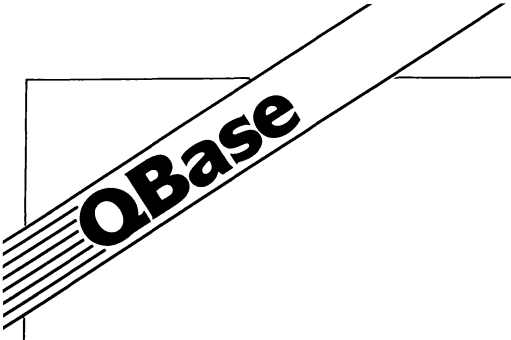



# Report

## Welcome to QBase Report

## Overview

QBase Report is a collection of programs that enable you to create a variety of reports based on QBase or Quick Screen data files. Besides providing standard output reports that may be sent to a printer, screen, or disk file, QBase Report also allows copying selected information from one file to another, and deleting or modifying selected records.

A unique Browse facility lets you view all or part of an entire database, and scroll that information in sorted order up or down, left or right. QBase report also includes several separate utility programs.

Reports are created either by using the QBase Report Assist program, or by entering commands and formatting information directly with the supplied full screen editor. We recommend the former approach, since this will insure that your report definition has the correct syntax and capitalization required by the report processor.

Assist also creates the final report appearance automatically, though you may of course edit the formatting manually. Once a report has been created, it is then loaded and run by the QBase Report program, QRPT.

Like QBase and Quick Screen, all of the BASIC source code for QBase Report and its accompanying utilities is included. Besides allowing you to modify and recompile the programs, this also gives you the ability to create additional reporting utilities.

QBase Report is comprised of a number of subprogram "primitives", which may be structured and re-combined in any way you'd like. Each of these subprograms is documented in detail, following the discussion of the main reporting and utility programs.

## Assist

The Assist program is intended to guide you through the entire report creation process. It begins by displaying a main menu of choices for creating, loading, saving, and formatting reports. Other main menu options allow editing the report, and sending a listing of the report definition to a printer.

Once you have selected *Create Report* from the menu, a list of all the QBase data files in the current directory is presented. Either select the file to be reported on from the menu, or press Escape to enter the name manually. You do not need to type the .DAT extension. As the report is being defined, Assist creates the text of the report definition in a separate window.

## Data Entry Forms

Next you are asked if a data entry form is to be presented when the report is run. A data entry form is useful when all of the selection criteria for the report is not known in advance. For example, if you want to report on all expenses that were made between a given starting and ending date, you will need to specify those dates just prior to running the report. Without a data entry form, the report definition would have to modified each time to specify new starting and ending dates.

A report data entry form is created like any other QBase screen, and the .SCR and .FRM files must be present when the report is run. In the sample time billing application provided with QBase Report, two date fields for exactly this purpose are on the screen named DATES.SCR.

It is not necessary to add the data entry forms to an application's .QSL library, and in fact Report would have no way to find it there if you did. By using the screen file directly, you don't have to specify a library *and* a screen name for the form, and additional memory to store the other screens in the library is not needed.

If you tell Report that a data entry form will be used, then a list of all the form definition files (.FRM) is presented in a menu. Either select the form, or press Escape to enter the name manually. Notice that you may use the same form for data entry as that being reported on if you'd like.

### Criteria

Next you are asked if there is to be any selection criteria. Selection criteria allows you to indicate which records are to be included in the report, for example all records with a Balance Due greater than $100. When no criteria is given, all of the records in the data file will be included. If a data entry form was specified, the selection criteria menu will be presented automatically.

To indicate that only certain records are to be included in the report, you will specify which fields must match a given criteria. For example,

```
BALDUE > 100
```

will include only those records with a balance due equal to or greater than $100. Field criteria in QBase Report operates exactly the same as when searching through a file in QBase with the F9 search key. That is, you may specify Equal, Equal or Greater, and Equal or Less with numeric or date fields, and use wild cards and substrings on string fields. As in QBase, using the Greater or Less symbols also includes fields that are equal.

QBase Report also lets you use an OR keyword, which allows two tables of matching criteria. For example, you may wish to get a listing of all records where the balance due is greater than $100 and the date paid is before 09/21/87 *or* where the balance due is greater than $50 and the sales category is "Merchandise". This would be done as follows, though of course your field names will be different from those shown here.

```
Criteria: BALDUE > 100
And DATE < 09/21/87
Or BALDUE > 50
And SALESCAT = Merchandise Choice 3
```

You may use only one OR when selecting records, though there is no limit to the number of AND statements. Notice that when a choice field is being compared, the choice number must also be indicated exactly as shown.

