Engine Test Data Collation and Reporting Database Package developed **For BICERI Ltd** (abridged)

Written by: Nigel Kitcher

Table of Contents

Page

Table of Figures

Page

1. Project specification and technical overview of solution

1.1 Project Overview

The Report Generator System had been attempted as a solution at BICERI on a number of different occasions. The most successful that being a system that ran on a Perkin-Elmer mini computer in the 1980s. This system was designed specifically for the needs of a single large OEM customer. With the move to the Lubricant and Fuels market in the late 1980s, together with the introduction of the "cheap" IBM PC, an alternative was required for these changes.

In 1994 the solution presented in here in summary was initiated, and within 2 years became the corporate system for producing all Lubricant and Fuels test reports, and finally became the hub of all the company databases held on the network.

1.2 Objectives

- **To de-skill the job of report production**. This enabled the engineers to be do what they were supposed to be doing i.e. engineering. It also permitted the spreading of work load as a report could easy be produced by anyone and only required the engineer for final approval.
- **To produce consistent documentation.** Reports were visually inconsistent between engineers and even a single engineers reports in the early days as there were no common application usage with engineers using whichever package they had become used to using.
- **To reduce errors in reports.** As each engineer has developed his own spreadsheets and macros to produce a report errors have arisen due to the lack of information when it is necessary for someone to produce a report (e.g. where macros don't quite work properly or need a special set up prior to running)
- **To provide information to those who require it.** Biceri made a living not by running engines but by producing information. This information is frequently lacking in areas of the company where was needed. For example the Sales team did not have access to any statistical data arising from tests as there was no central repository of data.
- **To meet the objectives as defined in the Quality procedures**. Reports did not meet NAMAS standards: no issue number and issue date on every page, rating methods not defined to a standard, no calibration dates for test equipment etc. With a vast array of report "templates" amending each and every one would have been an monumental task.

1.3 Technical overview of solution

The core to system is several ODBC database, which were implemented at the time using Access. Other database backends were considered including Dbase and SuperBase. The decision to implement using Access was made by the fact that it was considered easier to “upsize” to SQL Server when it became appropriate. The availability of ODBC was also a consideration as the MS Office suite, plus third party packages had to be able to communicate with the system.

Job numbers are accessed by the system via linked tables, which exist in the accounts package. A number of these tables are used by the timesheet application which a Visual Basic application available on all company PCs and is used by all staff.

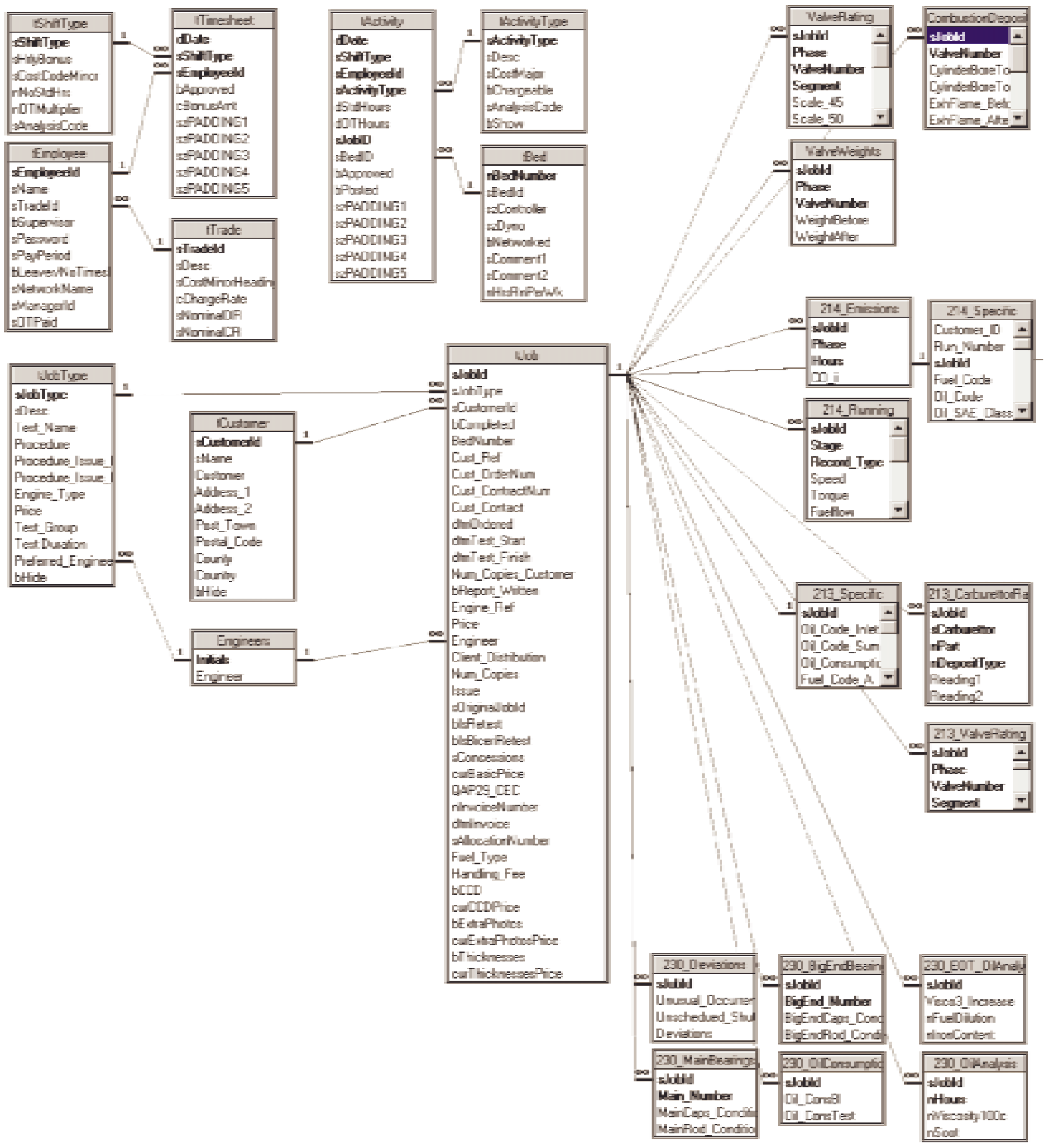
The creation of a job on the system results in the creation of many records through out many tables. Population of data within these records comes from a wide range of sources including automated engine test bed controllers and loggers, smoke meters and manual raters.

To create the customer report a suite of WordBASIC routines connect to the data source, retrieve the data relevant to the requested job, and populate the appropriate Word template with the data.

An engineer simply entering the job number when prompted can produce a report. The rest of the process is fully automated. Reports can be produced at any time during a test and the engineer is prompted if the report already exists so that “refreshing” of a partially completed report is possible.

The functionality of the Word File Save routines has been extended to include “Intelligent Documents” so that any documents produced by a user is automatically placed in the correct folder in the directory structure designed around the format of the job number. This eases use of Word and also ensures that documents are not misfiled in the wrong folder.

Figure 1 - Part of the Report Generator database Schema



2. User Manual

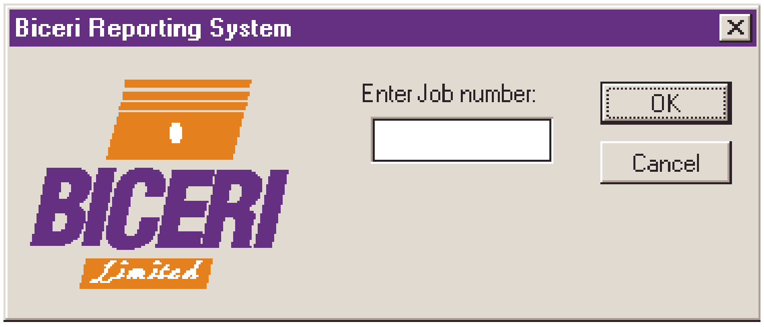
2.1 Building a report

All jobs are entered into the Access database. At this point the directory structure is created as shown in section on page 7-1.

The directory structure is used to check whether a job has been entered into the system at the initial point of report creation in Word (for performance reasons). However, if the directories have been created, illegally, by hand the Word code will alert the user when it can not find the appropriate data in the database tables.

The job number entry dialog box performs many validations and permits the job number to be entered in upper or lower case (converting it to upper if necessary), and the run number can be entered as one, two or three digits i.e. run number 3 may be entered as 3, 03, or 003. The run number is padded out to three numerics in all cases. The dividing character between the test Id and the run number may be any non-alphanumeric i.e. it will accept "/", "-", etc.

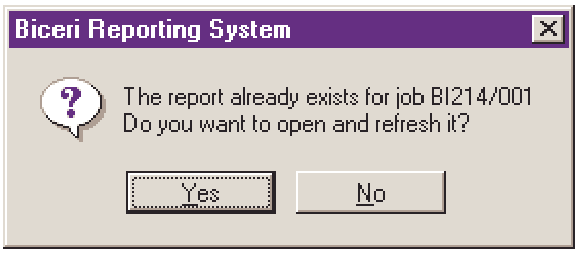
Figure 2 - Job Number Entry Dialog Box



2.2 Handling a previously generated report

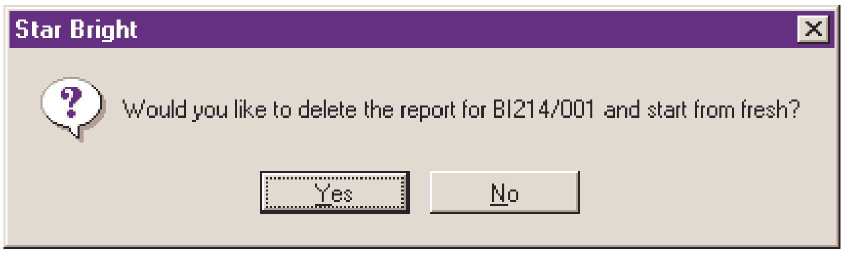
If a report already exists for the requested job number the dialog shown in Figure 3 appears. Selecting Yes will open the existing report and refresh ALL values in the report. Selecting No will present the user with the option of deleting the existing report and starting again (see Figure 4).

Figure 3 - Dialog when a report already exists



The report already exists for job BI214/001
Do you want to open and refresh it?

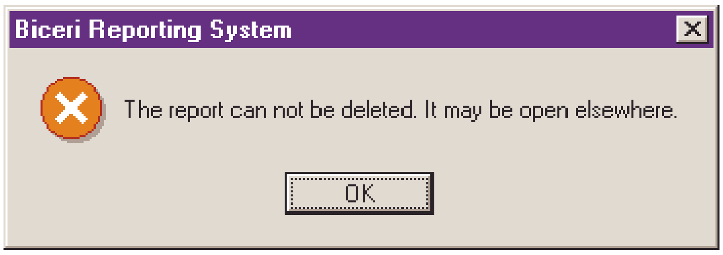
Figure 4 - Option to scratch report and start again



Would you like to delete the report for BI214/001 and start from fresh?

If the existing report can not be deleted, for example if it is already open some where else then the message box shown in Figure 5 appears. This aborts the reporting system.

Figure 5 - Failure to delete existing report

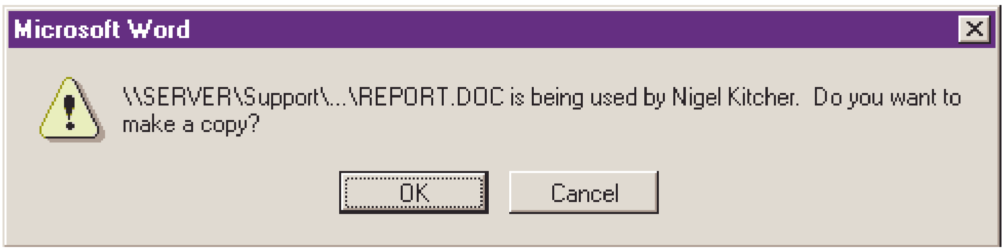

Biceri Reporting System

The report can not be deleted. It may be open elsewhere.

