Software Migrations Ltd

FermaT Workbench

User Guide

November 2008

Version: 138 - Issued

Status:
Confidential

Software Migrations Limited
3 Sandridge Park
Porterswood
St Albans
Herts
United Kingdom
AL3 6PH

Tel: +44 (0)1727 898699
Fax: +44 (0)1727 866212
www.smltd.com

Table of Contents

Key to standards used within this document:

Normal descriptive text

Important Term

Emphasised words

Tool Name

You try this

Reference to command / menu item / keyboard key

`Reference to example data / files`

1. The FERMAT Workbench System

This User Guide provides a technical reference manual for the SML Workbench. It is designed to explain how the Workbench is used and describes the underlying software which controls its operation. The User Guide is designed to be used with two sample projects DSA and Proj01, which were supplied with your copy of the Workbench. It details the features and functions of the Fermat Workbench and how the tool can be used to automatically document assembler systems and assist programmers and analysts to quickly comprehend even the most complex assembler systems.

1.1 What is included

With your copy of the Workbench you will have received the following files:

Documentation – This User Guide
Software – This contains the FERMAT software
Projects – Sample projects include DSA, Proj01. Both projects DSA and Proj01 are used in this guide to demonstrate various features of the workbench.

1.2 System Requirements

The Fermat Workbench is designed to work using Microsoft XP, Windows 2000 and NT; it also requires Microsoft Access 2000 or later. The software and sample projects will require approximately 150MB of disk space.

Some of the functions in the workbench such as flowchart generation are computationally intensive, the Fermat workbench will run on any PC but for optimum performance the minimum specification should be Pentium 4, 3.0GHz with 1024MB of RAM with at least 5Gb of free disk space. The PC should have Microsoft Office 2000 or above and Microsoft Internet Explorer version 6.0 or above installed.

1.3 Installation and set-up

When the software is originally loaded and unzipped, all the files in the Fermat directory will be read only. It is mandatory that you change the properties of the folder for full read-write access.

This is achieved by highlighting the Fermat Directory in Windows Explorer and right clicking with your mouse. Select properties and de-select the Read Only attribute, press **OK**. You will then be prompted to confirm attribute changes, check the “apply changes to this folder, subfolders and files”, press **OK**.

1.4 How to use this manual

The FermaT Workbench is supplied in various levels of sophistication; this user manual details the entire functionality of the Workbench. Depending on the actual level of Workbench you have purchased various options may not be available. The following table details the different levels of the Workbench and their functionality.

0 – Parser Only; 1 – Standard Workbench; 2 – Code Slicing; 3 - Business Rules Extraction

Function Description	**Manual Ref**	**0**	**1**	**2**	**3**
New Project	6	✓	✓	✓	✓
Import Files	7.1	✓	✓	✓	✓
Import Listings	7.2		✓	✓	✓
Delete Functions	8	✓	✓	✓	✓
Inventory Report	9	✓	✓	✓	✓
Function Catalogue	11	✓	✓	✓	✓
Call Graph	12		✓	✓	✓
Text Editor	14	✓	✓	✓	✓
Program Flowchart	15		✓	✓	✓
User Annotations	16		✓	✓	✓
Data Catalogue	17		✓	✓	✓
Batch Data Tracker	18		✓	✓	✓
Easy Data Tracker	19		✓	✓	✓
Sub-projects	21		✓	✓	✓
Generate Data Flow	23.4			✓	✓
Create Annotated Files	23.7			✓	✓
Business Rule Extraction	23.14				✓
Documenting Business Rules	23.15				✓
Export Function	24		✓	✓	✓
Reports	25	✓	✓	✓	✓
Customising the Workbench	30	✓	✓	✓	✓
Utilities	26	✓	✓	✓	✓
Migration					

2. Starting the Workbench

This document assumes you have loaded the Fermat Software into C:\FERMAT\ directory, if you have loaded the software in another location you must edit the FERMAT\Full (nnn)\Config\TempFile.tab.

By default the contents of this file will be:

```
TempLoc=c:
```

If you have installed the FERMAT software to another location on your system e.g. the D: drive then the contents of the TempFile.tab file must be changed to:

```
TempLoc=d:
```

Start the Workbench by double-clicking the following program file
C:\Fermat\Software\Eval\bin\Workbench.exe
(You could create a shortcut on your desktop to this program file.)

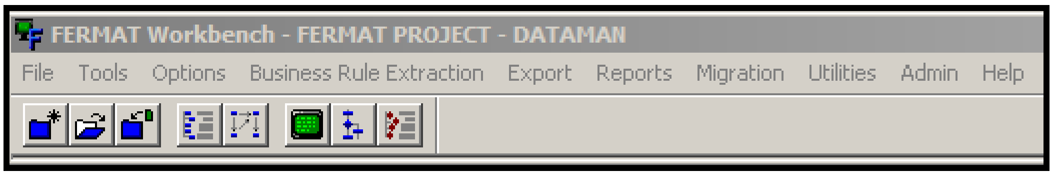
3. Using the Supplied FERMAT Projects

The sample FERMAT Projects DSA and Proj01 are required in order to demonstrate the FERMAT Workbench functions as described in this User Guide.

A FERMAT Project is a directory - named after the Project itself - containing a number of valid subdirectories and FermaT files.

4. The Workbench Environment

The FERMAT Workbench environment is represented to the user in the form of a Workbench Toolbar, from which all Workbench Tools are invoked.



The current release provides the following set of Tools to assist the Assembler programmer with code comprehension:

- ✓ **Function Catalogue**
- ✓ **Function Call Graph**
- ✓ **Text Editor**
- ✓ **Program Flowchart**
- ✓ **Data Catalogue**
- ✓ **Batch Data Tracker**
- ✓ **Easy Data Tracker**
- ✓ **Business Rules Extraction**
- ✓ **Export Function**
- ✓ **Reports**
- ✓ **Utilities**

This User Guide has been designed to introduce you to each of these Tools in turn as quickly as possible.

5. FERMAT Projects

Each FERMAT Project contains a collection of Assembler files, typically representing an Assembler system or sub-system. The Project also contains all the working files produced and required by the Workbench.

Before the Workbench Tools can be used on a particular Assembler system, the source and/or listing files must be imported into a FERMAT Project. See section 7.

FERMAT Projects may be stored anywhere that is accessible by the machine running the Workbench; i.e. either locally or on a shared network drive.

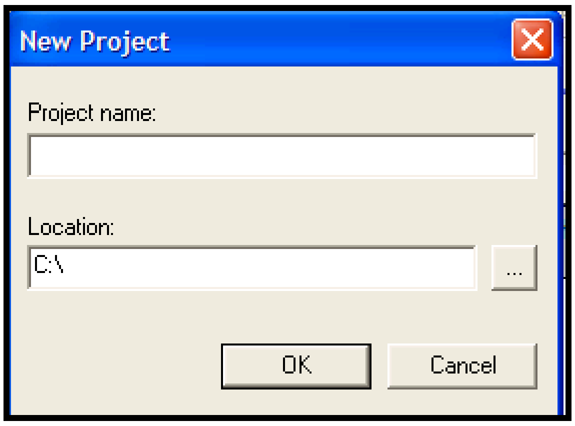
The Workbench supports a hierarchy of projects in the form of Master Projects and associated sub-projects – See section 20 Master Projects and Sub-projects for further details.

Start the Workbench by double-clicking the following program file
C:\Fermat\Software\Eval\bin\Workbench.exe
(Or create a shortcut on your desktop to this program file.)

FERMAT Workbench

6. Create a New Project

From the main menu bar select File | New Project, alternatively select the from the icons menu bar. This will bring up the New Project dialogue box, below:



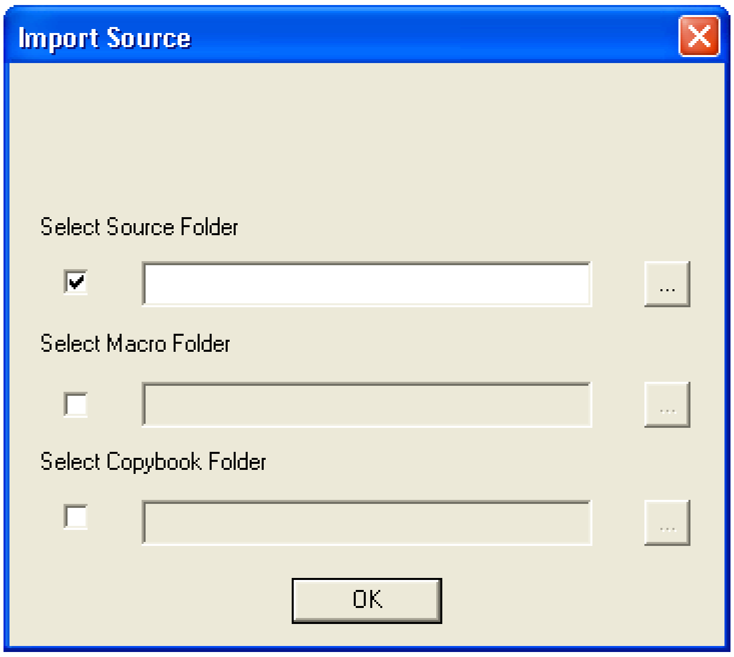
Enter the required project name in the Project Name, and then enter the location where the project is to reside, or use the browse button to navigate your system and select the required location.

7. Importing Files

The Workbench can import Assembler source or Assembler listings, depending on what functionality is required. Assembler source can be used for static analysis only, Assembler listings are required as input if code slicing and business rules processing are required.

7.1 Importing Assembler Source

From the main menu bar select File | Import Files, alternatively select from the icons menu bar. This will bring up the Import Files dialogue box, below:

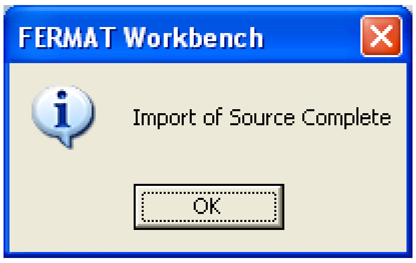


Enter the locations for the components you wish to import. The import source dialogue gives you the opportunity to specify directories for Source, Macros and Copybooks. For each type of element you should click the checkbox and enter the location required. Each component type has a browse button to help you locate the directory.

To process the files in the selected directory and make them available for use in the Workbench press **OK**. The workbench will import all files found in the specified locations into your project and regenerate the repository.

For an existing project, any elements which already exist in the workbench repository will be replaced with the newly imported version.

When the import process has completed, the Workbench will display the following message box:

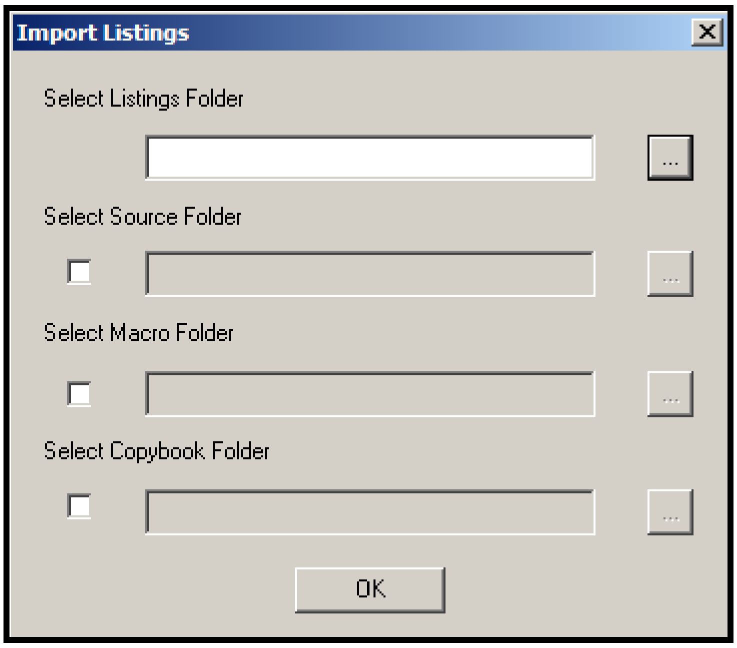

FERMAT Workbench

Import of Source Complete

The message box shows that the import process has completed and the repository is ready for use.

7.2 Importing Assembler Listings

To import Assembler listings select Business Rule Extraction | Import Listings from the main menu bar this will bring up the Import Files Dialogue Box, below:



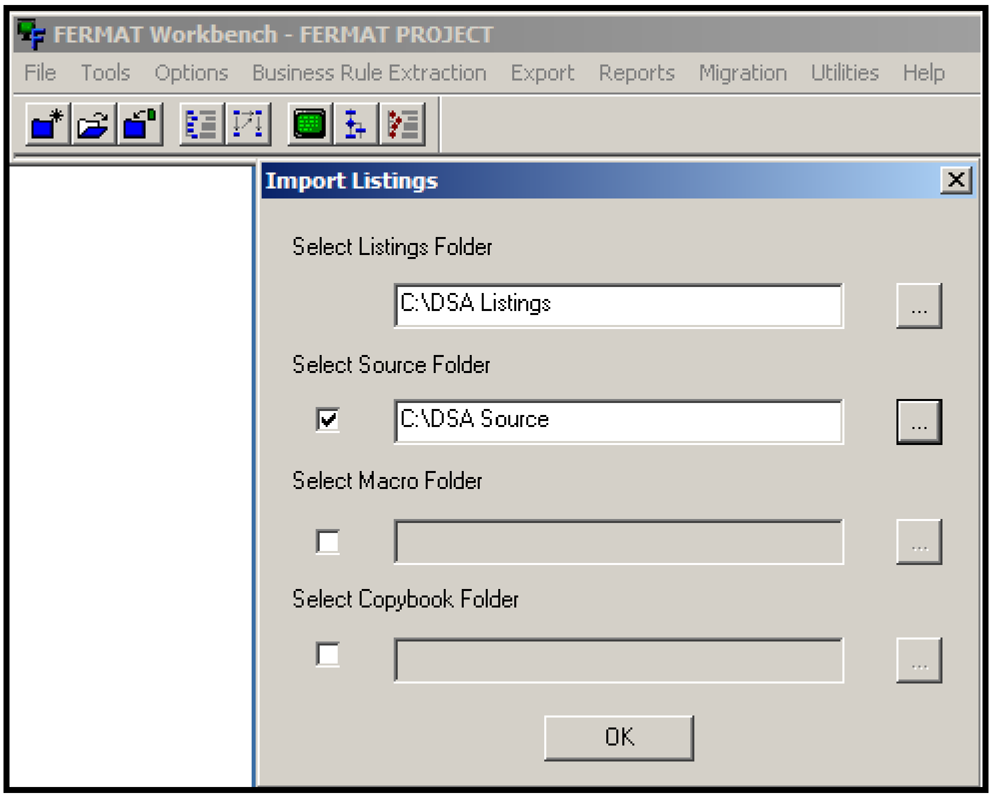
Enter the location of the file containing the Assembler listings, or use the browse button to navigate your directory structure to point to the Assembler Listings. Once selected press **OK**, the Workbench will now import the listings into the project it will automatically extract the source, copybooks and macros and populate the appropriate files in the Workbench repository.

The Workbench automatically recognises over 30 different Assembler listing types, including IBM mainframe variants and different levels of Tachyon listings. If the listings you are trying to import are not recognised please contact SML Support.

7.3 Importing Assembler Listings & Source

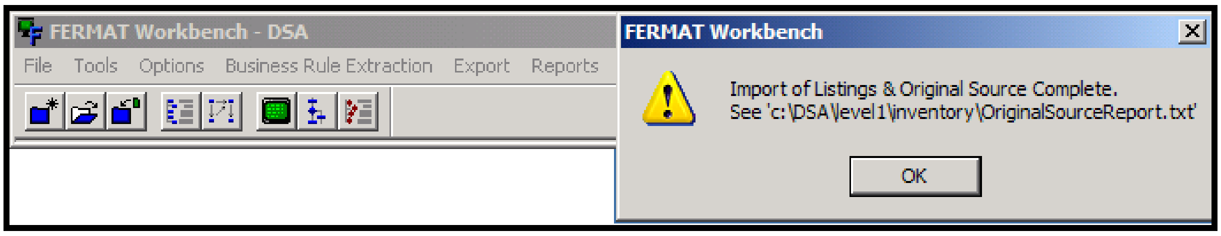
This is required when the Workbench is to be used as an integral part of the Applications Life Cycle, and where users want to include the FermaT comments and mark-ups made in the Text Editor and store these as comments in the original source of the application. See section 30 for further details of using the SML Workbench in the Application Life Cycle.

The original Assembler source files can be imported at the same time as the Assembler listings by ticking the Select Source Folder option and navigating to the appropriate folder. As shown in the following example:



In the above example the listings and source for the DSA project are imported at the same time.

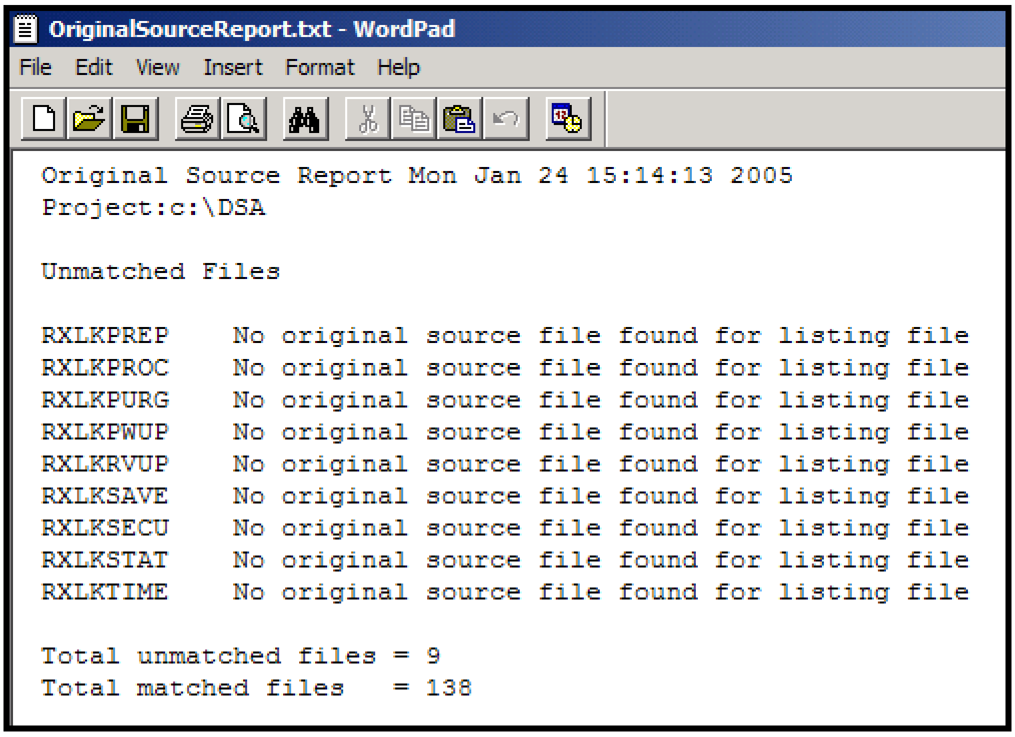
Once the import process is completed, the Workbench will display the following message box:



Import of Listings & Original Source Complete.
See 'c:\DSA\level1\inventory\OriginalSourceReport.txt'

This message shows the import process is complete, and has automatically produced a report located in the level1\Inventory\OriginalSourceReport.txt. This report details any differences in the listings and source files imported, as overleaf:

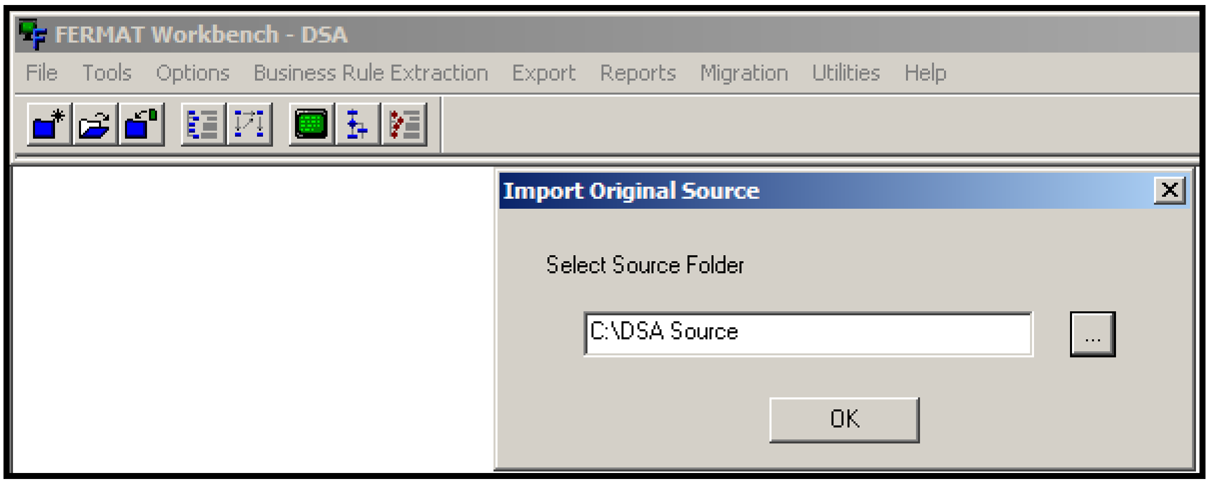
Software Migrations Ltd


OriginalSourceReport.txt - WordPad
File Edit View Insert Format Help

```
Original Source Report Mon Jan 24 15:14:13 2005
Project:c:\DSA

Unmatched Files

RXLKPREP    No original source file found for listing file
RXLKPROC    No original source file found for listing file
RXLKPURG    No original source file found for listing file
RXLKPWUP    No original source file found for listing file
RXLKRVUP    No original source file found for listing file
RXLKSAVE    No original source file found for listing file
RXLKSECU    No original source file found for listing file
RXLKSTAT    No original source file found for listing file
RXLKTIME    No original source file found for listing file

Total unmatched files = 9
Total matched files   = 138
```

In the above example the report has identified that several modules were missing from the source file, if source was present and there were no listings this would also be reported.

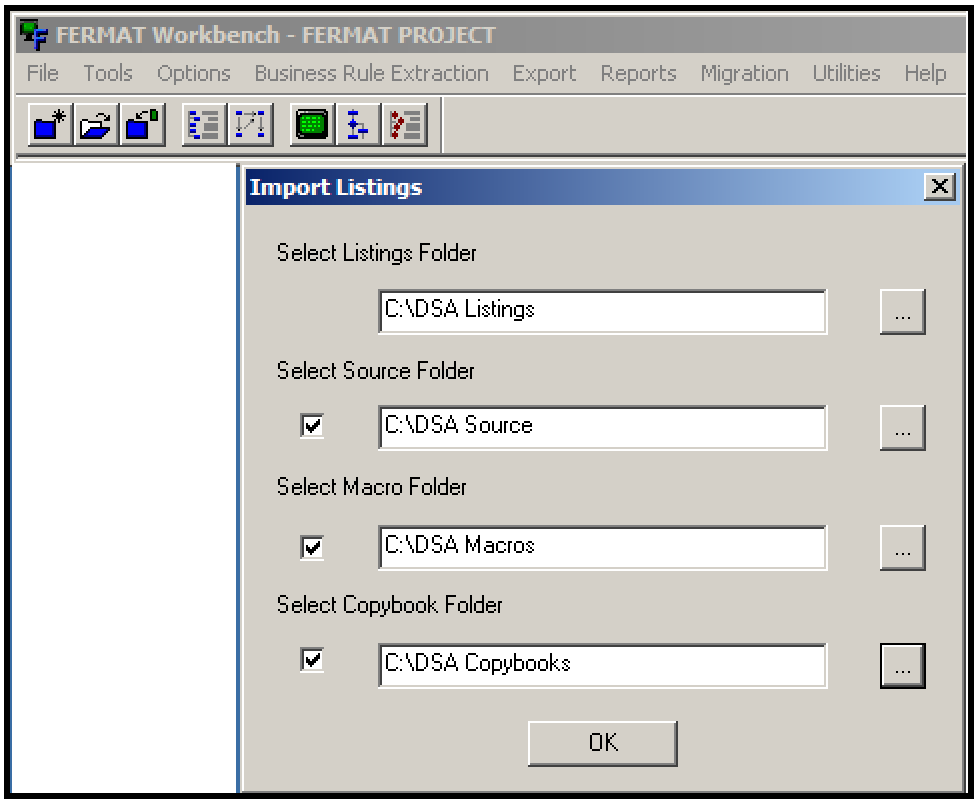
The original source files for a project can be imported separately from the listings at anytime by selecting Business Rule Extraction | Import Original Source from the main menu bar, this will bring up the Import Original Source Dialogue Box, below:


FERMAT Workbench - DSA
File Tools Options Business Rule Extraction Export Reports Migration Utilities Help
Import Original Source
Select Source Folder
...
OK

C:\DSA Source

When selected this process will also automatically generate the OriginalSourceReport.txt file as before. When importing the original source the process does not rebuild the internal Workbench Repository.

7.4 Importing Assembler Listings, Source, Macros & Copybooks


FERMAT Workbench - FERMAT PROJECT
File Tools Options Business Rule Extraction Export Reports Migration Utilities Help
Import Listings
...
OK

Select Listings Folder: C:\DSA Listings

Select Source Folder: [✓] C:\DSA Source

Select Macro Folder: [✓] C:\DSA Macros

Select Copybook Folder: [✓] C:\DSA Copybooks

As can be seen from the above dialogue box, it is possible to import multiple file types at the same time. Whenever listings, macros or copybooks are imported into the workbench, the internal repository must be updated, importing files using this process means that the import of the files is performed altogether and the repository only requires updating once.

7.5 Processing Listings

Using listings as import enables the FermaT Workbench to perform more detailed analysis of the assembler project. Listings are required for code slicing, detailed impact analysis and business rule extraction; as described later in this guide.

When source only is used as the import, the Fermat Workbench will scan all the modules, macros and copybooks and establish linkage between these entities, the Workbench will determine that a module calls certain macros and that macros can call other macros or modules, but cannot determine which module is actually called as the macro assembler language is conditional.

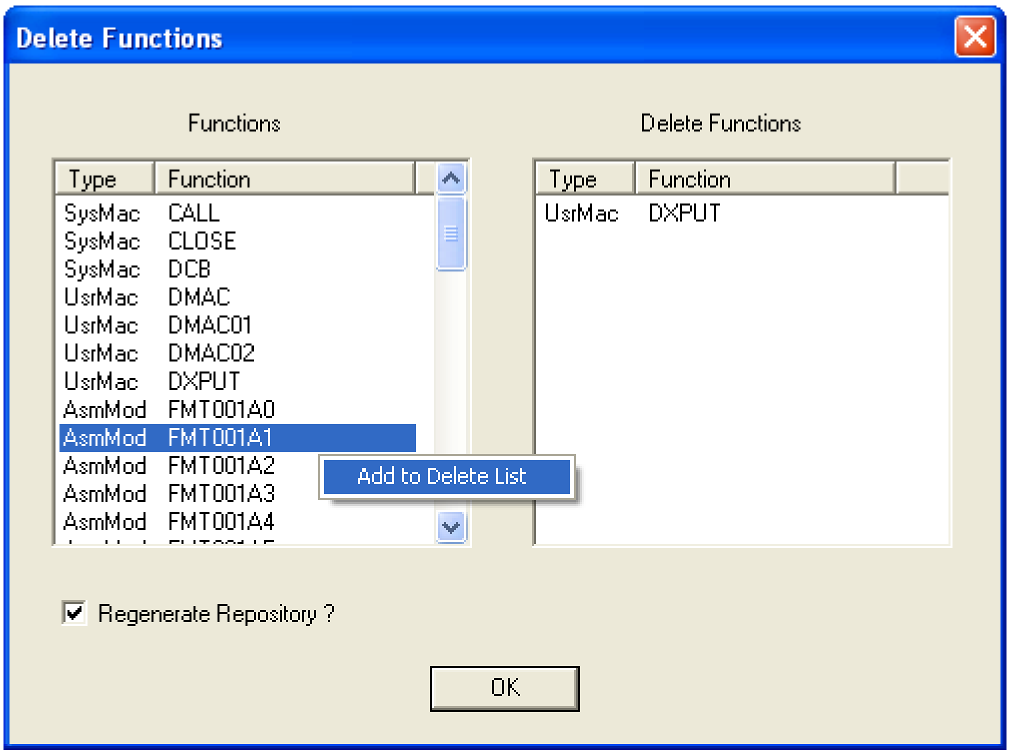
When listings are used as import, using the PCONTROL(GEN,MCALL,ON) compile directives, all macros are fully expanded including nested macros. As part of the import process the FermaT Workbench scans the listings searching for linkage between modules in the macro expansions and populates the repository with this information, providing a complete linkage map of the assembler application.

8. Delete Functions

Once a Fermat Project has been created and the repository populated with the imported Assembler source or listings, macros and copybooks, the user can remove unwanted functions from the project by deleting functions.

Select Tools | Delete Functions… from the main menu on the Workbench

The Workbench will display the following dialogue box.



Type	Function
SysMac	CALL
SysMac	CLOSE
SysMac	DCB
UsrMac	DMAC
UsrMac	DMAC01
UsrMac	DMAC02
UsrMac	DXPUT
AsmMod	FMT001A0
AsmMod	FMT001A1
AsmMod	FMT001A2
AsmMod	FMT001A3
AsmMod	FMT001A4

Type	Function
UsrMac	DXPUT

☑ Regenerate Repository ?

This dialog box displays all the functions in the current Project in the Functions panel. Right clicking on the module brings up the option “Add to Delete List”, selecting this will place that function in the Delete List in the Delete Functions panel.

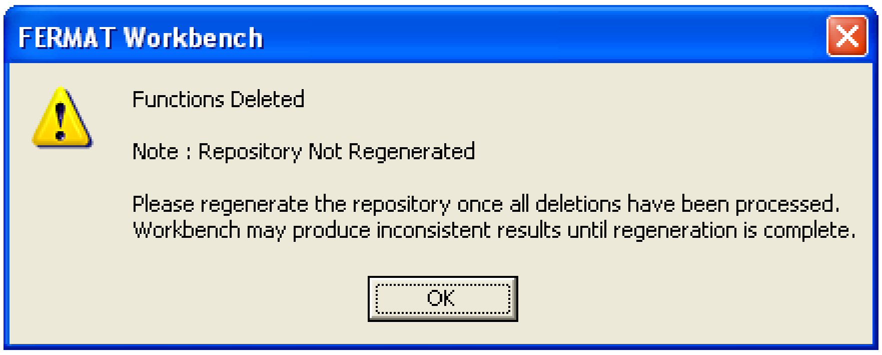
Functions can be removed from the Delete Functions Panel by right clicking on the Function and selecting Delete.

In the above example the User Macro DXPUT has already been selected and the Module FMT001A1 will be added to the Delete List.