Besides comparing fields to a constant amount or text string, you may also use values from the data entry form. For example, an expenses report where the starting and ending dates will vary each time the report is run might look like this:

```
Criteria: DATE > Form; DATE1
  And DATE < Form; DATE2
```

**Report Types**

Once any selection criteria has been specified, you are asked to choose the type of report to be produced. The report type indicates whether the report is to be sent to a printer or disk file, or if the report is to add, modify or delete records. For standard printed reports you may specify LPT1:, LPT2:, LPT3:, COM1:, COM2:, or SCRN: (the screen).

If one of the COM ports is specified, then a default parameter will be added to the report type statement. This way, you can edit the baud rate, stop bits, and so forth to match the type of serial printer being used. Details on the various COM options are given in your BASIC manual.

If you indicate that the report is to be sent to a file, Assist will prompt you for the file name to be written to. If you select the Defer option, you will instead be prompted for the report destination when the report is run. Other report types are Browse, Add, Modify, and Delete. Let's take a closer look at each report type in detail.

**Browse Reports**

Browse reports differ substantially from reports intended to be printed, viewed, or sent to a disk file. A major enhancement is that the selected and sorted data may be scrolled continually in any direction. One limitation, though, is that neither totals nor information from related files may be included—Browse is strictly for viewing data records.

Even though record selection and sorting are very fast in QBase Report, a fair amount of time is needed to format and prepare each line of data. For example, date fields must be converted from the special QBase integer form, floating point values need to be formatted to the correct number of places, and so forth. To eliminate the delay, Browse employs a clever multi-tasking technique that lets you begin to view records while they are still being prepared.

Record browsing is handled in one of two ways, depending on the number of records being viewed. If Report determines that all of the information (after formatting) will fit in memory, then a single large string array is created, and all of the data is read in from disk. Otherwise, data is taken in one screen-full at a time.

While the data is being read from disk and processed, the screen is constantly updated. This means that you may use the viewing and scrolling keys immediately, and the records will continue to be processed in the background. Browse recognizes all of the standard cursor keys, as follows.

Home displays the first screen-full of data, and End displays the last. If the first record is already at the top of the screen, then Home will scroll the screen right to make the upper left corner visible. Likewise, if the last record is on the bottom of the screen when End is pressed, the entire screen will be shifted to make the rightmost field visible.

The other cursor keys are handled pretty much as you'd expect, with PgUp and PgDn moving a screen-full at a time, and the four arrow keys scrolling by single lines. Two additional keys you may find useful are the Ctrl and Shift arrow keys. Ctrl-Left and Ctrl-Right scroll the screen horizontally by field, while the Shift arrow keys go directly to the far left and right.

**Add Reports**

Add Reports allow you to copy selected fields from one QBase data file, and append them as new data to the end of another file. This would be useful in a checkbook program, where both deposits and withdrawals must be combined into a single master transaction file.

Once you have told Report that you are defining an Add report, you are asked to choose a destination file that is to receive the copied field information. Next, you are asked to indicate which fields in the destination file are to be assigned, and from which fields in the original file they are to come. If you attempt to assign fields of different types, Report will ask you to confirm your intentions. Type conversion will be performed automatically when possible, though understand that assigning a string field to, say, an integer will cause the *value* of the string field to be used.

The syntax for a typical Add report looks like this:

```
File: XFILE
Criteria: XFIELD > 100
Type: Add YFILE
YFIELD1 Field 1 = XFIELD4 Field 4
YFIELD5 Field 5 = XFIELD15 Field 5
.Enddef
```

In this example, all of the records in XFILE with an XFIELD value equal to or greater than 100 will be appended as new data to YFILE. Only two fields in YFILE will be assigned—YFIELD1 and YFIELD5—and any remaining fields in YFILE will be assigned zero or null values.

Notice that Assist appends each field's number to the end of the field name as shown above. This minimizes the amount of work Report must perform, to determine which field in the source form is being copied, and to where.

**Modify Reports**

Modify reports allow you to alter the information in any or all fields of a group of selected records. Modifying field values can be very useful, for example when you want to increase the price of some of the items in an inventory file. The new field values may be either a constant, or the result of a calculation.

A typical Modify report might be:

```
File: INVENTRY
Criteria: CATEGORY = Hardware Choice 11
Type: Modify
PRICE = PRICE * 1.06
LASTDATE = 11/18/87
.Enddef
```

In this example, any hardware items in the inventory file will have the price increased by 6%, and the date of the last price increase will be marked as 11/18/87.

If you attempt to do field calculations on a string or choice field, Assist will warn you. However, it is then up to you to be sure you know what you're doing. As with Add reports, if you ask for calculations to be performed on string fields, Report will use the value of the field.

When modifying date fields, numbers that are added or subtracted will affect the date by the corresponding number of days. That is, if a date field that contains 03/12/88 has 5 added to it, the result will be 03/17/88 being placed in the field. You can also multiply a date field if you'd like, though it's not obvious why you'd want to do that.

