press Goto.

**L. Find**

1. This works in a similar way to Goto in the sense that it remembers where you came from, and offers that as a default next time you press Goto. The key that invokes Find is the question mark key, above the NUM key, which must be pressed in conjunction with the Shift key.

2. When you press '?' the screen says:

Find:

Type in the word, phrase or number you wish to find. Lucid will search for it only in the field (column) that the cursor is resting in. This enables you to

search for a zip without finding the same number in another field.

a. For example a street address may have a five digit number such as

13350 Wakefield Blvd.

that coincidentally matches the Zip 13350 in some other address in the database.

b. If you were searching for the zip 13350, you would not want to be shown the Wakefield street address.

3. Note that just like with data entry in View, Find assumes that what you are typing in is a label even if it starts with a digit. To force a search for a numeric value, you must start your typed in key with a '+' plus sign.

# LUCID DATABASE

## Section 2 - Reference

## For tutorial help, see section 1

## Chapter nine

## Reports in Lucid Data

## I. The Report Specifier block

### A. Accessing

1. Report can be accessed in two ways:

   a. From Lucid Spreadsheet by pressing F6 (Data) then F2 (Rprt).

   b. From View by pressing PRINT.

2. In both cases, the wide bar cursor must be resting at the head of a report specifier block. A report specifier block is a column of 10 cells that can reside anywhere in the data file spreadsheet.

### B. Where to place it

1. To make it easy to find but still keep it out of the way of your data, it is best to start the first report specifier block at cell DR1. This has the advantage of mnemonic ease (Data Report 1).

2. You can then start a *second* report specifier block at cell DS1 and so on.

There is no limit on the number of report specifier blocks that you can put in a single spreadsheet data file.

## C. Contents of a Report Specifier Block

1. Example

a. This is the report specifier block for the invoicing example in the tutorial section of this manual.

(1) Notice that notes about what is in each cell of the report specifier block have been typed in to the adjacent cells.

(2) This is for illustrative purposes only, but for the first few reports you specify you may like to include those notes to remind you of what is required.

```
        DR1        DR2        DR3      DR4
001 INVOIC.DO Report template file name
002       0.00 Selection criterion
003   *Data*   Break 1 criterion
004            Break 2 criterion
005       1.00 Number of columns of label
006       1.00 Starting break 1 count
007       1.00 Starting break 2 count
008       1.00 Starting page count
009      66.00 Number of lines per page
010       4.00 Left margin
```

2. Explanation of each cell

**a. 001 Report template file name**

(1) This is a regular .DO file name, just like any file that you can create with TEXT, and it can be accessed with F6-Text from View.

(2) It must be typed with the .DO extension in order to be recognized by Lucid Data as a text filename. No need for quotation marks or anything else like that.

(3) This file must have a specific format in order to produce a meaningful report. The format is similar in some respects to View file formats. The exact format is discussed in the next section.

**b. 002 Selection criterion**

(1) This cell contains a formula that determines which records are to be printed in the report, and which are to be ignored.

(2) A record is only selected for printing if the formula in this cell is true (non zero) for that record.

i. For example, if you leave this cell blank, nothing will be printed, since the criterion will be false (0) for every record in the file.

ii. If you were to put a constant (eg 1) into this cell, a line would be printed in the report for each of the 249 rows in the spreadsheet, even the blank ones, since the selection criterion would be true (non 0) for every record in the file.

(3) So it is best to put a formula into this cell, which includes a data referemce, for example:

+NOT(NUL(A#))

(4) This formula will only select those records for which field A is not empty.

(5) Another example:

+A#=AZ1

i. This will select those records for which the contents of field A match the contents in cell AZ1. This would allow interactive selection of records, since a Viewpoint could prompt the user to type in a key (eg customer ID).

ii. The view file screen would put this typed input into cell AZ1, and when the report was printed only records matching this key would be selected. This is the

technique used in the invoicing example in the tutorial section.

iii. Of course the selection criterion can be much more complicated than this, using as many relational and logical operators as you choose, for example:

+(NOT(NUL(A#))&(B#=AZ1))

**c. 003 & 004 Subtotal Break criteria**

(1) The concept of subtotal breaks is not a complicated one, but it seems mysterious until it has been carefully explained.