By default the Delete Functions process will regenerate the repository. If the repository is to be left without regeneration after the Function Delete process, the Regenerate Repository check box must be un-ticked. The only reason not to regenerate is if several calls to Delete Functions are required before any further use of the Workbench.

When the list is complete press **OK** to start the process.

If the repository has not been regenerated after deleting functions the following message is displayed:


FERMAT Workbench

Functions Deleted

Note : Repository Not Regenerated

Please regenerate the repository once all deletions have been processed.
Workbench may produce inconsistent results until regeneration is complete.

The repository must be regenerated after functions have been deleted to ensure workbench integrity. The repository can be regenerated manually as follows:

Select Tools | Regenerate repository… from the main menu on the Workbench

8.1 Importing New Versions of Modules

It is common practice for organisations to use version numbering for their functions e.g.

ASM00121 where ASM001 is the module name and 21 is the version number; the workbench uses the whole function name ASM00121 in the repository.

The Workbench supports versioning by using the Version.tab file in the Workbench Config directory, see Section 30 Customising the Workbench for further details.

If a new version of ASM001 version 22 is required to be imported into the workbench to replace the existing Function ASM00121.

The user should first delete the function ASM00121 using the Delete Function facility and regenerate the repository then import the new Listing (and Source if required) of ASM00122.

9. Inventory Report

As part of the process of importing the assembler applications, the Workbench automatically collects inventory information of the project. To view the inventory of the current project:

Select Tools | Inventory Report... from the main menu on the Workbench

This will invoke an Access application that will produce a series of 8 reports detailing the Assembler Modules, Copybooks, Macros and Listings supplied plus any ignored files, any missing or orphaned files and details of all files associated with the project.

The DSA project consists of 147 assembler modules. The Proj01 project has 9 assembler modules, 1 copy book and 10 macros.

10. Selecting a Module

Once a Project has been selected, you may select a Module to work with. A Module in the Workbench relates to a physical file imported into a FERMAT Project.

There are two project-level Tools in the Workbench that allow you to select the Current Module:

- The text-based **Function Catalogue**
- The graphical-based **Function Call Graph**.

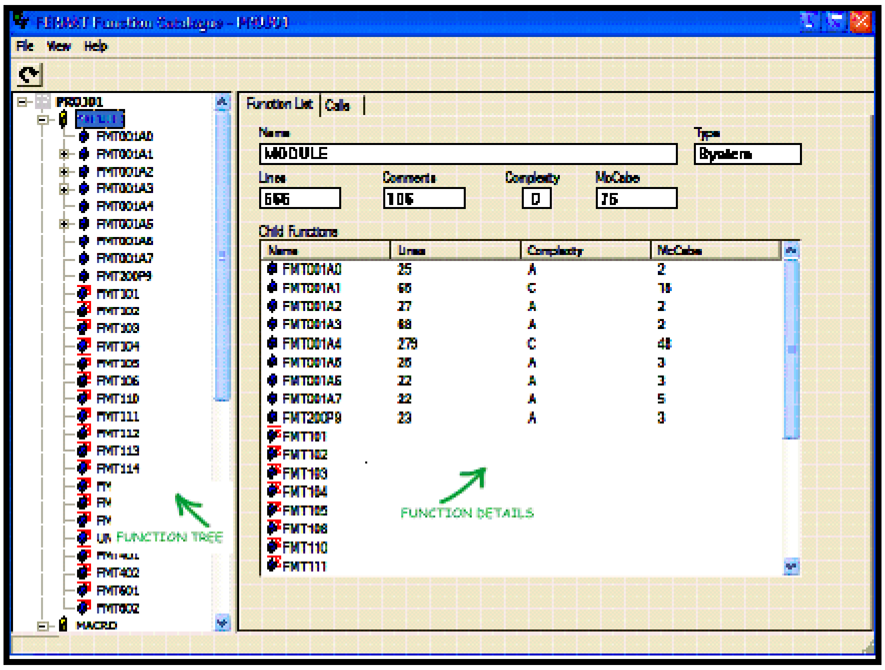
Once a Module has been selected as the Current Module, by either of these Tools, the Module name will appear alongside the name of the current Project in the Workbench Toolbar's title bar. From then on, any module-level Tool[1] that is opened – or is already open – will relate to this Current Module.

[1] i.e. the rest of the Tools: Text Editor, Program Flowchart and Data Catalogue.

11. The Function Catalogue

Open the **Function Catalogue** from the Workbench Toolbar, either by selecting Tools | Function Catalogue from the menu or by pressing the Function Catalogue button.

The **Function Catalogue** is split into two panes, the Function Tree and the Function Details (see screenshot below).



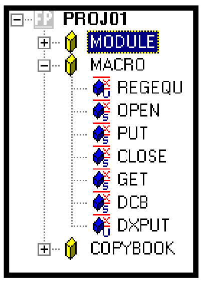
Name: MODULE
Type: Bysalem
Lines: 595
Comments: 106
Complexity: D
McCabe: 75

Child Functions

Name	Lines	Complexity	McCabe
FMT001A0	25	A	2
FMT001A1	65	C	18
FMT001A2	27	A	2
FMT001A3	68	A	2
FMT001A4	279	C	48
FMT001A5	26	A	3
FMT001A6	22	A	3
FMT001A7	22	A	5
FMT200P9	23	A	3
FMT101			
FMT102			
FMT103			
FMT104			
FMT105			
FMT106			
FMT110			
FMT111			

11.1 The Function Tree

The Function Tree pane shows the hierarchy of Functions contained within the FERMAT Project. A Function is the generic term for all source modules, macros, copybooks and groupings thereof. The Function Tree may be expanded and contracted by the user at will.



Software Migrations Ltd

Each Function in the tree is allocated an icon automatically by the Workbench, according to its type. The following Function types are supported in this release:

- ❑ System (yellow 3D box icon) - A high-level 'grouping' Function. For now there will only be three instances of the system type. These are `MODULE`, `MACRO` and `COPYBOOK`, which are automatically created when files are first imported into a new Project.
- ❑ Assembler Module (blue 3D box icon) - Assembler Modules generally relate to a physical file containing source of some kind. In addition to the standard Assembler Module, there are three special types of Module:
  - ♦ Copybook (blue 3D box icon, with 'C' in bottom right corner) - A common Module explicitly called using an Assembler `COPY` statement.
  - ♦ User Macro (blue 3D box icon, with 'U' in bottom right corner) - A non-system Assembler Macro utilised by one or more Modules in the Project.
  - ♦ Internal Macro (blue 3D box icon, with 'I' in bottom right corner) - A non-system Assembler Macro defined in one or more Modules in the Project but not stored in a User Macro file.
  - ♦ System Macro (blue 3D box icon, with 'S' in bottom right corner) - A reference to an Assembler System Macro utilised by one or more Modules in the Project.
- ❑ Routine (Red 3D box icon) - Assembler sub-routine. Note that no metrics are kept for this type.
- ❑ Entry Point (Light Blue 3D box icon) – Entry Point in an assembler module. Note that no metrics are kept for this type.

Furthermore, each Module in the **Function Catalogue** may be either internal or external. An internal Module is one that relates to a physical source file in the FERMAT Project; i.e. the source is available. Only internal Modules (shortened to just 'Modules') may be 'made current' in the Workbench, whether source code, macro or copybook.

- An external Module appears in the **Function Catalogue** to show that although it is utilised by one or more [internal] Modules, the source is not available in this Project. This has two uses: the identification of missing source or listings and the documentation of Project boundaries.

An external Function of any type is indicated by a red 'X' in the top right corner of its icon.

Note: Modules can be added to a FERMAT Project at any time. Where a newly imported module was previously identified as an external dependency, it will effectively be 'promoted' to be an internal module.

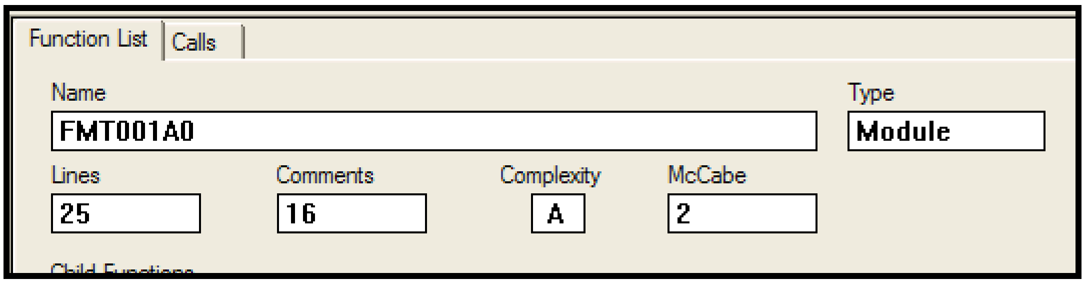
Finally:

- ✓ Single-clicking an item in the Function Tree will retrieve its details and present them in the Function Details pane (see next section).

- ✓ Double-clicking an item in the Function Tree - provided that it is an internal Module - will make it the Current Module in the Workbench.

11.2 Function Details

The Function Details pane includes two pages, accessed from the appropriate tab at the top of the pane. Each page has a standard header showing the name and type[2] of the selected Function.

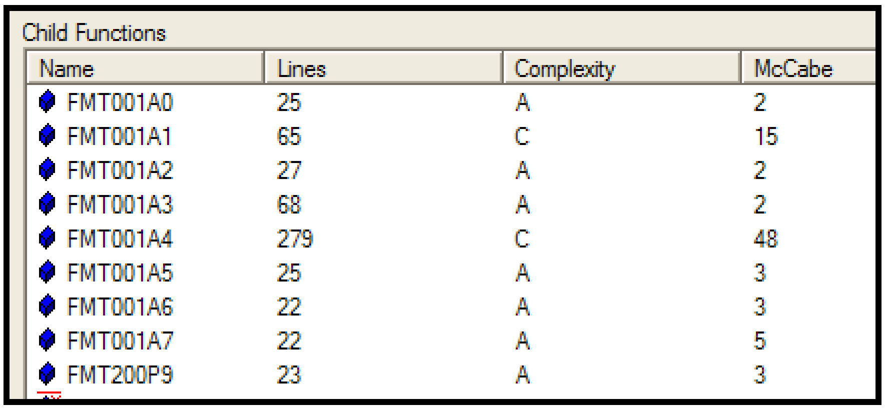


Name	Type
FMT001A0	Module

Lines	Comments	Complexity	McCabe
25	16	A	2

The first page - the Function List - displays the selected Function's immediate children (much like Windows Explorer).

[2] i.e. the textual description of the Function type icon.



Child Functions

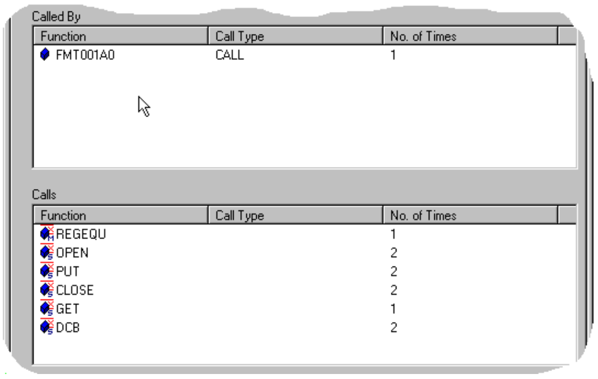
Name	Lines	Complexity	McCabe
FMT001A0	25	A	2
FMT001A1	65	C	15
FMT001A2	27	A	2
FMT001A3	68	A	2
FMT001A4	279	C	48
FMT001A5	25	A	3
FMT001A6	22	A	3
FMT001A7	22	A	5
FMT200P9	23	A	3

The sort order for each column toggles between ascending and descending; apart from in the Function Name column, where it toggles between the original order and ascending order (which may be the same in some cases).

11.3 Function Details Calls Page

The next page - the Calls page - displays two lists:

- a list of Functions that currently call the selected Function (i.e. 'Called By'),
- a list of Functions that the selected Function invokes (i.e. 'Calls').



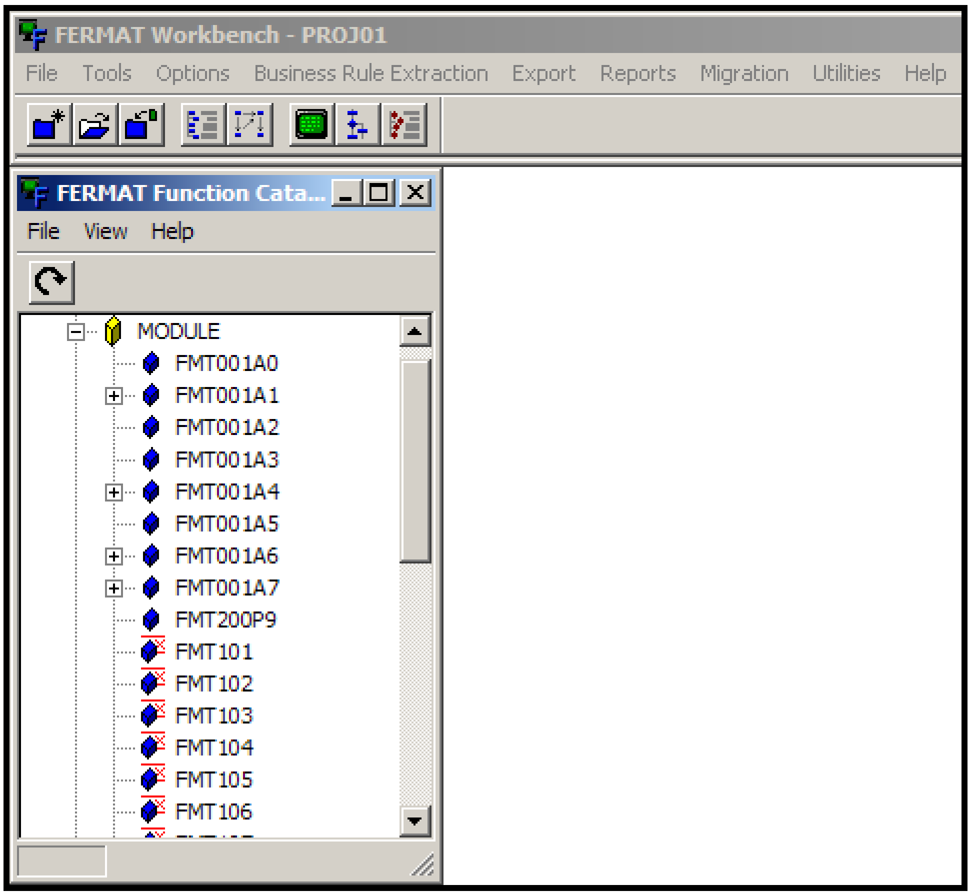
Called By

Function	Call Type	No. of Times
FMT001A0	CALL	1

Calls

Function	Call Type	No. of Times
REGEQU		1
OPEN		2
PUT		2
CLOSE		2
GET		1
DCB		2

The details panel on the Function Catalogue can be hidden from display to resize the Function Catalogue and make more room for other workbench tools on the desktop. Select View | Show Details and deselect it, this will display only the Function Tree information; as below.


FERMAT Workbench - PROJ01
File Tools Options Business Rule Extraction Export Reports Migration Utilities Help
FERMAT Function Cata...
File View Help

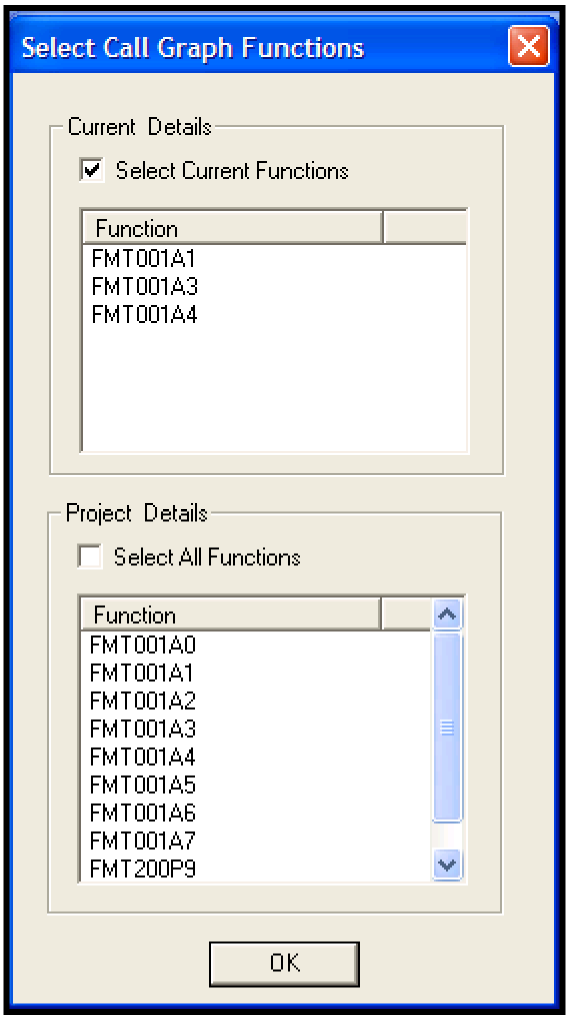
MODULE
FMT001A0
FMT001A1
FMT001A2
FMT001A3
FMT001A4
FMT001A5
FMT001A6
FMT001A7
FMT200P9
FMT101
FMT102
FMT103
FMT104
FMT105
FMT106

12. The Function Call Graph

Open the **Function Call Graph** from the Workbench Toolbar, either by selecting

Tools | Function Call Graph from the menu or by pressing the Function Call Graph button.

The **Function Call Graph** complements the **Function Catalogue** by graphically depicting the calling relationships between the Modules. When selected, the user is presented with a list as below:



Select Call Graph Functions

Current Details

☑ Select Current Functions

Function
FMT001A1
FMT001A3
FMT001A4

Project Details

☐ Select All Functions

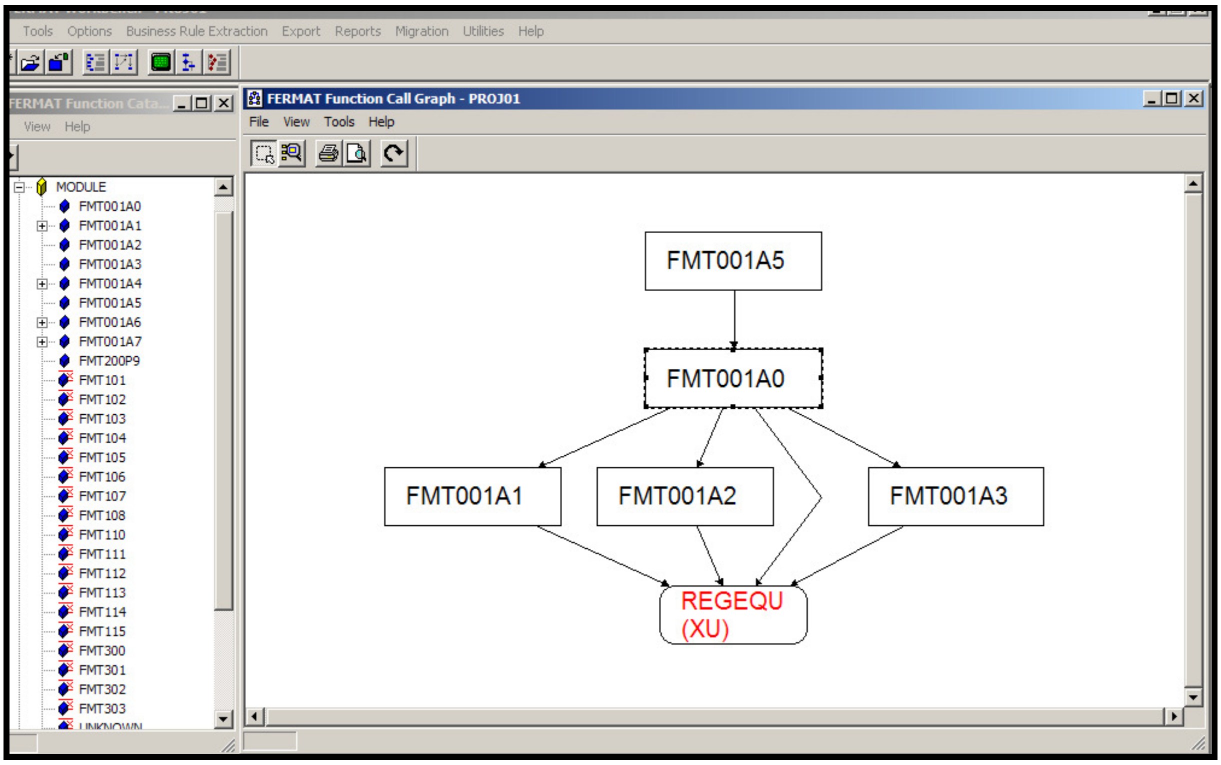
Function
FMT001A0
FMT001A1
FMT001A2
FMT001A3
FMT001A4
FMT001A5
FMT001A6
FMT001A7
FMT200P9

The display is divided into two sections: the first provides details of the current modules selected; on first entry this will be empty. If the Call Graph has been used before in this project, this section will be populated by the modules selected when previously used. Checking the Select Current Functions, will use these functions as the basis for the call graph, Modules can be deselected by right clicking on the modules and selecting the Delete option. Modules can be added to the Current Selection by selecting one or more modules from the second part of the display Project Details, modules are selected either individually or by selecting a block using shift left click or multiple individual entries by using control left click.

By default none of the modules are selected, the Select All Functions tick box can be checked; this will display the Call Graph for the entire project. Depending upon the complexity and size of the project this may result in large complex diagrams.

An arrow from one Module to another Module depicts the first calling the second. Note that by default, Copybooks and Macros are not shown on the graph for clarity; but in the View menu, Function Types offers you a choice of what to display, including copybooks, system macros, user macros and unknown modules.

Called Modules which are external to your Project are shown on the graph with red text and with an (X), while modules which are loaded are shown in a parallelogram.



Note that some organisations utilise a User Macro to call other Modules, in place of the standard `CALL` instruction. Such a User Macro may be designated as a calling macro in a special table, with the parameter taken to be the target Module. This enables the **Function Call Graph** to correctly show a call to the intended Module. See section 30 Customising the Workbench for further details.

12.1 Viewing the Call Graph

There are four methods of sizing the **Function Call Graph**:

- Scaling the graph using the available options in the View menu.
- Using the plus and minus keyboard keys for zooming in/out.

- Using the zoom cursor (second toolbar button) to rubber-band the area to be displayed in the window.
- Using the Overview Window (View | Overview Window from the menu or Ctrl+Shift+O)
  - in conjunction with the zoom cursor above, and/or
  - by resizing the marquee (shaded border) by dragging its corners.

There are also different ways of moving around the graph:

- Using the scrollbars, cursor keys, or Page Up and Page Down keys.
- Dragging the marquee around in the Overview window with the mouse.

12.2 Synchronising the Function Catalogue and the Function Call Graph

Right-click a Module in the **Function Call Graph** and select Show in Function Catalogue from the context menu. The **Function Catalogue**’s Function Tree highlights the selected Module.

Double-click an internal Assembler Module in the **Function Catalogue**’s Function Tree. The **Function Call Graph** centres on the selected Module. (If the **Function Call Graph** is hidden from view, simply click its button on the Workbench Toolbar to give it focus.)

When you performed the last action you may have noticed that not only did the **Function Call Graph** centre on the Module selected in the **Function Catalogue**, but the Workbench Toolbar now displays the name of the selected Module in its title bar.

Furthermore, if you have already opened some of the other Tools you will see that they now display details of the selected Module. This is because by double-clicking a Module in the Function Tree you have changed the Current Module.

13. Selecting the Workbench's current Module

The previous sections described the **Function Catalogue** and **Function Call Graph**, these two Tools are used to select the Current Module, against which the **Text Editor**, **Program Flowchart** and **Data Catalogue** will synchronise if opened, or are already open.

To select a Function (Module, Macro or Copybook)

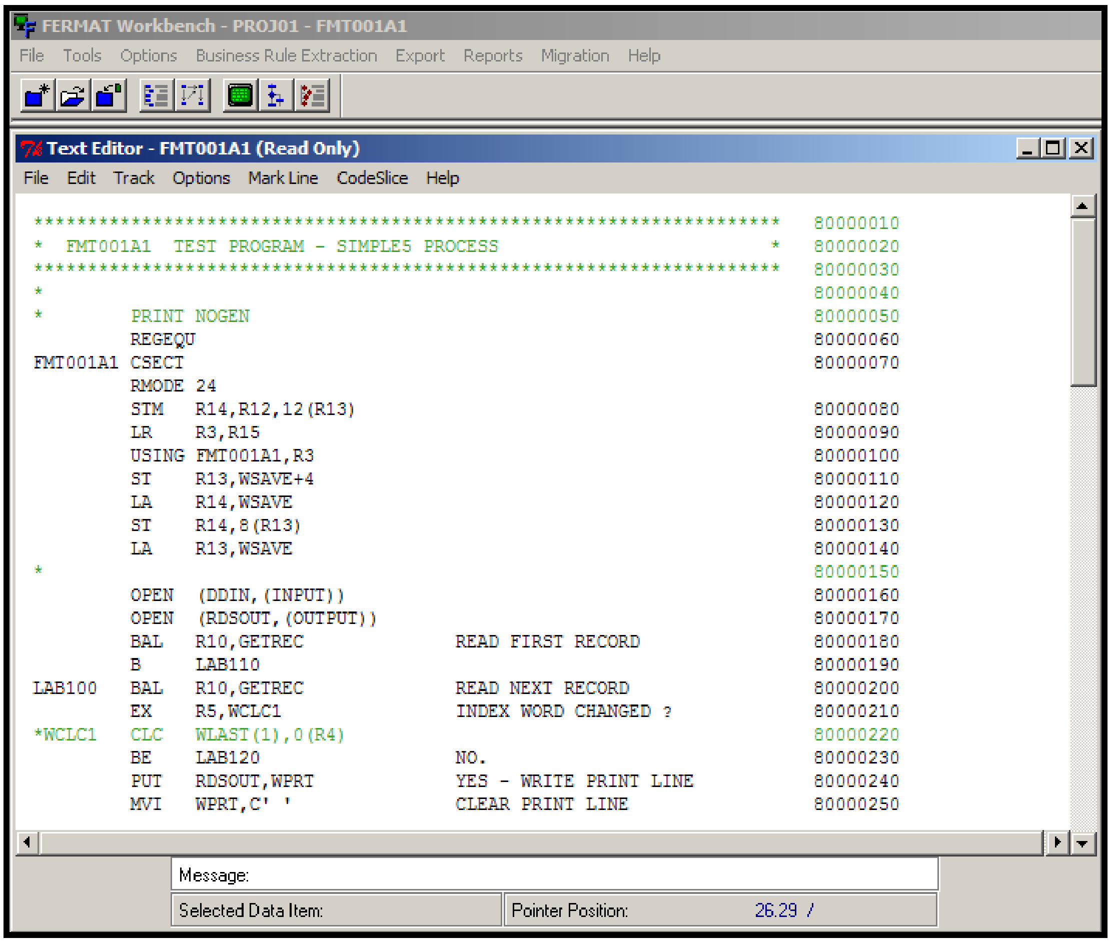
- From the **Function Catalogue**, double-click a Function in the Function Tree. (Only an internal one - it makes little sense to make an external Module current, because no information is known about it).

Or

- From the **Function Call Graph**, double-click a Module in the main graph window. (If instead you just want to retrieve its details in the **Function Catalogue** without re-synchronising the open Tools, select Show in Function Catalogue from the context menu).

14. The Text Editor

The TEXT Editor will by default open the module in Read Only mode, this feature enables multiple users to work on the same project simultaneously. A module opened in Read Only mode will have the message displayed on the Title Line of the Text Editor, as below:



Text Editor - FMT001A1 (Read Only)

```
***********************************************************************   80000010
*  FMT001A1  TEST PROGRAM - SIMPLE5 PROCESS                          *   80000020
***********************************************************************   80000030
*                                                                         80000040
*        PRINT NOGEN                                                      80000050
         REGEQU                                                           80000060
FMT001A1 CSECT                                                            80000070
         RMODE 24
         STM   R14,R12,12(R13)                                            80000080
         LR    R3,R15                                                     80000090
         USING FMT001A1,R3                                                80000100
         ST    R13,WSAVE+4                                                80000110
         LA    R14,WSAVE                                                  80000120
         ST    R14,8(R13)                                                 80000130
         LA    R13,WSAVE                                                  80000140
*                                                                         80000150
         OPEN  (DDIN,(INPUT))                                             80000160
         OPEN  (RDSOUT,(OUTPUT))                                          80000170
         BAL   R10,GETREC               READ FIRST RECORD                 80000180
         B     LAB110                                                     80000190
LAB100   BAL   R10,GETREC               READ NEXT RECORD                  80000200
         EX    R5,WCLC1                 INDEX WORD CHANGED ?              80000210
*WCLC1   CLC   WLAST(1),0(R4)                                             80000220
         BE    LAB120                   NO.                               80000230
         PUT   RDSOUT,WPRT              YES - WRITE PRINT LINE            80000240
         MVI   WPRT,C' '                CLEAR PRINT LINE                  80000250
```

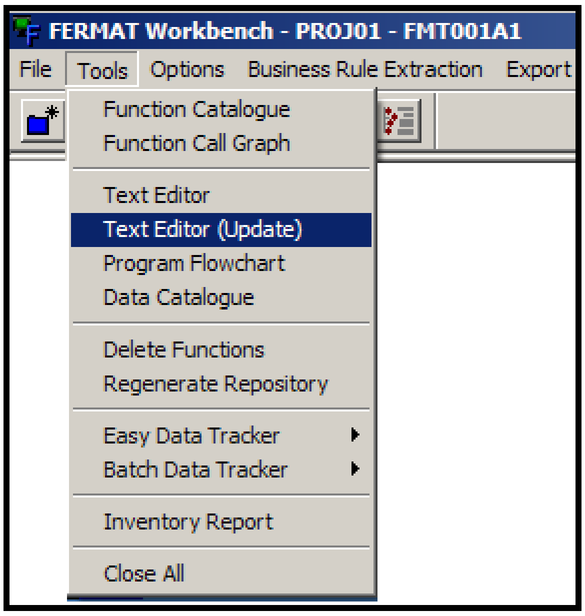
Message:

Selected Data Item: | Pointer Position: 26.29 /

When a module is opened in Read Only Mode all the Text Editor Functions are available to the user including marking lines and blocks to aid comprehension but the changes made to the module cannot be saved.

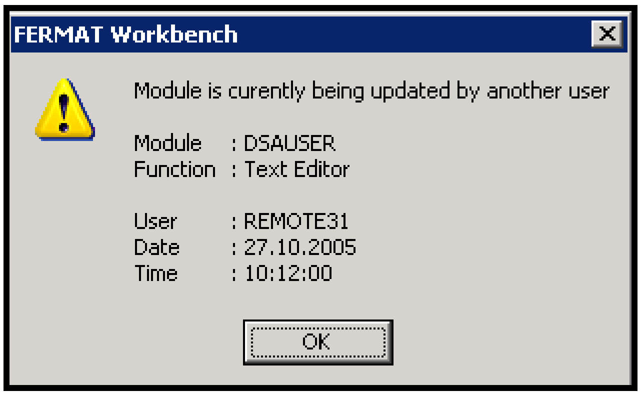
In order to make changes to a module (save and store) annotations marked lines etc the module must be opened for Update.