**Delete Reports**

If you ever need to purge obsolete information form a data file, the Delete report will do it for you automatically. As with other QBase Report types, you indicate which records are to be deleted with the selection criteria.

Understand that when records are deleted in a QBase file, they are not physically removed from disk. Rather they are simply marked as deleted by placing an exclamation point (!) into the first byte in the record. A single byte in each record is set aside specifically for this purpose. To actually remove deleted records from a file requires you to run the Rebuild program supplied with QBase.

**Column and Row Formatting**

Once the type of report has been determined, you are asked if the report is to be formatted into rows or columns. This is simply an initial formatting step, and you are free to make some changes to the location of the report fields. This is discussed in greater detail in the section entitled "Report Formatting".

**Printer Codes**

Next you are offered the opportunity to enter any printer initialization codes that are to be sent when the report is run. Rather than enter the characters directly, you will instead type the ASCII codes separated by commas or spaces. Thus, to place an Epson/IBM printer into enhanced mode with double strike you would enter:

```
Init: 27, 69, 27, 71
```

And to initiate condensed printing you would use:

```
Init: 15
```

**Report Fields**

Once all of the preliminary steps have been taken, you are ready to specify which fields are to be included in the report. A number of options are available, and each will be discussed in turn.

A menu of all of the possible fields is presented, waiting for your selection. Once you have selected a field for the report, you may indicate that it is to be grouped, sorted forward or reverse, or manipulated using four-function math. Sorting can be performed on either one or two key fields. For example, you may wish to sort all of your transactions first by date, and then by account number if several are on the same date.

**Groups**

Grouping is similar to sorting, except that repeated items are printed only once. Grouping also allows you to obtain totals within a group. As an example, you may need to get a list of all payments you are owed, broken down by customer. If the customer account number is grouped, then each separate outstanding bill will be listed, along with the total for each customer individually. The sample time/billing application shows this in detail, in the BALDUE.RPT report definition.

**Report Math**

Besides merely listing fields, you may also perform simple four-function math. Math is permitted between a field and a constant, or between two fields. This provides capabilities similar to calculated fields, for example adding a subtotal and a tax amount to obtain a single value.

**Other Report Options**

If *Other* is selected from the fields menu, you are presented with an additional list of reporting options. You may request a field from a related file, the total for a selected field, or a total for a related field.

**Related Fields**

Related fields are identified by first providing the name of the field in the main file that determines the relationship, followed by the field in the related file. In the BALDUE report, the report is processing the receivables file, but you also need to know the name and address of the customer who owes the money. Therefore, that information is obtained through the relation that has already been established by the account number field:

```
Related ACCOUNT NAME
Related ACCOUNT PHONE
```

**Total Fields**

If Total is selected, then the total for all the fields in the selected records will be printed at the end of the report. ~~If you specify Total Related, then all of the matching fields in the related file will be added together, and included along with any other totals.~~ IF YOU HAS GROUPING SUBTOTALS ALSO APPEAR FOR EACH...

**Today, Now, and Page Number**

Today and Now will list the current date and/or time on the report, which will be placed in the report header at the top of each page. Page Number instead prints the current page number. If you want these items to appear, though, they *must* be included in the list of report fields.

**Form Fields**

If a data entry form is being used as a selection criteria, you may want to also include that information in the report header. For example, EXPENSES.RPT lists all expenses paid out by category for a given period of time. In that report, the period being reported on is printed at the top of each page like this:

Expenses report for the period _____ to _____

Where the beginning and ending dates were actually on the DATES data entry form.

### Chain Report

Often, several operations may need to be performed successively. For example, before deleting a batch of records, you would probably want to keep a permanent printed listing for your files. Chain allows you to specify the name of a report that is to be run immediately after the current report finishes. There is no limit to the number of reports that may be consecutively run, however the Chain command must be the last item in the report.

### Editing

Once the report has been defined, you may make changes or correct errors by selecting Edit from the main menu. The editor supplied with QBase Report is a full screen editor that supports vertical and horizontal scrolling. Most reports that you create will probably be eighty columns across or less, though you have the capability to produce reports of any width.

Editing is also useful for making changes to the report appearance. (See *Report Formatting* below.)

### Load and Save

Load and Save let you edit an existing report, modify it, and then resave it under the same name or a new name. Reports are not run from within Assist, rather they are only created and edited there.

**Auto Format**

One of the most tedious parts of report creation has always been defining the layout and appearance. Rather than require you to manually position each field, Auto Format provides an initial level of formatting. You are then free to use the full-screen editor to center titles, adjust margins, etc.

Auto Format uses the format type indicated in the report definition to determine whether to place the fields into column or row order. You are free to alter the position of any report fields, as long as you follow the formatting guidelines given below.