Note: Selecting YES at the first dialog allows the user to generate a subsequent issue of the report. Choosing NO at the first and YES at the second allows the user to re-generate an issue #1.

If the user selects to open an existing file, but the file can not be opened because it already opened, for editing, elsewhere the dialog shown in Figure 6 appears. If Cancel is chosen the reporting system ends.

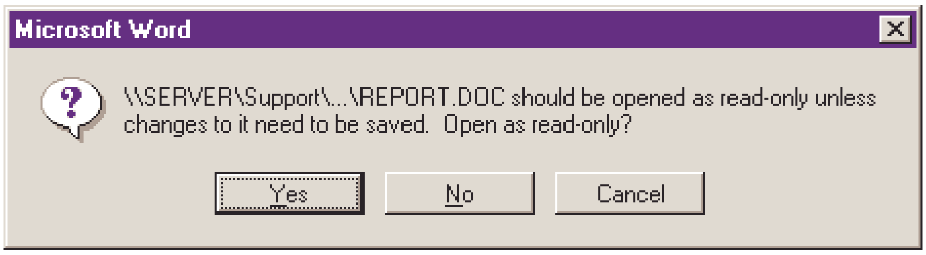
Figure 6 - Report has been opened elsewhere


Microsoft Word

\\SERVER\Support\...\REPORT.DOC is being used by Nigel Kitcher. Do you want to make a copy?

Note: Documents are saved as "Recommended Read Only". If a report is opened outside of the reporting system the dialog shown in Figure 7 appears. Editing a document directly may create problems later if the reporting system is used to refresh the document. You should always open report files in Read-Only mode to maintain integrity and allow other users access to the document.

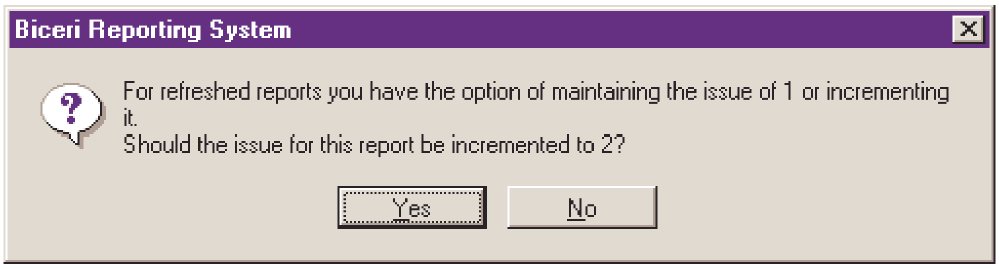
Figure 7 - Opening documents outside of the reporting system


Microsoft Word

\\SERVER\Support\...\REPORT.DOC should be opened as read-only unless changes to it need to be saved. Open as read-only?

If a report has previously been written and the user has chosen the option of opening the report and refreshing the data a dialog similar to Figure 8 appears offering the choice of maintaining the present issue number or automatically incrementing it.

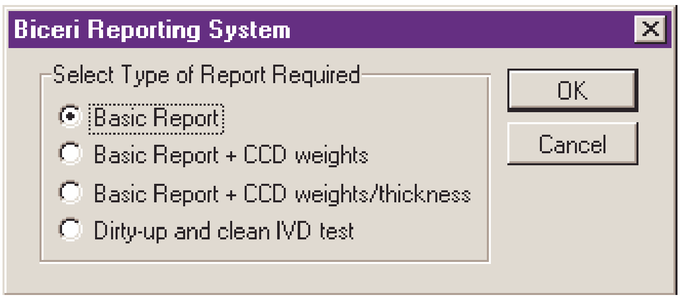
Figure 8 - Option to increment the issue number


Biceri Reporting System

For refreshed reports you have the option of maintaining the issue of 1 or incrementing it.
Should the issue for this report be incremented to 2?

As the Biceri test ID is not unique to an individual type of report the dialog box shown in Figure 9 will appear if there is more than one report format. In this case the dialog is prompting for reports for test type 214.

Figure 9 - Report type selection dialog box



Biceri Reporting System

Select Type of Report Required

(●) Basic Report
( ) Basic Report + CCD weights
( ) Basic Report + CCD weights/thickness
( ) Dirty-up and clean IVD test

Note: the text for this dialog box comes from the TEMPLATE.INI file - see section Error! Reference source not found.**.**

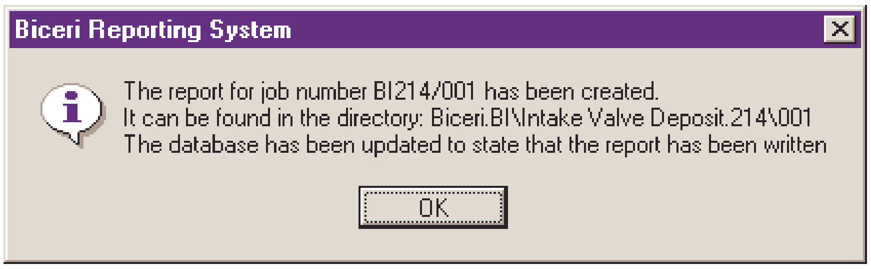
2.3 Report Building

Once the job number, and optionally the type of report, has been entered the process of generating the report is automatic. The status of the process can be seen at the bottom of the screen in the status bar and should proceed in a manner similar to the following (test dependant):

- ***Connecting to database…***
- ***Building SQL Select statement…***
- ***Waiting for database transaction...***
- ***Obtaining field names… retrieved (n)***
- ***Building Report... Fields Inserted (n)***
- ***Closing connection to database...***
- ***File … saved to disk***
- ***Done.***

When the report has been completed and the document saved in the correct directory the dialog box shown in Figure 10 will appear.

Figure 10 - Dialog box for completed report



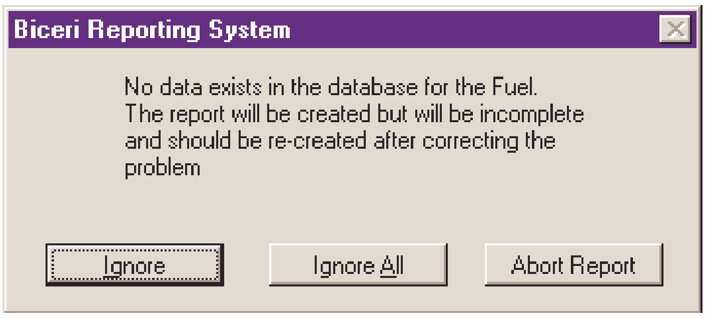
Biceri Reporting System

The report for job number BI214/001 has been created.
It can be found in the directory: Biceri.BI\Intake Valve Deposit.214\001
The database has been updated to state that the report has been written

2.4 Errors during Report building

During the report building phase a message box similar to one shown in Figure 11 will appear when an item of required information is missing. Choosing Ignore will just ignore the one item and still prompt for each subsequent missing item, whereas Ignore All will not prompt on subsequent missing information. Choosing Abort Report will halt report production.

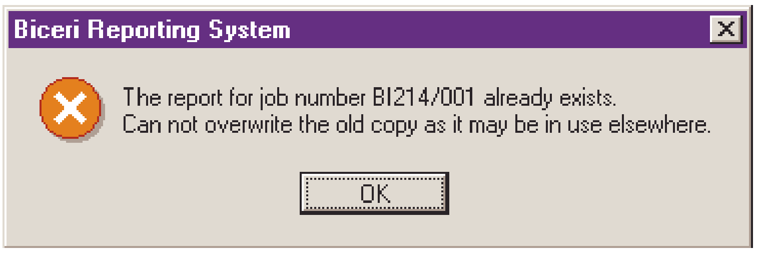
Figure 11 - Error message for missing information


Biceri Reporting System

No data exists in the database for the Fuel.
The report will be created but will be incomplete
and should be re-created after correcting the
problem

If a report is being recreated then the document will already exist in the correct directory. As the reporting system can not overwrite the file (it could be opened elsewhere on the network) the message box shown in Figure 12 will appear. If you know that it is safe to save the file then you may do so however this version will be outside of the reporting system.

Figure 12 - Error message when report already exists


Biceri Reporting System

The report for job number BI214/001 already exists.
Can not overwrite the old copy as it may be in use elsewhere.

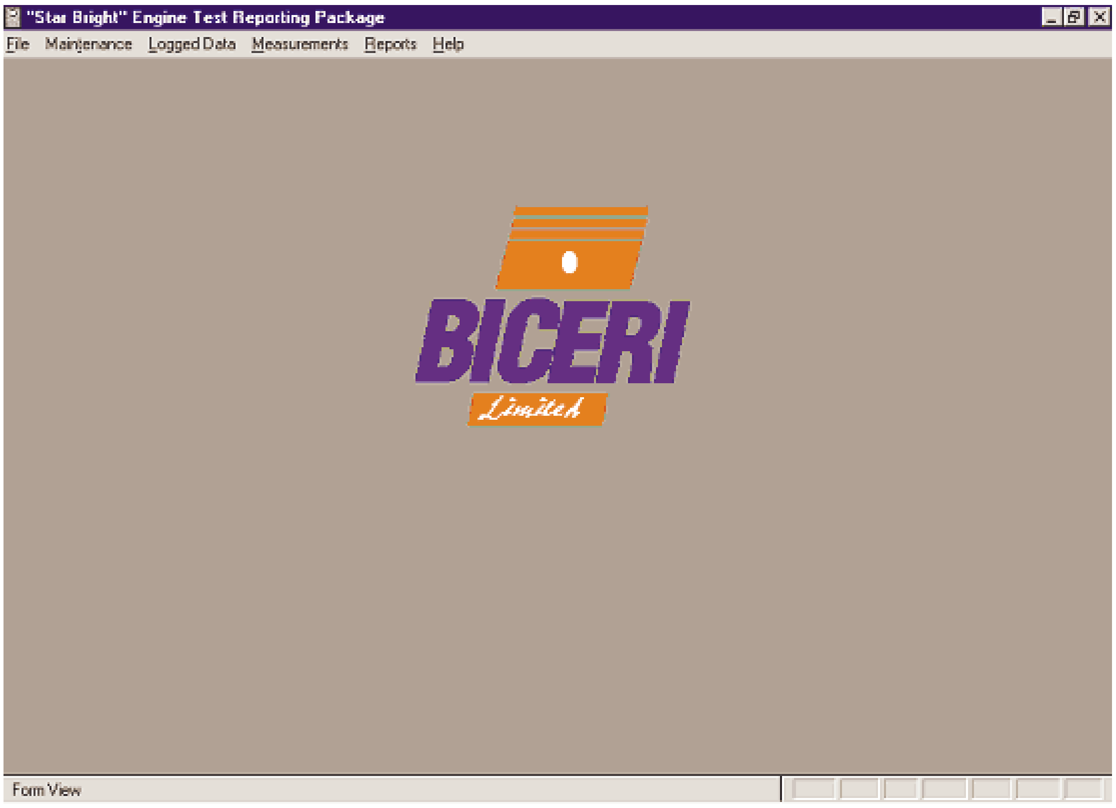
NOTE: Reports should not be recreated by simply choosing create report a second time. The report document should be opened and then create report should be chosen.

3. Using the Reporting System Database

3.1 Overview of Database

The terminology of the database is as follows: Tests are tests offered by the company, Jobs are specific instances of a test.

Figure 13 - "Star Bright" Main Database opening screen



3.2 Adding Jobs to the database

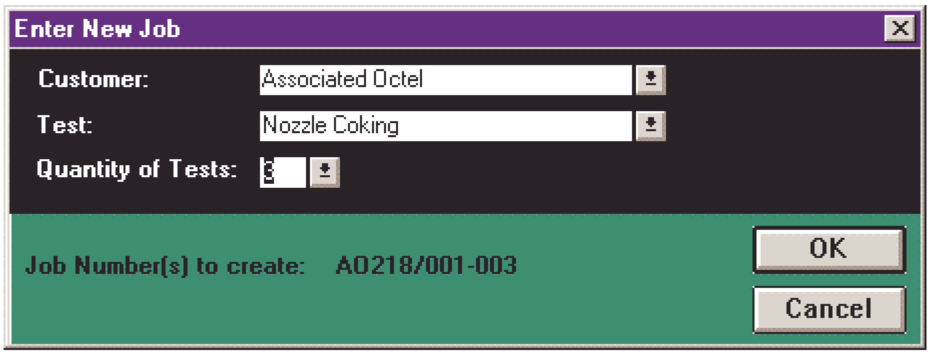
Jobs are added to the system via the “Add Job” menu option under “Maintenance”.