To open a module for Update Select Tools | Text Editor (Update) as below:



When a module is opened for Update this is recorded in the Workbench internal lock tables, any module locked for update will not be available for Text Editor Update for any other user or for other system wide functions that require access to locked module.

Note the internal lock tables function on a project wide basis. If a user attempts to open a module in Update Mode that is already checked out by another user the following message will be displayed.


FERMAT Workbench

Module is curently being updated by another user

```
Module    : DSAUSER
Function  : Text Editor

User      : REMOTE31
Date      : 27.10.2005
Time      : 10:12:00
```

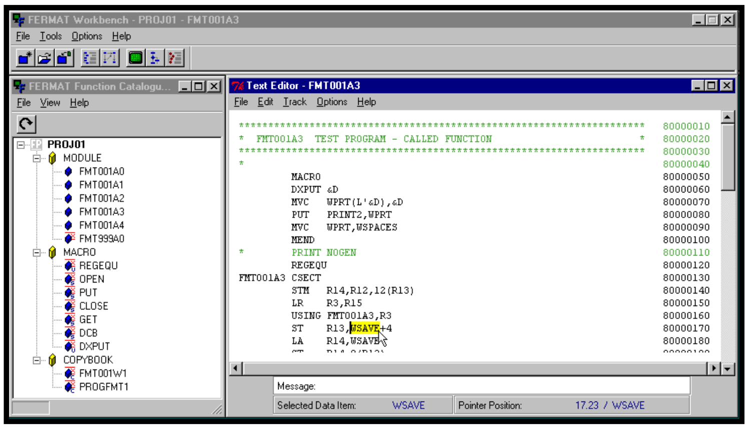
The message details who has checked the module out for updating together with the time and date, to view all modules currently locked use the View Lock Table command – See section 26 Admin for further details.

Select a Module, such as FMT001A1 (in Proj01); from either the **Function Call Graph** or the **Function Catalogue**.

Open the **Text Editor** (Update) from the Workbench Toolbar, Tools | Text Editor (Update) from the menu.

Software Migrations Ltd

If you performed the above actions, then the Text Editor will have opened the source for whichever module you selected as the Current Module.



```
*********************************************************************   80000010
*  FMT001A3  TEST PROGRAM - CALLED FUNCTION                         *   80000020
*********************************************************************   80000030
*                                                                       80000040
         MACRO                                                          80000050
         DXPUT &D                                                       80000060
         MVC   WPRT(L'&D),&D                                            80000070
         PUT   PRINT2,WPRT                                              80000080
         MVC   WPRT,WSPACES                                             80000090
         MEND                                                           80000100
*        PRINT NOGEN                                                    80000110
         REGEQU                                                         80000120
FMT001A3 CSECT                                                          80000130
         STM   R14,R12,12(R13)                                          80000140
         LR    R3,R15                                                   80000150
         USING FMT001A3,R3                                              80000160
         ST    R13,WSAVE+4                                              80000170
         LA    R14,WSAVE                                                80000180
```

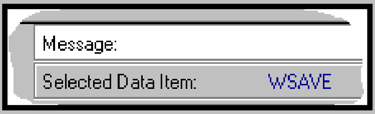
Message:

Selected Data Item: WSAVE   Pointer Position: 17.23 / WSAVE

Notice that the **Text Editor** is Assembler-aware:

- Comments are shown in green.
- The **Text Editor** is aware of the difference between Data Item[3] symbols and non-Data Item symbols (useful for Data Tracking - see next section).

Double-click a Data Item in the **Text Editor** (e.g. WSAVE). See that it is shown as the Selected Data Item in the bottom left of the editor window.



Message:

Selected Data Item: WSAVE

Now double-click a non-Data Item (such as an opcode, label or comment). See the message at the foot of the Text Editor informing you it is not a data item. The currently Selected Data Item is not changed.

14.1 Data Tracking in the Text Editor

The Data Tracking facilities in the **Text Editor** build on the strength that the editor can distinguish between symbols in the source that are Data Items and those that are not.

The Data Tracker consists of:

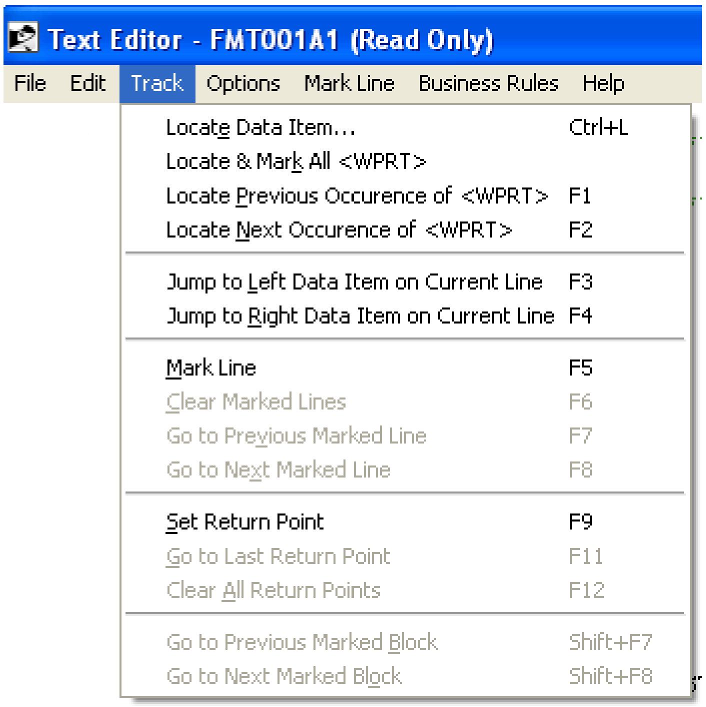
- ✓ a collection of 'one touch' find facilities that specifically work on Data Items,
- ✓ the facility to mark lines of source that need further work/investigation
- ✓ The facility to keep tabs on where you have been and where you still need to look, through the use of return points.

[3] Data Item is the generic term given to variables, constants, data structures, etc.

There are two scenarios when this assistance would be useful to the programmer:

- Debugging - starting with a Data Item whose value is known to be invalid, and working backwards through the code to find out where it might have got its erroneous value from.
- Enhancement - starting with a request to change the format of a given Data Item or the way that it is processed, and working forwards through the code to see what impact it would have before making the change.

The Track main menu item in the **Text Editor** contains the various 'locate' operations for Data Items, along with their Function Key assignments for 'one touch' operation.


Text Editor - FMT001A1 (Read Only)
File
Edit
Track
Options
Mark Line
Business Rules
Help
Locate Data Item... Ctrl+L
Locate & Mark All <WPRT>
Locate Previous Occurence of <WPRT> F1
Locate Next Occurence of <WPRT> F2
Jump to Left Data Item on Current Line F3
Jump to Right Data Item on Current Line F4
Mark Line F5
Clear Marked Lines F6
Go to Previous Marked Line F7
Go to Next Marked Line F8
Set Return Point F9
Go to Last Return Point F11
Clear All Return Points F12
Go to Previous Marked Block Shift+F7
Go to Next Marked Block Shift+F8

a. Ensure FMT001A1 is the Current Module.

b. Select Track | Locate Data Item… click in the dialogue box and type WPRT (you may have to overwrite the currently Selected Data Item). Click on Locate.

c. The first instance of WPRT is shown highlighted in yellow and set as the currently Selected Data Item. (It occurs on the previous line only as a macro argument.)

d. Use F9 (Set Return Point) to remember where we are. The start of the line is highlighted in green to show it is a Return Point. (Note: F9 would toggle it off again.)

e. Use F2 (Locate Next Occurrence...) to find the next occurrence of WPRT. Press F2 two more times until you find the instruction LA R8,WPRT on line 80000290.

f. At this point press F9 (Set Return Point) again to set where to come back to.

g. Use F1 to track WPRT backwards once from this point.

h. Use F11 to return to line 80000290. The return point (green start to the line) has been cleared now that you have looked down this branch. The currently Selected Data Item is WPRT to show us this is what we were tracking.

i. Use F5 (Mark Line) to mark a line of interest; then, as if changing your mind, press F5 again to unmark the line.

j. Use Track | Locate & Mark All <WPRT> to find every reference to WPRT.

14.2 Text Editors Shortcut Keys

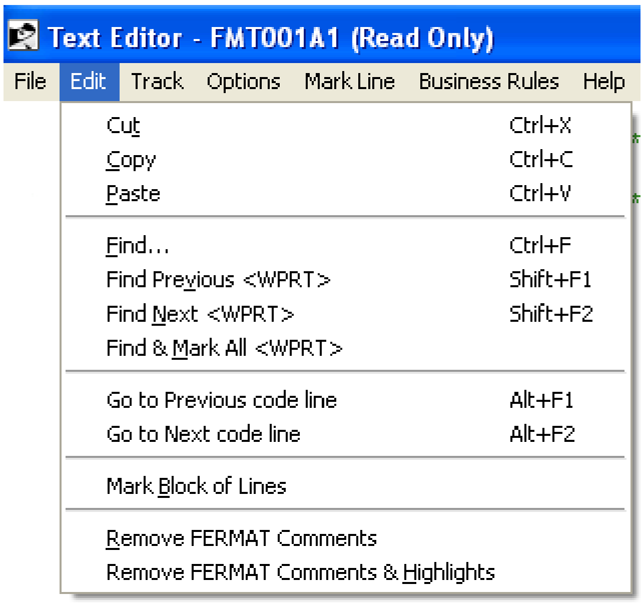
The following table shows the keyboard shortcuts available in the Text Editor:

Key	Function	Key	Function
F3	Jump left data item	Cntl-L	Locate Data item
F4	Jump right data item	F1	Locate Previous
F5	Mark Line with current mark	F2	Locate Next
F6	Clear marked lines	Cntl-F	Find Text
F7	Previous marked line	Shift-F1	Find Previous
F8	Next marked line	Shift-F2	Find Next
Shift-F7	Previous marked block	Alt-F1	Go to previous code line
Shift-F8	Next marked block	Alt-F2	Go to next code line
F9	Set return point		
F11	Goto previous return point		
F12	Clear all return points		
Cntl-X	Cut	Alt-F4	Close Editor
Cntl-C	Copy	Cntl-P	Previous Line
Cntl-V	Paste	Cntl-N	Next Line
Cntl-Z	Undo	Home	Start of line
Cntl-Y	Redo	End	End of line
		Cntl-Left	Previous non space char
Cntl-M	Mark block of lines	Cntl-Right	Next non space char
Cntl-U	Unmark block of lines	Cntl-Up	Top of file
		Cntl-Down	End of file
Cntl-*n*	Mark line as type *n* (0-9)	Cntl-Home	Top of file
Cntl-Shift-*n*	Mark block as type *n* (1-9)	Cntl-End	End of file
Mouse			
Double click	Select data item		
Single click	Remove highlight		
Right click	Popup menu		

Edit function of the Text Editor
The edit facility differs from the tracking facility, in that whilst it is still aware of the data elements it enables free string searching of all text including comments, labels in the source code itself.

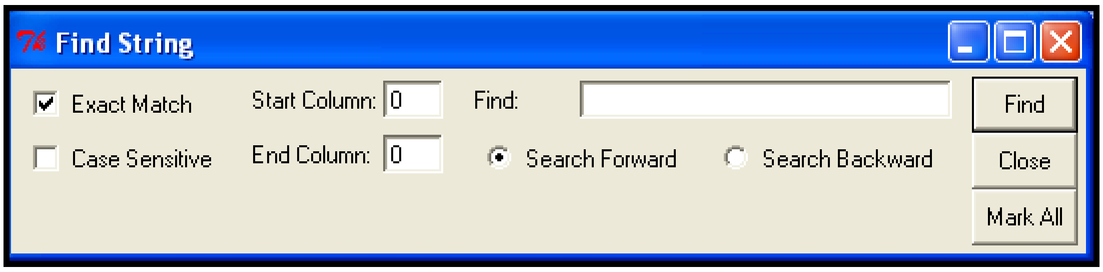
The context menu of the Edit function –

Software Migrations Ltd


Text Editor - FMT001A1 (Read Only)
File
Edit
Track
Options
Mark Line
Business Rules
Help
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Find... Ctrl+F
Find Previous <WPRT> Shift+F1
Find Next <WPRT> Shift+F2
Find & Mark All <WPRT>
Go to Previous code line Alt+F1
Go to Next code line Alt+F2
Mark Block of Lines
Remove FERMAT Comments
Remove FERMAT Comments & Highlights

Includes standard editing features such as Cut, Copy and Paste in addition to the workbench specific find and mark text.

Selecting the Find function will display the following dialogue box



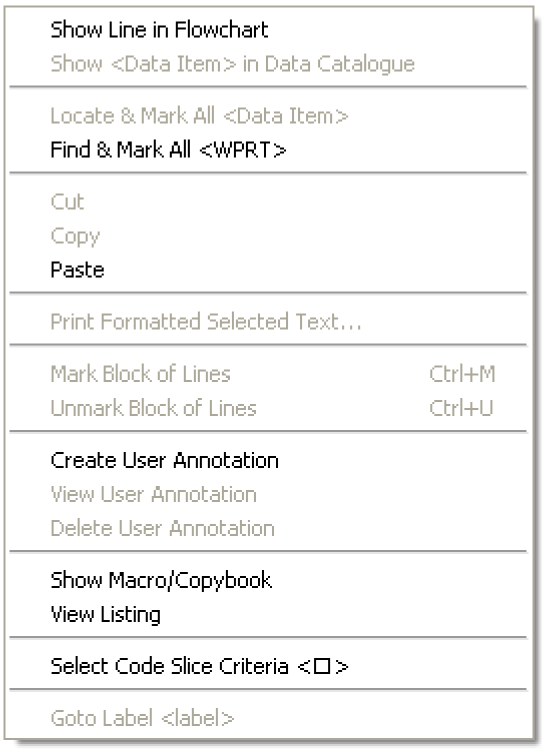
The string to be searched for should be entered in the find area, the search can be limited to search for strings within certain columns by entering the start and end column positions. The Exact Match checkbox is used to search for actual strings. When unchecked, it performs a wild card search for any strings containing the find string. This function also provides the facility for matching strings using case sensitivity. Searches can be either forward or backwards (Shift F2 and Shift F1 in the editor itself) and can either find the next occurrence or Mark All occurrences.

Going to the next or previous lines of code can be initiated from the edit menu or using the keyboard shortcut. This will position the cursor on the next or previous line that does not begin with an asterisk, enabling the skipping of large comment blocks.

Marking Blocks of code can also be achieved using the edit facility. With a block of code selected in the text editor, selecting the Edit function on the tool bar will enable these lines to be marked as a block of code and automatically highlighted. Note blocks of code marked using this function cannot be unmarked using the Edit function, they can only be removed using the right click function on selected text.

14.3 Right click function in the text editor

Simply right clicking the mouse button in the text editor will bring up the following context menu.


Show Line in Flowchart
Show <Data Item> in Data Catalogue
Locate & Mark All <Data Item>
Find & Mark All <WPRT>
Cut
Copy
Paste
Print Formatted Selected Text...
Mark Block of Lines Ctrl+M
Unmark Block of Lines Ctrl+U
Create User Annotation
View User Annotation
Delete User Annotation
Show Macro/Copybook
View Listing
Select Code Slice Criteria <☐>
Goto Label <label>

As no text was selected in the text editor, many of the options are not applicable and are greyed out.

Of those available,

Show Line in Flowchart is used to centre the flowchart on the line of code in the text editor where the cursor is pointing. If the flowchart is not currently opened this action automatically opens the flowchart utility and centres on the line of code (the flow charter is covered in detail later in this guide).

Find & Mark All <….> provides the option to mark all occurrences of the previous Edit | Find activity as described above.

Create User Annotation is detailed later in this guide.

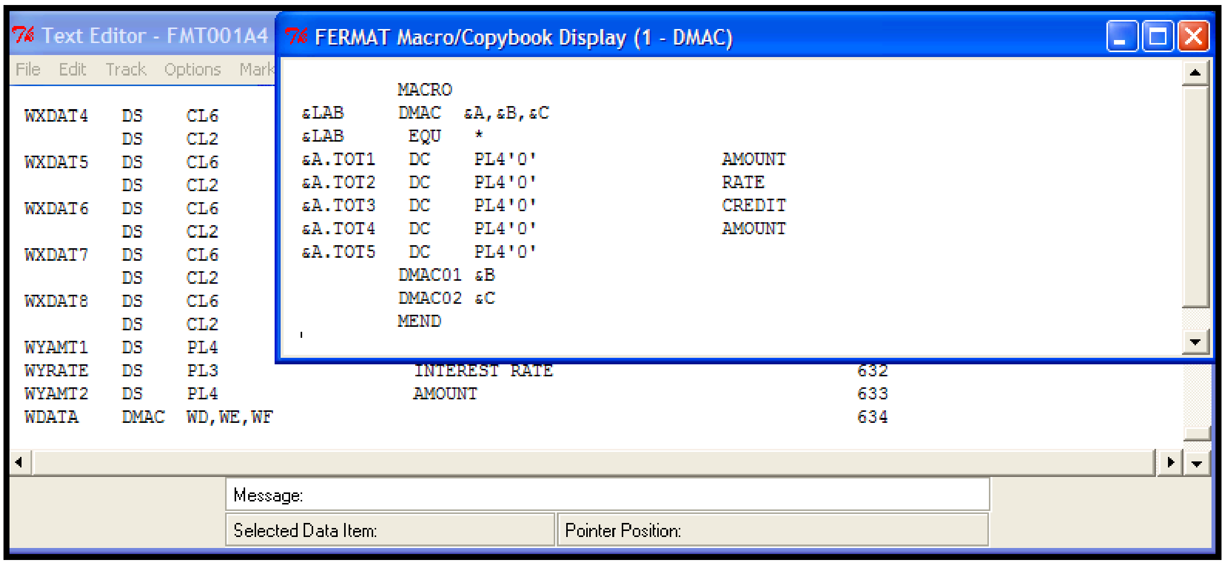
When one or more lines are selected in the text editor, by holding down the left mouse button and right clicking, the following options also become available.

- Cut, Copy and Paste
- Print Selected Text
- Mark block of Lines

14.4 Show Macro/Copybook

By selecting the name of a macro or copy book in the Text Editor it is possible to view the source of the macro or copybook directly. By selecting the Show Macro/Copybook option the Workbench will produce a pop up screen which will display the chosen item; this pop-up screen allows the user to view the Macro or Copy book only.

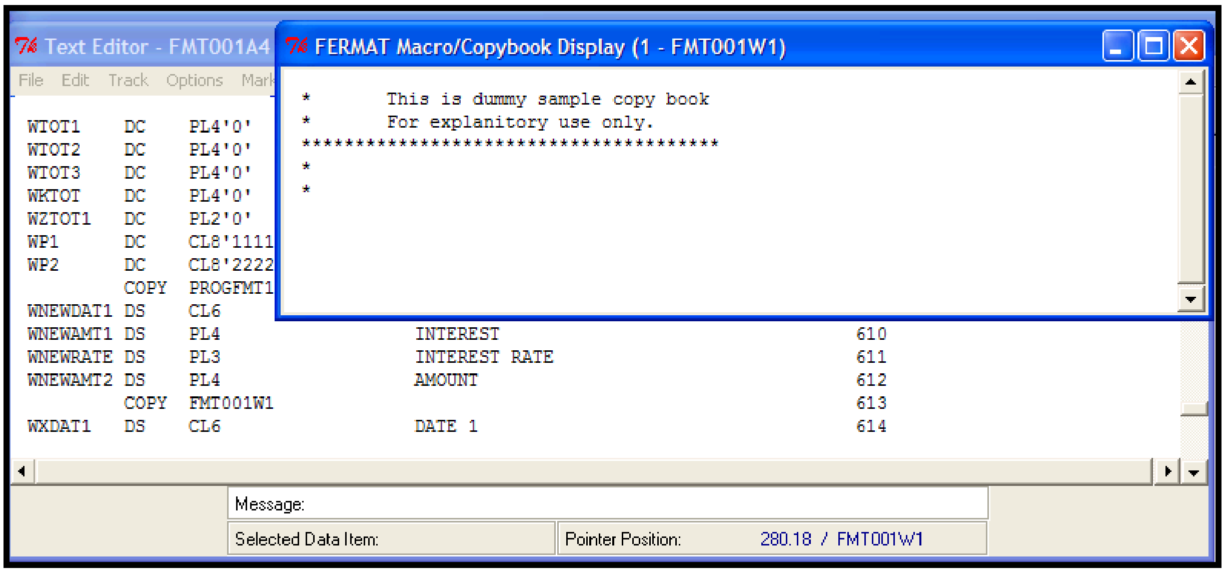
The Following example shows the Macro DMAC as referenced on Line 634 of the Module FMT001A4.


Text Editor - FMT001A4
File Edit Track Options Mark
FERMAT Macro/Copybook Display (1 - DMAC)
Message:
Selected Data Item:
Pointer Position:

```
         MACRO
&LAB     DMAC  &A,&B,&C
&LAB      EQU   *
&A.TOT1   DC    PL4'0'                AMOUNT
&A.TOT2   DC    PL4'0'                RATE
&A.TOT3   DC    PL4'0'                CREDIT
&A.TOT4   DC    PL4'0'                AMOUNT
&A.TOT5   DC    PL4'0'
         DMAC01 &B
         DMAC02 &C
         MEND
```

```
WXDAT4   DS    CL6
         DS    CL2
WXDAT5   DS    CL6
         DS    CL2
WXDAT6   DS    CL6
         DS    CL2
WXDAT7   DS    CL6
         DS    CL2
WXDAT8   DS    CL6
         DS    CL2
WYAMT1   DS    PL4
WYRATE   DS    PL3                INTEREST RATE                     632
WYAMT2   DS    PL4                AMOUNT                            633
WDATA    DMAC  WD,WE,WF                                             634
```

This example shows the Copybook FMT001W1 referenced on Line 614 of the Module FMT001A4.


Text Editor - FMT001A4

FERMAT Macro/Copybook Display (1 - FMT001W1)

File Edit Track Options Mark

```
*       This is dummy sample copy book
*       For explanitory use only.
***************************************
*
*
```

```
WTOT1     DC    PL4'0'
WTOT2     DC    PL4'0'
WTOT3     DC    PL4'0'
WKTOT     DC    PL4'0'
WZTOT1    DC    PL2'0'
WP1       DC    CL8'1111
WP2       DC    CL8'2222
          COPY  PROGFMT1
WNEWDAT1  DS    CL6
WNEWAMT1  DS    PL4                 INTEREST                                 610
WNEWRATE  DS    PL3                 INTEREST RATE                            611
WNEWAMT2  DS    PL4                 AMOUNT                                   612
          COPY  FMT001W1                                                     613
WXDAT1    DS    CL6                 DATE 1                                   614
```

Message:

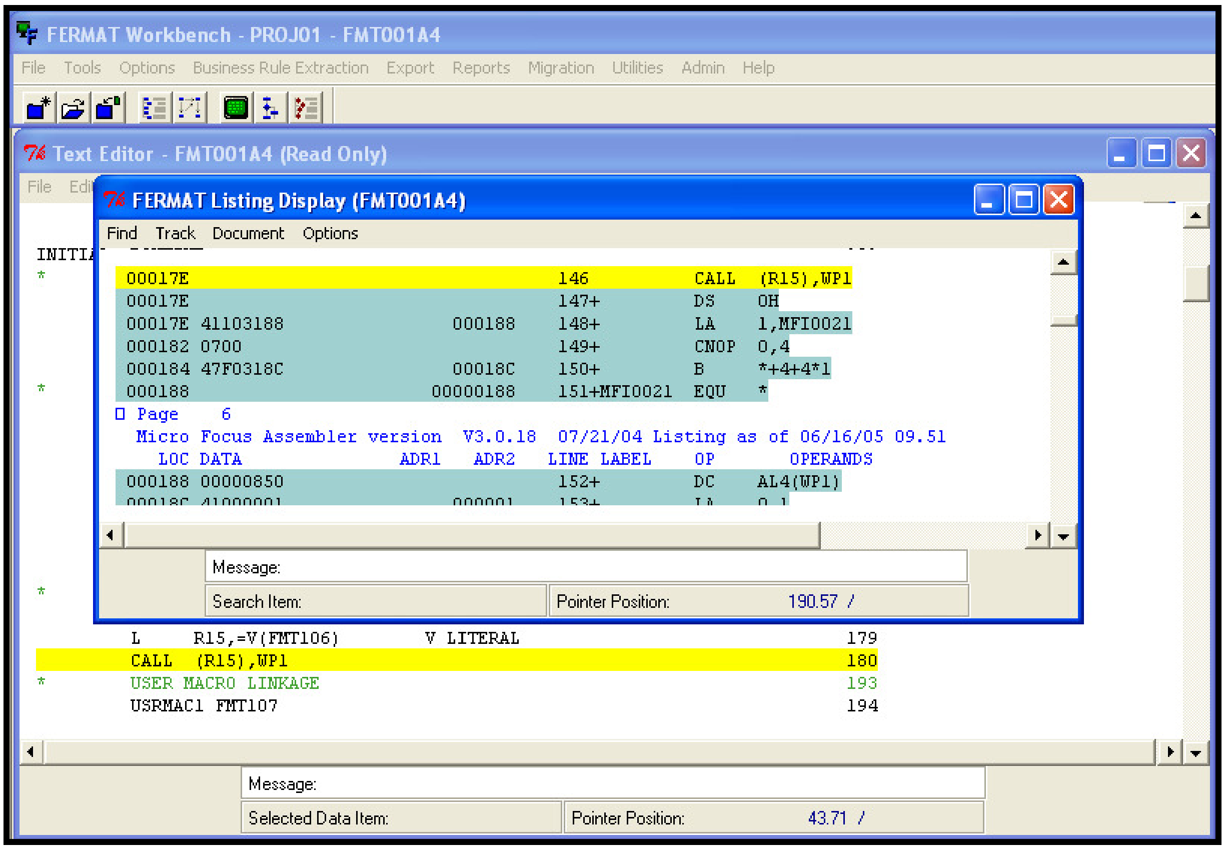
Selected Data Item:

Pointer Position: 280.18 / FMT001W1

14.5 View Listing

If listings have been used as import to the Workbench then selecting the View Listing will open a pop-up screen that will display the original listing, the display will start on the line selected in the Text Editor. This function is particularly useful for debugging where the user may be required to analyse the object code to understand displacements and lengths of elements.

The following example shows the original listing when line 180 is selected in the Text Editor for module FMT001A4.


FERMAT Workbench - PROJ01 - FMT001A4
File Tools Options Business Rule Extraction Export Reports Migration Utilities Admin Help
Text Editor - FMT001A4 (Read Only)
File Edi
FERMAT Listing Display (FMT001A4)
Find Track Document Options

```
00017E                                   146          CALL   (R15),WP1
00017E                                   147+         DS     0H
00017E 41103188               000188     148+         LA     1,MFI0021
000182 0700                              149+         CNOP   0,4
000184 47F0318C               00018C     150+         B      *+4+4*1
000188                      00000188     151+MFI0021  EQU    *
0 Page     6
  Micro Focus Assembler version  V3.0.18  07/21/04 Listing as of 06/16/05 09.51
    LOC DATA               ADR1    ADR2   LINE LABEL     OP        OPERANDS
000188 00000850                          152+         DC     AL4(WP1)
```

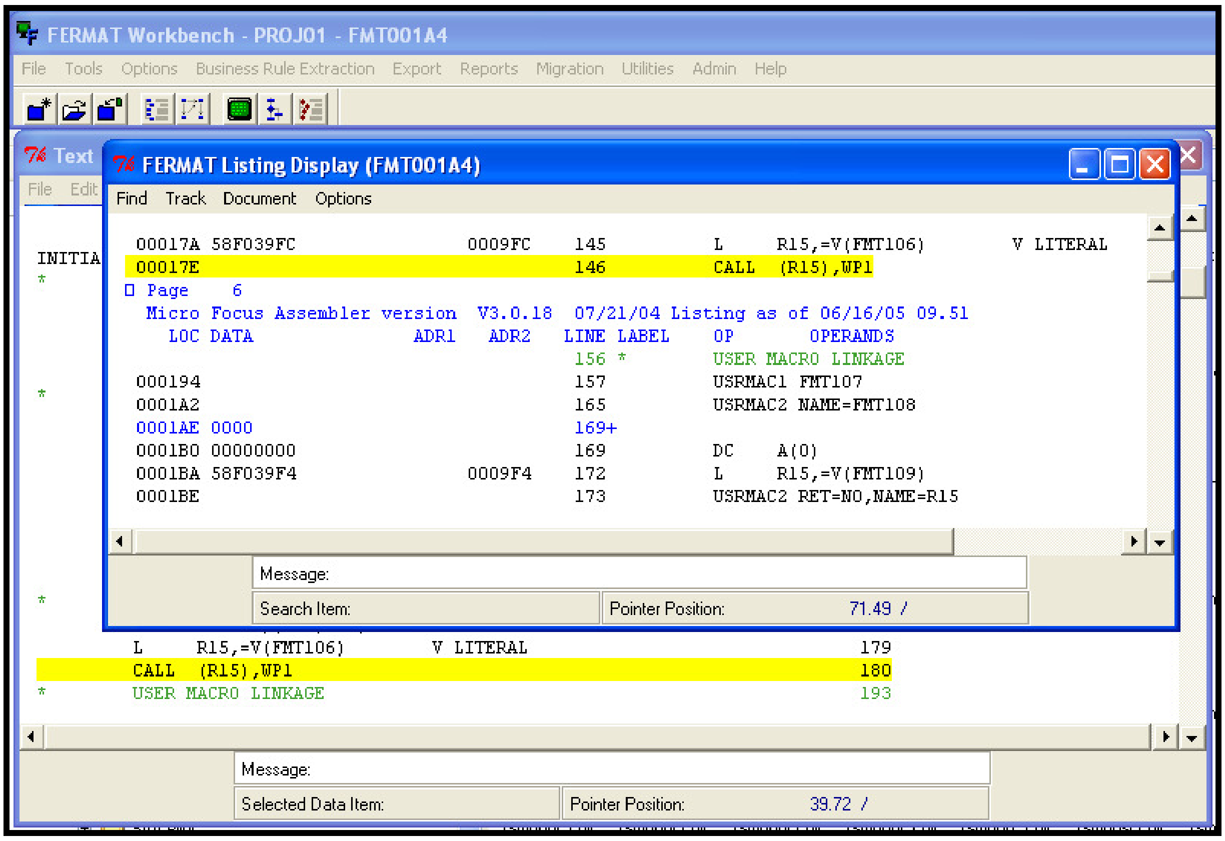
Message:
Search Item: Pointer Position: 190.57 /

```
INITIA
*
*
*
        L      R15,=V(FMT106)         V LITERAL                             179
        CALL   (R15),WP1                                                    180
*       USER MACRO LINKAGE                                                  193
        USRMAC1 FMT107                                                      194
```

Message:
Selected Data Item: Pointer Position: 43.71 /

As can be seen from the above example user comments appear as in the Workbench Text Editor in green, whilst lines generated in the listings such as page headers etc are displayed in Blue text.