(2) Suppose that you have an organization with 5 areas and 8 salesmen in each area. The database contains an area ID in field A, a salesman ID in field B, and his sales in field C. At the end of the month, you could sort the database by salesman, then sort it again by area, so the records were sorted by salesman within each area.

(3) Lucid Data Report can give you totals for each salesman, and totals for each area, and a grand total.

(4) It is done like this. Lucid Report keeps track of the total in each field since the last break 1,

since the last break 2 and the grand total since the beginning.

i. If you put the formula

+CHG(B#)

into the Break 2 criterion, you will force a subtotal break every time Report encounters a new salesman I.D. So if at this point you print the break 2 subtotal for this field, you will get a total figure for that salesman's sales. The formula for this would be

+TL2(C#)

ii. By the way, this formula would be in a virtual field referred to by the break 2 template in the report template file.

iii. More about this in the section on report template file formats.

(5) In order to get the area sales total, you would need another kind of subtotal break, this time on the area I.D. field, field A. The formula for the break 1 criterion would look like this:

+CHG(A#)

(6) So when the contents of field A, the area I.D. field changed it would

cause a subtotal break 1. For example, if there were 21 records for Area 1, the contents of this field would be different in the 22nd record, and the subtotal break 1 template would be printed between the 21st and the 22nd line items. To print the break 1 subtotal of sales for the entire area you would put the formula

+TL1(C#)

(7) In a cell pointed to by the break 1 template in the report template file. More about the report template file format later.

**d. 005 Number across**

(1) This is how you can print mailing labels across a page. As you probably know, sticky mailing labels come on fanfold paper, in various widths. It is possible to get them one label wide, but more common to have three or four labels next to each other on each row.

(2) If the value in the number across cell is anything but 1, the subtotal breaks and page headers and footers are disabled, since Report assumes that you are printing mailing labels.

(3) Normally this value would be kept at 1.

**e. 006, 007, 008 Starting counts**

(1) These three values are the base values for the three functions NBP, NB1 and NB2. These are page count, count of break 1's and count of break 2's.

i. The reason Lucid provides you with the option of changing the base is in case you are generating forms that need serial numbers. For example in our invoice example in the tutorial section we use the page count as the invoice number.

ii. The maximum value for these counts is 65535, after which it wraps back to 0.

(2) Your invoices would not be very professional if every time you printed invoices the first was number 1! By setting these initial values each time you print, you can provide effective serial numbering of your forms.

**f. 009 Lines per page**

(1) This is the same as the lines per page function in Lucid spreadsheet under the PRINT function key. The value you put in here does not affect the one you set with the function key, except it overrides it while

the report specified by this specifier block is being printed.

(2) Lucid Data provides this feature because you may want to print out several reports from a single database file, on different sized preprinted forms. Setting the lines per page for the particular report saves you having to go into the PRINT function of Lucid Spreadsheet each time you want to print on a form with an unusual length.

**g. 010 Left Margin**

(1) This feature is provided in Report Specifier Blocks for the same reasons as the lines per page specification.

(2) You may have a preprinted form with a unique layout in the same data file as a report on a completely different form.

(3) With this feature you don't have to keep resetting your PRINT settings.

3. Other Printer Specifications

You can redirect your output to any device or file by changing the F3-Outp setting under the PRINT function key in Lucid Spreadsheet.

## II. Report Template File Structure

### A. Common features of template files

1. For a discussion of template files in general, see the section entitled 'Template Files'. Report template files have the same features as View templates, but you need to know a few additional characteristics that are unique to report templates.

2. To refresh your memory, here are the special codes used by all template files:

[A# ] Regular field
[A1 ] Virtual field (cell has a data formula)
[# ] Record number
Code-0 Template separator

### B. Special features of a Report Template File

1. A report template file has exactly six templates in it, each separated by a Code-0 paragraph marker. This is what they are used for:

a. Line item template
b. Grand Total template
c. Page Header template
d. Page Footer template
e. Break1 header template
f. Break1 footer template
g. Break2 header template
h. Break2 footer template

2. Report templates can be as many lines in length as you wish. When they are printed in a report, they are copied exactly onto the report, but with the fields filled in from the database. There is no limit on the number of lines you can have in an individual template, but of course you will get into trouble if you have for example a page length of 20 and more than 20 lines in your page header template.