In order to allow auto-formatting of a report at a later date, the length of each report field is appended to the end of the field's name. Once Assist has formatted a report, the length information is no longer needed, and it is not necessary for you to change the "Len =" statements if you alter the field length on the screen. You might want to change a field's length on the report to print only the first few characters of a long field, or to allow extra digits in a total field.

**Report Formatting**

Most reports will be formatted in either a column arrangement, or in rows. Formatting in columns results in a "spreadsheet" style layout, while row formatting would be used when printing mailing labels. Assist will automatically create either format for you, though you may also make certain changes.

For example, mailing labels will be essentially in a row format, but you'll probably want to put the City, State, and Zip all on one line. The only requirement that you *must* observe is that the fields have to be in the same order in which they were originally specified.

A typical mailing label report might begin like this:

```
Form: CUSTOMER
Criteria: BALDUE > 100
Type: Lpt1:
Fields: NAME
COMPANY
STREET
CITY
STATE
ZIP
.Enddef
```

Then once Assist has formatted this into rows, the fields will be placed each on a separate line:

```
_____________________________________
_____________________________________
_____________________________________
_____________________________
____
__________
```

You are free to move the State and Zip fields to the same line as the City field, but the state must follow the city, which must in turn be followed by the zip code. If you want to alter the order in which fields appear, you must also change the order in which they have been selected.

**Print Report**

This menu option merely sends a printed listing of the report definition to the printer connected as LPT1. Comments in the program listing show how to enable condensed printing (for very wide reports) for both Epson/IBM and HP LaserJet printers.

**QBase Report—the main program**

QBase Report is invoked from the DOS command line by entering:

QRPT [*reportname*]

Where *reportname* is an optional parameter indicating the name of a report to load and run automatically. You do not need to include an extension, as long as the report name uses the normal default extension of .RPT.

If a report name is not given, Report will display a menu of existing reports within the current directory. Either select the report to be run, or press Escape to enter the name manually. You will need to enter the name only if the report does not have an .RPT extension, or if it is located in another directory.

**Calling Report from QBase**

Most programmers will probably prefer to invoke reports directly from the QBase main menu. This is not difficult to do, however you will need to modify QBase and recompile it.

In QBase, the main pulldown menu subprogram returns two values when a selection has been made. One is the menu number that was active, and the other is the choice selected within that menu. QBase uses On/Goto to branch to the portion of the code that handles a particular menu, and (in the case of the File menu) another On/Goto to execute the chosen selection.

The simplest way to invoke report from within QBase is to add a label such as Report: to the QBase program, branch there when "Report" is selected, and finally add code to run QRPT. As shipped, QBase contains the following branch instructions from the main menu, just under the label Main.Menu:

```
On Menu Goto _
   FileMen, Main.Menu, Quit
```

Since "Report" is the second item on the QBase main menu, selecting that choice simply branches back to the main menu, thereby ignoring the selection. Thus, you should add the new "Report" label and Run command before the "Quit" label:

```
.
.
.
Report:
    Run "QRPT"


Quit:
    Color 7, 0
    Cls
.
.
.
```

QBase Report ends by running QBase, so you don't need to do anything else to get back again.

If you would like to be able to put a list of reports on the QBase Report menu and run them from QBase, a few additional steps will be required.

First you must add the report names that are to appear on the Report menu to the menu's data statements that appear earlier in the program. As shipped, the Report menu has a title only—which ends up in the array's zero element—followed by a group of null data items. Simply add the appropriate report names, and when QBase branches to your new Report label, the variable Choice will contain the selection number that was made from the report menu.

**Passing a command line argument**

The only problem remaining is that BASIC provides no method for passing command line arguments to a subsequent program. It would be great if we could do something like this:

```
Run "QRPT Report1"
```

Where Report1 is the name of the report to run. Unfortunately, this simply doesn't work. While you could always create a file containing the name of the report to run, and then modify QRPT to read that file and obtain the name, it's a messy solution at best.

Instead, we have provided a BASIC subprogram named StuffBuf that will place a DOS command directly into the keyboard buffer before QBase is exited, allowing you to run Report with a command line argument automatically.