By default all macro expansions lines are displayed and colour coded, if required the user can turn off the display of macro expansion lines by using the Options facility on the View Listings menu. This would result in following display.


FERMAT Workbench - PROJ01 - FMT001A4
File Tools Options Business Rule Extraction Export Reports Migration Utilities Admin Help
Text
File Edit
FERMAT Listing Display (FMT001A4)
Find Track Document Options

```
INITIA
 00017A 58F039FC              0009FC    145          L     R15,=V(FMT106)        V LITERAL
 00017E                                 146          CALL  (R15),WP1
 Page     6
   Micro Focus Assembler version  V3.0.18   07/21/04 Listing as of 06/16/05 09.51
     LOC DATA                 ADR1    ADR2    LINE  LABEL     OP        OPERANDS
                                          156 *         USER MACRO LINKAGE
 000194                                   157           USRMAC1  FMT107
 0001A2                                   165           USRMAC2  NAME=FMT108
 0001AE 0000                              169+
 0001B0 00000000                          169           DC     A(0)
 0001BA 58F039F4              0009F4      172           L      R15,=V(FMT109)
 0001BE                                   173           USRMAC2 RET=NO,NAME=R15
```

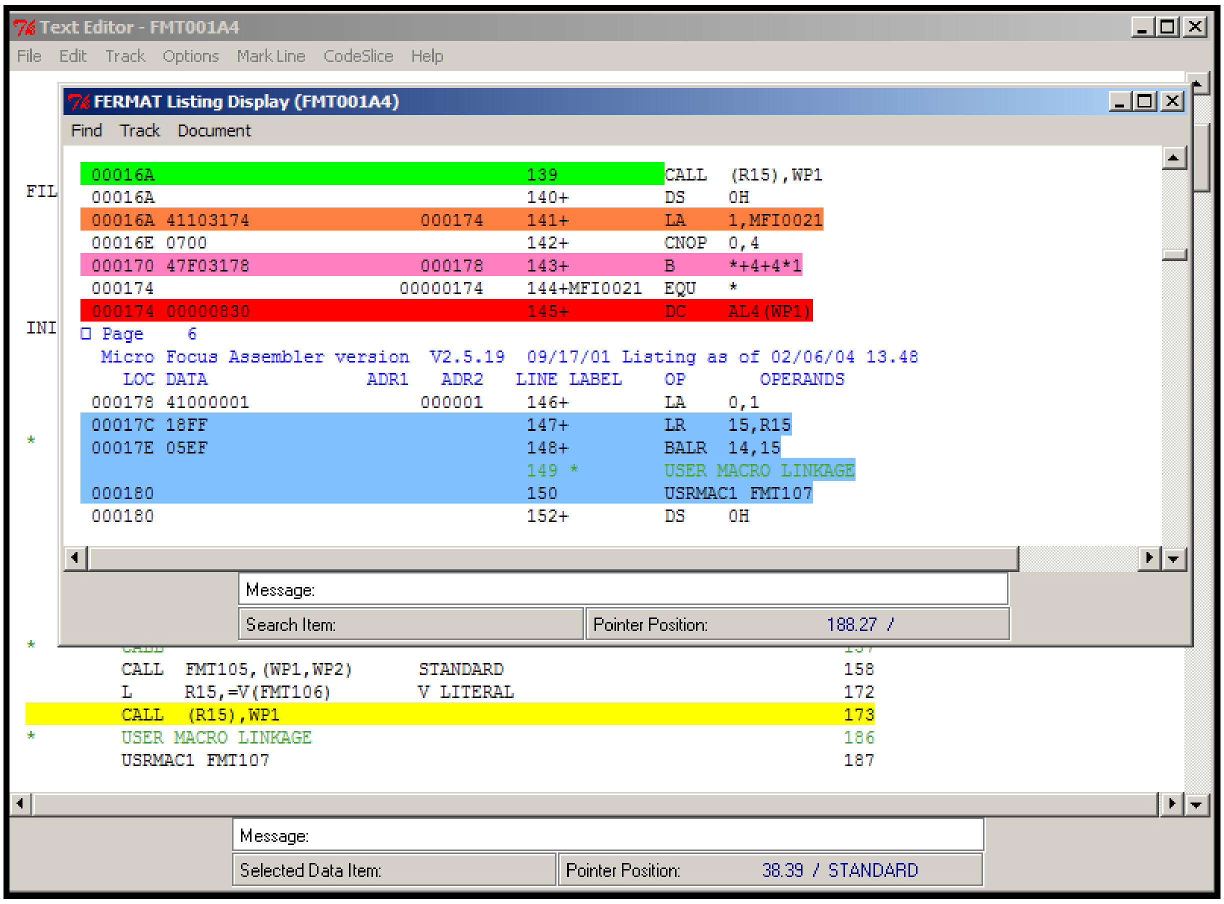
Message:
Search Item:
Pointer Position: 71.49 /

```
        L      R15,=V(FMT106)        V LITERAL                    179
        CALL   (R15),WP1                                          180
*       USER MACRO LINKAGE                                        193
```

Message:
Selected Data Item:
Pointer Position: 39.72 /

14.6 Analysing a Listing

To aid programmers in comprehending a listing of a module, the View listing option allows for certain local editing and marking of lines. As in the Text Editor, users can set return points in the code F9 key, mark individual lines in various colours Cntrl + 0 to 9 will mark a line in the colour defined for that marked line type as specified in the Text Editor. As below:



```
00016A                                    139      CALL   (R15),WP1
00016A                                    140+     DS     0H
00016A 41103174              000174       141+     LA     1,MFI0021
00016E 0700                               142+     CNOP   0,4
000170 47F03178              000178       143+     B      *+4+4*1
000174                     00000174       144+MFI0021  EQU  *
000174 00000830                           145+     DC     AL4(WP1)
□ Page     6
  Micro Focus Assembler version  V2.5.19  09/17/01 Listing as of 02/06/04 13.48
    LOC DATA               ADR1    ADR2   LINE LABEL     OP        OPERANDS
000178 41000001              000001       146+     LA     0,1
00017C 18FF                               147+     LR     15,R15
00017E 05EF                               148+     BALR   14,15
                                          149 *    USER MACRO LINKAGE
000180                                    150      USRMAC1  FMT107
000180                                    152+     DS     0H
```

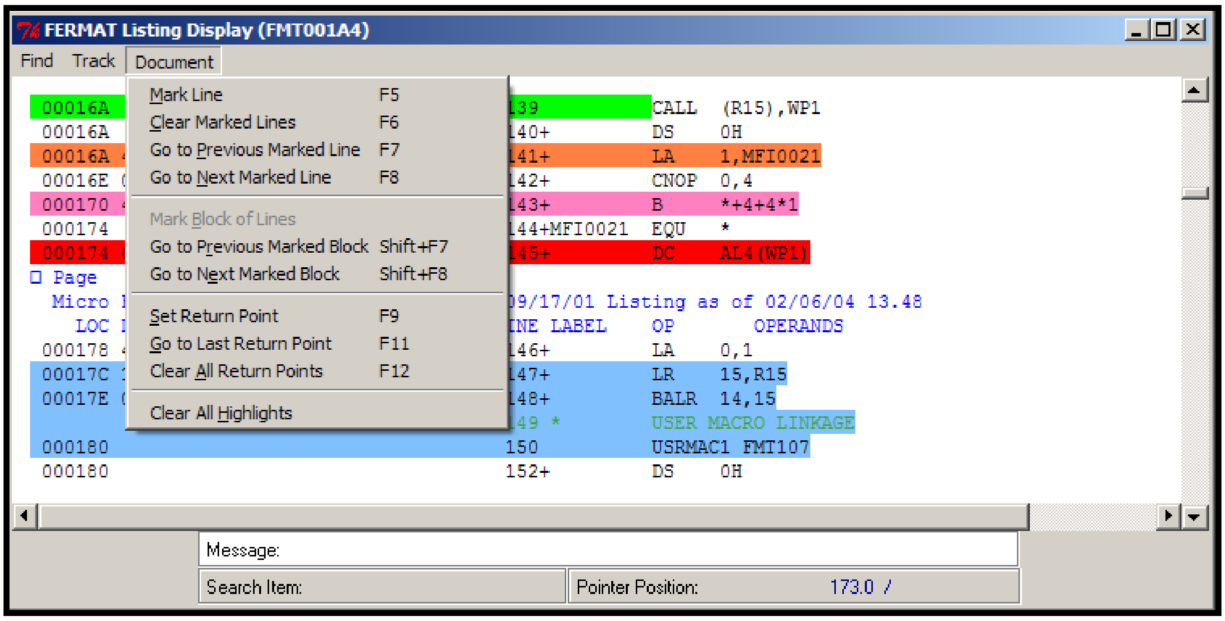
Message:
Search Item: Pointer Position: 188.27 /

```
      CALL   FMT105,(WP1,WP2)      STANDARD                    158
      L      R15,=V(FMT106)        V LITERAL                   172
      CALL   (R15),WP1                                         173
*     USER MACRO LINKAGE                                       186
      USRMAC1  FMT107                                          187
```

Message:
Selected Data Item: Pointer Position: 38.39 / STANDARD

Text Editor - FMT001A4
File Edit Track Options Mark Line CodeSlice Help
FERMAT Listing Display (FMT001A4)
Find Track Document

The Document option on the menu bar allows setting and navigation of marked lines and blocks.

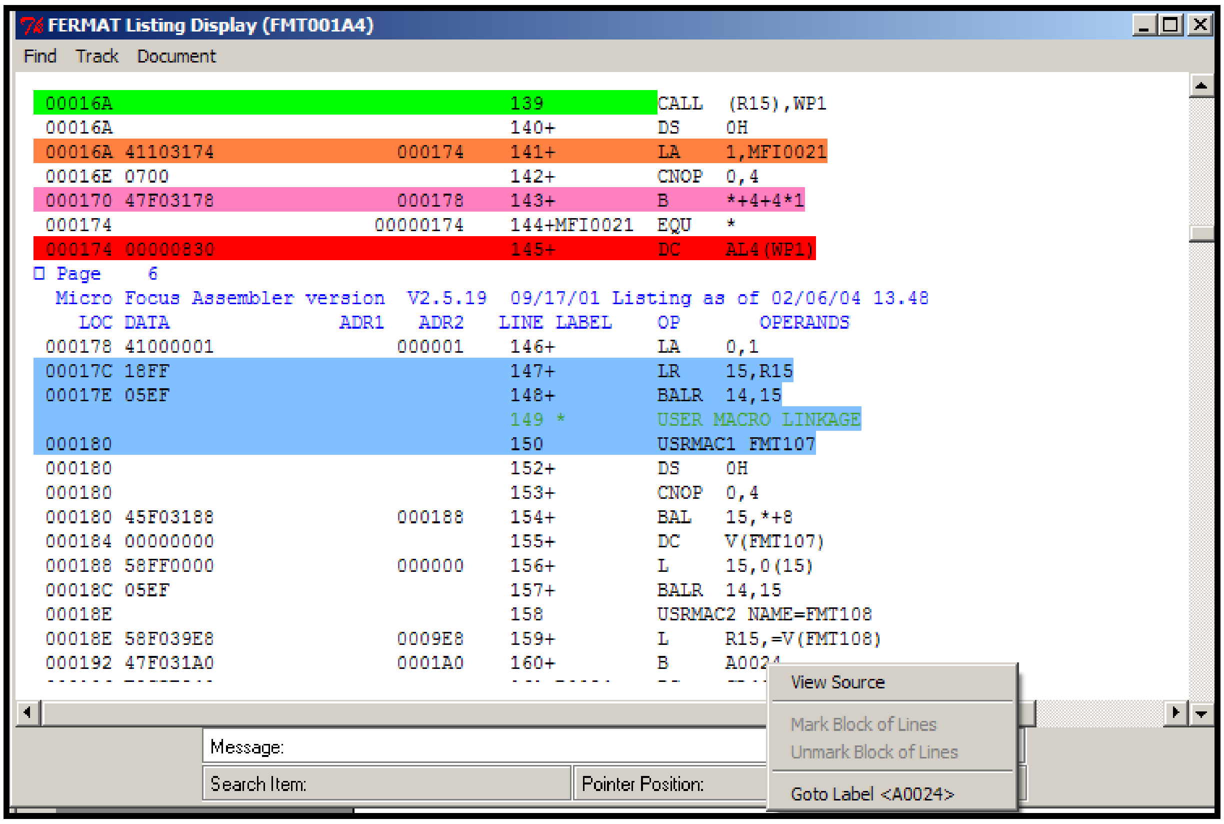


FERMAT Listing Display (FMT001A4)
Find Track Document
Mark Line F5
Clear Marked Lines F6
Go to Previous Marked Line F7
Go to Next Marked Line F8
Mark Block of Lines
Go to Previous Marked Block Shift+F7
Go to Next Marked Block Shift+F8
Set Return Point F9
Go to Last Return Point F11
Clear All Return Points F12
Clear All Highlights

Message:
Search Item: Pointer Position: 173.0 /

The Find and Track enable searching for stings and finding and marking data variables as in the Text Editor.

The right click function when used in the View Listing will display the following options.



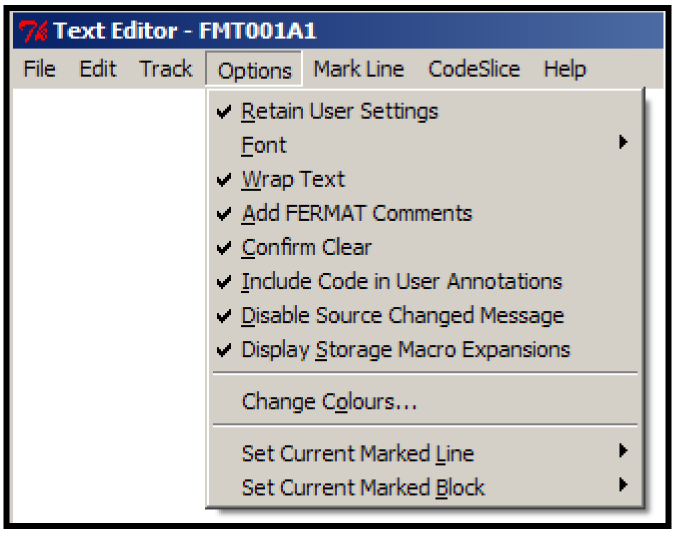
```
00016A                              139        CALL   (R15),WP1
00016A                              140+       DS     0H
00016A 41103174          000174     141+       LA     1,MFI0021
00016E 0700                         142+       CNOP   0,4
000170 47F03178          000178     143+       B      *+4+4*1
000174                 00000174     144+MFI0021 EQU   *
000174 00000830                     145+       DC     AL4(WP1)
□ Page     6
  Micro Focus Assembler version  V2.5.19  09/17/01 Listing as of 02/06/04 13.48
    LOC DATA             ADR1   ADR2   LINE LABEL     OP      OPERANDS
000178 41000001          000001     146+       LA     0,1
00017C 18FF                         147+       LR     15,R15
00017E 05EF                         148+       BALR   14,15
                                    149 *      USER MACRO LINKAGE
000180                              150        USRMAC1 FMT107
000180                              152+       DS     0H
000180                              153+       CNOP   0,4
000180 45F03188          000188     154+       BAL    15,*+8
000184 00000000                     155+       DC     V(FMT107)
000188 58FF0000          000000     156+       L      15,0(15)
00018C 05EF                         157+       BALR   14,15
00018E                              158        USRMAC2 NAME=FMT108
00018E 58F039E8          0009E8     159+       L      R15,=V(FMT108)
000192 47F031A0          0001A0     160+       B      A002
```

Selecting View Source will switch the user back to the Text Editor on the corresponding source line. If you are pointing at a label in the listing which is the destination of a branch statement then Goto Label <xxxxx> will navigate through the listing and stop at the label in the listing.

Please note that unlike the Text Editor you cannot save the Listing with the lines marked up, the facility for making lines in the Listing is purely to assist in analysis of the listing. Closing the View Listing will lose any marked up lines.

14.7 Customising the Text Editor

The text editor is fully customisable in terms of the colours and the labels used for marking, when analysing code. Selecting **Options** from the tool bar will produce the following function list.

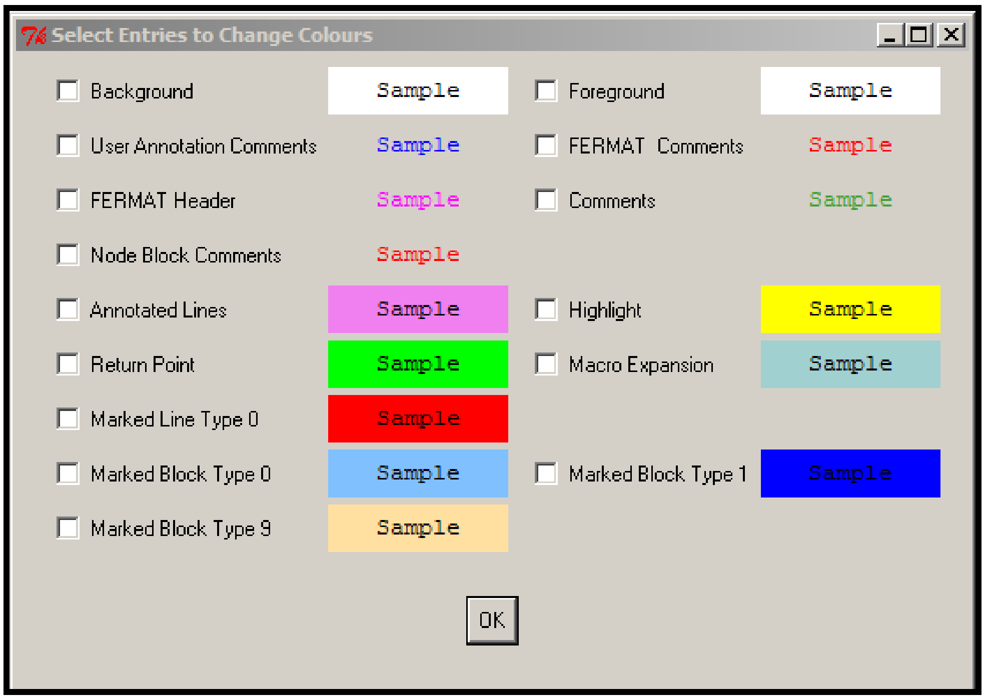


Retain User Settings will keep all user setting for colours and labels for future use; this is particularly useful for establishing documentation standards.

Add FERMAT Comments. When checked this will add special Fermat comments before the highlighted line/s, lines can be marked and unmarked using F5. Fermat comments aid the understanding of the code by allowing relevant comments to be placed describing why a particular line was highlighted. For further details see Set Current Marked Line.

Include Code in User Annotations. When checked this will include the original source code in the annotations file together with user entered comments (Annotations are detailed later in this guide).

Change Colours, selecting this option will produce the following dialogue box.

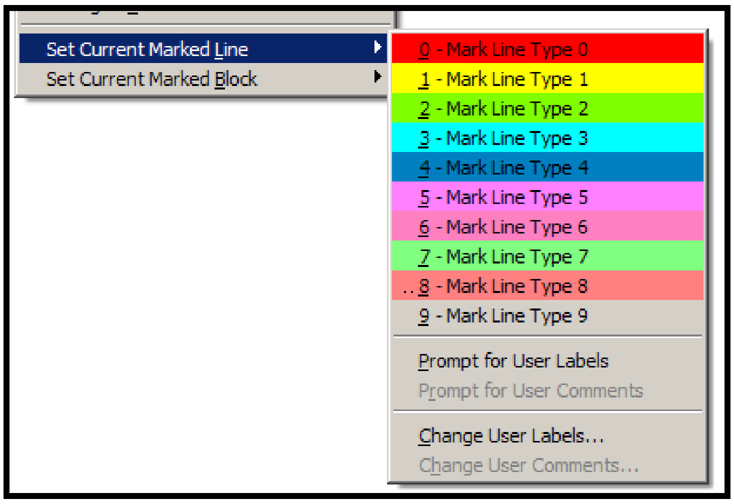


Select Entries to Change Colours

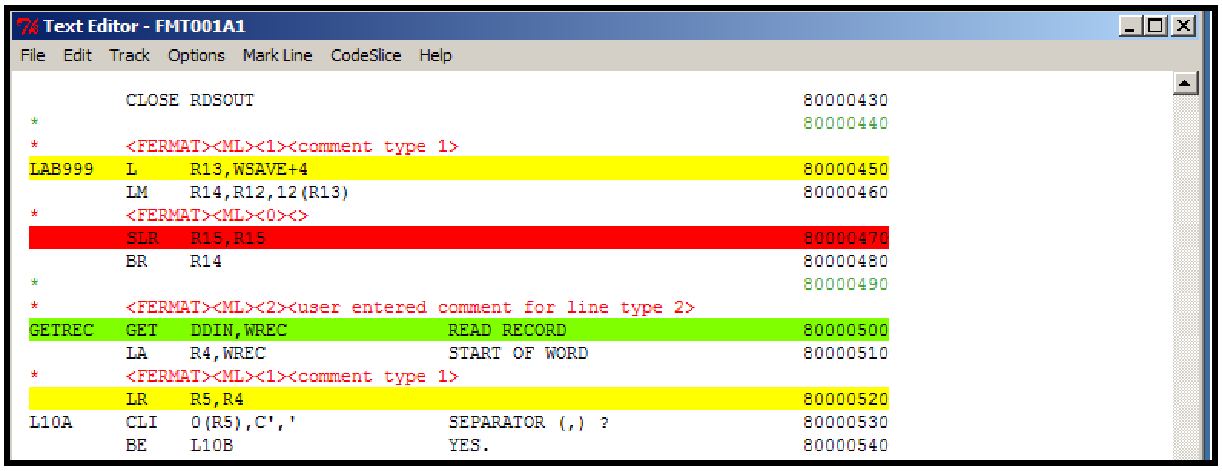
☐ Background	Sample	☐ Foreground	Sample
☐ User Annotation Comments	Sample	☐ FERMAT Comments	Sample
☐ FERMAT Header	Sample	☐ Comments	Sample
☐ Node Block Comments	Sample		
☐ Annotated Lines	Sample	☐ Highlight	Sample
☐ Return Point	Sample	☐ Macro Expansion	Sample
☐ Marked Line Type 0	Sample		
☐ Marked Block Type 0	Sample	☐ Marked Block Type 1	Sample
☐ Marked Block Type 9	Sample		

This enables customisation of the colours used within the text editor. By selecting one or more or features, then OK, the user is presented with the standard Windows colour options to select the desired colours for that function.

Set Current Marked Line will produce the following context menu.



Using this facility is it possible to select the current mark line parameter, change its colour and assign labels and user comments. When a line is highlighted in the text editor with the Display Fermat Comments selected the Fermat comment line will include the user defined comments.



```
         CLOSE RDSOUT                                             80000430
*                                                                 80000440
*        <FERMAT><ML><1><comment type 1>
LAB999   L     R13,WSAVE+4                                        80000450
         LM    R14,R12,12(R13)                                    80000460
*        <FERMAT><ML><0><>
         SLR   R15,R15                                            80000470
         BR    R14                                                80000480
*                                                                 80000490
*        <FERMAT><ML><2><user entered comment for line type 2>
GETREC   GET   DDIN,WREC                READ RECORD               80000500
         LA    R4,WREC                  START OF WORD             80000510
*        <FERMAT><ML><1><comment type 1>
         LR    R5,R4                                              80000520
L10A     CLI   0(R5),C','               SEPARATOR (,) ?           80000530
         BE    L10B                     YES.                      80000540
```

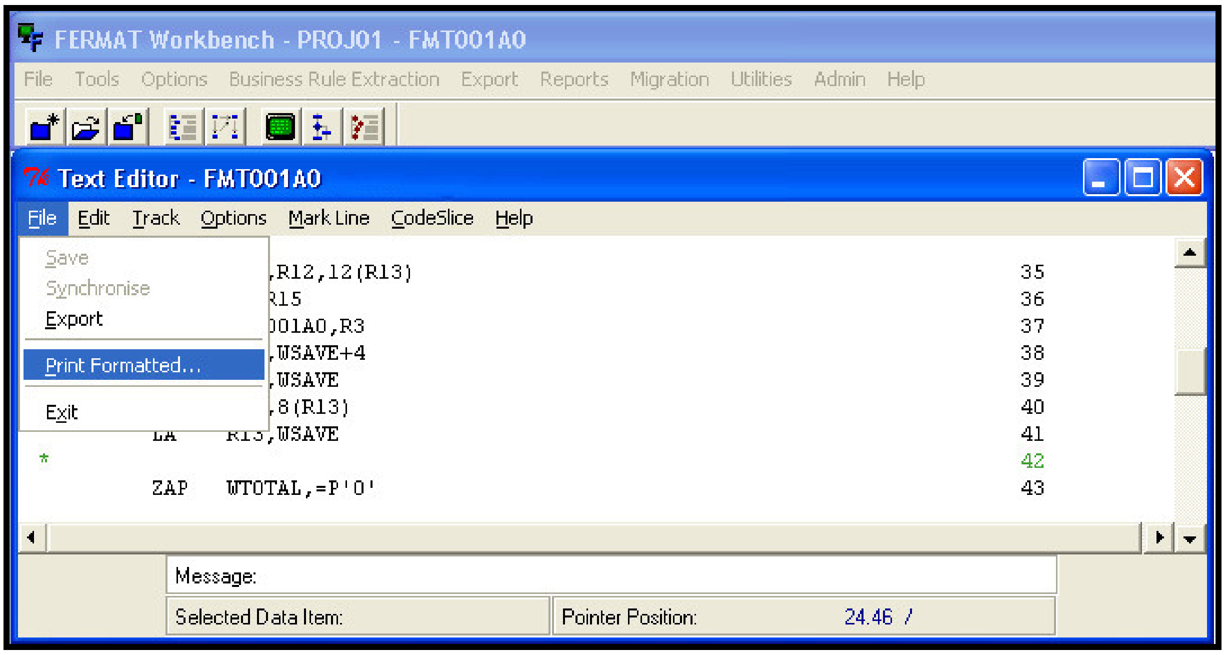
The same facilities are available for setting the colours and labels for marked blocks of code using the Set Current Marked Block option.

The Mark Line option on the tool bar provides a quick way to select the mark line type to be used for highlighting a line.

14.8 Print Function

The Workbench provides a print function that will output the entire module or selected text as it appears on the screen including all colour highlights of marked lines, blocks and annotated code.

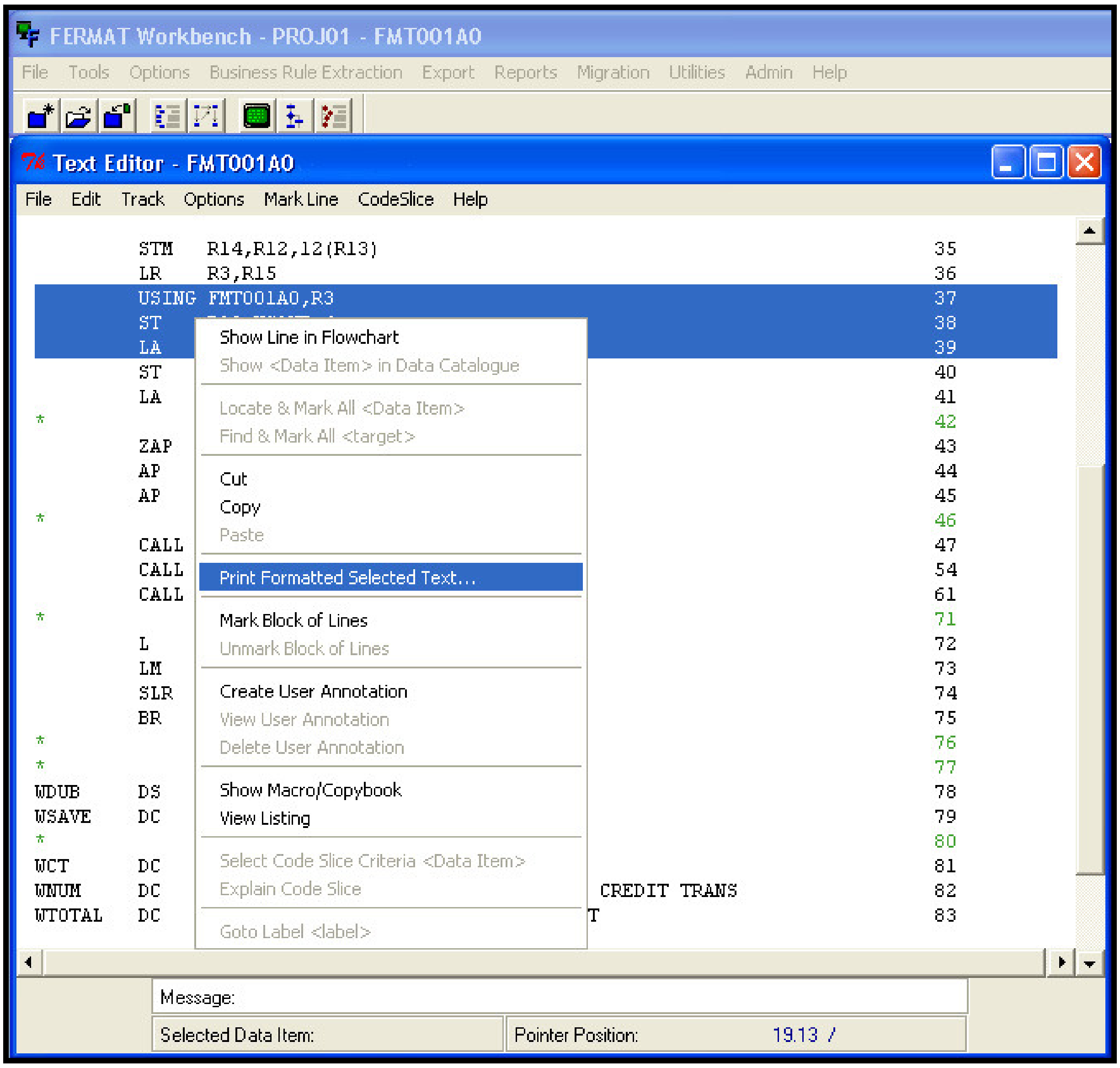
To print the entire module use the Print Formatted function from the drop down list on the File option on the Text Editor Menu Bar. As below:


FERMAT Workbench - PROJ01 - FMT001A0
File Tools Options Business Rule Extraction Export Reports Migration Utilities Admin Help
Text Editor - FMT001A0
File Edit Track Options Mark Line CodeSlice Help
Save
Synchronise
Export
Print Formatted...
Exit

```
                ,R12,12(R13)
                R15
                 001A0,R3
                ,WSAVE+4
                ,WSAVE
                ,8(R13)
         LA     R13,WSAVE
*
         ZAP    WTOTAL,=P'0'
```

Message:
Selected Data Item:
Pointer Position: 24.46 /

To print selected text, select the lines to print using the Text Edit left mouse key, then click the right mouse key, as below:

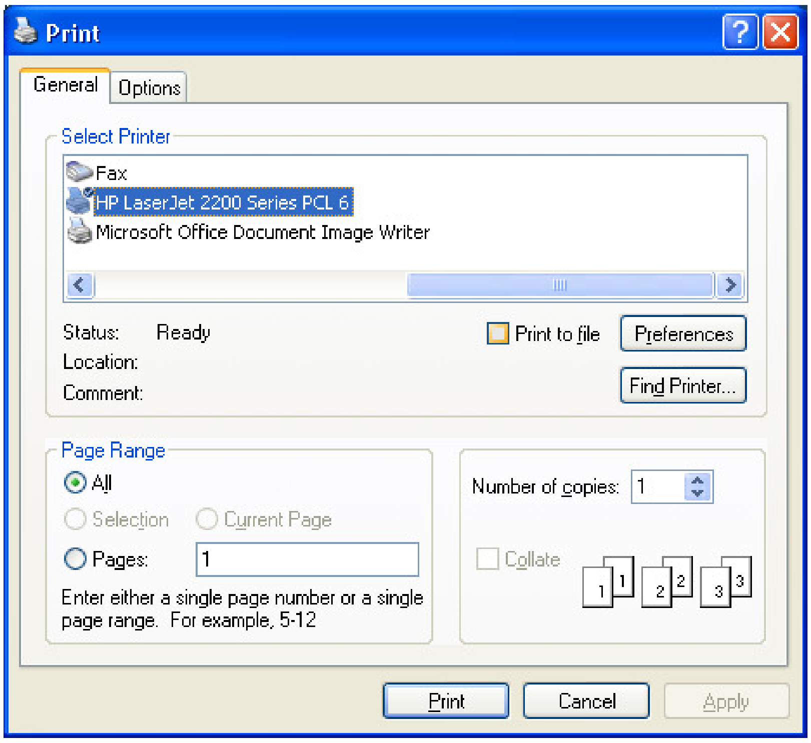

FERMAT Workbench - PROJ01 - FMT001A0
File Tools Options Business Rule Extraction Export Reports Migration Utilities Admin Help
Text Editor - FMT001A0
File Edit Track Options Mark Line CodeSlice Help
Show Line in Flowchart
Show <Data Item> in Data Catalogue
Locate & Mark All <Data Item>
Find & Mark All <target>
Cut
Copy
Paste
Print Formatted Selected Text...
Mark Block of Lines
Unmark Block of Lines
Create User Annotation
View User Annotation
Delete User Annotation
Show Macro/Copybook
View Listing
Select Code Slice Criteria <Data Item>
Explain Code Slice
Goto Label <label>
Message:
Selected Data Item:
Pointer Position: 19.13 /

```
         STM   R14,R12,12(R13)
         LR    R3,R15
         USING FMT001A0,R3
         ST
         LA
         ST
         LA
*
         ZAP
         AP
         AP
*
         CALL
         CALL
         CALL
*
         L
         LM
         SLR
         BR
*
*
WDUB     DS
WSAVE    DC
*
WCT      DC
WNUM     DC                            CREDIT TRANS
WTOTAL   DC                           T
```

This process will generate a HTML file in the project level1/*Username* directory called *module-name*.htm, which is then printed. This file can then be used to be printed or emailed using standard facilities.