3. The invoice example in the tutorial section shows how you can use page headers and footers to produce special forms.

4. When a report template is actually printed in the report, we call it an *item*, so when a break1 template is printed out on the report with the blanks filled in, we call it a break1 item on the report.

5. If you leave a template empty, nothing will print. This is useful for printing summary reports where you want to select line items but print only subtotal information.

**C. Explanation of each feature**

1. Line item template

This is printed each time a record fulfills the selection criterion. If it is left blank (nothing between the Code-0 paragraph markers), nothing is printed.

2. Grand total template

   This is printed after the last line item, and after the last break footer, but before the last page footer.

3. Page Header template

   a. This is printed before the first record, and again after each page feed.

   b. Note that because it is triggered by a page feed, and a page feed is triggered by a line being printed that fills up a previous page, a page header can be printed in the middle of a line item, a break item or a grand total item.

4. Page Footer template

   This is printed at the foot of each page or form.

5. Break1 header template

   a. This is printed before the first line item selected, and again after each Break 1 footer item.

   b. The idea of break headers and footers is somewhat sophisticated, but very useful. Think of Lucid Data as having a 'pointer' to the line item currently being printed.

   c. A break falls between two line items, so which record number would be the one

for Report to use when evaluating a data reference such as '+A#'?

d. Say the subtotal break fell between record numbers 8 and 9, would this reference (+A#) return the contents of cell A8 or cell A9? Depending on your requirements for a particular report, either might be preferrable.

(1) For example suppose you wanted to preface each area's sales figures with the line

'Sales for the Southwest Area'

(2) You would need to take the word 'Southwest' from the record following the break and substitute it into the template.

(3) On the other hand if you wanted to end the list of sales for an area with the subtotal line

'Southwest Area Total $210,000'

(4) You would need to take the word 'Southwest' from the record preceding the break and substitute it into the template.

e. Lucid Data provides you with the flexibility to do either of these options, by offering you two different templates for each break.

f. The Break Header takes information from the line item *following* the subtotal break.

g. The Break Footer takes information from the line item *preceding* the subtotal break.

6. Break1 footer template

   See the 'Break 1 header template' for a discussion of break headers and footers.

7. Break2 header template

   a. See the 'Break 1 header template' for a discussion of break headers and footers.

   b. This template is printed before the first record and again each time the Break 2 criterion is fulfilled.

8. Break2 footer template

   a. See the 'Break 1 header template' for a discussion of break headers and footers.

   b. This template is printed after each type 2 break, and again after the last data record, before the break 1 footer template.

**D. Illustration of the order in which report templates are printed.**

```
Page Header template
Break 1 header template
Break 2 header template
Line item
Line item....  Break 2 occurs here
Break 2 footer template
Break 2 header template
Line item
Line item.... Break 1 occurs here
Break 1 footer template
Break 1 header template
Line item
Line item.... Both break 1 and break 2
                occur here
Break 2 footer template
Break 1 footer template
Break 1 header template
Break 2 header template
Line item
Line item.... Last line item in report
Break 2 footer template
Break 1 footer template
Grand total template
                Empty lines to fill page
Page footer template
```

Note that the break 2 templates are nested inside the break 1 templates whenever both types of break occur at once.

# LUCID DATABASE

## Section 2 - Reference

## For tutorial help, see section 1

## Chapter ten

## Quickref

### I. Logical Operators:

& - Logical AND. Example A2&B2
| - Logical OR. Example A2|B2

### II. Relational operators:

=, <, >, =>, <=, <>

### III. Data Functions:

**A. JYN("filename",dataref,dataref,dataref)**

Join files. Example:

```
JYN("MPLYEE",A#,B#,D#)
```

**B. NOT**

Logical NOT. Example:

```
NOT(A1)
```

C. NUL

Test empty field. Example:

```
NOT(NUL(A#))
```

D. CHG

Test field for change. Example:

```
CHG(C#)
```

E. TLG,TL1,TL2

Field totals:

TLG(A#) Field A grand total
TL1(A#) Field A Break 1 total
TL2(A#) Field A Break 2 total

F. LNG,LN1,LN2

Selected record counts:

LNG Total number of records selected
LN1 Records selected since last Break 1
LN2 Records selected since last Break 2