The user is then presented with the dialog shown in Figure 14 - The Enter New Job(s) Dialog. To enter a new job or jobs the customer and test type must be selected from the drop down lists. After these have been selected the job number that will be allocated is displayed at the bottom of the dialog box. If the user wishes to add more than one job then the “quantity of tests” should be changed in the edit box. In this case the range of job numbers to be allocated is then displayed.

Selecting “OK” will start the process of creating entries for the chosen job(s). Part of this process is the construction of the directory structure for the reporting system (see

Reporting Directory Structure on page 7-1).

Figure 14 - The Enter New Job(s) Dialog


Enter New Job

Customer: Associated Octel

Test: Nozzle Coking

Quantity of Tests: 3

Job Number(s) to create: AO218/001-003

3.3 *Entering/Amending Job Specific data*

At the time a job is created by the system a number of fields are automatically set to predetermined defaults as follows:

Bed	This is set the preferred bed for the given test type[1]
Engineer	Set the `Preferred_Engineer` entry as listed in the table Tests.
Order Date	Set to the current date (when job is created in system)

To ease inputting of dates into the system the user may use the CTRL+D key combination to insert the current date into a date field.

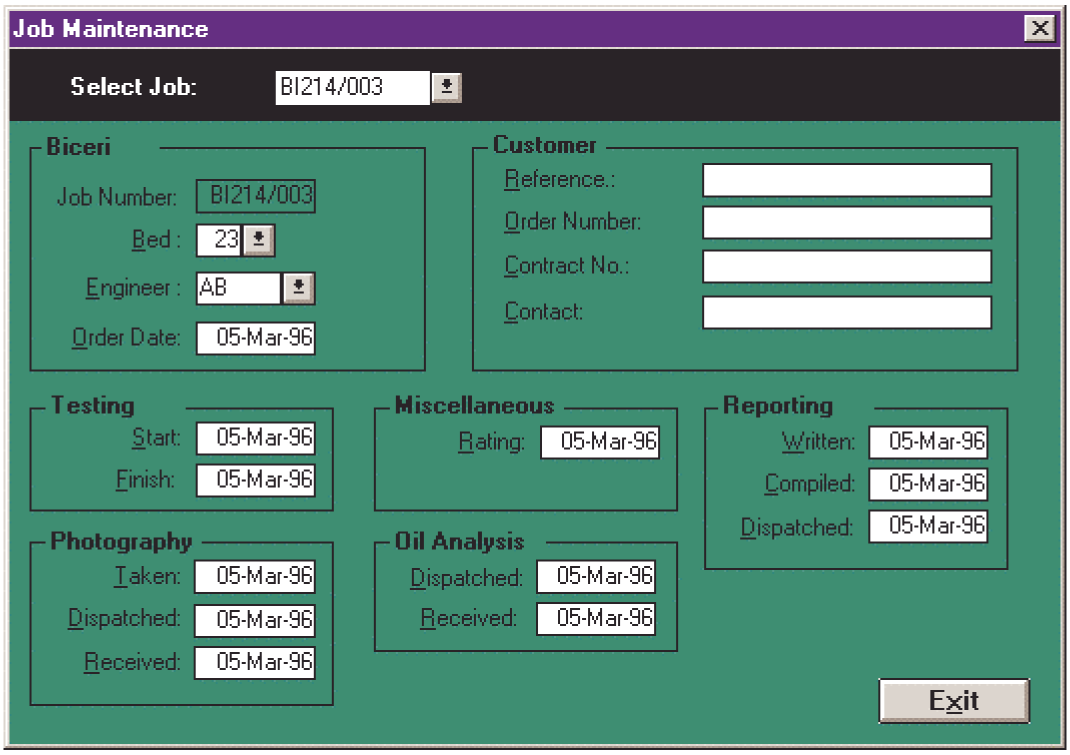
Note that the following information is automatically updated by other components in the system:

Reporting: Written	This date is updated by the system when a report has been created via the "Report Builder" in Microsoft Word (see 2.3 Report Building on page 2-3)
Reporting: Dispatch	This date is updated by the system when a covering letter is created via the "Letter Writer" in Microsoft Word (see 5

[1] This is held in the database table tBedTest. At the time of writing this table is not readily available to average user.

The Covering letter writing system on page 6-1)

Figure 15 - Customer and date information Dialog Box



Job Maintenance

Select Job: BI214/003

Biceri

Job Number:	BI214/003
Bed :	23
Engineer :	AB
Order Date:	05-Mar-96

Customer

Reference.:	
Order Number:	
Contract No.:	
Contact:	

Testing

Start:	05-Mar-96
Finish:	05-Mar-96

Miscellaneous

Rating:	05-Mar-96

Reporting

Written:	05-Mar-96
Compiled:	05-Mar-96
Dispatched:	05-Mar-96

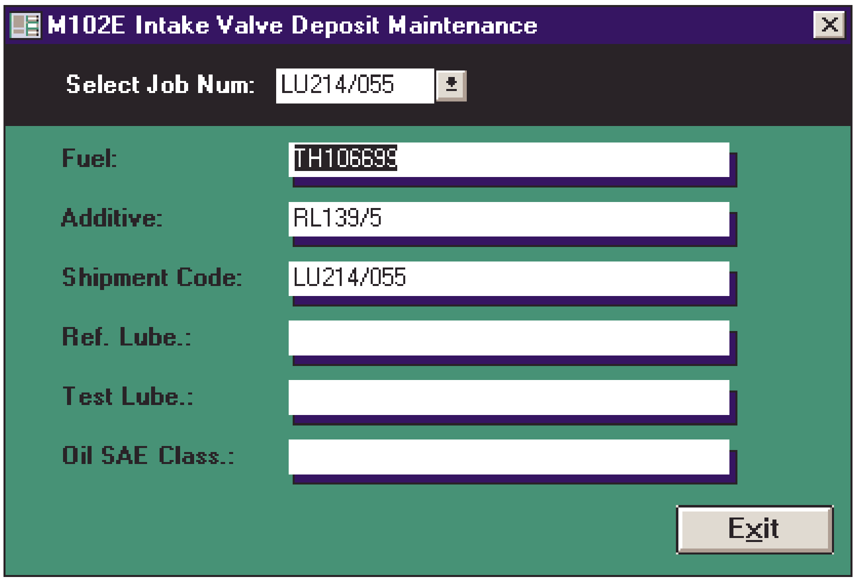
Photography

Taken:	05-Mar-96
Dispatched:	05-Mar-96
Received:	05-Mar-96

Oil Analysis

Dispatched:	05-Mar-96
Received:	05-Mar-96

Figure 16 - The Maintenance dialog for M102E IVD Jobs



M102E Intake Valve Deposit Maintenance

Select Job Num: LU214/055

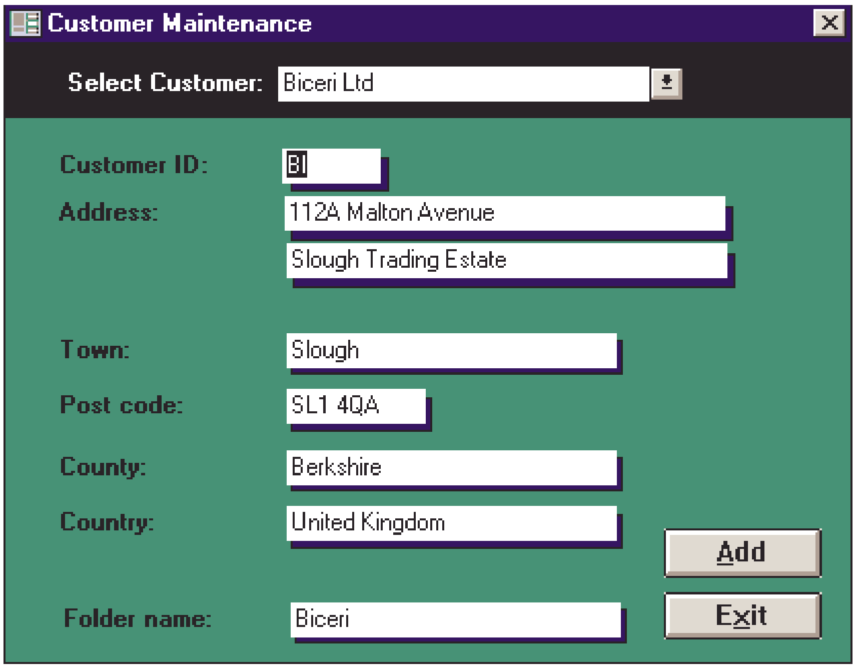
Fuel:	TH106699
Additive:	RL139/5
Shipment Code:	LU214/055
Ref. Lube.:	
Test Lube.:	
Oil SAE Class.:	

3.4 Customer Maintenance

Customer information can be amended in the customer maintenance dialog box (see Figure 17).

To add a customer select the “Add” button.

Figure 17 - The Customer Maintenance Dialog box



Customer Maintenance

Select Customer: Biceri Ltd

Customer ID: BI

Address: 112A Malton Avenue

Slough Trading Estate

Town: Slough

Post code: SL1 4QA

County: Berkshire

Country: United Kingdom

Folder name: Biceri

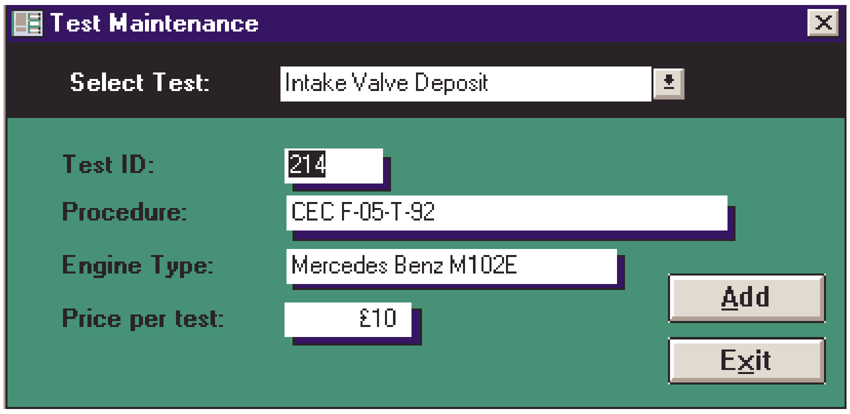
NOTE: The entry labelled "Folder name" is the directory name used for this customer. After migration to Windows'95 and long file names, this entry will no longer be applicable as the full customer name will used instead.

3.5 Test Maintenance

Test information can be amended in the test maintenance dialog box (see Figure 18).

To add a test select the "Add" button.

Figure 18 - The Test Maintenance Dialog box



Test Maintenance

Select Test: Intake Valve Deposit

Test ID: 214

Procedure: CEC F-05-T-92

Engine Type: Mercedes Benz M102E

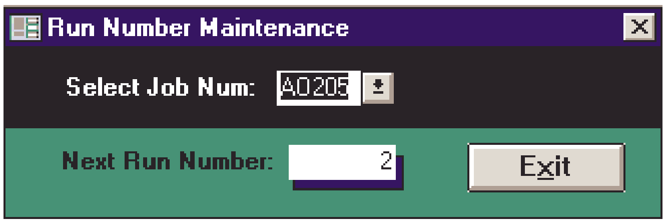
Price per test: £10

3.6 Run Number Maintenance

The system keeps a note of which run or sequence numbers have been already been allocated for each combination of Customer *and* Test type. If for any reason the next run number is required to be different it may be manually adjusted in the run number maintenance dialog box (see Figure 19).