Once you have clicked on Print Formatted… or Print Formatted Selected Text… a printer selection dialog box will appear:

Software Migrations Ltd



Select the printer that you want the print directed to, and click OK.

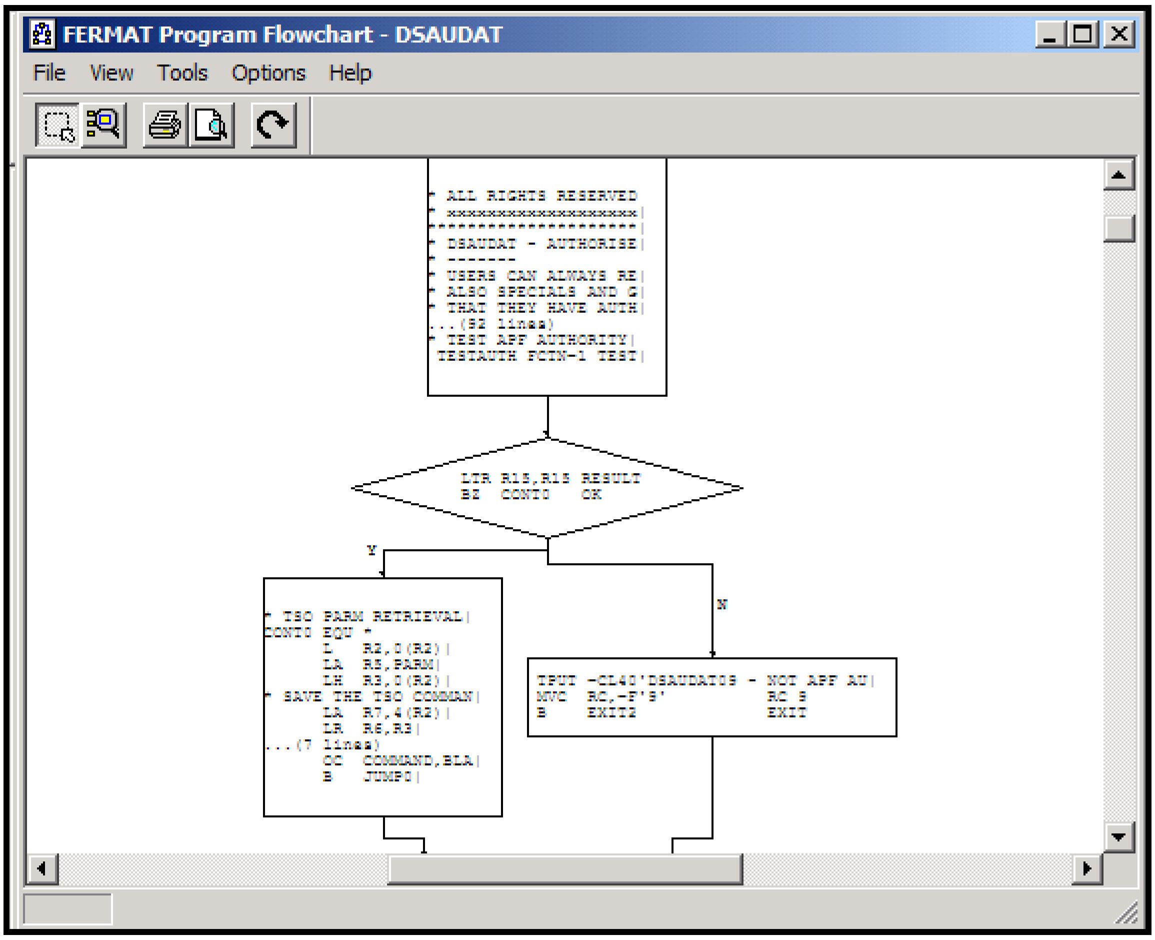
14.9 Export Function in the Text Editor

The Export function is available as an option on the File function of the Text Editor Menu bar, when selected the Workbench prompts the user to select the location where the exported file will be written. The Export process is used to merge the recorded annotations and FermaT comments in the assembler listing with the original source imported as part of the Full Life Cycle process. When using this function it is mandatory to save the module using the File | Save option or to close the module and save the annotations, any unsaved annotations will not be exported to the export file.

15. The Program Flowchart

The **Program Flowchart** depicts the control flow of a Module's logic.

Open the **Program Flowchart** from the Workbench Toolbar, either by selecting Tools | Program Flowchart from the menu or by pressing the Program Flowchart button.



```
* ALL RIGHTS RESERVED
* xxxxxxxxxxxxxxxxxxxx|
**********************|
* DSAUDAT - AUTHORISE|
* --------
* USERS CAN ALWAYS RE|
* ALSO SPECIALS AND G|
* THAT THEY HAVE AUTH|
...(92 lines)
* TEST APF AUTHORITY|
 TESTAUTH FCTN-1 TEST|
```

```
LTR R15,R15 RESULT
BZ  CONT0   OK
```

```
* TSO PARM RETRIEVAL|
CONT0 EQU *
      L   R2,0(R2)|
      LA  R5,PARM|
      LH  R3,0(R2)|
* SAVE THE TSO COMMAN|
      LA  R7,4(R2)|
      LR  R6,R3|
...(7 lines)
      OC  COMMAND,BLA|
      B   JUMP0|
```

```
TPUT  -CL40'DSAUDAT09 - NOT APF AU|
MVC   RC,=F'9'                RC 9
B     EXIT2                   EXIT
```

If you have any lines already marked in the Editor, they will be shown marked in the Flowchart.

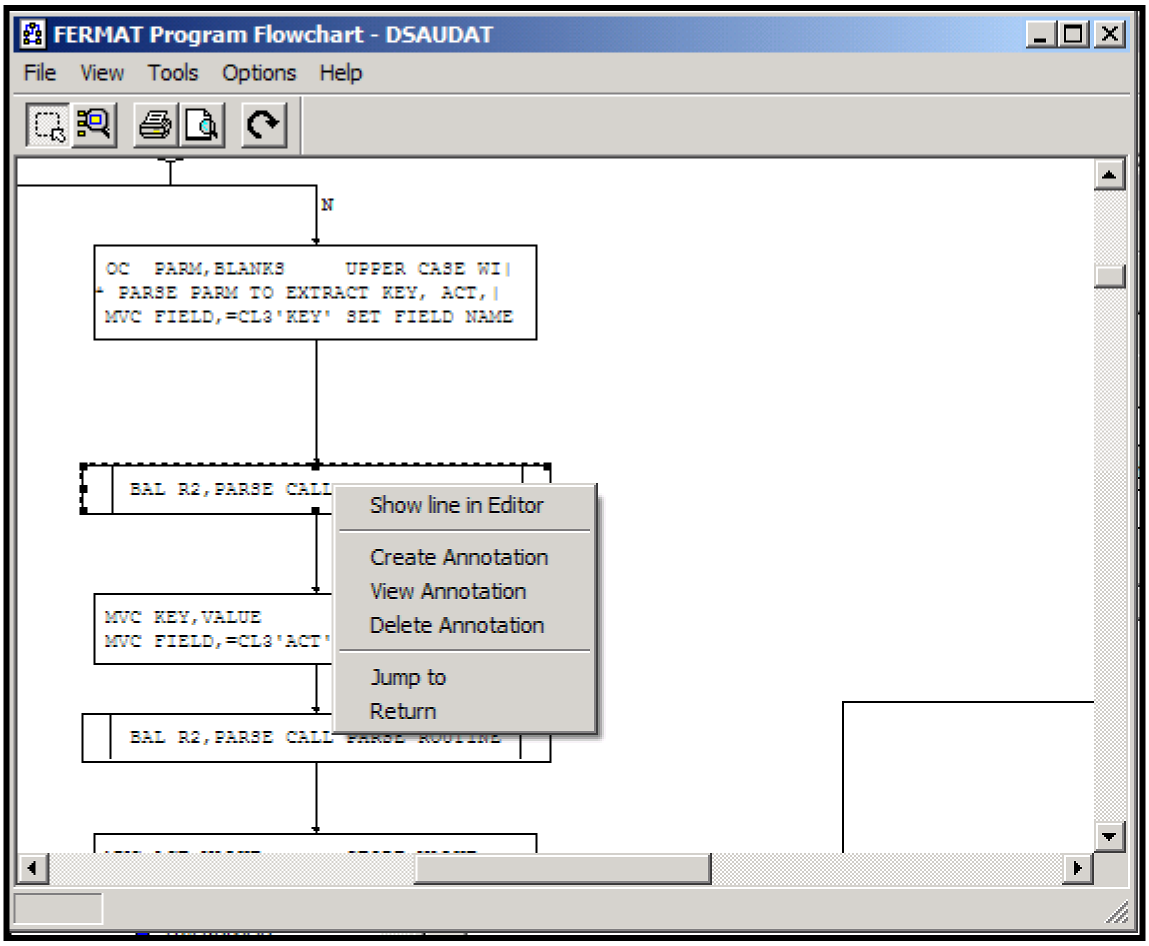
Navigation around the **Program Flowchart** is as for the **Function Call Graph** - i.e.:

- Scaling the graph using the available options in the View menu.
- Using the plus and minus keyboard keys for zooming in/out.

- Using the zoom cursor (second toolbar button) to rubber-band the area to be displayed in the window.
- Using the Overview Window (View | Overview Window from the menu or Ctrl+Shift+O) to select and size the area to be displayed in the main window.
- Using the scrollbars, cursor keys, and/or Page Up and Page Down keys to move around.

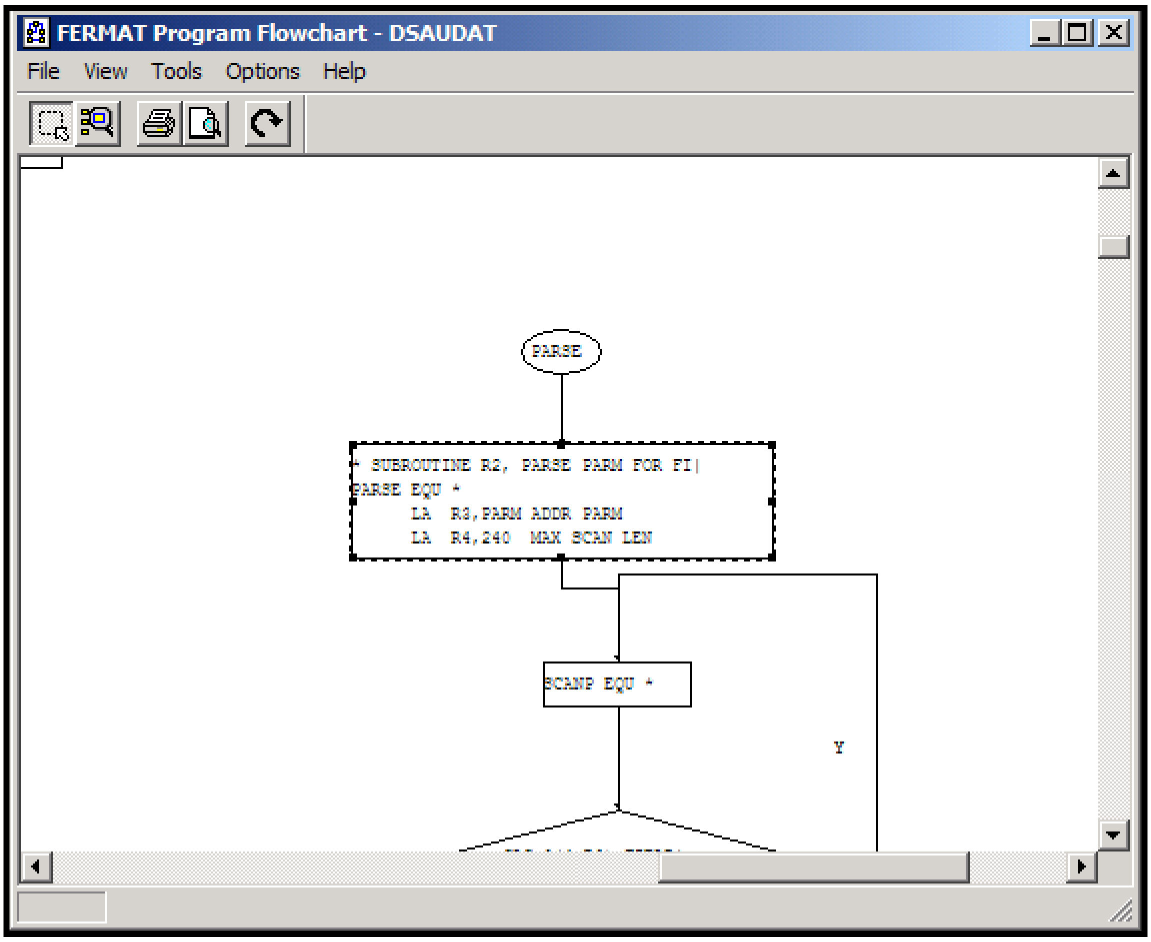
♦ Select View | Actual Size from the **Program Flowchart** menu to show the source code in the flowchart nodes more clearly.
♦ Select View | Fit In Window to show the entire module in the flowchart in one window.

In Module DSAUDAT in the DSA project Notice how the **Program Flowchart** has picked out the subroutine `PARSE` and it is represented as subroutine calls in the main flowchart, indicated as process boxes. The user can navigate within the flowchart by selecting a subroutine box and right clicking, this will then produce the dialogue box with the Jump to and return options in, as below.



```
OC   PARM,BLANKS      UPPER CASE WI|
* PARSE PARM TO EXTRACT KEY, ACT,|
MVC FIELD,=CL3'KEY' SET FIELD NAME

BAL R2,PARSE CALL

MVC KEY,VALUE
MVC FIELD,=CL3'ACT'

BAL R2,PARSE CALL PARSE ROUTINE
```

Selecting the Jump to option will cause the Program Flowchart to go to the appropriate routine, as below:


FERMAT Program Flowchart - DSAUDAT
File View Tools Options Help
PARSE
Y

```
* SUBROUTINE R2, PARSE PARM FOR FI|
PARSE EQU *
      LA   R3,PARM ADDR PARM
      LA   R4,240   MAX SCAN LEN
```

```
SCANP EQU *
```

If the user right clicks in the Parse subroutine and takes the return option the Program Flowchart will return to program where it was jumped from.

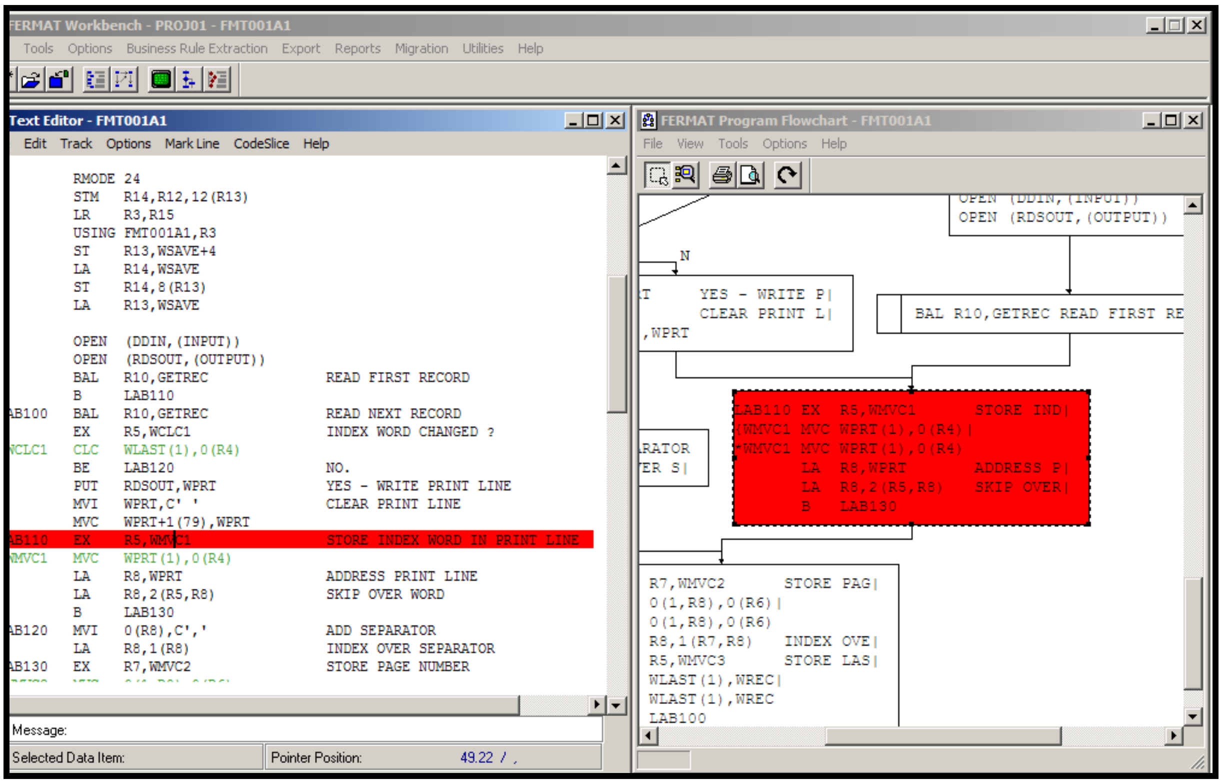
15.1 Synchronising the Program Flowchart and the Text Editor

Perhaps the most powerful combination of Tools for module analysis in the Workbench is using the **Program Flowchart** in conjunction with the **Text Editor**.

The **Program Flowchart** can be used interactively with the **Text Editor** to navigate through the logic of a module.

A selected line in the **Text Editor** can have its node highlighted in the **Program Flowchart**, and the code for a selected node in the **Program Flowchart** can be highlighted in the **Text Editor**.

Size the Program Flowchart and the Text Editor so that you can see both.
In the Editor, Clear All Marked Lines (F6) and Clear All Return Points (F12).


FERMAT Workbench - PROJ01 - FMT001A1
Tools Options Business Rule Extraction Export Reports Migration Utilities Help
Text Editor - FMT001A1
Edit Track Options Mark Line CodeSlice Help
FERMAT Program Flowchart - FMT001A1
File View Tools Options Help

```
         RMODE 24
         STM   R14,R12,12(R13)
         LR    R3,R15
         USING FMT001A1,R3
         ST    R13,WSAVE+4
         LA    R14,WSAVE
         ST    R14,8(R13)
         LA    R13,WSAVE

         OPEN  (DDIN,(INPUT))
         OPEN  (RDSOUT,(OUTPUT))
         BAL   R10,GETREC                 READ FIRST RECORD
         B     LAB110
AB100    BAL   R10,GETREC                 READ NEXT RECORD
         EX    R5,WCLC1                   INDEX WORD CHANGED ?
WCLC1    CLC   WLAST(1),0(R4)
         BE    LAB120                     NO.
         PUT   RDSOUT,WPRT                YES - WRITE PRINT LINE
         MVI   WPRT,C' '                  CLEAR PRINT LINE
         MVC   WPRT+1(79),WPRT
AB110    EX    R5,WMVC1                   STORE INDEX WORD IN PRINT LINE
WMVC1    MVC   WPRT(1),0(R4)
         LA    R8,WPRT                    ADDRESS PRINT LINE
         LA    R8,2(R5,R8)                SKIP OVER WORD
         B     LAB130
AB120    MVI   0(R8),C','                 ADD SEPARATOR
         LA    R8,1(R8)                   INDEX OVER SEPARATOR
AB130    EX    R7,WMVC2                   STORE PAGE NUMBER
```

Message:
Selected Data Item:
Pointer Position: 49.22 / ,

```
OPEN  (RDSOUT,(OUTPUT))

N
T       YES - WRITE P|
        CLEAR PRINT L|
,WPRT

BAL R10,GETREC READ FIRST RE

LAB110 EX  R5,WMVC1        STORE IND|
{WMVC1 MVC WPRT(1),0(R4)|
*WMVC1 MVC WPRT(1),0(R4)
       LA  R8,WPRT         ADDRESS P|
       LA  R8,2(R5,R8)     SKIP OVER|
       B   LAB130

RATOR
ER S|

R7,WMVC2       STORE PAG|
0(1,R8),0(R6)|
0(1,R8),0(R6)
R8,1(R7,R8)    INDEX OVE|
R5,WMVC3       STORE LAS|
WLAST(1),WREC|
WLAST(1),WREC
LAB100
```

From within the **Text Editor**, select a line, right-click and choose Show Line in Flowchart from the context menu. The **Program Flowchart** will centre on the node that contains the selected line.

From within the **Program Flowchart**, with the select cursor double-click a flowchart node. The **Text Editor** will display the first line of code in the selected node. Alternatively, right-click in a **Program Flowchart** node and choose Show Line in Editor.

15.2 Printing from the Program Flowchart & Function Call Graph

The diagrams contained in both the **Program Flowchart** and **Function Call Graph** can be printed from the File | Print... menu of the Tool.

By default the entire diagram will be printed on a single page, though by experimenting with

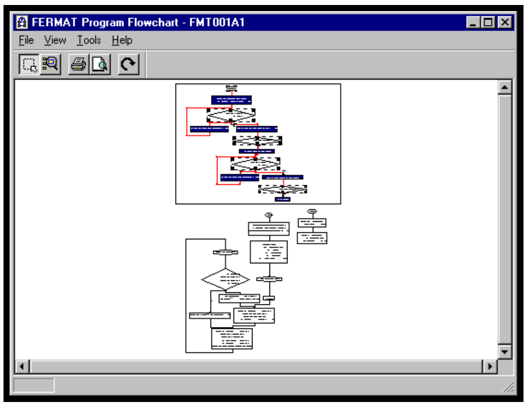
File | Print Setup... you can scale the diagram to your own requirements. From the Print Setup dialog choose the Printer... button to access the Windows Print Dialog, where you can select the printer, page orientation and number of copies, etc.

What may not be immediately obvious is how you can print just the section of the diagram you are interested in.

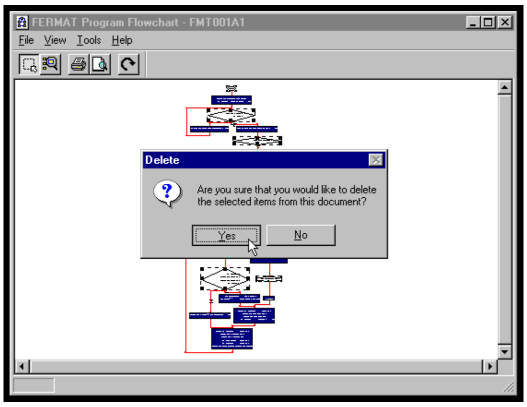
i. Select the part of the diagram you wish to print, with the Select cursor (i.e. 'rubber-band it').
ii. Press Ctrl+C to copy it to the Windows clipboard.
iii. Press Ctrl+A to select all of the diagram.
iv. Press the delete key and confirm deletion of all elements of the diagram.
v. Press Ctrl+V to activate the paste cursor, and click anywhere in the graph window to paste the selected part of the diagram back in the window.
vi. By default, the selected part of the diagram will be scaled to print on a single page.

Note: to restore the original flowchart/call graph, click on the Refresh button.

i. & ii.

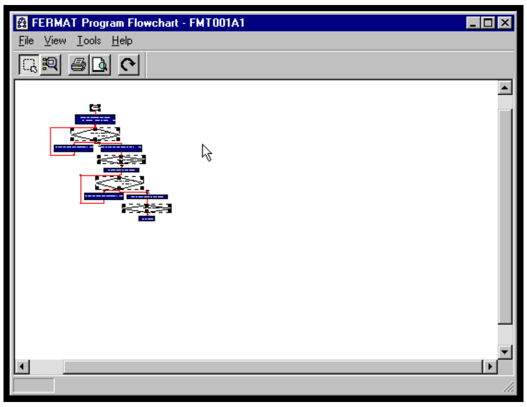


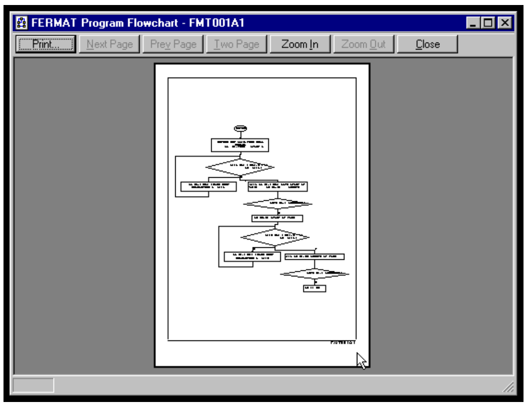
iii. & iv.



Are you sure that you would like to delete the selected items from this document?

Yes No





16. User Annotations

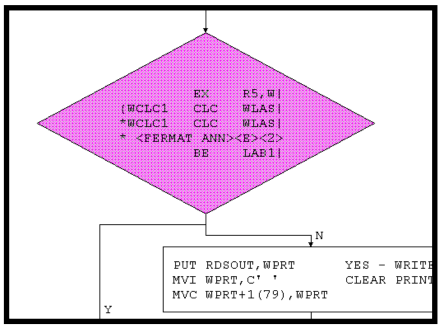
The **Text Editor** and **Program Flowchart** tools share a common Code Annotation Facility. This facility allows the user to annotate a block of code from either the **Program Flowchart** or **Text Editor**.

A user may wish to annotate a block of code for a number of reasons, such as: a reminder to do something (like an electronic 'post it' note), a warning to a fellow programmer about a difficult piece of code, or a way of documenting a piece of code without cluttering the source with comments.

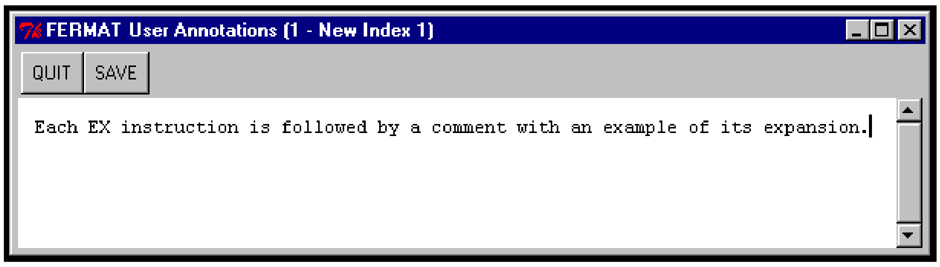
The Code Annotation Facility works by inserting special FERMAT comments in the source code to mark the start and end points of the block being annotated, with a reference to the associated annotation that is held separately in the Project repository (see example below).

```
* <FERMAT ANN><S><1>

         EX     R5,WCLC1                INDEX WORD CHANGED ?               80000210

*WCLC1   CLC    WLAST(1),0(R4)                                             80000220

 <FERMAT ANN><E><1>
```

Wherever a block of code contains a special Annotation comment, the **Text Editor** and the **Program Flowchart** will indicate this by marking the lines/node in a predefined colour (the default is purple).



An Annotation is created by selecting a number of lines in the **Text Editor,** or a node in the **Program Flowchart**, and invoking the Code Annotation Facility from the context menu. Subsequently, the Annotation is accessed by either selecting an annotated source line or an annotated flowchart node and selecting View/Create Annotation from the context menu.



Each EX instruction is followed by a comment with an example of its expansion.

Note: Annotations may be used in combination with other marked lines (see Section 14.1, Data Tracking in the Text Editor). For example, one or more source lines may be marked in, bright red to indicate a problem area. The user may then choose to annotate these same lines describing the issue.