G. NBP,NB1,NB2

Break counts:
NBP Number of page breaks to date
NB1 Number of Break 1's to date
NB2 Number of Break 2's to date

## IV. Data function key tree

```
View Rprt                    Text      Exit
  1    2    3    4    5    6    7    8

Edit Dupe Prev Next Vwpt Text Lucd Exit
  1    2    3    4    5    6    7    8
```

## V. View file templates

[A#  ] Regular field
[A1  ] Virtual field (display only)
[#   ] Record number (display only)
Code-0 Viewpoint separator
Comments Text beyond line 7 is ignored
by View

# VI. Report specification

### A. Report specifier block

```
       DR1        DR2        DR3       DR4
001 INVOIC.DO Report template file name
002      0.00  Selection criterion
003   *Data*   Break 1 criterion
004            Break 2 criterion
005      1.00  Number of columns of label
006      1.00  Starting break 1 count
007      1.00  Starting break 2 count
008      1.00  Starting page count
009     66.00  Number of lines per page
010      4.00  Left margin
```

### B. Selection criterion

Only records matching the criterion are selected. Example: +NOT(NUL(A#))

### C. Break criteria

A subtotal break occurrs whenever the criterion is true
Example: +CHG(A#)

### D. Starting counts

These are the base values for NBP, NB1, NB2.

### E. Number across

Set to 1 for reports, up to 4 for mailing labels.

## VII. Report File Templates

[A# ] Regular field
[A1 ] Virtual field (cell has a data formula)
[# ] Record number
Code-0 Template separator

## VII. Order of report templates

Line Item
Grand Total
Header Item
Footer Item
Break 1 Header
Break 1 Footer
Break 2 Header
Break 2 Footer

# OTHER PRODUCTS FROM PCSG

After experiencing SUPER ROM, most users ask " What else do you have". PCSG is the foremost developer of software for the Model 100 and 200. You should have received some literature describing some of our other excellent packages, but if you did not please call or write for an information packet.

One of the difficulties of buying software that you haven't seen, is that often it simply is not as advertised. Many amateur programmers will offer their wares with professional advertising with claims of functionality and ease of use that simply are not true. At PCSG you can be assured that the programs are well designed, well documented and they really perform the way you would expect them to.

**Here are some of our other programs:**

**DISK+**

This program lets you instantly transfer files from your Model 100 or 200 to a desktop computer. We have versions for virtually every desktop computer manufactured, from IBM PC's to the Commodore 64. We have versions for all Radio Shack desktops.

This program is remarkable. You use the desktop's disk drive as a drive for the M-100 or 200. All text files are made compatible on the other machine. Transfering files in either direction is as easy as using your Model 100's Main Menu.

Some people use DISK+ to make different desktops compatible with each other. For example, using the 100 or 200 as intermediary you can take an Apple II's text files and use them on your IBM PC.

*Works over the telephone*

If you have an IBM PC or other MSDOS desktop, DISK+ works over the phone. You can leave your IBM unattended back at the office or your home. From a remote location DISK+ works just the same as if you were in the same room with it.

All you do is connect the phone jack, and press one function key for automatic dial up and connect. Then the rest is exactly the same; Main Menu type screens and ENTER key loads and saves. The only difference is that it is slower in transfer, but when you are in a distant place it is so nice to have access to your files, that a little wait isn't a bother.

The great thing is that all files or programs are transfered ready to use or run, and done with an error checking mechanism that guarantees that what is transferred is a perfect copy.

ANALYST+

Business financial analyst program. Gives you very powerful analytical tools to make business decisions based on breakeven analysis, return on investment, interest costs, IRR, NPV, FV, modified IRR and many other financial functions available at the touch of a function key. It also prints out detailed reports including amotizations schedules for any time period.

TUTOR+

Typing instruction program. Feeds through your Model 100 or 200 screen a comprehensive course as detailed as any high school typing curriculum. Has a unique way of presenting graded typing drills to

also hones your touch typing abilities.

**TYPE+**

Interesting program for those who need a typewriter. Actually turns Model 100 plus any printer into a memory typewriter. Shows last line you typed on the screen as a vitrual window of 80 columns or up to 132 if you prefer. Prints immediately to the printer as you type (last character, word or line) and writes a formatted or unformatted copy to RAM at the same time.