First select the customer/test combination in the "Select Job Num" drop down list and then type the next required run number. It should be noted that entries for an unused customer/test combination will not exist until at least one job has been run. Hence it is not possible to start a sequence from a number other than "1" until a job has been added.

Figure 19 - The Run Number Maintenance Dialog box



NOTE: THIS SHOULD BE USED WITH EXTREME CARE AND ONLY UNDER THE GUIDANCE OF THE I.T. DEPARTMENT AS CHANGING THE NEXT RUN NUMBER MAY CAUSE DATA INTEGRITY ERRORS.

3.7 Tips and shortcuts

The following is a list of tips and shortcuts to aid inputting of data:

- To ease inputting of dates into the system the user may use the CTRL+D key combination to insert the current date into a date field.
- The action of the [ENTER] key may be adjusted through the state of the [SCROLL LOCK] key for tabular forms. With the [SCROLL LOCK] key is down pressing [ENTER] will move the cursor to the next cell downwards. With [SCROLL LOCK] key off pressing [ENTER] will move the cursor to the next cell to the right.
- Where an edit box on a form is not large enough to display all of the contents of the field use [SHIFT]+[F2] to bring up the zoom box.

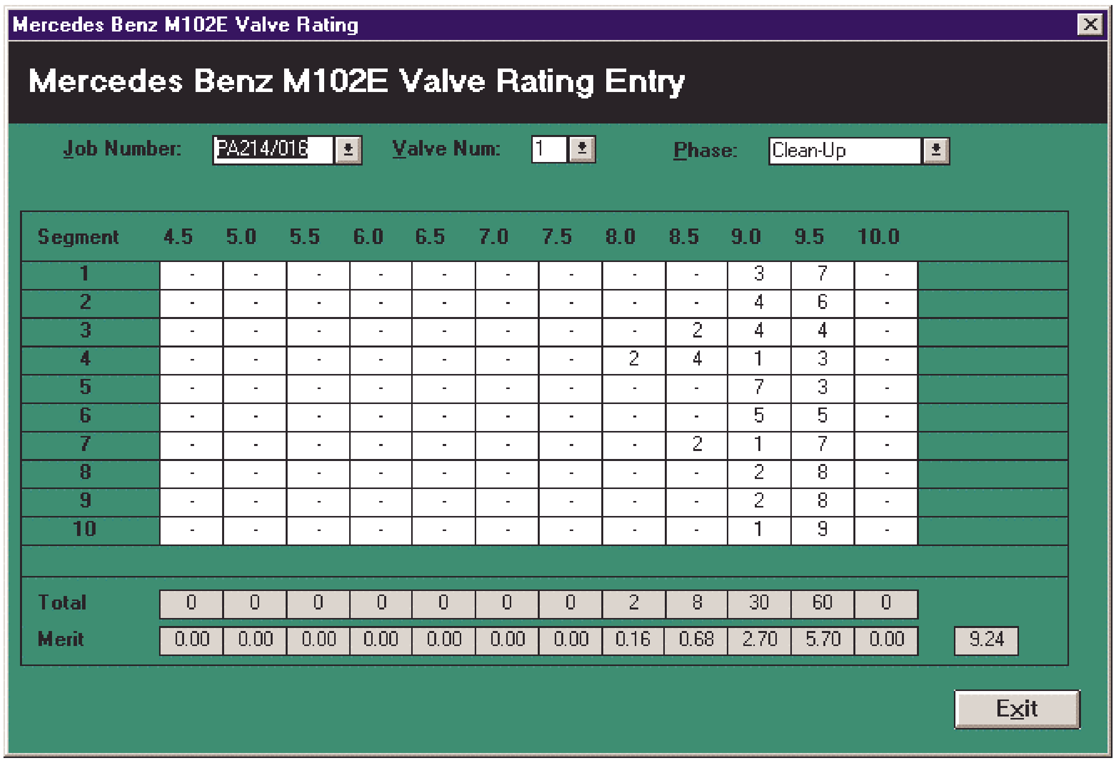
4. The Mercedes M102E Data Entry

4.1 *Valve Rating Entry*

The M102E valve rating entry screen is shown in Figure 20. To select the appropriate entry sheet choose the required job number, valve number and phase from the drop down lists at the top of the screen. The cursor may be moved around the screen by use of the arrow keys.

The use of the [ENTER] key may be adjusted through the state of the [SCROLL LOCK] key. With the [SCROLL LOCK] key is down pressing [ENTER] will move the cursor to the next cell downwards. With [SCROLL LOCK] key off pressing [ENTER] will move the cursor to the next cell to the right.

Figure 20 - M102E valve rating entry dialog box



Mercedes Benz M102E Valve Rating

Mercedes Benz M102E Valve Rating Entry

Job Number: PA214/016 Valve Num: 1 Phase: Clean-Up

Segment	4.5	5.0	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0	9.5	10.0	
1	-	-	-	-	-	-	-	-	-	3	7	-	
2	-	-	-	-	-	-	-	-	-	4	6	-	
3	-	-	-	-	-	-	-	-	2	4	4	-	
4	-	-	-	-	-	-	-	2	4	1	3	-	
5	-	-	-	-	-	-	-	-	-	7	3	-	
6	-	-	-	-	-	-	-	-	-	5	5	-	
7	-	-	-	-	-	-	-	-	2	1	7	-	
8	-	-	-	-	-	-	-	-	-	2	8	-	
9	-	-	-	-	-	-	-	-	-	2	8	-	
10	-	-	-	-	-	-	-	-	-	1	9	-	
Total	0	0	0	0	0	0	0	2	8	30	60	0	
Merit	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.16	0.68	2.70	5.70	0.00	9.24

Whilst entering a number into a cell the left and right arrow keys will not move to cursor to a new cell - in this case use the ENTER key to move the cursor to the next cell or use the up and down arrow keys.

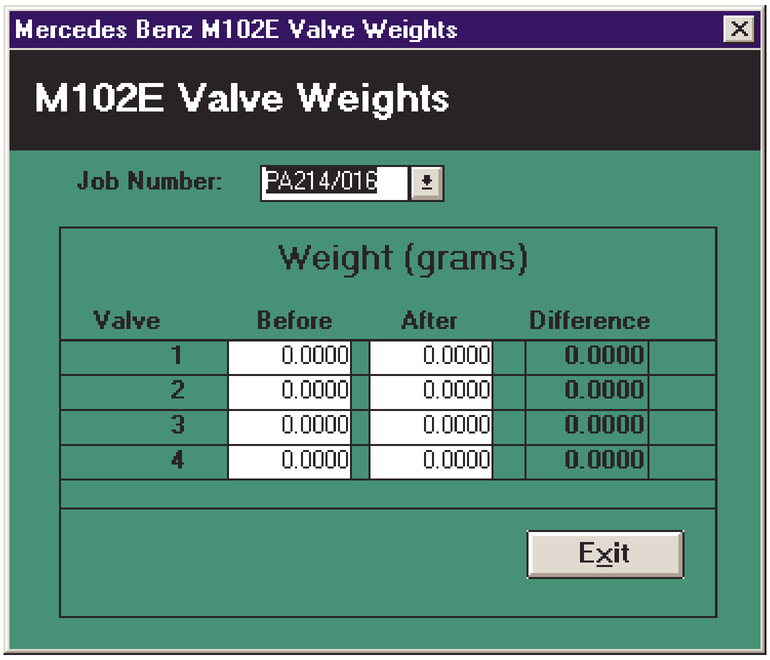
A number of calculated items are placed around the entry table.

To the right of the table is the sum of each segment shown as a red number. When the total equals 10 the number will disappear.

At the bottom of each merit column is the total of the column, and the merit. The total merit for the current valve is also displayed.

The total for each segment is re-calculated each time a number is entered. However the sums for the columns, and the merit are only calculated once a complete segment has been entered. As such once the final number has been entered for the final segment the cursor must be moved to different segment for the calculations to take place.

Figure 21 - M102E Valve Weight Entry



Mercedes Benz M102E Valve Weights

M102E Valve Weights

Job Number: PA214/016

Weight (grams)

Valve	Before	After	Difference
1	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000
4	0.0000	0.0000	0.0000

5. Intelligent Documents

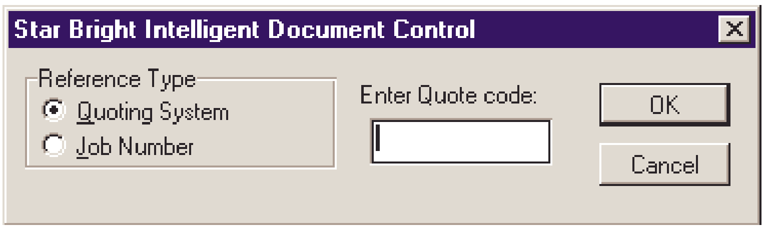
5.1 Overview

The “Star Bright” package extends the functionality of Microsoft Word via a process named “Intelligent Documents”. As documents are created by the reporting system the documents are labelled with the relevant job number and information as to the type of document. This intelligence may be overridden or adjusted at a later date by use of the “Intelligence” button on the “Star Bright Toolbar”.

5.2 Purpose of document intelligence

As documents maintain some sort of intelligence they can aid the general usage. This is most notably when performing a FILE/SAVE AS. At this point the document adjusts the default folder so that the user does not have to navigate the file structure to find the relevant folder.

Figure 22 - The Intelligence Control Dialog box



6. The Covering letter writing system

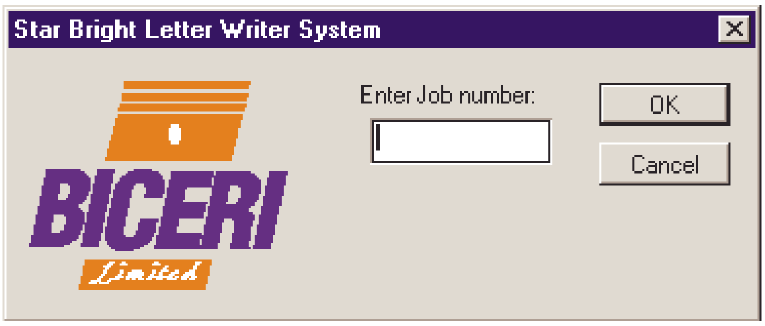
6.1 Overview

The “Star Bright” reporting package includes the facility to produce a cover letter for one or more report(s). This facility extracts the customer and test information for a given job or jobs and inserts them into the body text of the letter. Once it is complete the system writes it the file to the appropriate directory and offers the user the chose of printing immediately.

6.2 Using the Letter Writer

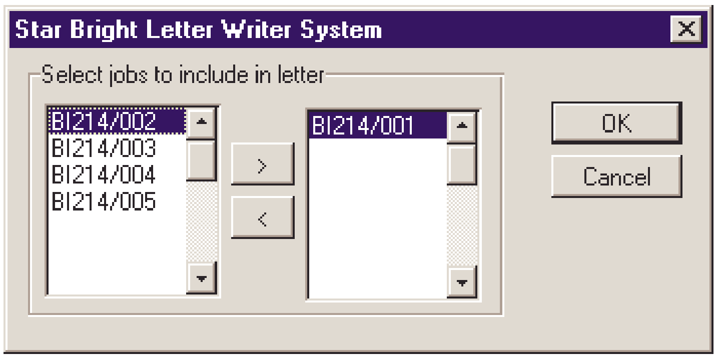
To start the letter chose the “Write Letter” button on the “Star Bright Reporting” toolbar (alternatively choose FILE/NEW and “Report Covering Letter”). A dialog box similar to the one in Figure 23 will appear and prompt for the required job number. If you wish to write a letter covering more than one report you should enter the job number with the lowest run number.