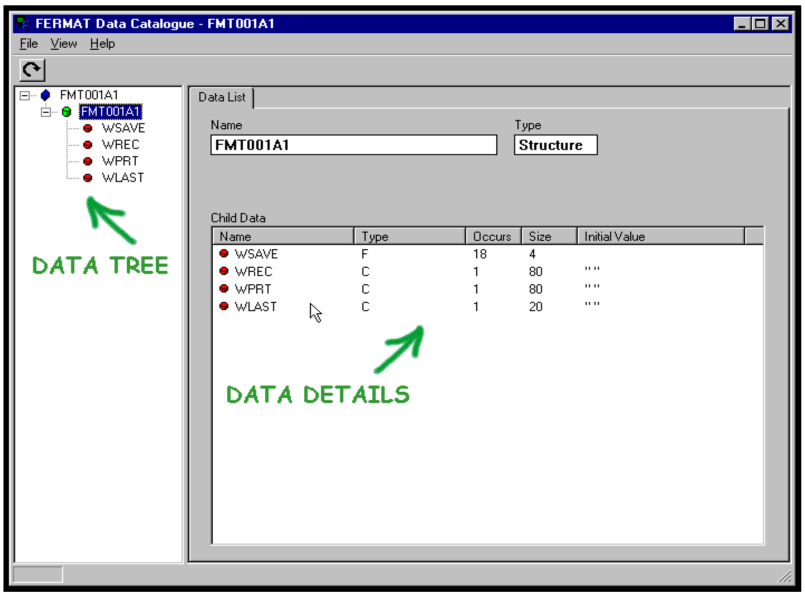
- Select one or more code lines in the **Text Editor** that you wish to annotate.
- Right-click in the selection and choose Create User Annotation.
- Type a suitable message in the dialog box.
- Click on SAVE.
- Right-click one of the annotated lines (shown highlighted as purple by default) and select Show Line in Flowchart to highlight the corresponding node in the **Program Flowchart**.
- Right-click the current node in the **Program Flowchart** and select View Annotation. The text you typed in will be displayed from the **Program Flowchart**. Double-click the node to show the related code in the **Text Editor**.

A **User Annotation Report** file can be produced. This will contain details of all the User annotations contained in a module. The report is selected from the Reports | Annotation Report menu. The report file is called Annotation_*modulename*.txt and is output to the project\level1\inventory directory.

17. The Data Catalogue

Open the **Data Catalogue** from the Workbench Toolbar, either by selecting Tools | Data Catalogue from the menu or by pressing the Data Catalogue button.

Like the **Function Catalogue**, the **Data Catalogue** is split into two panes, the Data Tree and the Data Details (see screenshot below).



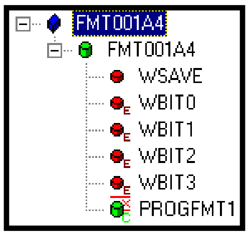
Name: FMT001A1
Type: Structure

Child Data

Name	Type	Occurs	Size	Initial Value
● WSAVE	F	18	4	
● WREC	C	1	80	" "
● WPRT	C	1	80	" "
● WLAST	C	1	20	" "

17.1 The Data Tree

The Data Tree pane shows the hierarchy of Data Items contained in the current module as they were declared in the source.



Each Data Item in the tree is allocated an icon according to its type. The following data types are supported in this release:

Software Migrations Ltd

- Data Structure (green cylinder icon) - Data Structures are non-atomic Data Items which contain others and may be nested.

- Data Element (red cylinder icon) - Data Elements are atomic Data Items and thus may not be nested. Note that an Assembler Equate is shown as a Data Element with a trailing ‘E’.

17.2 Data Defined in Copybooks and Macros

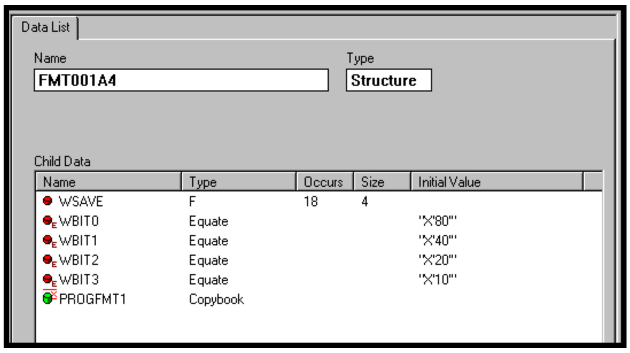
The **Data Catalogue** may contain external Data Structures[4], where the data in a module is defined in one or more Copybooks and/or User Macros. Such an ‘external’ Data Structure is indicated by a red ‘X’ in the top right corner of the Data Structure icon with a trailing ‘C’ or ‘U’ to show whether it was defined in a Copybook or a User Macro respectively.

Note: To enable the **Data Catalogue** to display details of externally declared data without having to import the relevant copybooks/macros into the FERMAT Project, the Workbench Toolbar Options | Project Settings... menu allows you to specify the path to your copybook and macro libraries.

17.3 Data Details

The Data Details pane currently includes one page - the Data List. This corresponds directly to the **Function Catalogue**’s Function List page.

The Data List displays the selected Data Item’s immediate children.



Name: FMT001A4
Type: Structure

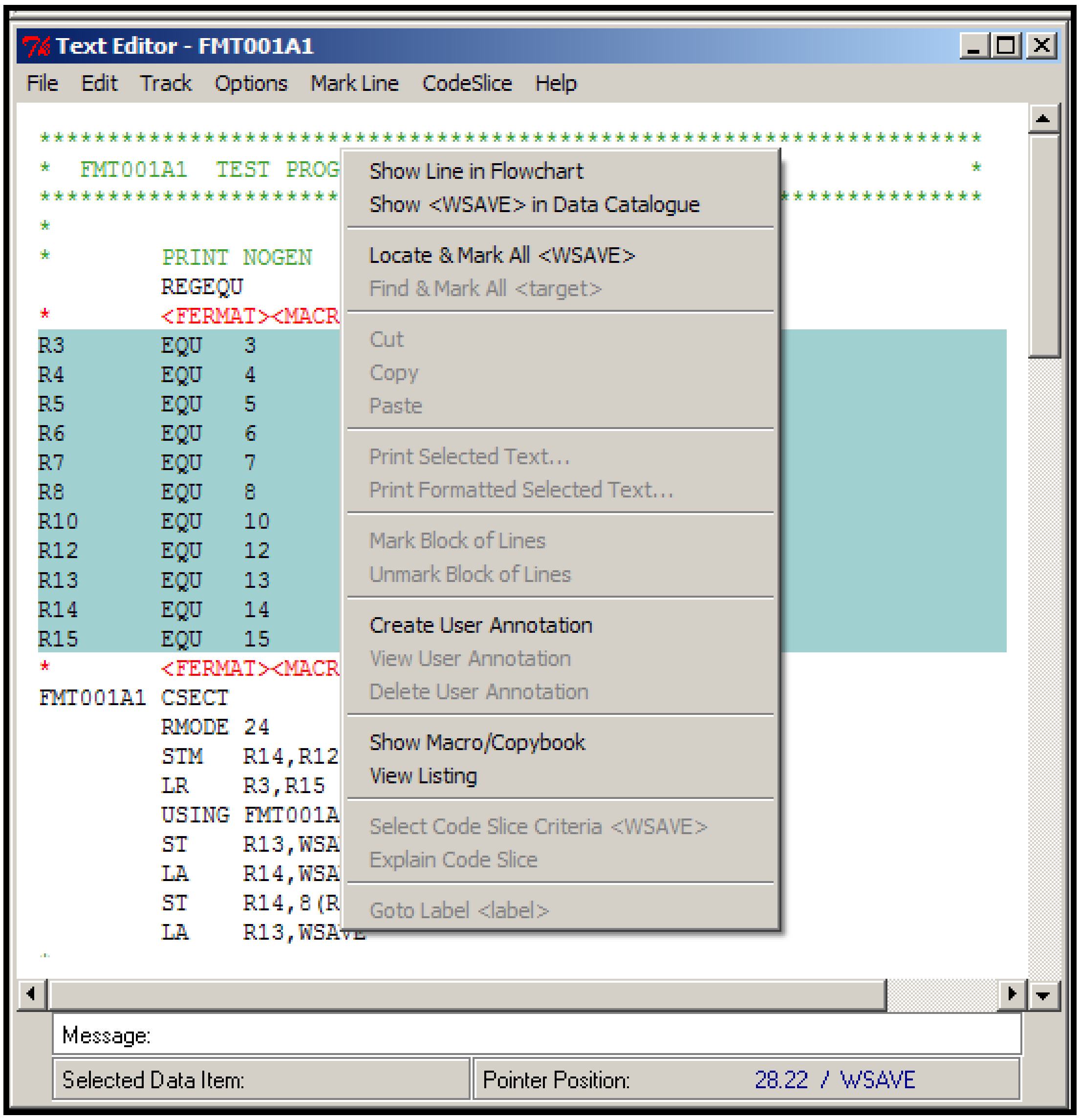
Child Data

Name	Type	Occurs	Size	Initial Value
WSAVE	F	18	4	
WBIT0	Equate			"X'80'"
WBIT1	Equate			"X'40'"
WBIT2	Equate			"X'20'"
WBIT3	Equate			"X'10'"
PROGFMT1	Copybook			

The sort order for each column toggles between ascending and descending, apart from the Name column which toggles between the order in which the Data Items were declared and ascending order.

[4] Within the Data Catalogue this means external to the Module, not external to the Project as with the Function Catalogue.

17.4 Synchronising the Data Catalogue and the Text Editor



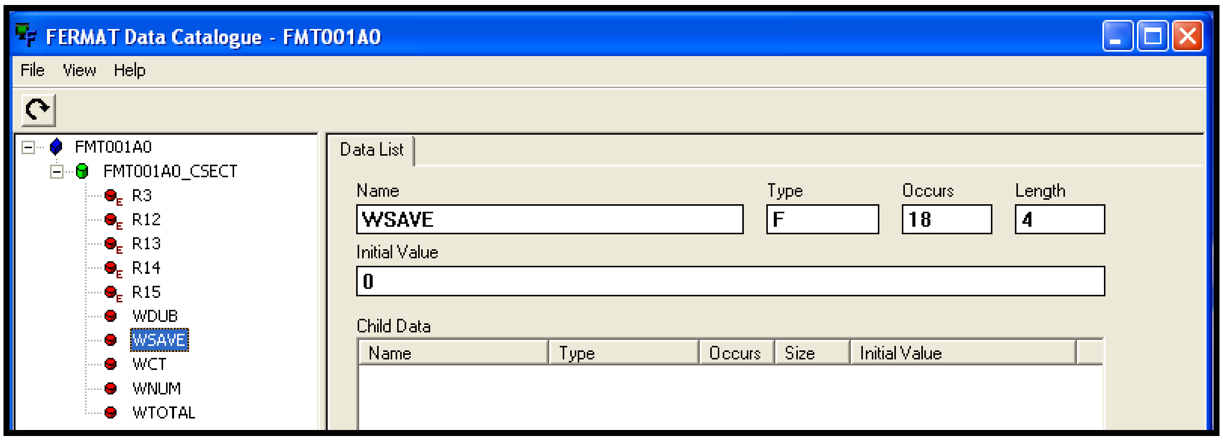
```
**********************************************************************
*   FMT001A1   TEST PROG                                             *
*********************                               ****************
*
*          PRINT NOGEN
           REGEQU
*          <FERMAT><MACR
R3         EQU    3
R4         EQU    4
R5         EQU    5
R6         EQU    6
R7         EQU    7
R8         EQU    8
R10        EQU    10
R12        EQU    12
R13        EQU    13
R14        EQU    14
R15        EQU    15
*          <FERMAT><MACR
FMT001A1   CSECT
           RMODE  24
           STM    R14,R12
           LR     R3,R15
           USING  FMT001A
           ST     R13,WSA
           LA     R14,WSA
           ST     R14,8(R
           LA     R13,WSAVE
```

Message:

Selected Data Item:

Pointer Position: 28.22 / WSAVE

Right clicking on the data item WSAVE will display the dialogue box, selecting the option Show <WSAVE> in data catalogue as above will result in Data Catalogue being displayed focused on the WSAVE Data element as below.



Name: WSAVE
Type: F
Occurs: 18
Length: 4
Initial Value: 0

18. The Batch Data Tracker

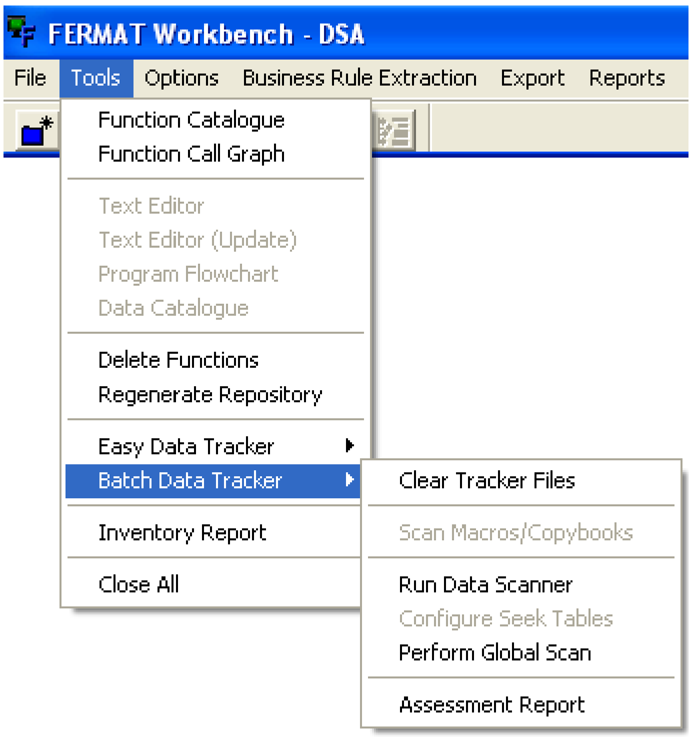
Ensure that the current project selected is DSA for this section of this guide.

The Batch Data Tracker is a specialist tool within the Workbench and is used to locate and identify data items in a project to assist in estimating the impact of change on a system or for creating a sub-project.

The Batch Data Tracker is a Microsoft Access application in which the assembler listings or source are parsed to generate Access tables containing all defined data items with their associated comments.

The programmer or analyst searches these tables using the Batch Data Tracker for data items that match the Boolean search criteria. Once all required data items have been selected, a Global Scan can then be performed against the Assembler source listings, all data items found are highlighted for use in the text editor and flow charter, the impact of a change can then be easily identified and quantified.

The Batch Data Tracker is invoked from Tools option of the main menu, select Batch Data Tracker as below.



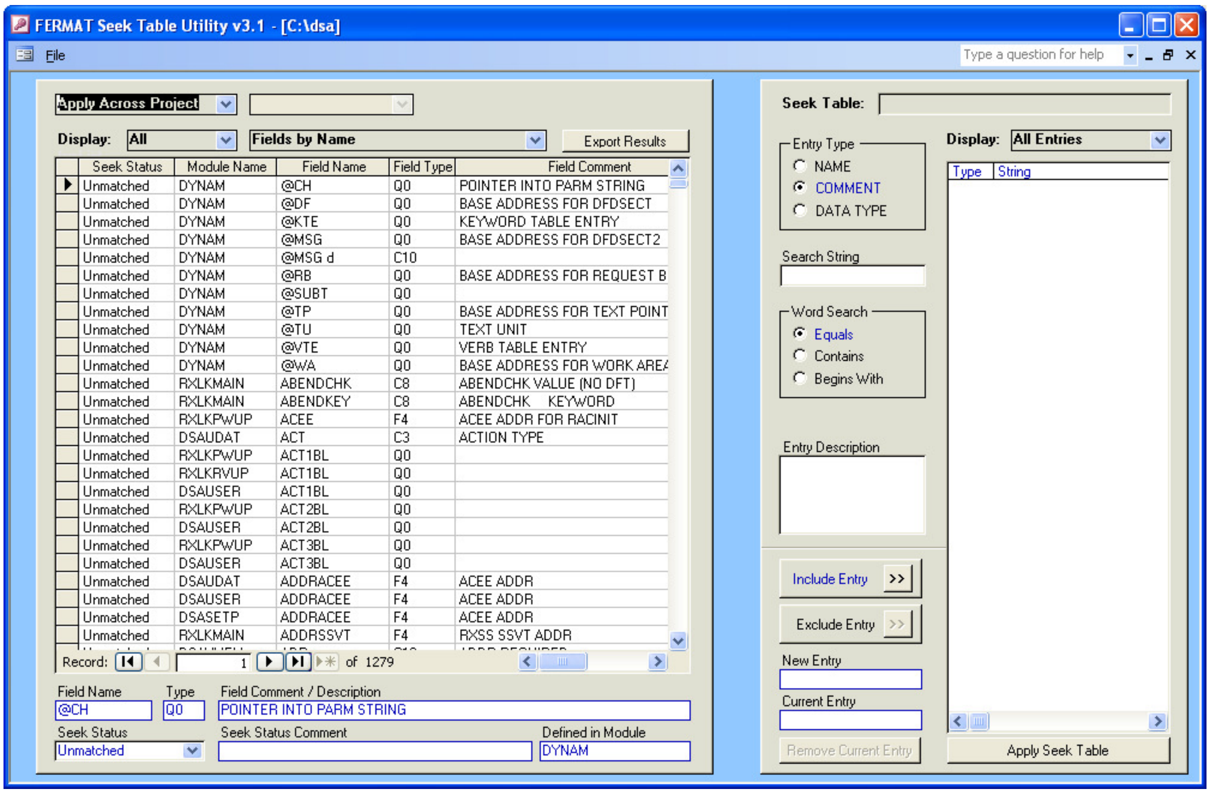
The first requirement is to Run Data Scanner.

Select Run Data Scanner

This starts a process that parses all the assembler modules in the project and populates the Access tables with the data items and any associated comments. This only needs to be rerun when new or updated modules are imported, so the option will be disabled until a run is required.

Once the Data Scanner has completed **select Tools Batch Data Tracker|Configure Seek Tables** from the drop down list, as before.

You will now be presented with the Configure Seek Tables Access Screen as below.



Apply Across Project

Display: All — Fields by Name

Seek Status	Module Name	Field Name	Field Type	Field Comment
Unmatched	DYNAM	@CH	Q0	POINTER INTO PARM STRING
Unmatched	DYNAM	@DF	Q0	BASE ADDRESS FOR DFDSECT
Unmatched	DYNAM	@KTE	Q0	KEYWORD TABLE ENTRY
Unmatched	DYNAM	@MSG	Q0	BASE ADDRESS FOR DFDSECT2
Unmatched	DYNAM	@MSG d	C10	
Unmatched	DYNAM	@RB	Q0	BASE ADDRESS FOR REQUEST B
Unmatched	DYNAM	@SUBT	Q0	
Unmatched	DYNAM	@TP	Q0	BASE ADDRESS FOR TEXT POINT
Unmatched	DYNAM	@TU	Q0	TEXT UNIT
Unmatched	DYNAM	@VTE	Q0	VERB TABLE ENTRY
Unmatched	DYNAM	@WA	Q0	BASE ADDRESS FOR WORK ARE
Unmatched	RXLKMAIN	ABENDCHK	C8	ABENDCHK VALUE (NO DFT)
Unmatched	RXLKMAIN	ABENDKEY	C8	ABENDCHK KEYWORD
Unmatched	RXLKPWUP	ACEE	F4	ACEE ADDR FOR RACINIT
Unmatched	DSAUDAT	ACT	C3	ACTION TYPE
Unmatched	RXLKPWUP	ACT1BL	Q0	
Unmatched	RXLKRVUP	ACT1BL	Q0	
Unmatched	DSAUSER	ACT1BL	Q0	
Unmatched	RXLKPWUP	ACT2BL	Q0	
Unmatched	DSAUSER	ACT2BL	Q0	
Unmatched	RXLKPWUP	ACT3BL	Q0	
Unmatched	DSAUSER	ACT3BL	Q0	
Unmatched	DSAUDAT	ADDRACEE	F4	ACEE ADDR
Unmatched	DSAUSER	ADDRACEE	F4	ACEE ADDR
Unmatched	DSASETP	ADDRACEE	F4	ACEE ADDR
Unmatched	RXLKMAIN	ADDRSSVT	F4	RXSS SSVT ADDR

Record: 1 of 1279

Field Name: @CH Type: Q0 Field Comment / Description: POINTER INTO PARM STRING

Seek Status: Unmatched Seek Status Comment: Defined in Module: DYNAM

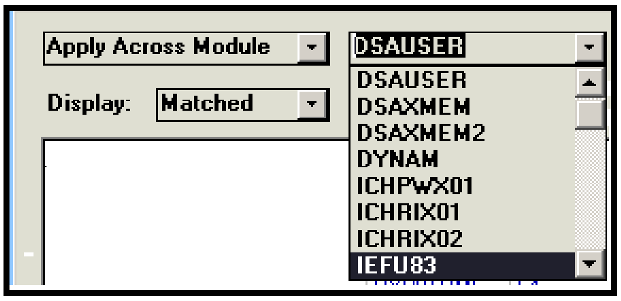
Entry Type: NAME [✓ COMMENT] DATA TYPE

Word Search: [✓ Equals] Contains Begins With

Display: All Entries

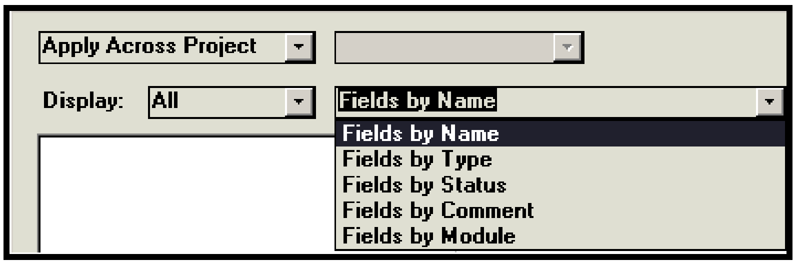
The screen is divided into 2 sections the left-hand pane gives details of the contents of the Access tables. The main panel of which displays the detailed information of the items in the tables which are to be searched.

Searches can be carried out against the whole project or selected individual modules, within a project, by choosing the appropriate settings in the top 2 selection boxes. As below:-

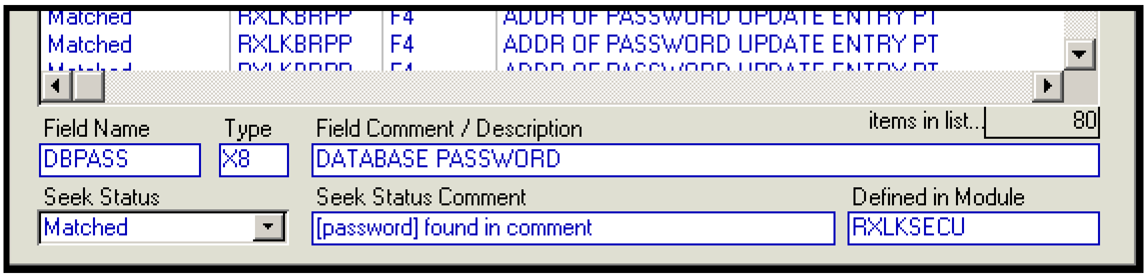


Using the display selection list, the user can either list ALL the data items or only those that MATCH the selection criteria, those UNMATCHED or those items EXCLUDED.

The listed items can be sorted by using the next selection list, see below:-



The lower section of the left panel details information on any selected data item in the list providing details of the field name, field type, details of any associated comment, the seek status following a search with match, unmatched or excluded, details of why matched and in which module the match was found. See below for example of a matched data item:-



Matched	RXLKBRPP	F4	ADDR OF PASSWORD UPDATE ENTRY PT

items in list.. 80

Field Name: DBPASS
Type: X8
Field Comment / Description: DATABASE PASSWORD
Seek Status: Matched
Seek Status Comment: [password] found in comment
Defined in Module: RXLKSECU

Software Migrations Ltd

The following table describes the Data types identified by the Batch Data Tracker.

Code	Constant Type	Machine Format
C	Character	8-bit code for each character
G	Graphic	16-bit code for each character
X	Hexadecimal	4-bit code for each hexadecimal digit
B	Binary	Binary format
F	Fixed-point	Signed, fixed-point binary format; normally a fullword
H	Fixed-point	Signed, fixed-point binary format; normally a halfword
E	Floating-point	Short floating-point format; normally a fullword
D	Floating-point	Long floating-point format; normally a doubleword
L	Floating-point	Extended floating-point format; normally two doublewords
P	Decimal	Packed decimal format
Z	Decimal	Zoned decimal format
A	Address	Value of address; normally a fullword
Y	Address	Value of address; normally a halfword
S	Address	Base register and displacement value; a halfword
V	Address	Space reserved for external symbol addresses; each address normally a fullword

The right hand panel of the Configure Seek Tables screen is used to select or create the seek table, and generate the selection criteria. Selections can be made on data names, comments and data type. The search can be on **equals** (exact match), **Contains** same as using wild cards before and after or **Begins With** same as wild card after the string.

An entry description is optional which acts as a prompt as to why this element was searched by the programmer or analyst.

The option is then available to Include or Exclude the entry from the selection process. Finally there is the option to remove the current entry from the selection criteria.

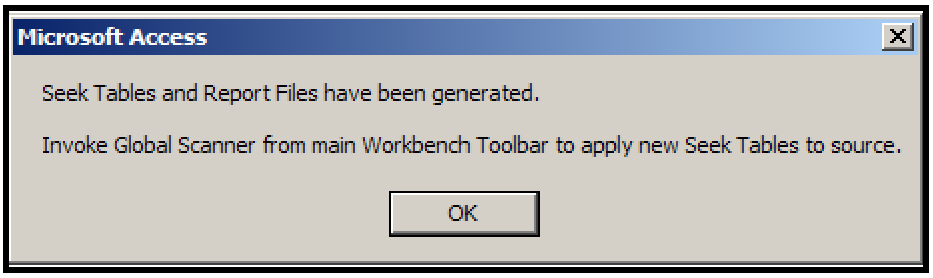
The search criteria are stored in the tracker folder of the project. It is possible to build as many different search criteria as required. The name and path of the seek table can be entered at the end of the directory displayed in the Seek Table box (top right), **the name of the Seek table can be any valid file name but must have the file extension .txt.** If no seek table is specified the batch data tracker will use a default (*Project*\tracker\UserName \TrackerDefault.txt), if no path is specified it will be saved in My Documents and will have .txt added to it if not entered.

The Apply Seek Table button is used to execute the search; the search is carried out against the access tables not the actual assembler source and is therefore far more efficient.

Generate an enquiry that will search the entire project for any data associated with the processing for new passwords.

With the Configure Seek Table screen open (if not Select Tools|Batch Data Tracker| Configure Seek Tables). Click within the Seek Table dialogue box and add a new seek table, at the end of the file name location add the name "test.txt", (ensure you use the .txt extension it is a requirement of Microsoft Access).
Select the radio button Comment
In Search String enter **new password**
Select the Contains radio button
Click Include Entry Button
Click the Apply Seek table Button to execute the search
Once the scan is complete
Change the Display function on the left-hand panel to display Matched Items
Using the DSA project you will see 10 items listed
Display the Fields by Module to see in which modules the data elements were discovered
Click the Export Results to write the selected information for further processing.

The workbench will then display the following dialogue box.



Seek Tables and Report Files have been generated.

Invoke Global Scanner from main Workbench Toolbar to apply new Seek Tables to source.

This confirms that the seek tables and reports have been generated and prompts the programmer or analyst to run the Global Scanner.

18.1 The Global Scanner

The **Global Scanner** uses the internal tables generated to identify and highlight the hits of the search in the assembler source, for use in the text editor or flow charter.

Close the Configure Seek Table Screen by clicking on the close (x) button
Select Tools|Batch Data Tracker|Perform Global Scan

The Global Scan performs two functions, as stated it marks (highlights) the source for use with the text editor and flow charter, it also generates the data required for the Assessment Report.

Select Tools|Batch Data Tracker|Assessment Report
The browse for folder dialogue box appears select the project folder …..\Projects\DSA

This process will now generate the Data Change Impact Assessment Reports.

The first report is a summary report detailing the number of hits, number of modules and total lines of code affected.

The second report, Module Summary by Name, details number of hits in each module, total and executable lines of code and complexity of the module.

The third report, Module Summary by Hits, is similar to the second report but sorted by number of hits and includes a percentage of the executable lines of code affected.

The fourth report, Module Hits Detail, provides details of which hits where found in each module and the reason for the hit.

Now view the affected modules in the text editor to see the hits highlighted in the source.

Close the reports
Open the Function Catalogue
Double Click on the Module DSAUSER to make it the active module
Open the Text Editor
Click on the source in the Text Editor
Press Shift F8
To find next marked line
Shift F8 will find the next etc
Right click and select show in Flowchart
The flowchart will now be started and centred on the highlighted code.
Perform the same on the module RXLKPWUP and ZICH001

The Batch Data Tracker will locate, highlight and report on the searches, detail the hits in each module and provide an assessment of the work that would be involved in making a change, the workbench will provide a more detailed assessment when the Batch Data Tracker is used in combination with the Business Rules Extraction (batch BRE Processing), detailed later in this guide.

18.2 Clear Tracker Files

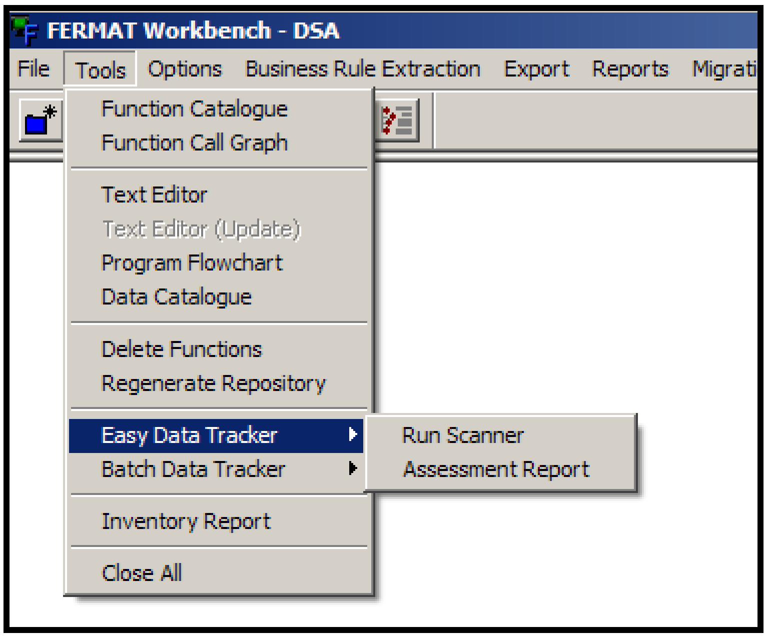
This process will clear any tracker files held in the projects repository, this option is used in conjunction with creating sub-projects described later in this manual.