Simply $Include STUFFBUF.BAS anywhere in the main QBase program, and then call it with a string argument representing the name of the program to run, plus the command line argument. Then execute an End instruction. The code fragment below shows how to do this.

```
Report:
    Arg$ = "QRPT " + MMenu$(Choice, Menu)
    Call StuffBuf(Arg$)
    End
```

MMenu$ is a two-dimensional array that holds all of the main menu choices, thus the report name will be passed along to Report. If you plan to word your menus with descriptive terms rather than report names, you will probably want to create a parallel array containing the actual report names, and pass those instead. Then you could add something like this earlier in the program:

```
NumReports = 4                  'how many reports there are
Dim Report$(NumReports)         'make array to hold each name
For X = 1 To NumReports         'read them into the array
    Read Report$(X)
Next
Data "Report1", "Sales", "Inventry", "Report4"
```

At the report menu you would specify which report was run by using the Report$() array, rather than the main menu titles:

```
Report:
    Arg$ = "QRPT " + Report$(Choice) + CHR$(13)
    Call StuffBuf(Arg$)
    End
```

**Time & Billing Application**

Included with QBase Report is a complete time and billing application intended for use by computer professionals. Seven separate screens are provided, with one each for your client information, billings, payments, and expenses. Two additional screens are used to hold the next available invoice number, which is incremented automatically each time you send a bill, and the expenses category multiple choice field. (NEXTNUM and CATEGORY respectively.) The last screen contains two date fields, which allows you to specify a starting and ending date on the reports.

Only one small modification must be made to QBase to accommodate the billing form, which simply calculates a total balance derived from the number of hours entered, and sales tax if appropriate. An $Include file containing the necessary calculations is also on the QBase Report disk, named CALC.BAS.

To add the code in CALC.BAS to QBase, load QBASE.BAS into the QuickBASIC editor (QBASE2.BAS in the Turbo Basic version), and locate the line labeled ExitCodes. Immediately below that line is the statement:

```
If FChanged Then Changed = 1
```

Just below that line add the $Include meta statement:

```
' $Include: 'CALC'
```

Once QBase has been recompiled, total hourly calculations will be performed automatically whenever billing information is entered.

**Clients**

The Clients form is used to hold the name and address of each of your customers, along with their hourly and tax rate. By using a field to hold the hourly charges, you may have different billing rates for different clients.

The Account field is intended to be used to keep track of each client, and must be different for each entry. That is, the first client you enter should have, say, the number 1, the second will be number 2, and so forth. Once you have entered all of your clients this way, QBase can relate the various billing and payment information you enter to the customers they belong to.

**Billing and Payments**

The Billing form is where you will enter the services you have performed. To create billing information, simply enter the client number, the number of hours worked, and a brief description that is to appear on the bill. When an invoice is payed, you should marked the "Paid Y/N" field "Y". The payments form is to record payments received from each client to simplify obtaining income totals at year's end.

**Expenses**

The expenses form is not really related to the customer, billing, or payment information, but you'll find it useful on April 15. Fourteen different payment categories have been established (which are contained in the Category screen), though you may of course modify them to suit your own individual needs.

By entering each expense item as it is payed, you will have instant access to a complete breakdown with the Expenses report.

**Time/Billing Reports**

Six separate reports are included with the Time/Billing application. Clients simply lists each client by number, Billing and Payments provide a detailed report on money billed and payments received, Invoice sends an invoice to the selected client, and Expenses presents all of your expenses.

## QBase Report Utilities

In addition to the Main QBase Report module, three separate stand-alone utility programs are also provided. Import lets you bring external data in standard ASCII form into the QBase environment, MSortQB will rearrange a QBase data file into sorted order, and MSort is intended for sorting *any* BASIC random file. A complete description of each utility follows.

## Import

The QBase Import facility is intended for importing standard ASCII comma-delimited data into an existing QBase database. This format was chosen because it is the least common denominator for all database, spreadsheet, and word processor programs.

The incoming information for each record should be contained all on the same line, with a comma used to separate each field. You are not required to place quotes around string data, unless the strings themselves contain commas, colons, or semicolons.

The entire import process is fully menu driven, however a few ground rules should be observed. First, you must have already defined fields in the database where the imported data is to go. Second, imported data cannot be used to update information in existing records—that is what the Modify report type is for. When data is being imported into a QBase database, new records are always appended to the end of any existing data.

Third, the data being brought into the QBase environment must already be in the correct field order, though you don't have to account for every field in the existing database. (More on that in a moment.) Finally, you should not attempt to import data for more fields than actually exist in the file being added to. That is, if you have a form containing, say, twenty fields, then each line of the incoming data must contain twenty items or less.

Rather than belabor a verbal description of the import data format, several examples are given below. This example shows a few lines from a typical ASCII data file to be imported:

```
Smith, John, M, 34, 80286 Maim Street, Dallas, TX, 75231
Rogers, Joanne, F, 28, 15 Spittoon Blvd., Memphis, TN 38119
Gilbey, Oliver, M, 64, 142 Oak Place, New Shmaltz, NY, 12561
```

Where "Smith" will be placed in the first field, "John" will go in the second field, and so forth. If there is any possibility that the incoming data will contain any of BASIC's reserved punctuation characters, then those fields must use surrounding quotes:

```
"Jones, Robert P.", 1200.43, "Hawthorne, MN", 55117
```

Notice that all of the fields in the database do not need to be accounted for. That is, if the QBase database form contains 27 fields but the incoming data only includes the first 15, the data will still be imported correctly. However, this means that each line of imported data must be identical.