Figure 23 - Letter Writer Job Number Entry Dialog



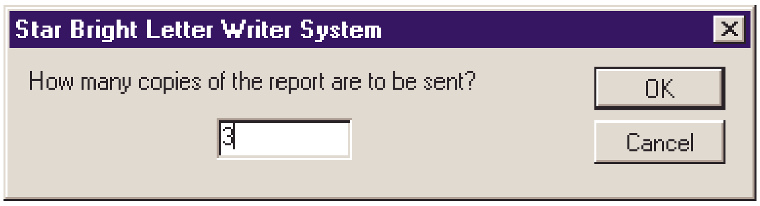
On choosing OK the system will check if there are any reports of that test type, for that customer that have higher run numbers than the one entered. If there are, a dialog similar to Figure 24 will appear to allow selection of additional jobs. The job first entered will appear already in the right hand list box which is for those jobs to be included in the letter. Select additional job numbers in the left hand list box and chose “>” to include them. To exclude jobs select them in the right hand side and chose “<”.

Figure 24 - Additional Job Selection


Star Bright Letter Writer System
Select jobs to include in letter
BI214/002
BI214/003
BI214/004
BI214/005
>
<
BI214/001
OK
Cancel

Once the job(s) have been entered the user will be prompted (see Figure 25) to enter the number of copies of the report which will be sent with the letter. Valid numbers are between 1 and 10.

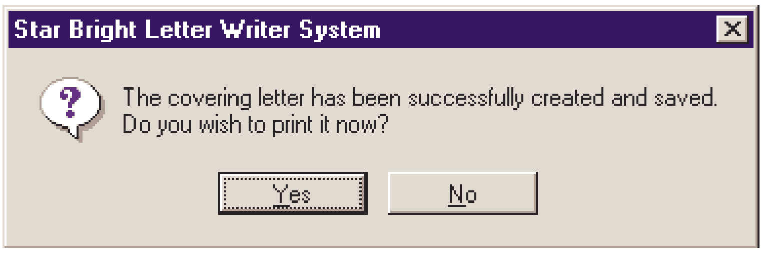
Figure 25 - Number of Report Copies Entry


Star Bright Letter Writer System
How many copies of the report are to be sent?
3
OK
Cancel

All the information required by the system has then been entered and a letter will be automatically produced with the correct information regarding the customer, the test type, each job(s) and the number of copies which will accompany the letter.

The letter will saved in the directory relating to the first job number. The last message given to the user allows them to print the letter immediately.

Figure 26 - Option to print letter immediately


Star Bright Letter Writer System

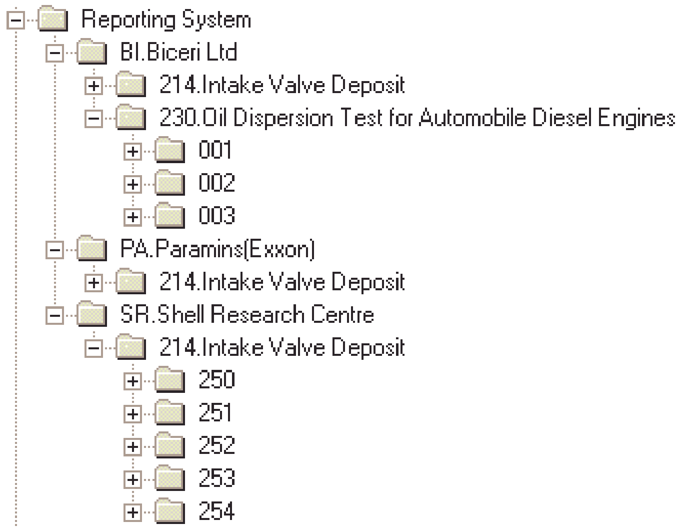
The covering letter has been successfully created and saved.
Do you wish to print it now?

Once the letter has been created the dispatch date for the report is automatically updated to the current date.

7. Reporting Directory Structure

The customer name is shortened to eight characters and the Biceri customer id is used as the extension. Under each customer each test name is also shortened to eight characters and the Biceri test id used as the extension. The run number is then simply used as the last directory name.

Figure 27 - Directory (Folder) Structure under Windows '95/NT



```
Reporting System
    BI.Biceri Ltd
        214.Intake Valve Deposit
        230.Oil Dispersion Test for Automobile Diesel Engines
            001
            002
            003
    PA.Paramins(Exxon)
        214.Intake Valve Deposit
    SR.Shell Research Centre
        214.Intake Valve Deposit
            250
            251
            252
            253
            254
```

This method allows anyone given a job number to easy find the files for the job by simply taking each part of the id (which is the extension), or the description (base directory name).

For additional information see section 5.2 of Biceri Information Management Plan dated March 95

8. Issue Controller

8.1 Overview of issues

All documents created by the reporting system contain a limited amount of issue control. The issue control keeps track of the date of issue changes, the initials of the person making the change and a user comment.

8.2 Automatic handling of issue numbers

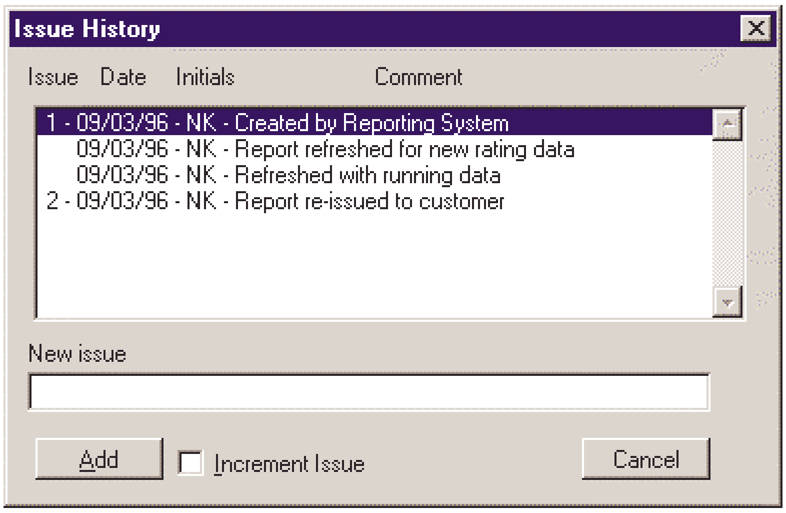
When a test report is created by the reporting system the issue number is automatically set to one, the date is set to the current date, the users initials are logged, and the comment set to “Created by Reporting System”. If the document contains a bookmark named “Issue” then the text at that location is automatically set to read “1”. As the issue number is incremented the text at the bookmark will be automatically incremented.

8.3 Manually handling issue numbers

To increment the issue number you must type a comment in the “New issue” edit box, and then choose Add. Note that by default the issue number is not incremented. To do this the user must first check the “Increment Issue” check box. When the issue number is incremented the text at the bookmark “Issue” (if it exists) is automatically incremented.

If a document does not contain the required issue controlling additions the message box shown in Figure 29 appears. Choosing “YES” will make the required amendments.

Figure 28 - Document History Control Dialog box



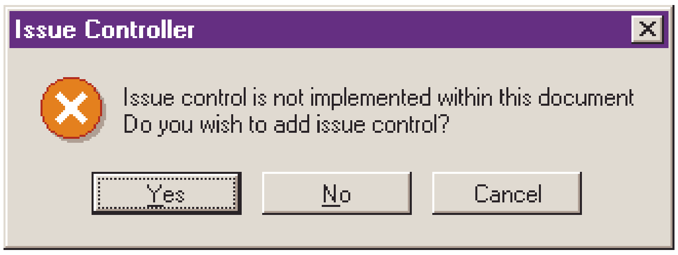
Issue History

Issue Date Initials Comment

```
1 - 09/03/96 - NK - Created by Reporting System
    09/03/96 - NK - Report refreshed for new rating data
    09/03/96 - NK - Refreshed with running data
2 - 09/03/96 - NK - Report re-issued to customer
```

New issue

Figure 29 - Message box when document has no issue control



Issue Controller

Issue control is not implemented within this document
Do you wish to add issue control?