18.3 Scan Macros/Copybooks

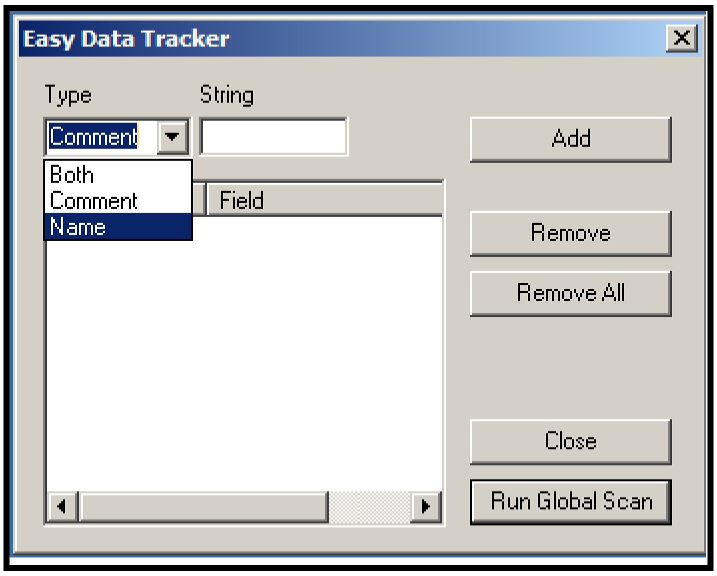
This option is only available for projects that only contain Macros and Copybooks, which are used in the Options | Project Settings of the Workbench. This facility allows for multiple projects to share Copybooks and Macros, which reduces the size of the workbench repository and also aids on-going management as only one copy of the Macros and copybooks are required to be stored.

19. The Easy Data Tracker

The Easy Data Tracker is a simplified version of the Batch Data Tracker, and is invoked from the main Workbench menu bar as follows:



Two options are available the Run Scanner and the Assessment Report. The Run Scanner presents the following dialogue box:



The options available are to search within the comments or data name of a project for elements that contain the string being searched on (same as wild cards at the start and the end of the string). The comments being searched are those on the same line as a data item.

Press the add button to add the search type and string to the search list in the large window. There is no limit to the number of fields to search on.

Once the search list is complete Run Global Scan, this matches the search criteria against the source modules and highlights the hits as with the Batch Data Scanner, then run the assessment report as before.

The Easy Data Scanner is used when programmers know exactly what fields they wish to search upon and for improved performance, but for increased functionality use the Batch Data Tracker.

20. Master Projects and Sub-projects

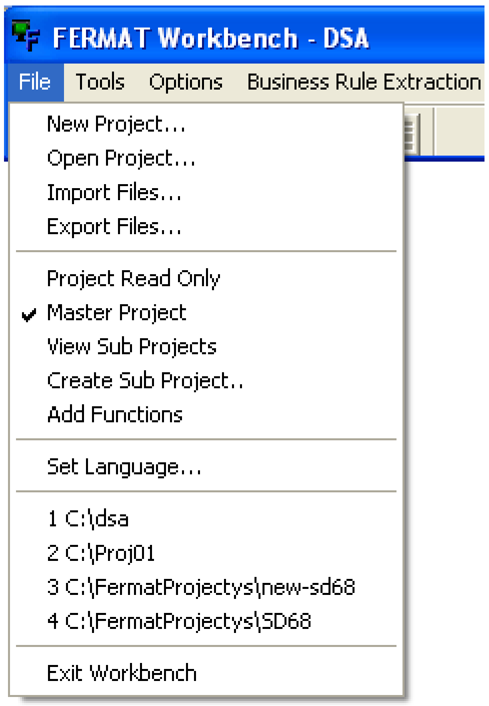
The use of Master and Sub-projects provides a convenient process for managing the Workbench projects in a system.

A Master project can have any number of Sub-projects associated with it; the benefit of the Master/Sub-project relationship is that any changes to the Master project are also replicated in the associated Sub-projects.

As an example, Master project ABC has 2 Sub-projects which have been created called DEF and XYZ. If we have a new or updated module that requires importing into the workbench, the user only needs to import the module into the Master Project and the Workbench system will automatically import the new or updated module into the associated Sub-projects and update their repository.

If the three projects ABC, DEF and XYZ were not designated as Master/Sub-projects, then the new or updated modules would require importing into each project individually.

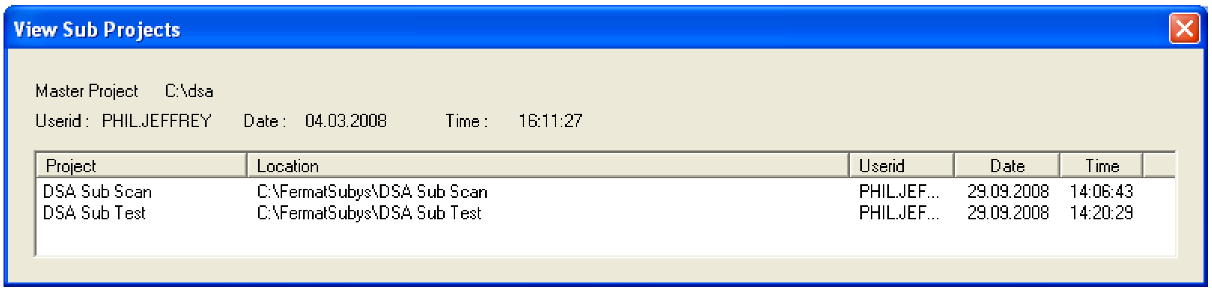
A project can be designated a master project by selecting the File | Master Project from the main title menu bar, as below:



Once the project has been designated as a Master Project, any Sub-projects created from the project will become “Managed” sub-projects and will be automatically updated when new modules are imported.

To view which Sub-projects are associated with a master project, select File | View Sub-Projects in the main title menu bar.

This will display the following information:



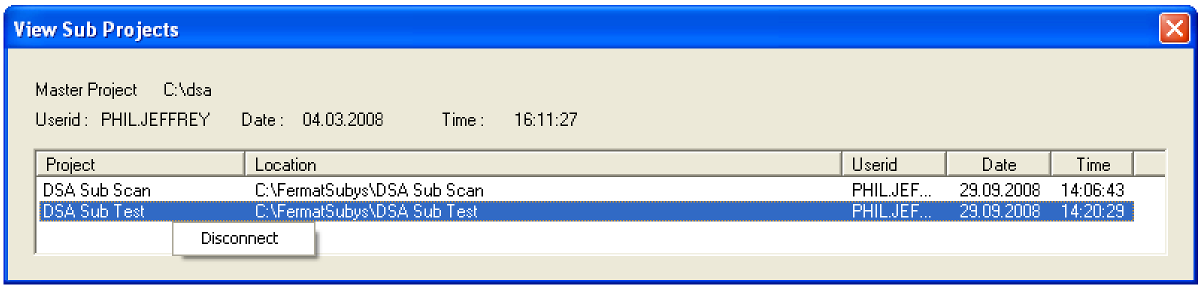
Master Project C:\dsa

Userid : PHIL.JEFFREY Date : 04.03.2008 Time : 16:11:27

Project	Location	Userid	Date	Time
DSA Sub Scan	C:\FermatSubys\DSA Sub Scan	PHIL.JEF...	29.09.2008	14:06:43
DSA Sub Test	C:\FermatSubys\DSA Sub Test	PHIL.JEF...	29.09.2008	14:20:29

This shows the name of the Master project and location, Sub-project name and location along with the user id of the user who created the Sub-project and the creation date.

To remove the logical link between a Sub-Project and its Master project - whilst in the View Sub-Projects (above); right clicking on the Sub-project will give the option to **Disconnect** which ends the projects association, as shown below:

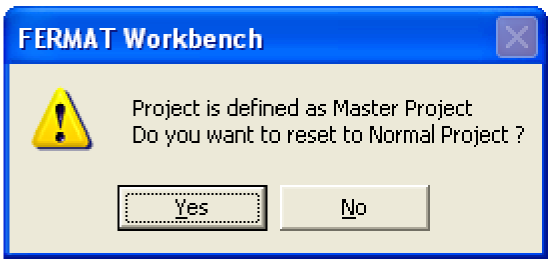


Master Project C:\dsa

Userid : PHIL.JEFFREY Date : 04.03.2008 Time : 16:11:27

Project	Location	Userid	Date	Time
DSA Sub Scan	C:\FermatSubys\DSA Sub Scan	PHIL.JEF...	29.09.2008	14:06:43
DSA Sub Test	C:\FermatSubys\DSA Sub Test	PHIL.JEF...	29.09.2008	14:20:29

A Master project can be returned to the status of a “normal” project by selecting File | Master Project, this will display the following warning box:



Project is defined as Master Project
Do you want to reset to Normal Project ?

Note that any association with sub-projects will then be lost and they will no longer be “managed”, changing the status of the project back to Master would lose all associations with any previous Sub-projects.

21. Creating Sub-projects

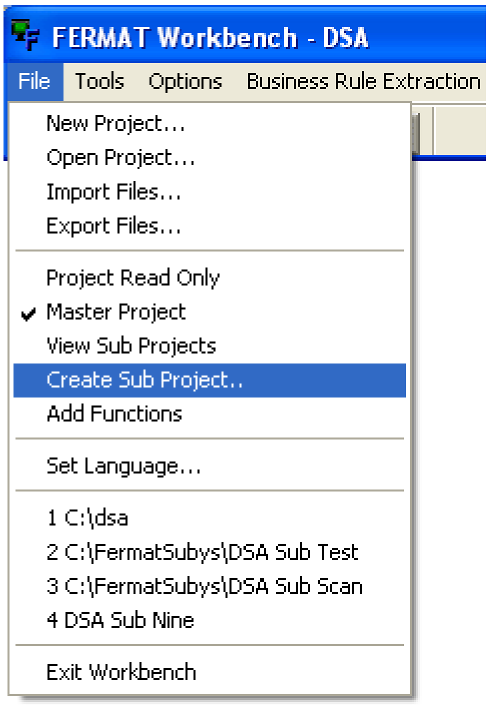
A Sub-project is a smaller section of a main project – see section 20 Master Projects and Sub-projects. There are significant benefits for the user in creating a Sub-project, the new project will be smaller in size, and therefore the analysis of the project will be more efficient as items such as the Seek Tables used in the Batch Data Tracker will only hold the data for the modules in the Sub-project not the entire project.

The modules to be included in a sub-project can be identified in the following ways:

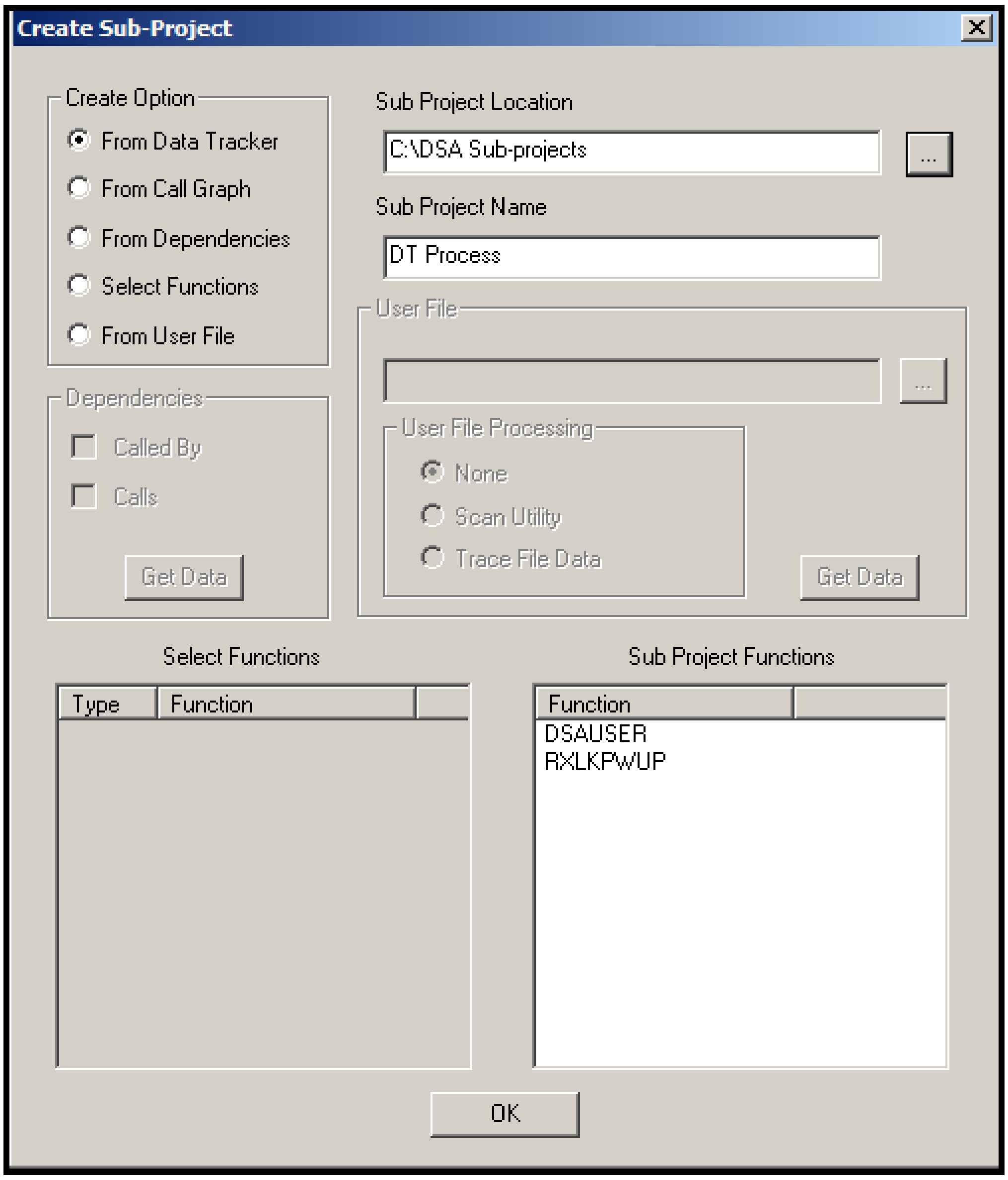
- Modules identified using the Batch Data Tracker
- Modules identified using the Call Graph
- Via Dependencies
- Select Functions
- Using User Files

To create a Sub-project select File | Create Sub-project as below:

Note to create a sub-project the current project must be declared a master project see Section 20.



This will produce the following dialogue box.



The Create Sub-project dialogue box requires the user to specify the name and location of the new sub-project.

The Sub-project Location can be any valid directory on the system

The Sub-project Name is required to be unique within the Sub-project Location directory.

When the Workbench creates the Sub-project it will create a new project and copy over all the files associated with the selected functions; including dependencies such as Macros and Copy Books. If the Data Flow Analysis has been performed on the selected modules then the Data Flow Tables will also be copied to the new project, it will automatically execute Run Data Scanner and when complete change the current project to the newly created sub-project.

21.1 Creating a Sub-project from the Data Tracker

If the as in the above example the user has run the Batch Data Tracker previously in this project, then this information can be used to create the sub-project. In the above example this would create a sub-project containing just the modules DSAUSER and RXLKPWUP and their associated dependent macros and copy books. The new Sub-project would be placed in the directory C:\DSA Sub-projects and call the new Sub-project DT Process.

21.2 Creating a Sub-project from the Call Graph

The user can create the new sub-project from modules identified in the Call Graph in exactly the same way as the previous example, by selecting the Call Graph radio button at the top left of the screen. Note if the Text Editor or the Call Graph have not previously been used then these options are greyed out.

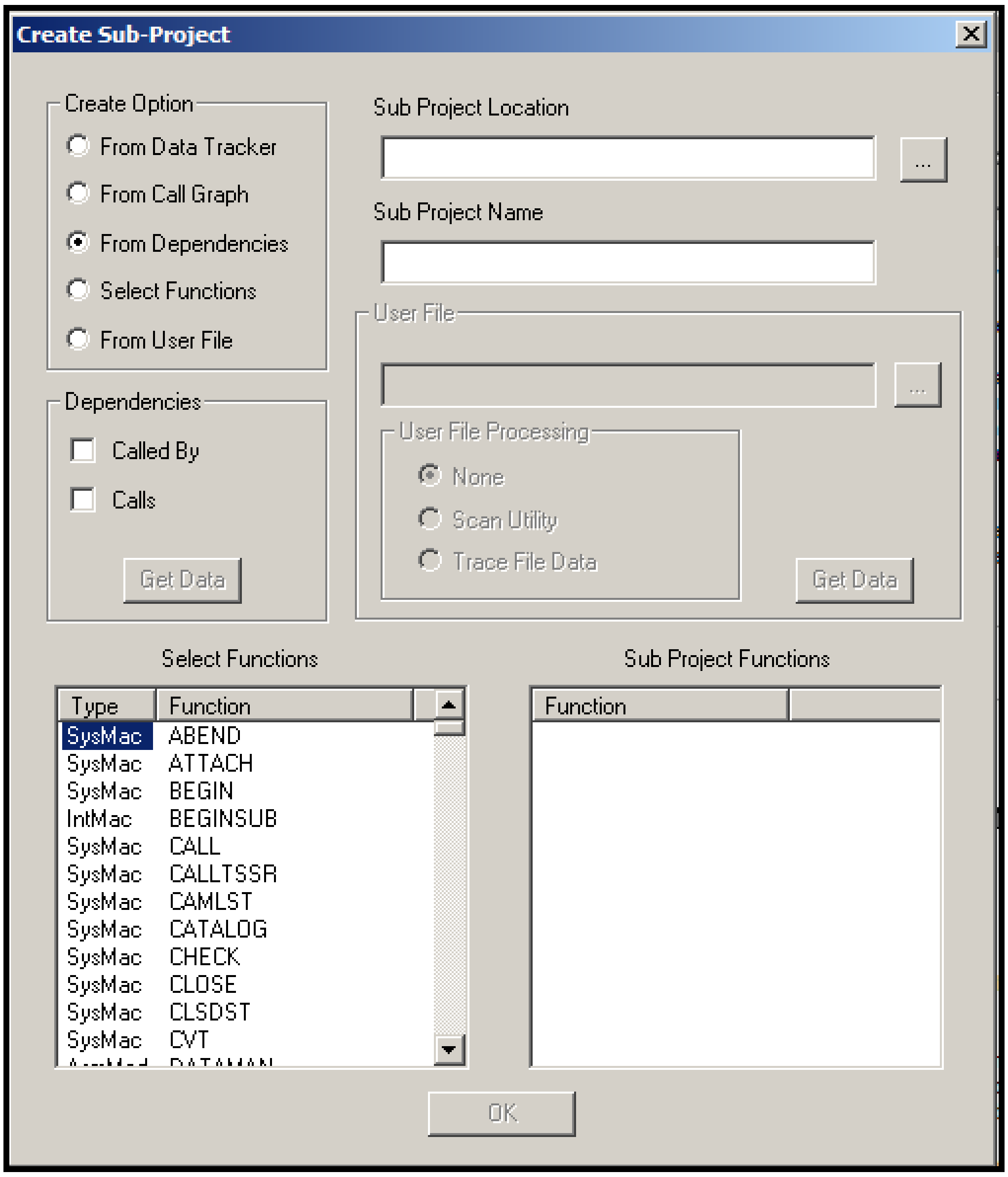
Pressing the OK button will create the sub-project using the modules in the lower right panel.

21.3 Creating a Sub-project from a function's dependencies

To create a sub-project from dependencies means to select one or more components of the master project (modules, macros and copybooks) and create the Sub-project using modules that the components call or are called by.

Example of usage, if you want to change the a variable of a file definition that is declared in a macro and you want to create a sub-project of all the modules that use that macro for further investigation, then use this option.

The dialogue box will display all components of the master project, as below.



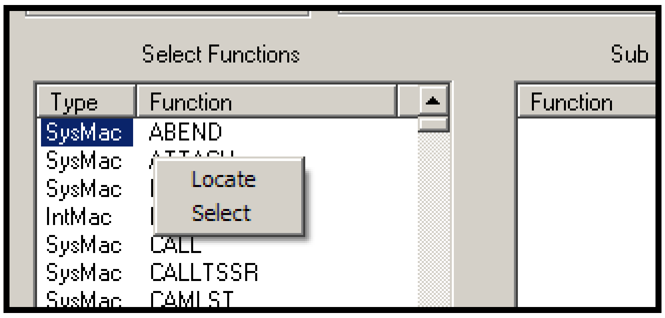
Create Option: [✓ From Dependencies]

User File Processing: [✓ None]

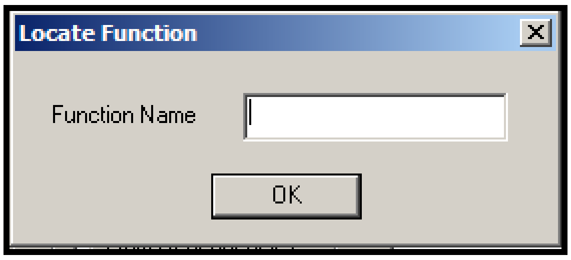
Type	Function
SysMac	ABEND
SysMac	ATTACH
SysMac	BEGIN
IntMac	BEGINSUB
SysMac	CALL
SysMac	CALLTSSR
SysMac	CAMLST
SysMac	CATALOG
SysMac	CHECK
SysMac	CLOSE
SysMac	CLSDST
SysMac	CVT

The bottom left panel will display all the functions available in the project.

By right clicking on the Type field for a function provides the following options:



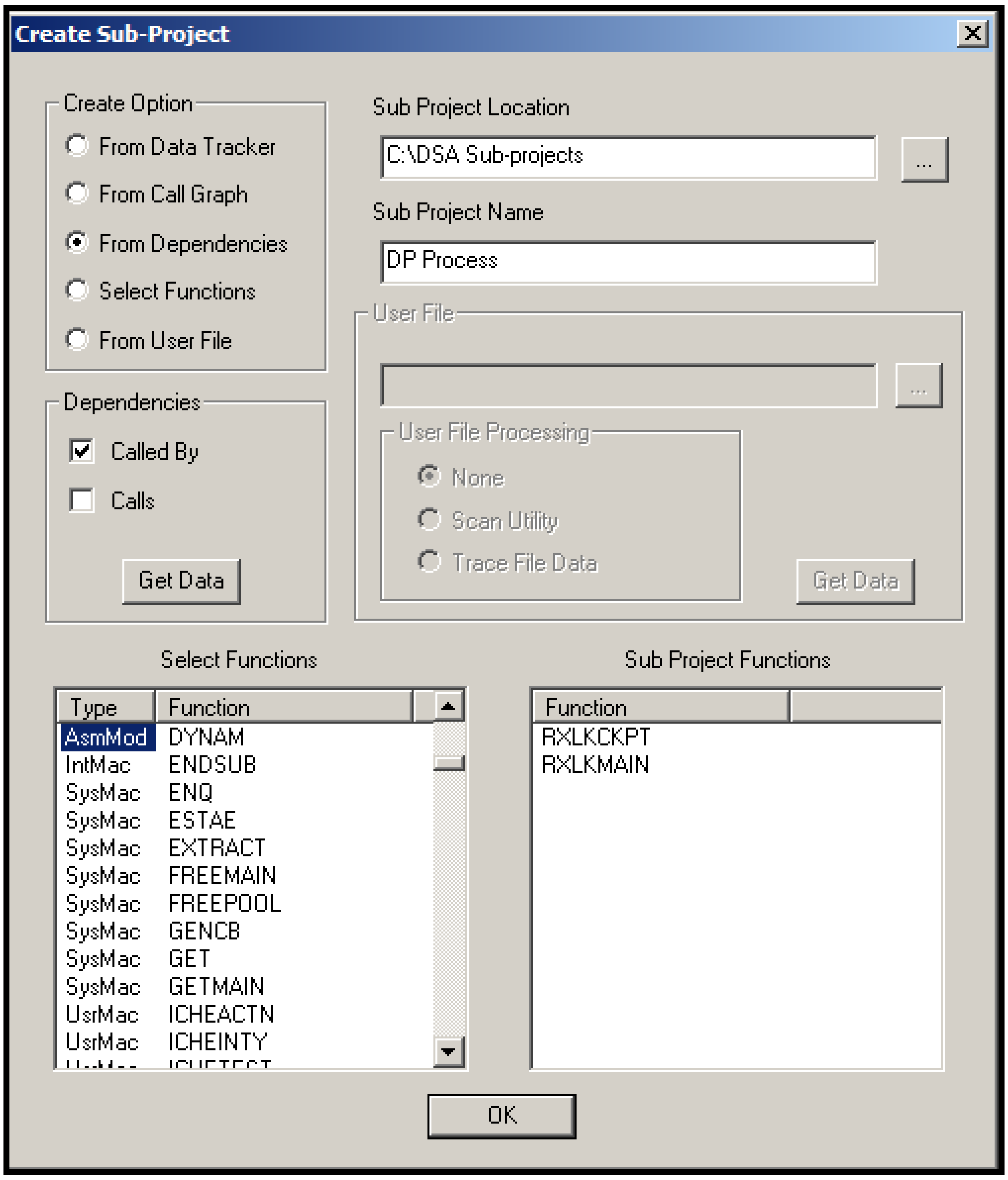
Select will select the Function for further processing, Locate will show the following dialogue box:


Locate Function

Function Name

This provides a quick way of navigating the functions within a large module, enter in the first character(s) of the function you are searching for and the Function list will be adjusted accordingly.

Once the required functions have been selected, the user must declare the dependency either; what the function calls or what it is called by:


Create Sub-Project

Create Option
- ○ From Data Tracker
- ○ From Call Graph
- ● From Dependencies
- ○ Select Functions
- ○ From User File

Sub Project Location: C:\DSA Sub-projects

Sub Project Name: DP Process

User File

User File Processing
- ● None
- ○ Scan Utility
- ○ Trace File Data

Dependencies
- ☑ Called By
- ☐ Calls

Select Functions

Type	Function
AsmMod	DYNAM
IntMac	ENDSUB
SysMac	ENQ
SysMac	ESTAE
SysMac	EXTRACT
SysMac	FREEMAIN
SysMac	FREEPOOL
SysMac	GENCB
SysMac	GET
SysMac	GETMAIN
UsrMac	ICHEACTN
UsrMac	ICHEINTY

Sub Project Functions

Function
RXLKCKPT
RXLKMAIN

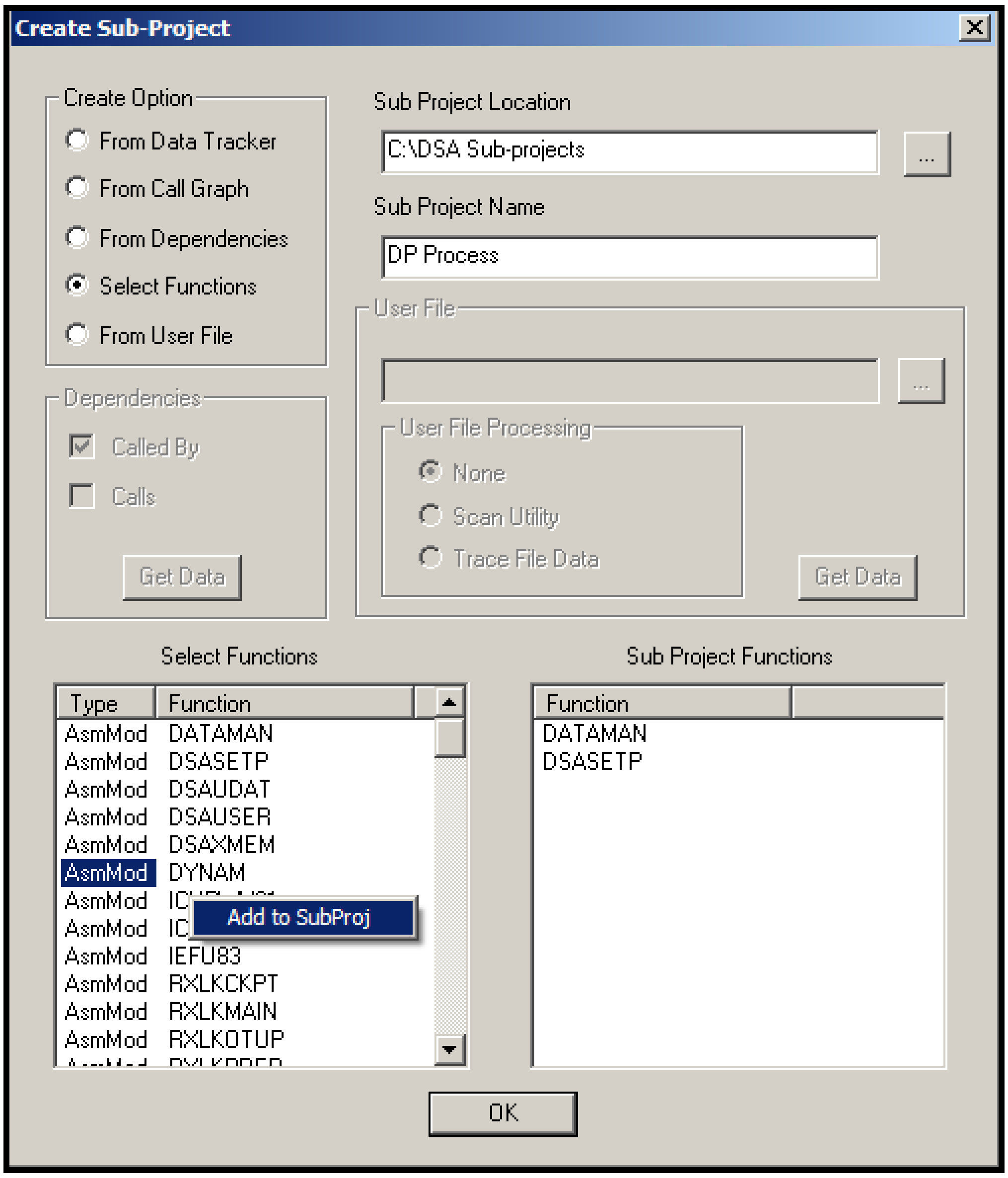
Software Migrations Ltd

In the above example the function DYNAM an assembler module has been selected, the dependency has been selected as Called By, the Workbench will then search the repository to find all modules that call the module DYNAM and place the names of the modules in lower right panel. If required the user can delete modules from the identified Sub Project Functions by right clicking the module name an selecting the delete option.

Pressing the OK button will create the sub-project using the modules in the lower right panel.

21.4 Creating a Sub-project from Select Functions

To create a sub-project by selecting the modules from a list of all modules in the master project select the Select Functions radio button, this will show the following dialogue box.