One of the first things Import does is to count the number of items in the first line of the incoming file. It then uses that count for all subsequent lines. Notice that extra commas may be used to skip over fields that are not to be filled, when other subsequent fields will be:

```
Anderson, John,,,, 100.98,, Sales,,,,,, 56
```

There are very few restrictions on the type of data that may be imported, and when possible, type conversion will be performed automatically. For example, an incoming date in MM/DD/YY form will be converted to the special QBase date format, and a digit intended for a choice field will be converted to an integer choice number. Any remaining fields in the QBase form that are not included in the imported file will be automatically set to a null value.

**MSortQB**

Unlike the normal QBase reports where a file may be reported on in sorted order without re-arranging the data, MSortQB is used to create a new file, sorted in any new arrangement you would like. MSortQB uses a Merge Sort technique, which allows an unlimited amount of data to be sorted, regardless of BASIC's string space limitations.

You tell MSortQB what file is to be sorted, and how, in one of two ways. If there is no command line parameter given when you start the program, you will be prompted for the type of sort to be performed, the QBase data file name, which fields are to be used for the sort keys, and whether to sort ascending or descending. If you intend to sort the same file the same way fairly often, you may instead place the sort parameters in a *response* file, and specify that file when you start MSortQB:

```
MSortQB [FileName]
```

These parameters should be either comma delimited, or with each contained on a new line:

```
SortType, DatFile, Field1, Dir1 [, Field2, Dir2]
```

SortType is either the letter "S" to create a new sorted file, or the letter "I" to create a BLoadable integer index array of sorted records. If you specify the "S" parameter, a new file will be created from the original one, but with an .SRT extension. It is then up to you to delete the original file, and rename the sorted file with a .DAT extension. Comments near the end of the program listing show how to have this done for you automatically if you prefer.

If the "I" parameter is used, an index array is created and saved to disk, instead of a new data file. This allows access to the original data in sorted order, but without actually creating a new file. Comments at the beginning of the program explain how to BLoad this array, and access the file based on that index.

DatFile is the name of a QBase data file either with or without the .DAT extension, Field1 is the field number for the primary sort, and Dir1 is the sort direction. (0 = ascending, and 1 = descending). The Field2 and Dir2 parameters are optional and may be omitted. Be sure to specify the key fields by their *number*, and not by their name.

It is important to understand that if a new data file is created, you must erase any corresponding QBase index files. (Index files are given the name of the *field* they are associated with, but with an .NDX extension.) Since the data is now in a different order, the entries in the index file will no longer be valid. However, each time QBase is started, it looks for all of the index files that it needs. If they are missing or the wrong size, then new indexes are created automatically.

### MSort

MSort is very much like MSortQB, except it will sort *any* random access file, not just those that are used by QBase. This means that you must specify additional information, since the program has no way to determine the record length, field layout, and so forth.

Like MSortQB, MSort is invoked either with or without the name of a response file on the DOS command line. If the file name is omitted, you will be prompted for the information, which is as follows:

```
InFile, OutFile, RecLen, Fld1, Len1, Dir1 [,Fld2, Len2, Dir2]
```

InFile is the full name of the file to sort, and OutFile is the name of the new sorted file—there is no option to create a sorted index array. RecLen is the length of each record, Fld1 is a number indicating where in the record the first sort key begins, Len1 is the length of that field, and Dir1 is either 0 or 1 as above. Fld2, Len2, and Dir2 are optional, and allow sorting on a secondary key if needed.

Be aware that MSort assumes that each of the sort key fields are normal ASCII strings. Where MSortQB knows what type of fields are being sorted and can process them accordingly, MSort can make no such assumptions. For example, if one of the key fields happens to be an MKI$() integer, then negative numbers will be considered as *greater* than 32767.

**QBase Report Modules**

QBase Report is comprised of a main *shell* program that interprets your report and format instructions, and then calls upon a number of lower level *primitives* to do the actual record selection, sorting, and so forth. These low level routines may be accessed separately if needed, for example to create special reports that go beyond the capabilities of the main reporting program.

This section describes each of the subprograms, what they do, and the type of input and output each performs. A demonstration program is included for many of these routines, to illustrate the steps you must perform when calling them, and any mandatory arrays or other setup requirements. Notice that all of the routines that manipulate a file uses a file number greater than 50, so they can be freely called without fear of a conflict with files you may already have open.

All of the assembler routines that are used by the QuickBASIC version of QBase Report are contained in two separate library files. QBR.EXE is intended to be loaded into the QuickBASIC editing environment during development, and QBR.LIB is meant for use with LINK once your programs are complete and ready to be compiled to disk.