9. Source Code

9.1 NORMAL.DOT (SQL)

```
'*************************************************************
'*      SQL ROUTINES
'*      These are tried And tested routines. Note that in
'*      coordance With Q48115 variable names have been kept
'*      to minimum length, And no remarks exist in the Function
'*      to provide maximum speed
'*
'*      Note: The entire UNC name must be used in declares to the
'*      Word add-in library As it would Not have been loaded at this
'*      time If called via a macro at start-up
'*
'*************************************************************

Declare Function SQLExecQuery Lib "\\SERVER\DATABASE\WBODBC.WLL"(c As Integer, q As String) As
Integer
Declare Function SQLCountErrors Lib "\\SERVER\DATABASE\WBODBC.WLL"() As Integer
Declare Function SQLErrorText$ Lib "\\SERVER\DATABASE\WBODBC.WLL"(e As Integer) As String
Declare Function SQLErrorClass$ Lib "\\SERVER\DATABASE\WBODBC.WLL"(e As Integer) As String
Declare Function SQLErrorCode Lib "\\SERVER\DATABASE\WBODBC.WLL"(e As Integer) As Integer
Declare Function SQLOpen Lib "\\SERVER\DATABASE\WBODBC.WLL"(c As String, o As String, p As Integer)
As Integer
Declare Function SQLGetSchema Lib "\\SERVER\DATABASE\WBODBC.WLL"(c As Integer, t As Integer, q As
String) As Integer
Declare Function SQLGetSchemaItem$ Lib "\\SERVER\DATABASE\WBODBC.WLL"(c As Integer, i As Integer) As
String
Declare Function SQLCloseAll Lib "\\SERVER\DATABASE\WBODBC.WLL"() As Integer
Declare Function SQLClose Lib "\\SERVER\DATABASE\WBODBC.WLL"(c As Integer) As Integer
Declare Function SQLRetrieveRows Lib "\\SERVER\DATABASE\WBODBC.WLL"(cn As Integer) As Integer
Declare Function SQLRetrieveColumns Lib "\\SERVER\DATABASE\WBODBC.WLL"(cn As Integer) As Integer
Declare Function SQLRetrieveItem$ Lib "\\SERVER\DATABASE\WBODBC.WLL"(cn As Integer, col As Integer,
row As Integer) As String

'*************************************************************
'*      Parse SQL Errors And display in Message Boxes
'*************************************************************

Sub ParseErrors(b$)
n = SQLCountErrors
For i = 1 To n
        m$ = "Class(" + SQLErrorClass$(i) + ") "
        m$ = m$ + "Code(" + Str$(SQLErrorCode(i)) + ")" + Chr$(13)
        m$ = m$ + SQLErrorText$(i)
        If (SQLErrorCode(i) = - 1305) Then
                m$ = "Attempt to extract spurious data. "
                m$ = m$ + Chr$(13) + Chr$(13) + "This report Is requesting " + Chr$(34) + b$ +
Chr$(34) + " data "
                m$ = m$ + "although this test does Not require that "
                m$ = m$ + " information. Please remove bookmarks pertaining "
                m$ = m$ + " to " + Chr$(34) + b$ + Chr$(34) + "."
                MsgBox m$, "SQL Error", 48
        Else
                MsgBox m$, "SQL Error", 16
        End If
Next
End Sub

'*************************************************************
'*      Connects to a DBMS returning connection code
'*      This calls ParseErrors If there any any
'*************************************************************

Function ConnectToDBMS(o$)
Print "Connecting to database..."
ConnectToDBMS = SQLOpen(o$, r$, 4)
End Function

'*************************************************************
'*      Closes connection to DBMS
'*************************************************************

Function CloseDBMS(cn)
Print "Closing connection to database..."
CloseDBMS = SQLClose(cn)
End Function
```

```
'***********************************************************
'*      Closes all connections to DBMS
'***********************************************************

Sub CloseAllDBMS
SQLCloseAll
End Sub

'***********************************************************
'*      Execute And wait For transaction to be completed
'***********************************************************

Function WaitForTransaction(cn, s$)
Print "Waiting For database transaction..."
r = - 2
While (r = - 2)
        r = SQLExecQuery(cn, s$)
Wend
WaitForTransaction = r
End Function

'***********************************************************
'*      Retrieve the number of rows And columns
'*      Returns True If there are more than one row And col
'***********************************************************

Function RetrieveNumberOfRowsCol(cn, r, c)
r = SQLRetrieveRows(cn)
c = SQLRetrieveColumns(cn)
RetrieveNumberOfRowsCol = (r > 0) And (c > 0)
End Function

'***********************************************************
'*      Retrieve an item from a database
'***********************************************************

Function RetrieveItem$(cn, c, r)
RetrieveItem$ = SQLRetrieveItem$(cn, c, r)
End Function

'***********************************************************
'*      Get todays Date in valid format For SQL
'***********************************************************
Function LTrimStr$(n)
LTrimStr$ = LTrim$(Str$(n))
End Function

Function GetSQLDate$
GetSQLDate$ = "#" + LTrimStr$(Month(Now())) + "/" + LTrimStr$(Day(Now())) + "/" +
LTRimStr$(Year(Now()) - 1900) + "#"
End Function
'***********************************************************
'*      Update the JOBS table to say that a report has been
'*      written For a given job (Date Is in MM/DD/YY)
'*      Note that this presently only sets the Date Not the Boolean!
'***********************************************************

Function UpdateJobDate(cn, j$, dbf$)
Print "Building SQL update statement..."
cID$ = Library.ExtractCustomerID$(j$)
tID$ = Library.ExtractTestID$(j$)
rn$ = Library.ExtractRunNumber$(j$)

s$ = "UPDATE DISTINCTROW Jobs SET Jobs." + dbf$ + " = "
s$ = s$ + GetSQLDate$ + " WHERE"
s$ = s$ + "((Jobs.Customer_ID='" + cID$ +  "') "
s$ = s$ + "AND (Jobs.Test_ID=" + tID$ + ") "
s$ = s$ + "AND (Jobs.Run_Number=" + LTrimStr$(Val(rn$)) + " ));"

UpdateJobDate = WaitForTransaction(cn, s$)
End Function

'***********************************************************
'*
'*      Gets the unique columns headings (into h$) from a
'*      number of tables (t$) For a SQL connection (cn)
'*      returns 0 (FALSE) If Error
```

```
'*
'***********************************************************

Function GetUniqueColumnsFromTables(cn, t$(), h$())
Print "Obtaining field names..."
r = 1
i = 1
j = 1
While ((t$(j) <> "") And (r > 0))
        r = SQLGetSchema(cn, 5, t$(j))
        If r <= 0 Then
                ParseErrors
        Else
                For k = 1 To r Step 2
                        a$ = SQLGetSchemaItem$(cn, k)
                        '*** The following couple of lines are VERY IMPORTANT!
                        '*** They make sure that a field Is Not held twice
                        m = 1
                        While ((m < i) And (h$(m) <> a$))
                                m = m + 1
                        Wend
                        If (h$(m) <> a$) Then h$(i) = a$
                        i = i + 1
                        Print "Obtaining field names... retrieved (" + Mid$(Str$(i), 2) + ")"
                Next
                j = j + 1
        End If
Wend
GetUniqueColumnsFromTables = (r > 0)
End Function

'***********************************************************
'*
'*      Query the given database connection, get the data
'*      And insert it at the appropriate positions
'*
'*      Returns:        TRUE  - OK, continue
'*                              FALSE - Something nasty happened so stop
'*
'***********************************************************
Function GetAndInsertData(con_num, con_num_hdgs, s$, Tbl$(), comment$, bIgnore, Job$, title$)

'*      Execute the SQL Select command
TRUE = (1 = 1)
FALSE = Not(TRUE)

bAbort = FALSE

ret = WaitForTransaction(con_num, s$)

NumColumns = ret

If ret <= 0 Then
        MsgBox "QUIT" :
        ParseErrors :
        bAbort = TRUE :
        Goto EndFunction
End If

'*******************************************************************
'*      Get names of the columns For the  tables we are going to use
'*******************************************************************

Redim ColumnArray$(NumColumns)
Redim FieldValue$(NumColumns)

If (Not GetUniqueColumnsFromTables(con_num_hdgs, Tbl$(), ColumnArray$())) Then bAbort = TRUE : Goto
EndFunction

ret = RetrieveNumberOfRowsCol(con_num, rows, cols)

If (rows < 1) Then
        a$ = "There Is a problem retrieving data from one of the tables For this job number (" +
Job$ + ")." + Chr$(13) + "The most likely cause Is that the job has Not been registered into the
system using MS Access."

        MsgBox a$, title$, 16
        bAbort = TRUE

        Goto EndFunction :

End If

'******************************************************************
'*      Interrogate database For the real data And put it in
'*      the correct place
'******************************************************************


r = 1

NumInserted = 0
Print comment$ + ": Fields Inserted (0)"


'bIgnore = FALSE

c = 1
While (c <= cols) And (bAbort = FALSE)

        item$ = RetrieveItem$(con_num, c, r)
        item$ = RTrim$(item$)

        '***********************************************************
        '*      Formatting does Not appear to be taken from Access
        '*      so we convert any Date And time strings
        If (Left$(ColumnArray$(c), 3) = "dtm") Then
                item$ = Library.ConvertMSAccessDate$(item$)
        End If
        '***********************************************************

        FieldValue$(c) = item$

        If ExistingBookmark(ColumnArray$(c)) Then
                '*** If there Is a place holder For this field Then go to it
                '*** And fill it in
                '*** Note the item$ MUST be tidied up using RTRIM$() Function
                '*** The "EditClear" Is required to delete the initial
                '*** bookmark holder ie. the space

                If item$ <> "" Then

                        Library.InsertAtBookmark(ColumnArray$(c), item$)

                        NumInserted = NumInserted + 1
                        Print comment$ + ": Fields Inserted (" +     Mid$(Str$(NumInserted), 2) + ")"

                ElseIf (Not bIgnore) Then
                        '** Inform user that field in database Is blank
                        '** but that the template requires it.

                        a$ = "No data exists in the database For the " + ColumnArray$(c) + "." +
Chr$(13) + "The report will be created but will be incomplete And should be re-created after
correcting the problem"

                        Select Case Dialogs.DataMissingDialog(title$, a$)
                                Case 1
                                Case 2
                                        bIgnore = TRUE
                                Case Else
                                        bAbort = TRUE
                        End Select
                End If
        End If
        c = c + 1
Wend


EndFunction:

GetAndInsertData = (bAbort)

End Function

'**********************************************************
```

```
'*      Insert the data from a multi-row selection into a Word
'*      table (tbl$). Insert Zero values If b=True
'*      n - Is column to start at (zero based)
'*   d - Is number of decimal places (column array)
'*
'*      Returns:       TRUE  - OK, continue
'*                            FALSE - Something nasty happened so stop
'*
'***********************************************************
Function InsertSQLTable(con, r, c, tbl$, msg$, b, n, d())

For x = 1 To (c - n)
        msg$ = msg$ + "." : Print msg$
        EditGoTo .Destination = tbl$
        i = 1
        While i < x
                NextCell
                i = i + 1
        Wend

        For y = 1 To r
                SelectCurSentence
                v = Val(RetrieveItem$(con, x + n, y))
                If ((b) Or (v <> 0)) Then junk = Library.InsertWithDP(v, d(x))
                LineDown 1
        Next y
Next x

EditGoTo .Destination = tbl$
TableSelectTable
UpdateFields

InsertSQLTable = (1 = 1)


End Function
```