This program replaces TEXT as a way to generate a text file. It is an interesting compliment to SUPER ROM.

**RESTORE**

This is a program that everyone needs. It will restore lost text files anytime you ever cold start your Model 100. As amazing as it seems it will bring back your files even though your Main Menu is completely blank.

**LIGHTNING**

Disk speedup program. This is not for the portables but for the IBM PC and its clones. Makes disk intensive programs such as databases function up to four times faster. It works equally well on the Tandy 1000 and other MSDOS machines.

This is a very special development that works by means of what is called an intelligent cache, providing all the advantages of a RAM disk but with out the dangers of data loss.

This is one of our most sucessful products. It is use in thousands of companies across the country.

The results are astounding.

# HARDWARE PRODUCTS FROM PCSG

**Portable Disk Drive**

This is the Holmes Engineering Chipmunk, the result of joint developmental effort by Holmes and PCSG.

We offer both Model 100 and 200 versions.

*Superior to the Tandy drive*

There is no comparison between this unit and the Tandy disk drive. We can show you that the lower priced Tandy unit is not only vastly inferior, but it is actually the more costly drive.

Features of the PCSG/Holmes drive:

- No complicated startup procedure. The operating system comes active instantly when you touch one button.

- Works just like the Main Menu. To save a file just move the wide bar cursor to the filename and press one function key.

- Loading is the same way. The disk files show on your screen like the Main Menu. Put the cursor on the file to load and press a function key.

- You don't have to load a file or program to use it. Just look at the disk menu and put the cursor on the filename and press ENTER and you are using that file or program just like RAM.

- It is as fast as RAM. No cumbersome slow cassette-like speeds with this one, you access files or programs just as fast as you do from your Main Menu.

-354K storage on a single diskette. Think of it! Ten diskettes and you have over 3.5 megabytes. The Tandy diskette only stores 100K.

- Nicad rechargable battery all self contained. Because it has a sleep mode that is in effect when you are not actually reading or writing data, it will last for a week or more between charges.

- Random access. Just by adding a few simple instructions (explained in the manual) to any BASIC program you can have true random access. This means that you are not limited by the limited RAM space of the 100 or 200. All the data files can then be on the diskette.

- Inexpensive. Costs more to purchase than the Tandy drive, but a few boxes of diskettes (3 and 1/2 boxes of Tandy diskettes to get the same storage as one box of ours- at $50.00 a box), and the batteries that you waste with the cheaper unit will end up making it more costly in a short while. Then what do you have? An inferior product that you paid more and will continue to pay more to own.

Marketed exclusively by PCSG.

**128K RAM expansion**

This the answer to the commonest request of Model 100 users; more memory. On a single tiny board is packed an amazing 96K of additional memory for your Model 100. Combined with the 32K you already have, it gives you four banks of 32K each. The ROM socket is left free and available so you can run SUPER ROM in any of the banks. Additionally a program is provided with the RAM expansion called RAM+. It comes on a Snap-in ROM and provides menu file management in any bank, and enables you to copy any files from bank to bank with

just a function key. The RAM module was designed and is manufactured by Cryptronics Inc. utilizing space age techniques and components, notably a multi-layer PC board of a density conventionally supposed to be impossible. Marketed exclusively by PCSG.

## ROM Bank

When you have SUPER ROM, DISK+, RAM+ and other ROM programs, the idea of being abe to turn a switch to access any of them instantly was formerly just a dream. ROM Bank makes that dream a reality, but more so. Not only can you plug in up to 6 ROMS into the Model 100 at the same time, but you also have massive battery backup, rechargable thousands of times, so you will never suffer low power problems again. A single 6 hour charge gives 30 hours of continuous use, four times the average life of a set of AA cells.

If you plug your Model 100 power suppy into the ROM Bank it charges while you are using your computer. ROM Bank fits under the back of the Model 100 and props it up similar to those little legs sold in the Radio Shack stores. Lugs on the ROM Bank engage in the screw recesses at the back of the computer. Assembly is simplicity itself. Just snap off the option ROM door and plug in the cable. Plug in your ROMS into the sockets in the ROM Bank, and plug the ROM bank into the back of your Model 100.

No dissasembly is needed. The ROM Bank is the result of a joint development by PCSG and Cryptronics, Inc. Marketed exclusively by PCSG.