The Turbo Basic assembler routines are each in a separate .COM file, and can be used in both the editor and your final .EXE programs. Turbo Basic handles assembler routines differently from QuickBASIC, and instead uses .COM files that are included as $Inline procedures.

**AddRecs**

AddRecs will retrieve selected information from one QBase data file, and append it as new records to the end of another data file. It is called by specifying the "From" file, the "To" file, an array of record numbers, and the number of records in that array. The To and From file names do not require the .DAT extensions, which will in fact be stripped off if they are included. The array of record numbers indicates which records in the From file will be used as the source of data appended to the To file, and in what order.

A second array is also expected, to indicate which fields have been selected from the source file, and into which fields in the destination file they are to go. The Selected Fields array must be dimensioned to the number of fields in the From file, and each selected element will contain the target field number.

As an example, imagine there are ten fields in the From file, and fields three and four are to be entered into fields five and six in the To file. In this case, all of the elements in the array must be set to zero, *except* for elements three and four. Element three will then contain the number 5, while element four will hold the number 6.

Type conversions between forms will be performed automatically (within reason), allowing you to copy, say, a single precision field over to a currency field. Some error checking is also performed—if you attempt to place too large or small a number into an integer or date field, the request will be ignored. Any fields in the new records being created that are not filled will be set to a null or zero.

**Browse**

This is the core of the QBase Report Browse facility. It handles all aspects of stepping through a file forward or backward, and also allows scrolling horizontally to view more fields than will fit across the display screen.

Browse is called with the name of the file to view, an array of record numbers to include, and an array of field numbers to be displayed. The field numbers do not need to be in sequential order, rather the order dictates how the fields will be arranged on the screen.

**Delete**

Delete is a very simple module that merely marks all of the indicated records as being deleted. It is called by specifying the name of the file, an array of selected record numbers, and the total number of records in the array. Notice that when a record is deleted in a QBase data file, an exclamation point (!) is placed into the first byte in the record. The only way to reclaim the space wasted by deleted records in a file is to use the QBase Rebuild program.

**FindMtch**

FindMtch is used to select certain records in a file, based on one or more matching fields. It is called with the name of the QBase data file, and two arrays that have been filled with the selection criteria. Two arrays are needed to handle both string and numeric field types, for each of two match tables. (The second table handles the other side of any OR criteria.)

The numeric arrays are double precision, since that type can be used to represent any numeric field. It is also a two-dimensional array, to allow specifying both upper and lower range limits. Regardless of which type of fields the selection is to be based on, you must assign both the string and numeric array elements for each field being examined.

If the matching is to be based on a string, then simply assign the search string or sub-string into the array element that corresponds to the field being examined. Then put the value 1 into the corresponding numeric array. The asterisk (*) wild-card is supported, and in fact the match criteria is identical to that used when searching within the QBase database.

If the field is numeric, then the lower limit to search is placed into the first subscript, and the higher limit is assigned to the second subscript. But remember to put something—anything—into the corresponding string array element. If an exact match is needed, then put the value into both the first and second array subscripts.

When only an upper or lower limit is being used for numeric matching (ie: <=10 or =>2.8) FindMtch expects a particular very large or small number to be used as that limit. For integers the limits are simply 32767 and -32768, though for single and double precision values the limits are 999999999999# and -999999999999#.

Because this scheme is not immediately obvious (though it is very efficient), several examples are given below.

Search for all occurrences of "Smith" in the CUSTOMER file:

```
Nam$ = "CUSTOMER"
NumFields = 10
NumCrit = 1                          'not using an OR
ReDim Match$(NumFields)
ReDim Match#(NumFields, 1)
Match$(3) = "Smith"                  'field 3 is the name field
Match#(3, 0) = 1                     'must also set this element
Call FindMtch(Nam$, NumCrit, Match$(), Match#())
```

Search for all ID numbers between 10 and 50 in the CUSTOMER file:

```
Nam$ = "CUSTOMER"
NumFields = 10
NumCrit = 1
ReDim Match$(NumFields)
ReDim Match#(NumFields, 1)
Match$(1) = " "                      'must be non-null
Match#(1, 0) = 10                    'the lower limit
Match#(1, 1) = 50                    'the upper limit
Call FindMtch(Nam$, NumCrit, Match$(), Match#())
```

Search for records in the BOAT database with ID number <= 100, BALANCE => $250.00, and COVER = "Y":

```
Nam$ = "BOAT"
NumFields = 36
NumCrit = 1
Dim Match$(NumFields)
Dim Match#(NumFields, 1)
Match$(1)  = " "                    'must be non-null
Match#(1, 0) = -32768               'the lower limit
Match#(1, 1) = 100                  'the upper limit
Match$(27) = " "                    'must be non-null
Match#(27, 0) = 250#                'the lower limit
Match#(27, 1) = 999999999999#       'the upper limit
Match$(31) = "Y"                    'the string to match
Match#(31, 0) = 1                   'must set this to 1
Call FindMtch(Nam$, NumCrit, Match$(), Match#())
```

One final note is that FindMtch can be used to select *all* of the records in a data file by merely calling it with null arrays. If each element in the Match#() array is zero, and each Match$() element is a null, FindMtch will select every record except those that are deleted.