9.2 NORMAL.DOT (Library)

```
'***********************************************************
'*      LIBRARY ROUTINES
'*      These are tried And tested routines. Note that in
'*      coordance With Q48115 variable names have been kept
'*      to minimum length, And no remarks exist in the Function
'*      to provide maximum speed
'*
'***********************************************************

'***********************************************************
'*      Takes a MS Access Date & time String And converts it
'*      to the format "dd-mmm-yy"
'***********************************************************
Function ConvertMSAccessDate$(d$)
If Len(d$) > 10 Then
        mn = Val(Mid$(d$, 6, 2))
        months$ = "JanFebMarAprMayJunJulAugSepOctNovDec"
        mn$ = Mid$(months$,(mn - 1) * 3 + 1, 3)
        ConvertMSAccessDate$ = Mid$(d$, 9, 2) + "-" + mn$ + "-" + Mid$(d$, 3, 2)
Else
        ConvertMSAccessDate$ = d$
End If
End Function

Sub ClearNumDP(dp(), n)
For i = 1 To n
        dp(i) = 0
Next i
End Sub

'***********************************************************
'*      Insert the data With the correct number of dp's
'*   padded With training zeros If needed
'*
'*       PARAMETERS:    n - Value
'*                                    d - number of dp's
'***********************************************************
Function InsertWithDP(n, d)

If (n = 0) And (d > 0) Then
       l$ = " 0." + String$(d, "0")
Else
       ndp = ToolsCalculate("10^" + LTrim$(Str$(d)))

       n$ = Str$(Int(n * ndp + 0.5))
       l$ = Left$(n$, Len(n$) - d)
       If l$ = " " Then l$ = " 0"
       If (d > 0) Then l$ = l$ + "." + Right$(n$, d)
End If
Insert l$

End Function

'***********************************************************
'*     Insert text at a bookmark, removing any old text
'***********************************************************
Sub InsertAtBookmark(b$, t$)
EditGoTo .Destination = b$
n = Len(Selection$())
CharLeft 1
Insert t$
WW6_EditClear n
End Sub

'***********************************************************
'*     Insert text at a bookmark, removing any old text
'***********************************************************
Sub InsertAtBookmarkWithDP(b$, t$, d)
EditGoTo .Destination = b$
n = Len(Selection$())
CharLeft 1
junk = InsertWithDP(Val(t$), d)
WW6_EditClear n
End Sub

'***********************************************************
'*     Return the top level drive+directory For reports
'*     This MUST be the ONLY reference to reduce redundancy
'***********************************************************
Function GetBaseReportingDirectory$
GetBaseReportingDirectory$ = "\\SERVER\Engineer\Reporting System"
End Function

'***********************************************************
'*     Return the template For the letter - may expand For
'*     Each test type...
'***********************************************************
Function GetLetterTemplate$
GetLetterTemplate$ = "\\SERVER\DATABASE\DOCUMENT\LETTER.DOT"
End Function

'***********************************************************
'*     Return the template For a given test ID
'***********************************************************
Function GetReportTemplate$(t$, title$)
Dim d$(10)
x = - 1
GetReportTemplate$ = ""
INIFile$ = "\\SERVER\DATABASE\TEMPLATE.INI"
tp$ = t$ + ".DOT" : n = 1
k$ = "Type" + Mid$(Str$(n), 2)
a$ = GetPrivateProfileString$(t$, k$, INIFile$)
While a$ <> ""
       d$(n) = a$ : n = n + 1 : k$ = "Type" + Mid$(Str$(n), 2)
       a$ = GetPrivateProfileString$(t$, k$, INIFile$)
Wend
If (n > 1) Then
       n = n - 1
       Begin Dialog UserDialog 428, 40 + (n * 17), title$
              GroupBox 19, 4, 293, 23 + (n * 17), "Select Type of " + t$ + " Report Required"
              OptionGroup  .TypeRequired
              For i = 1 To n
                     OptionButton 29, 4 + (i * 17), 270, 16, d$(i)
```

```
               Next i
               OKButton 322, 7, 88, 21
               CancelButton 322, 31, 88, 21
        End Dialog
        Dim dlg As UserDialog
        x = Dialog(dlg)
        If (dlg.TypeRequired > 0) Then
               tp$ = t$ + Chr$(64 + dlg.TypeRequired) + ".DOT"
        End If
End If
If (x = - 1) Then GetReportTemplate$ = "\\SERVER\DATABASE\DOCUMENT\" + tp$
End Function

'***********************************************************
'*      Is a one character String alphabetic?
'***********************************************************
Function IsAlpha(a$)
a$ = UCase$(a$)
If ((a$ < "A") Or (a$ > "Z")) Then
        IsAlpha = 0
Else
        IsAlpha = - 1
End If
End Function

'***********************************************************
'*      Is a one character String numeric?
'***********************************************************
Function IsNumeric(a$)
If ((a$ < "0") Or (a$ > "9")) Then
        IsNumeric = 0
Else
        IsNumeric = - 1
End If
End Function

'***********************************************************
'*      Get current Users' name
'***********************************************************
Function GetUserName$
Dim dlg As ToolsOptionsUserInfo
GetCurValues dlg
GetUserName$ = dlg.Name
End Function

'***********************************************************
'*      Get current Users' Initials
'***********************************************************
Function GetUserInitials$
Dim dlg As ToolsOptionsUserInfo
GetCurValues dlg
GetUserInitials$ = dlg.Initials
End Function

'***********************************************************
'*      Get Standard Reporting Database Open String
'***********************************************************
Function GetReportingDatabaseOpen$
i$ = GetUserInitials$
GetReportingDatabaseOpen$ = "DSN=Star_Bright;UID=" + i$ + ";PWD=" + i$ + " "
End Function

'***********************************************************
'*      Extract Customer ID from Job number
'***********************************************************
Function ExtractCustomerID$(j$)
ExtractCustomerID$ = Left$(j$, 2)
End Function

'***********************************************************
'*      Extract Test ID from Job number
'***********************************************************
Function ExtractTestID$(J$)
ExtractTestID$ = Mid$(j$, 3, 3)
End Function

'***********************************************************
'*      Extract Run Number from Job number
```

```
'***********************************************************
Function ExtractRunNumber$(j$)
ExtractRunNumber$ = Right$(j$, 3)
End Function

'***********************************************************
'*      Given a number convert it to a 3 char String With
'*      leading zeros
'***********************************************************
Function MakeRunNum$(i)
        a$ = Mid$(Str$(i), 2)
        MakeRunNum$ = Left$("0000", 3 - Len(a$)) + a$
End Function

'***********************************************************
'*      Given a jobnumber returns the jobnumber With the
'*      run number fixed to 3 characters With leading
'*      zeroes. Also fixes back slash
'***********************************************************
Function PadoutRunNumber$(j$)
n$ = j$
If Len(j$) < 9 Then
        n$ = Left$(j$, 5) + "/" + MakeRunNum$(Val(Mid$(j$, 7)))
End If
PadoutRunNumber$ = n$
End Function

'***********************************************************
'*      Given a starting directory (d$) And a run num$ (r$)
'*      check that it exists
'*      Return -1 For TRUE And 0 For FALSE
'***********************************************************
Function CheckThatJobExists(d$, r$)
m = CountDirectories(d$)
i = 1
a$ = ""
While (i <= m) And (r$ <> a$)
        a$ = GetDirectory$(d$, i)
        i = i + 1
Wend
If (r$ <> a$) Then
        CheckThatJobExists = 0 'FALSE
Else
        CheckThatJobExists = - 1      'TRUE
End If
End Function

'***********************************************************
'*      From a given job number (j$) of the format AANNN/NNN
'*      check whether a directory exists For it And If so
'*      return the directory. If it does Not exist Then return
'*      an Empty String. Job Number MUST be UPPER CASE
'*      r$ Is the root directory For reports
'***********************************************************
Function DirectoryFromJobNumber$(j$)
r$ = GetBaseReportingDirectory$ + "\"
DirectoryFromJobNumber$ = ""
cID$ = Left$(j$, 2)
tID$ = Mid$(j$, 3, 3)
rn$ = Mid$(j$, 7, 3)
d$ = MatchDirectory$(r$, cID$)
If d$ <> "" Then
        r$ = r$ + d$
        d$ = MatchDirectory$(r$, tID$)
        If d$ <> "" Then
                r$ = r$ + d$
                If (CheckThatJobExists(r$, rn$) = - 1) Then
                        DirectoryFromJobNumber$ = r$ + rn$
                End If
        End If
End If
End Function

'***********************************************************
'*      From a given Ref number (r$) of the format AANNNN/C
'*      check whether a directory exists For it And If so
'*      return the directory. If it does Not exist Then return
'*      an Empty String. TBD: Change f$ On /C code
'*      5=Products, 6=OEM, 7=L&F
'***********************************************************
Function DirectoryFromRef$(r$)
f$ = "\\SERVER\MARKETING\CLIENTS\"
Select Case Val(Right$(r$, 1))
        Case 5
                s$ = "Products"
        Case 6
                s$ = "OEM"
        Case 7
                s$ = "Lubes"
End Select
f$ = f$ + s$ + "\"
DirectoryFromRef$ = ""
cID$ = Left$(r$, 2)
d$ = MatchDirectory$(f$, cID$)
If d$ <> "" Then DirectoryFromRef$ = f$ + d$
End Function


'===========================================================
'=      NOTE: Functions below here required changes to support
'=      the directory structure round the other way where
'=      the ID Is the first part And Not the extension
'===========================================================

'***********************************************************
'*      Given a starting directory (d$) And an extension (x$)
'*      find a given directory And return it Else
'*      return "".
'*      W95 changes - Right$'s became Left$'s
'*  CURRENT SET UP FOR WIN3.11 (i.e. 8DOT3 FILENAMES)
'***********************************************************
Function MatchDirectory$(d$, x$)
m = CountDirectories(d$)
xLn = Len(x$)
i = 1
a$ = ""
While (i <= m) And (x$ <> Left$(a$, xLn))
        a$ = GetDirectory$(d$, i)
        i = i + 1
Wend
If (x$ <> Left$(a$, xLn)) Then
        MatchDirectory$ = ""
Else
        MatchDirectory$ = a$ + "\"
End If
End Function

'***********************************************************
'*      From a given directory return the job number
'*      This Is the opposite of DirectoryFromJobNumber$()
'*
'***********************************************************
Function JobNumberFromDirectory$(d$)
        i = InStr(d$, ".")
        cID$ = Mid$(d$, i - 2, 2)' Was +1
        i = InStr(i + 1, d$, "\")' Was "."
        tID$ = Mid$(d$, i + 1, 3)
        i = InStr(i + 1, d$, "\")' Added line
        rn$ = Mid$(d$, i + 1, 3)' Was + 5
        JobNumberFromDirectory$ = cID$ + tID$ + "/" + rn$
End Function
```