## ISorts

Because QBase Report must be able to manipulate many thousands of bytes worth of data, some reports will require sorting to be performed using multiple passes. Further, some means is needed to tie the sorted field information to the original record numbers. Therefore, the main sorting module sorts an *index* into the data file, rather than exchanging the actual data. This is where the ISort routines come into play.

There are four different versions of ISort—one for strings, and one each for integers, single, and double precision numeric fields. Each version of ISort uses two arrays to perform the sorting, with one containing the data, and the other containing an integer index array that is actually manipulated. ISort3 is designed to sort string data, while ISortI, ISortS, and ISortD sort integer, single, and double precision values respectively.

All of the ISort routines are in a single file named ISORTS.BAS. In the QuickBASIC version of QBase Report, ISort3 is written in assembly language and is contained in the QBR library files. All four versions of ISort use the Quick Sort algorithm, which is the fastest method currently known.

Each of the BASIC ISort versions is called with five parameters. The first parameter is a string array that contains the values in MKI$, MKS$, or MKD$ form. The next two parameters indicate where in the array to begin sorting, and how many elements to include. The fourth parameter is the direction to sort, with 0 meaning ascending and 1 meaning descending. The final parameter tells ISort whether or not to initialize the parallel index array.

When a primary key field is being sorted, ISort must first set all of the elements in the index array to sequential numbers. Then as the data is examined, the index elements are sorted to be a table of pointers into the data. But when ISort is sorting the *second* key field, the original numbers are important, and must not be initialized again.

### Modify

Modify is the heart of the QBase Modify report type. It is used to modify information in a QBase data file, by replacing the current contents of a field with new information. This new information may be derived in one of several ways.

The new field contents can be either a fixed value or string, or a computed value. If a string or numeric constant is used, then the original field value is simply replaced. If the new value is to be computed, then simple four-function math may be performed.

The field calculations may be based on either a numeric value, or the contents of a second field. Only numeric fields may be used for computations, since attempting to, say, multiply one string by another would be pointless. The only exception is numeric strings, which may be used in a calculation.

A number of parameters are required to be passed to Modify as shown below, and each will be described in turn.

```
Call Modify(FileName$, SelRecs(), NumRecs, SelFields(), Operator(), _
     Value$(), Value#(), MathFields())
```

FileName$ is the file to be modified, SelRecs() is an array of selected record numbers that tell Modify which records to alter, and NumRecs is the number of records in that array. SelFields() contains the table of fields to be modified, while Operator() holds the type of math operations to be performed.

Each element in the Operator() array can hold one of five possible values. A 1 means the field is to be added, 2 indicates subtraction, 3 multiplies, and 4 means divide. Any other value tells Modify that the original field contents are to be replaced.

Value$() and Value#() contain the replacement strings and replacement/math values to be used, and the MathFields() array indicates which field is to be used for calculations when appropriate.

### MSortA and MSort

MSortA is really just a "front-end" to the main MSort sorting routine, and it is responsible for figuring out where the various key fields are located in each record.

MSort is the key sorting module, and is also the basis for the stand-alone MSort utilities included with QBase report. This is a fairly complex routine, and the comments throughout the program listing explain it much better than a written description here ever could.

**RelType**

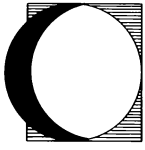
All of the information about each QBase database field is stored in a field definition file with an .FRM extension. Whenever one of the QBase Report modules needs to obtain a field's type, it is a simple matter of loading the form definition and examining it.

Relational fields, however, assume the same type as the field from which they are derived in the *related* form. Therefore, RelType is needed to determine which file and field contains the actual type for each related field. It then loads the related file's form definition and gets the type.

RelType is called with a field number, an integer variable that is to receive the type, and an error flag. Though QBase Report checks to be sure that all of the necessary files are present before beginning a report, it would be too much trouble to also search for every possible *related* file in advance. Thus, the error flag will be set if RelType is unable to locate the necessary related .FRM form definition file.

**Totals**

The Totals subprogram will read through a selected group of records, and obtain the total value for one or more fields by adding. Totals is called with the name of the file to be processed, an array of selected record numbers, and the number of records contained in that array. Two additional arrays are used to indicate which fields are to be totaled, and to hold the final results. The array that receives the totals is double precision.



**CRESCENT SOFTWARE**
64 Fort Point Street, East Norwalk, CT 06855
(203) 846-2500