9.3 NORMAL.DOT (GetDataFromDatabase)

```
Dim Shared TRUE
Dim Shared FALSE

Declare Function SQLOpen Lib "\\SERVER\DATABASE\WBODBC.WLL"(c As String, o As String, p As Integer)
As Integer

'*******************************************************************
```

```
'*
'*      "STAR BRIGHT" ENGINE TEST REPORTING PACKAGE
'*      ===========================================
'*
'*
'*  Oct '98 -   Flavour of the month *again*
'*                           Operational Summary - Emissions d.p.'s
'*                           Exhaust value - d.p's
'*                           Added support For any test ID (starting With the 236)
'*                           i.e. coding Is more generic
'*                           Migration code added to support moving to single
'*                           field job number rather than composite key
'*                           dp's now work For all values
'*                           A few "fudges" are in place For migration
'*                           these will need removing
'*
'*
'*      Mar '96 -       Added emissions table
'*
'*      Feb '96 -       Moved to Word'95. NOTE: SQLOpen stopped working
'*                           And it Is now necessary to call func. directly
'*                           from the MAIN macro Function
'*
'*      Jan '96 -       Removed e-mail And mmsound.
'*
'*      Dec '95 -       Major modifications For editing existing reports etc.
'*
'*      Jan '95 -       Updated table names to follow Access naming
'*      convention from MSDN CD#10. (THIS HAS BEEN UNDONE!)
'*
'*      Aug '94 -       Alpha version of ODBC accessing a database
'*
'*******************************************************************

'*******************************************************************
'*
'*      Main Procedure
'*
'*******************************************************************
Sub MAIN
Dim NumDP(20)
MacroTitle$ = "Star Bright Reporting System"
CR$ = Chr$(13)
TRUE = (1 = 1)
FALSE = Not TRUE

DEBUG = 0

MB_ICONINFORMATION = 64
MB_ICONSTOP = 16
MB_ICONEXCLAMATION = 48
MB_ICONQUESTION = 32
MB_YESNO = 4
MB_DEFAULT2 = 256
MB_YES = - 1
MB_NO = 0

IDC_NORMAL = 0
IDC_WAITCURSOR = 1

'* End of Constants

Print "Running " + MacroTitle$ + "..."

'*      In the live system uncomment the Next line
'On Error Goto CloseIt :

ScreenUpdating 0        '*** Don't update the screen While running
DisableInput 1          '*** Don't allow ESC to abort macro

bNewReport = TRUE
bRefreshing = FALSE     '*      Are going to Set Issue#
bAbort = FALSE          '*      If bAbort Then the report Is Not scrapped

JobNumber$ = Dialogs.EnterJobNumber$(TRUE, MacroTitle$)

If JobNumber$ = "" Then Goto MyEnd :
```

```
'*******************************************************************
'*      Check report to see If Is exists, And offer various options
'*      to delete And start again, Or refresh it As existing/New issue
'*******************************************************************

Directory$ = Library.DirectoryFromJobNumber$(JobNumber$)
ReportFileName$ = Directory$ + "\Test Report.DOC"

If (Files$(ReportFileName$) <> "") Then
        If (MsgBox("The report already exists For job " + JobNumber$ + CR$ + "Do you want to open
And refresh it?", MacroTitle$, MB_ICONQUESTION + MB_YESNO) = MB_YES) Then
                '***********************************
                '* Try to open file to refresh it
                '***********************************
                FileOpen ReportFileName$
                bNewReport = FALSE
                IssueNum = GetDocumentProperty("IssueNum")

                a$ = "For refreshed reports you have the Option of maintaining the issue of" +
Str$(IssueNum)
                a$ = a$ + " Or incrementing it." + CR$ + "Should the issue For this report be
incremented to" + Str$(IssueNum + 1) + "?"
                If (MB_YES = MsgBox(a$, MacroTitle$, MB_ICONQUESTION + MB_YESNO)) Then
                        bRefreshing = TRUE
                        IssueNum = GetDocumentProperty("IssueNum") + 1
                End If

        Else
                '***********************************
                '* Do we want to delete & Restart
                '***********************************
                If (MsgBox("Would you like to delete the report For " + JobNumber$ + " And start from
fresh?", MacroTitle$, MB_ICONQUESTION + MB_YESNO) = MB_YES) Then
                        Kill ReportFileName$
                Else
                        Goto MyEnd
                End If
        End If
End If

WaitCursor IDC_WAITCURSOR

'*******************************************************************
'*      Build strings that will be used several times through out
'*      the course of the code
'*      The table specific to this test Is built up from the Test_ID
'*******************************************************************

'*      Only reference these strings.
Customer_ID$ = Library.ExtractCustomerID$(JobNumber$)
Run_Number$ = Library.ExtractRunNumber$(JobNumber$)
Test_ID$ = Library.ExtractTestID$(JobNumber$)

'*      Choose the required report template (If New report)
If (bNewReport = TRUE) Then
        ReportingTemplate$ = Library.GetReportTemplate$(Test_ID$, MacroTitle$)
        If ReportingTemplate$ = "" Then Goto MyEnd :
End If

' SpecificTable$ = "tbl" + Test_ID$ + "Det"
'### CHANGED ABOVE LINE FOR DEMO
SpecificTable$ = Test_ID$ + "_Specific"
RatingTable$ = Test_ID$ + "_ValveRating"
ValveWeightTable$ = Test_ID$ + "_ValveWeights"
EmissionsTable$ = Test_ID$ + "_Emissions"
OpsSummaryTable$ = Test_ID$ + "_Running"
CombustionDepTable$ = Test_ID$ + "_CombustionDeposits"

'*******************************************************************
Dim Table$(10)

'*** These are the tables that will be referenced
'### CHANGED LINES BELOW BACK FROM tblNAME SINGULAR
Table$(1) = "Jobs"
Table$(2) = "Tests"
Table$(3) = "Customers"
Table$(4) = ""
```

```
'********************************************************************

'
' Delete any gratuitous connections from previous runs, that may Not have been open
'
SQL.CloseAllDBMS

'********************************************************************
'*      Open the connection to the database.
'*      Note that users are logged On under their initials As UID And
'*      As their password
'********************************************************************

DBOpen$ = Library.GetReportingDatabaseOpen$

'*** NB: Change to work With Word'95
'connect_num = SQL.ConnectToDBMS(DBOpen$)
connect_num = SQLOpen(DBOpen$, r$, 4)

If connect_num <= 0 Then
        SQL.ParseErrors("")
        a$ = "There Is a problem connecting to the database. The most likely cause Is that the
database has been opened elsewhere."

        MsgBox a$, MacroTitle$, MB_ICONSTOP

        Goto Myend
End If

'*** NB: Change to work With Word'95
'connect_num_headings = SQL.ConnectToDBMS(DBOpen$)
connect_num_headings = SQLOpen(DBOpen$, r$, 4)

If connect_num_headings <= 0 Then Goto Myend

'********************************************************************
'*      Create a New report (If Not an existing report)
'********************************************************************

If (bNewReport = TRUE) Then
        Print "Opening New report..."
        FileNew .Template = ReportingTemplate$
End If

ViewNormal

'********************************************************************
'*      Special Case to insert the job number without using
'*      three components
'********************************************************************

If ExistingBookmark("JobNumber") Then
        Library.InsertAtBookmark("JobNumber", JobNumber$)
End If

'********************************************************************
'*      Build the main SQL Select statement
'********************************************************************

Print "Building SQL Select statement..."

Select$ = "SELECT * FROM "

'*** List the tables to be referenced...
i = 1
While (Table$(i) <> "")
        Select$ = Select$ + Table$(i)
        i = i + 1
        If (Table$(i) <> "") Then Select$ = Select$ + ", "
Wend

Select$ = Select$ + " WHERE (Tests.Test_ID = Jobs.Test_ID) And "

'********************************************************************
'*      Build up the WHERE of the SELECT from dialog input
'********************************************************************

Select$ = Select$ + "(Jobs.Customer_ID = '" + Customer_ID$ + "' ) And "
```

```
Select$ = Select$ + "(Customers.Customer_ID = '" + Customer_ID$ + "' ) And "
Select$ = Select$ + "(Jobs.Test_ID = " + Test_ID$ + " ) And"
Select$ = Select$ + "(Jobs.Run_Number = " + Run_Number$ + " )"

bIgnore = FALSE

comment$ = "Common Data"

If (SQL.GetAndInsertData(connect_num, connect_num_headings, Select$, Table$(), comment$, bIgnore,
JobNumber$, MacroTitle$) = TRUE) Then Goto CloseIt :

'*******************************************************************
'*      Build the SQL Select statement For test specific data
'*******************************************************************

Select$ = "SELECT * FROM " + SpecificTable$
Select$ = MakeSelect$(Select$, Customer_ID$, Test_ID$, Run_Number$)

Table$(1) = SpecificTable$
Table$(2) = ""

comment$ = "Test Specific Data"

If (SQL.GetAndInsertData(connect_num, connect_num_headings, Select$, Table$(), comment$, bIgnore,
JobNumber$, MacroTitle$) = TRUE) Then Goto CloseIt :

'*******************************************************************
'*      VALVE RATING DATA
'*      Note that valve rating data Is ONLY inserted If a table exists
'*      For that valve
'*******************************************************************

Phase$ = "C"'* Set Phase initially to Clean-Up
Library.ClearNumDP NumDP(), 20

NextPhase:

If ExistingBookmark(Phase$ + "RatingTable1") Then

        Select$ = "SELECT * FROM " + RatingTable$
        Select$ = MakeSelect$(Select$, Customer_ID$, Test_ID$, Run_Number$)
        Select$ = Select$ + " AND (Phase = '" + Phase$ + "' ) "

        valve = 1

        TableID$ = Phase$ + "RatingTable" + Mid$(Str$(valve), 2)

        While (ExistingBookmark(TableID$))

                msg$ = "Inserting rating data For valve" + Str$(valve)

                s$ = Select$ + "AND (ValveNumber = " + Str$(valve) + " ) ORDER BY Segment "

                ret = SQL.WaitForTransaction(connect_num, s$)
                If ret <= 0 Then SQL.ParseErrors("Rating")

                ret = SQL.RetrieveNumberOfRowsCol(connect_num, rows, cols)

                If (Test_ID$ = "214") Then
                        startCol = 5
                Else
                        startCol = 4
                End If

                junk = SQL.InsertSQLTable(connect_num, rows, cols, TableID$, msg$, FALSE, startCol,
NumDp())'Changed from 2 to 0 - Oct98

                valve = valve + 1
                TableID$ = Phase$ + "RatingTable" + Mid$(Str$(valve), 2)

        Wend


        Select$ = "SELECT * FROM " + ValveWeightTable$
        Select$ = MakeSelect$(Select$, Customer_ID$, Test_ID$, Run_Number$)
        Select$ = Select$ + " AND (Phase = '" + Phase$ + "' ) "

        ret = SQL.WaitForTransaction(connect_num, Select$)
        If ret <= 0 Then SQL.ParseErrors("Valve Weights")

        For v = 1 To 4
                b$ = phase$ + "WeightBefore" + Mid$(Str$(v), 2)
                If ExistingBookmark(b$) Then
                        item$ = SQL.RetrieveItem$(connect_num, 5, v)
                        Library.InsertAtBookmark(b$, item$)
                End If
                b$ = phase$ + "WeightAfter" + Mid$(Str$(v), 2)
                If ExistingBookmark(b$) Then
                        item$ = SQL.RetrieveItem$(connect_num, 6, v)
                        Library.InsertAtBookmark(b$, item$)
                End If
        Next v

End If

If Phase$ = "C" Then
        Phase$ = "D"'* Set Phase to Dirty-Up And re-run valve tables
        Goto NextPhase :
End If

'*******************************************************************
'*      EMISSIONS DATA
'*      Note that emissions data Is ONLY inserted If a table exists
'*******************************************************************

TableID$ = "emis"'* name of bookmark in document
Library.ClearNumDP NumDP(), 20
NumDP(2) = 2

If ExistingBookmark(TableID$) Then

        Select$ = "SELECT * FROM " + EmissionsTable$
        Select$ = MakeSelect$(Select$, Customer_ID$, Test_ID$, Run_Number$)

        msg$ = "Inserting Emissions Data"

        ret = SQL.WaitForTransaction(connect_num, Select$)
        If ret <= 0 Then SQL.ParseErrors("Emissions")

        ret = SQL.RetrieveNumberOfRowsCol(connect_num, rows, cols)

        If (Test_ID$ = "214") Then
                startCol = 3
        Else
                startCol = 2
        End If

        junk = SQL.InsertSQLTable(connect_num, rows, cols, TableID$, msg$, TRUE, startCol, NumDP())

End If

'*******************************************************************
'*      OPERATIONS SUMMARY DATA
'*******************************************************************

Library.ClearNumDP NumDP(), 20
NumDP(2) = 1' Torque
NumDP(3) = 2' Fuel kg/hr

If ExistingBookmark("Operations_Summary1") Then

        Select$ = "SELECT * FROM " + OpsSummaryTable$
        Select$ = MakeSelect$(Select$, Customer_ID$, Test_ID$, Run_Number$)

        Stage = 1

        TableID$ = "Operations_Summary" + Mid$(Str$(Stage), 2)

        While (ExistingBookmark(TableID$))

                msg$ = "Inserting running data For stage" + Str$(Stage)

                s$ = Select$ + "AND (Stage = " + Str$(Stage) + " ) "

                ret = SQL.WaitForTransaction(connect_num, s$)
                If ret <= 0 Then SQL.ParseErrors("Operations Summary")
                ret = SQL.RetrieveNumberOfRowsCol(connect_num, rows, cols)

                'DEBUG MsgBox "Rows:" + Str$(rows) + cr$ + "Cols:" + Str$(col)

        If (Test_ID$ = "214") Then
                startCol = 4
        Else
                startCol = 3
        End If

                junk = SQL.InsertSQLTable(connect_num, rows, cols, TableID$, msg$, TRUE, startCol,
NumDP())

                Stage = Stage + 1
                TableID$ = "Operations_Summary" + Mid$(Str$(Stage), 2)

        Wend

End If

'*******************************************************************
'*      Total Fuel Flow
'*******************************************************************
b$ = "TotalFuelFlow"

If ExistingBookmark(b$) Then

        Select$ = "SELECT * FROM q" + Test_ID$ + "Totalfuelflow"
        Select$ = MakeSelect$(Select$, Customer_ID$, Test_ID$, Run_Number$)

        comment$ = "Calculating total fuel flow"

        ret = SQL.WaitForTransaction(connect_num, Select$)
        If ret <= 0 Then SQL.ParseErrors("Total fuel flow")

        item$ = (SQL.RetrieveItem$(connect_num, 3, 1))
        Library.InsertAtBookmarkWithDP("TotalFuelFlow", item$, 0)
End If

'*******************************************************************
'*      Set the issue control to issue n If we are refreshing Or 1 If
'*      If this Is a New issue
'*******************************************************************

If (bRefreshing = TRUE) Then
        Issues.SetIssue("Refreshed by Reporting System", 0)
Else
        Issues.SetIssue("Created by Reporting System", 1)
End If

EditSelectAll
UpdateFields
UpdateFields
StartOfDocument

'*******************************************************************
'*      Save the report - but recommend read-only For re-opening
'*******************************************************************

'* Note: FileName$ Is a reserved word so use ReportFileName$ instead

ScreenUpdating 1'*** Allow the screen to update

'*** Set the document properties ***
SetDocumentProperty("JobNum", 0, JobNumber$, 2)
SetDocumentProperty("DocType", 0, "Report", 2)
'************************************

If ((Files$(ReportFileName$) <> "") And (bRefreshing = False)) Then
        a$ = "The report For job number " + JobNumber$ + " already exists." + CR$ + "Can Not
overwrite the old copy As it may be in use elsewhere."

        MsgBox a$, MacroTitle$, MB_ICONSTOP
Else

'* THIS IS THE LINE THAT WRITES THE REPORT TO DISK!!!
        FileSaveAs .Name = ReportFileName$, .RecommendReadOnly = 1
```

```
'*******************************************************************
'***
'***    As the report has now been safely written to disk must
'***    update the database to say that this report has been
'***    produced
'***
'*******************************************************************

       ret = SQL.UpdateJobDate(connect_num, JobNumber$, "dtmReport_Written")

       If ret <= 0 Then SQL.ParseErrors("")

       Print "File " + ReportFileName$ + " saved to disk."
       a$ = "The report For job number " + JobNumber$ + " has been created." + CR$ + "It can be
found in the directory: " + CR$ + Directory$ + CR$ + "The database has been updated to state that
the report has been written"
       MsgBox a$, MacroTitle$, MB_ICONINFORMATION

End If


'*******************************************************************
'*** Close the database connection
'*******************************************************************
CloseIt:
If Err = 75 Then
       MsgBox "The report can Not be deleted. It may be open elsewhere.", MacroTitle$, MB_ICONSTOP
ElseIf (Err <> 0) Then
       MsgBox "The macro encountered an Error" + CR$ + "The Error number Is " + Str$(Err), "Error
Message", MB_ICONSTOP
End If

ret = SQL.CloseDBMS(connect_num)

'** Should move this nearer to where it Is used
ret = SQL.CloseDBMS(connect_num_headings)

'*******************************************************************
'*     Normal Exit point, And Exit Point For Error handling
'*******************************************************************
MyEnd:

If (bAbort = TRUE) Then FileClose 2'* Close - Don't Save

WaitCursor IDC_NORMAL

Print "Done."

End Sub

'*******************************************************************
'*
'*  NOTE: This Function Is simply whilst the migration from a
'*  composite primary key to a single field primary key takes place
'*  This Is previously the customer And run numbers were in
'*  separate fields And Not simply a job number field
'*
'*  When the 214 tables have been amended this Function should
'*  be removed
'*******************************************************************
Function MakeSelect$(s$, c$, t$, r$)

If t$ = "214" Then
       s$ = s$ + " WHERE (Customer_ID = '" + c$ + "')"
       s$ = s$ + " AND (Run_Number = " + r$ + " ) "
Else
       s$ = s$ + " WHERE (sJobID = '" + c$ + t$ + r$ + "') "
End If

MakeSelect$ = s$

End Function
```