Sub Project Location: C:\DSA Sub-projects

Sub Project Name: DP Process

Select Functions

Type	Function
AsmMod	DATAMAN
AsmMod	DSASETP
AsmMod	DSAUDAT
AsmMod	DSAUSER
AsmMod	DSAXMEM
AsmMod	DYNAM
AsmMod	IC
AsmMod	IC
AsmMod	IEFU83
AsmMod	RXLKCKPT
AsmMod	RXLKMAIN
AsmMod	RXLKOTUP

Sub Project Functions

Function
DATAMAN
DSASETP

All the modules in the project are displayed in the lower left panel, one or more modules are selected by right clicking on the type field of the module concerned, once clicked the Add to

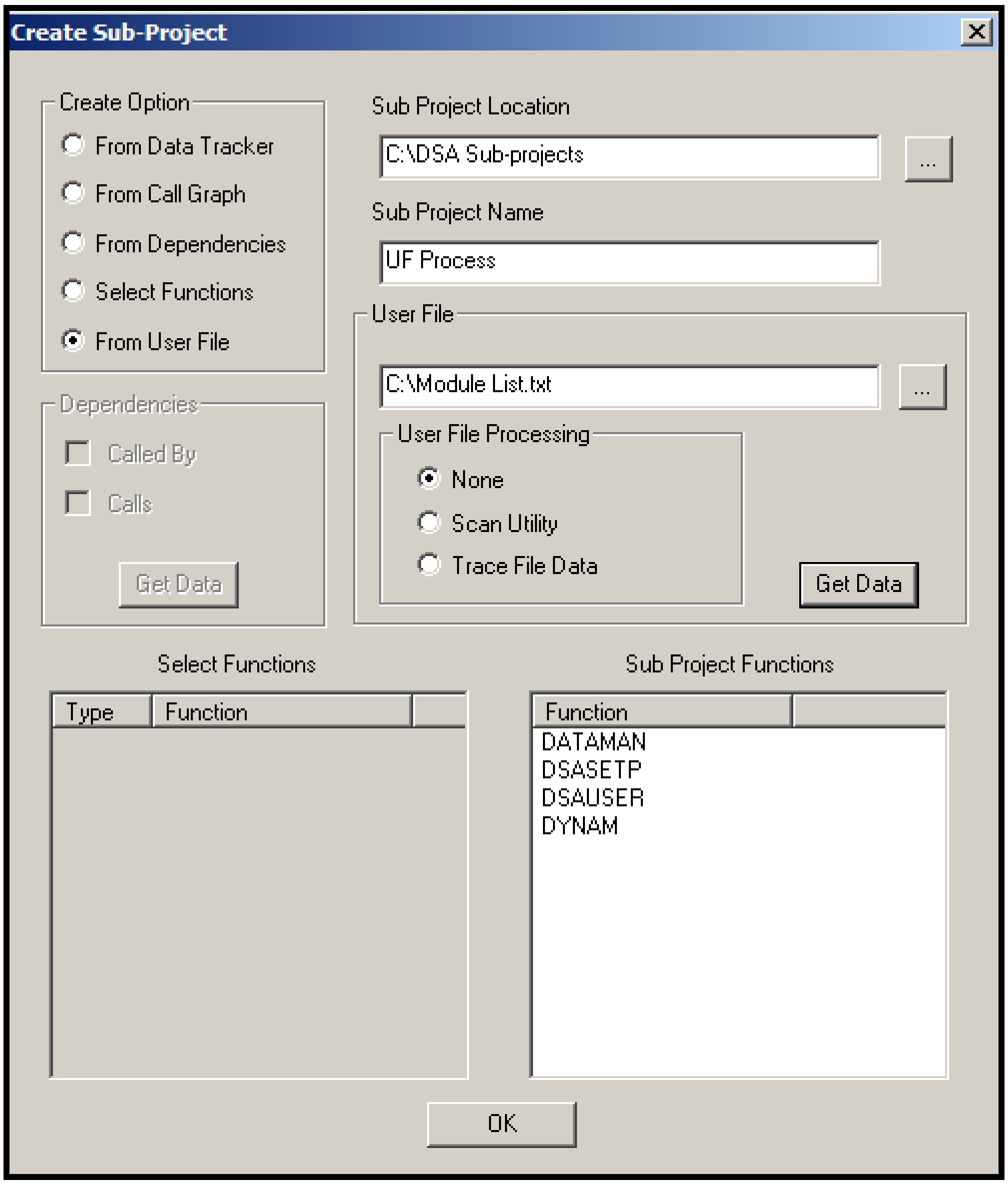
Subproj option appears selecting this will place the name of the module in the lower right panel. If required the user can delete modules from the identified Sub Project Functions by right clicking the module name an selecting the delete option.

Pressing the OK button will create the sub-project using the modules in the lower right panel.

21.5 Creating a Sub-project from a User File

This option allows a user to create a sub-project from a pre-generated list of modules which have been compiled outside the actual workbench.

The user is required to enter the location of the file to process or use the browse button to right of the field. Once the file has been located the user has 3 other options under the User File Process


Create Sub-Project

Create Option: From Data Tracker / From Call Graph / From Dependencies / Select Functions / [✓ From User File]

Dependencies: Called By / Calls

Sub Project Location: C:\DSA Sub-projects

Sub Project Name: UF Process

User File: C:\Module List.txt

User File Processing: [✓ None] / Scan Utility / Trace File Data

Select Functions

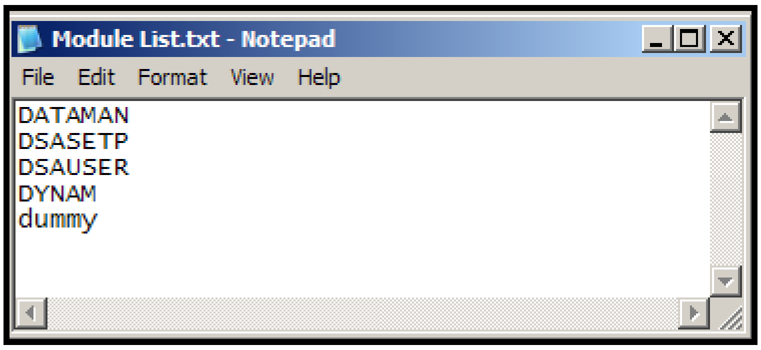
Type	Function

Sub Project Functions

Function
DATAMAN
DSASETP
DSAUSER
DYNAM

The **None** option will not perform any pre-processing on the file, and expects to find a .txt file with valid module names.

The following example is the .txt file used in the above diagram.



```
DATAMAN
DSASETP
DSAUSER
DYNAM
dummy
```

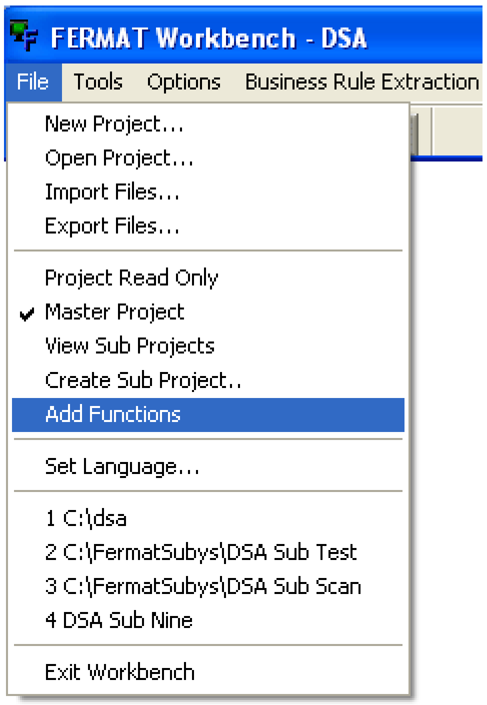
As can be seen the file contains a simple list of module names any invalid module names e.g. dummy will be ignored by the process.

The **Scan Utility** option will pre-process the output file generated by Workbench Text Scanner process in the Short Form report format – See section 25 Utilities for further information on the Text Scanner. The pre-processing will extract the module names from the report and use this as the module names to create the subproject. Identified valid module names will be displayed in the lower right panel and can then be deleted from the list if required.

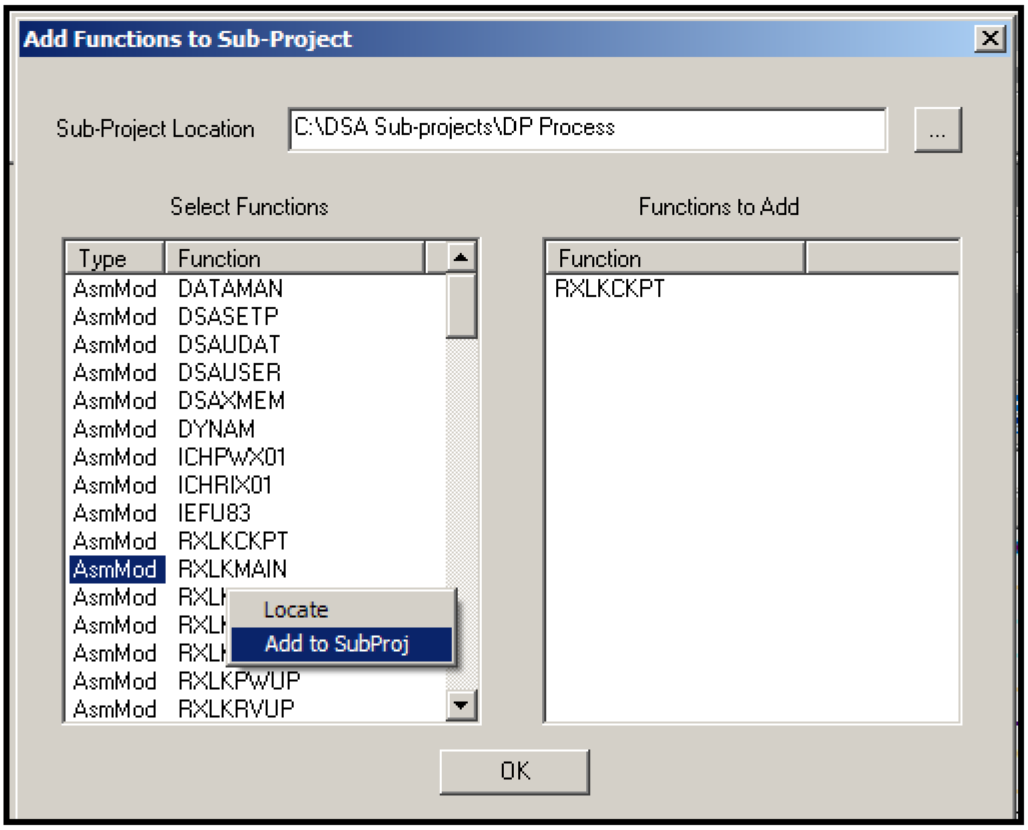
The **Trace File Data** option is only valid for TPF Assembler systems; this process uses the output from the system SST Summary Display execution trace utility, and pre-processes this text file to extract the module names to be used in the creation of the sub-project.

21.6 Adding Functions to a Sub-project

The user has the ability to add functions to an existing sub-project, by selecting the File | Add Functions from the main menu, as below:



The Current project must be the original master project which was used to generate the sub-project; this action will display the following dialog box.



Sub-Project Location: C:\DSA Sub-projects\DP Process

Type	Function
AsmMod	DATAMAN
AsmMod	DSASETP
AsmMod	DSAUDAT
AsmMod	DSAUSER
AsmMod	DSAXMEM
AsmMod	DYNAM
AsmMod	ICHPWX01
AsmMod	ICHRIX01
AsmMod	IEFU83
AsmMod	RXLKCKPT
AsmMod	RXLKMAIN
AsmMod	RXLK
AsmMod	RXLK
AsmMod	RXLK
AsmMod	RXLKPWUP
AsmMod	RXLKRVUP

Function
RXLKCKPT

The user must specify the sub-project location or navigate to it using the browse button. All the functions in the master project are displayed in the left hand panel, the user selects which modules to add by right clicking on the function, with the options to add the module to the sub-project or to enter a string to locate an entry in the list of modules in the master project.

In the above example the module RXLKCKPT has been added to Functions to Add list and the module RXLKMAIN will be added. Functions can be deleted from the Functions to Add list by right clicking the function name and selecting delete.

When all additions to the sub-project list are complete press **OK**. The Workbench will then import the new functions into the sub-project, rebuild the repository and run the Data Scanner.

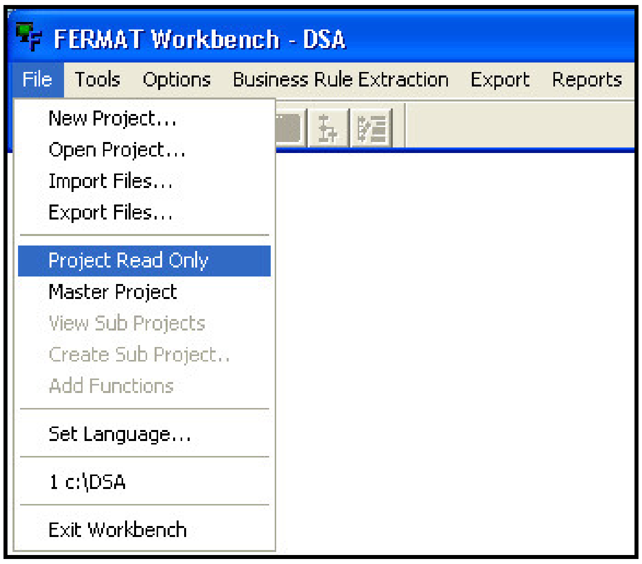
22. Read-only Projects

By default, new projects created in the workbench have update capabilities. This will allow any user to mark modules with highlights, add annotations, change modules and import new modules etc.

For most installations this method of working allows complete flexibility over the use of the workbench.

In some circumstances it may be desirable to prevent users from making changes within a Project. For example where a Workbench Project is defined which will be used as an accurate copy of the mainframe live assembler code base. Such a project should be made read-only to prevent users from making amendments which would make the project out-of-step with the mainframe version of the source library.

The capability to make a project read-only is available only to FermaT Workbench administrators, and is found on the File menu item from the main Workbench window.



Once Read Only status is enabled, any user opening the project will only be allowed to view modules and information about the project and any mark-ups made as a result of data tracker or code slicing will mark up a copy of the source visible only to the logged in user.

Access to the following functions will be either disallowed, or will have functionality changed in a read-only project to ensure that the contents of the project are not updated.

File->Import Files	Disabled
File->Master Project	Disabled

Tools->Text Editor (update)	Disabled
Tools->Delete Functions	Disabled
Tools->Regenerate Repository	Disabled
Tools->Easy Data tracker->Run Scanner	When running the scan, the Workbench will only mark a user copy of the modules.
Tools->Batch Data Tracker->Perform Global Scan	When running the scan, the Workbench will only mark a user copy of the modules.
Business Rules Extraction->Import Listings	Disabled
Business Rules Extraction->Import Original Source	Disabled
Business Rules Extraction->Reset Source File	Removes the user copy of the module.
Business Rules Extraction->Batch Reset Source File.	Removes the user copies of the modules.
Business Rules Extraction->Batch BRE Processing	Disabled
Business Rules Extraction->Assessment Report	Disabled
Reports->User Macro Report	Check box to allow user macro lines to be marked up in modules is disabled

A FermaT Workbench administrator can re-instate normal project behaviour by again selecting File-> Project Read Only from the main Workbench window. This will, of course, remove all user specific mark-ups.

22.1 Using a Read-Only project

The principal use of a read-only project is to maintain synchronisation between the Fermat Workbench Project and assembler source located on the mainframe.

Since importing of source and listings is disabled within a read-only project, only a Fermat Workbench Administrator can import additional modules into the project, by first switching off the read-only indicator, importing new modules or listings as required, and then re-enabling the read-only indicator.

Alternatively periodic importing of new and updated source, listings, macros and copybooks can still be performed on a read-only project using the Fermat Workbench Batch Processing facility.

A read-only project will allow modules to be viewed, and will allow the use of the Data Tracker and code slicing facilities to identify scope and impact of changes. The information gathered can then be used to create a sub-project containing the modules identified. This sub-project can then be used to perform any required detailed analysis since it will not have the read-only indicator set.

23. Business Rule Extraction

This process is only available when assembler listings, with all macros fully expanded, are imported into the workbench. The use of full assembler listings enables the Import Listings process and subsequent Generate Dataflow process to perform sophisticated Dataflow and Control Flow analysis of assembler modules, to understand how the assembler actually works.

This process generates a series of files that are stored in the LEVEL02 folder in the project directory; the analysis process automatically identifies dead code candidates[5] and enables the Business Rule Extraction process and Code Slicing to be achieved.

The Business Rule Extraction process uses the Fermat Migration process to represent the assembler code in an intermediate language WSL (Wide Spectrum Language)[6], the WSL code is then transformed using thousands of self verifying proven transformations to restructure the code and analyse the code in terms of Dataflow and control flow dependencies. See section 32 Understanding Data Flows and Control Flows.

23.1 What is a Business Rule?

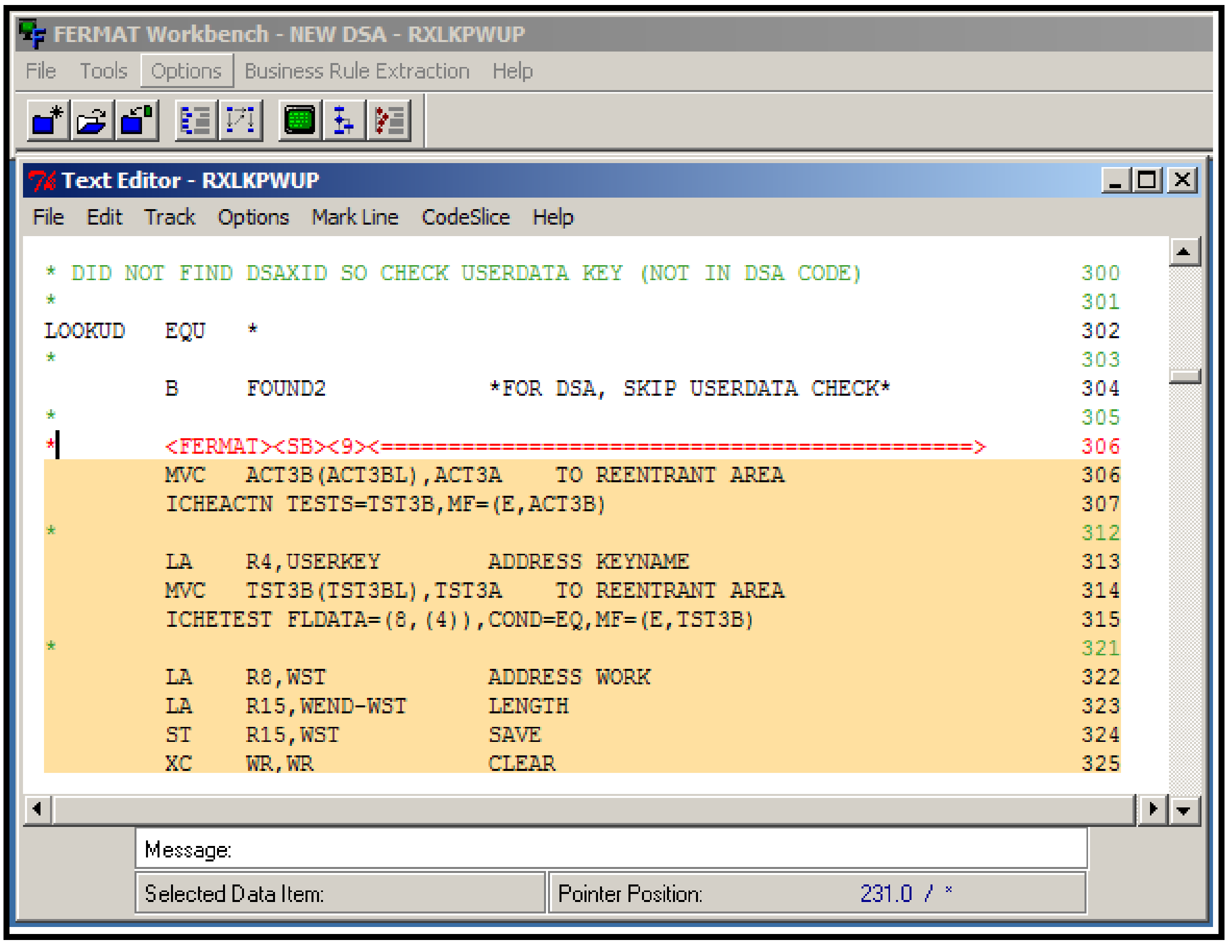
The term Business Rule can mean many things to many people depending on its use and context. In the Workbench a Business Rule is defined as code that is executed in a module that affects the referenced data element. Using the Workbench this code can be identified and documented in business terms to assist organisations; who need to document their existing systems prior to re-engineering them in new languages.

23.2 Dead Code Candidates

Dead Code Candidates are automatically identified by the Workbench dataflow generation process, they are highlighted as a special marked block type 9, coloured in this example as light brown, to view an example of dead code.

Make the module RXLKPWUP the current active module from the Function Catalogue or Function Call Graph
Open the Text Editor
Click in the text and press shift F8 to go to the next marked block
The following text will be displayed

[5] Dead Code Candidates is code that cannot logically be referenced from within the module typical of evolved legacy code and systematic of quick system fixes and patches.
[6] Detailed information on WSL can be found on the SML web site www.smltd.com


FERMAT Workbench - NEW DSA - RXLKPWUP
File Tools Options Business Rule Extraction Help
Text Editor - RXLKPWUP
File Edit Track Options Mark Line CodeSlice Help

```
* DID NOT FIND DSAXID SO CHECK USERDATA KEY (NOT IN DSA CODE)     300
*                                                                 301
LOOKUD   EQU    *                                                 302
*                                                                 303
         B      FOUND2            *FOR DSA, SKIP USERDATA CHECK*  304
*                                                                 305
*        <FERMAT><SB><9><============================================> 306
         MVC    ACT3B(ACT3BL),ACT3A     TO REENTRANT AREA         306
         ICHEACTN TESTS=TST3B,MF=(E,ACT3B)                        307
*                                                                 312
         LA     R4,USERKEY        ADDRESS KEYNAME                 313
         MVC    TST3B(TST3BL),TST3A     TO REENTRANT AREA         314
         ICHETEST FLDATA=(8,(4)),COND=EQ,MF=(E,TST3B)             315
*                                                                 321
         LA     R8,WST            ADDRESS WORK                    322
         LA     R15,WEND-WST      LENGTH                          323
         ST     R15,WST           SAVE                            324
         XC     WR,WR             CLEAR                           325
```

Message:

Selected Data Item:

Pointer Position: 231.0 / *

As can be seen from this code on line 304 there is a branch to the label FOUND2 and the code between lines 306 and 365 including the labels GOTKEY, LENKOK and MOVEID are not referenced in the code, therefore this block is marked as a dead code candidate. There are other examples of dead code within this module on lines 456 and 476 – 489.

What is code slicing? This provides the ability to trace the logic within an assembler module from given start point, the Workbench has the facility to perform different types of code slicing:

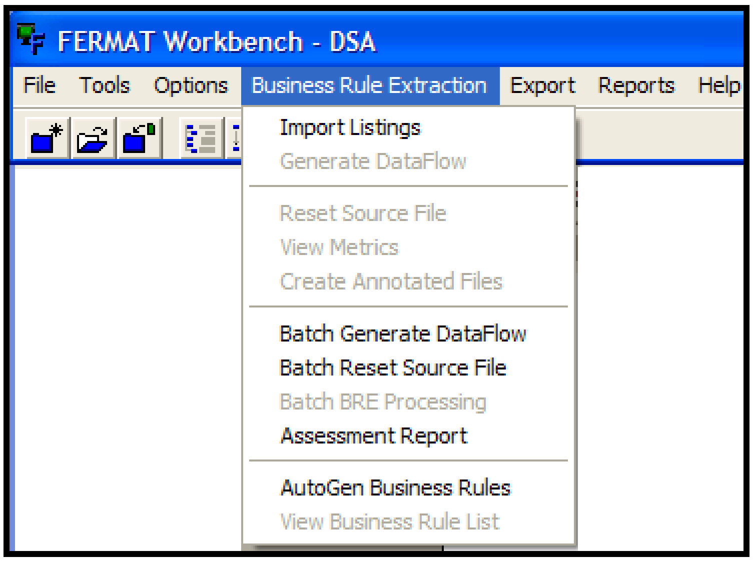
Data Flow – this analysis will identify the where data is being moved, stored and referenced.

Control Flow – this analysis will identify the control dependency flows within a module.

Both Control Flow and Data Flow – this will produce different results than simply combining the output of individual Data Flow and Control Flow as a control flow link may will lead to some data flow links, which in turn may lead to other control flow links and so on.

Using the text editor the developer identifies a data element or register to be the start point for a code slice. The developer can perform a backwards slice to highlight all the instructions which could have contributed to the value of the variable at this point in the module; a forward slice highlights all the instructions which could be affected if the variable is changed at this point in the module. A developer can also perform a bi-directional code slice of a data element. This is a combination of both backward and forward code slicing. This will highlight all code executed in affecting the data element. From the highlighted code the developer and analyst can determine the business rules incorporated within the module. This ability to highlight which code is executed is a very powerful facility in assembler, since any data element can be referenced in any number of ways, such as off-set addressing, which may not be immediately apparent to the developer.

The Business Rules Extraction processes are selected from the main menu bar as follows:-



The following is a description of the options available.

23.3 Import Listings

Import Listings - this process is used to import the assembler listings as detailed in section 7.2 Importing Assembler Listings.

23.4 Generate Data Flow

Generate Dataflow - this is the process of representing the assembler code in WSL, applying the transformations and performing the Dataflow and control flow analysis required for identifying dead code performing code-slicing or extracting Business Rules.

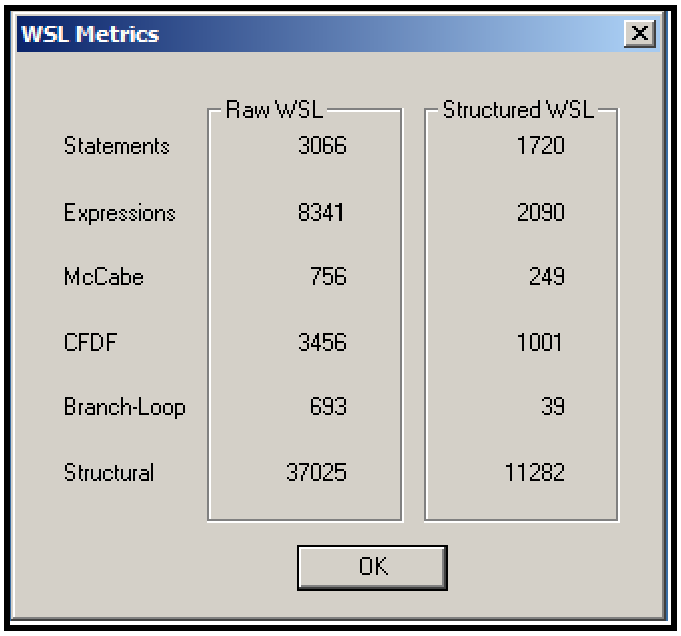
Selecting this option will generate the Data flow and Control Flow analysis tables for the currently selected module.

23.5 Reset Source File

When using Assembler listings as import to the workbench two copies are kept of the listings. One, the current, is used to mark highlights and annotations etc and the other is the original listing. This process is used to reset the current selected module back to the original listing, resetting it and therefore losing any highlights or annotations. For a read only project only the highlights and annotations for the logged in user are lost.

23.6 View Metrics

During the import and dataflow generation process the workbench generates sophisticated metrics on each module; the metrics for the currently selected module can be viewed. As below:



	Raw WSL	Structured WSL
Statements	3066	1720
Expressions	8341	2090
McCabe	756	249
CFDF	3456	1001
Branch-Loop	693	39
Structural	37025	11282

The following is a description of these metrics, the Raw WSL is equivalent to the original assembler as represented in WSL, and the structured WSL represents the same code after restructuring.

Statements: Number of statements in the module.

Expressions: Number of expressions (including sub-expressions).

McCabe: McCabe Cyclometric Complexity. Basically this is the number of branches in the control flow graph for the program plus one. Straight line code has a McCabe of 1, a single binary IF statement has a McCabe of 2 and so on.

CFDF: Control Flow Data Flow metric. This is the total number of variable references plus procedure calls and branches (gotos).

Branch-Loop: The number of procedure calls, branches (gotos) and loop structures in the program.

Structural: A "weighted sum" of various features in the program, for example a simple expression counts as 1, a complex expression counts as 4, a goto counts as 10, other statements count 5 and so on.

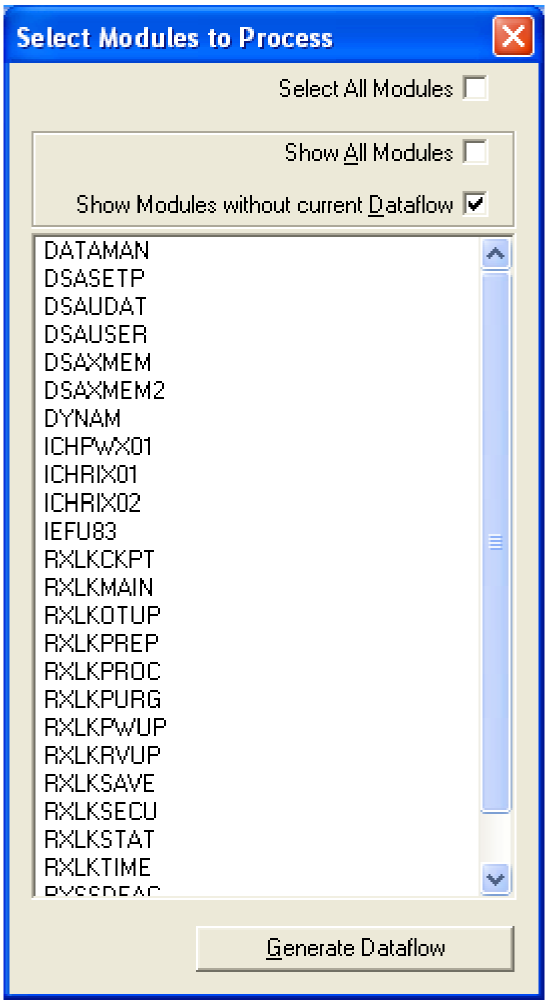
The purpose of these metrics is firstly to show the effect of the WSL to WSL transformations: typically all the metrics are reduced by a factor of 5 to 10, and secondly to show which modules have the highest complexity and can be used to estimate the effort required to make a system change or enhancement.

23.7 Create Annotated Files

This is used to perform the code slicing for selected data elements or registers for the currently selected module; see section 23.14 for further details.

23.8 Batch Generate Dataflow

This process is used when the user needs to generate the Dataflow and Control Flow files for a number of modules; it performs the same function as Generate Dataflow but for a batch of modules. When Selected this option will produce the following dialogue box.


Select Modules to Process
Select All Modules
Show All Modules
Show Modules without current Dataflow
Generate Dataflow

Select All Modules [ ]
Show All Modules [ ]
Show Modules without current Dataflow [✓]

DATAMAN
DSASETP
DSAUDAT
DSAUSER
DSAXMEM
DSAXMEM2
DYNAM
ICHPWX01
ICHRIX01
ICHRIX02
IEFU83
RXLKCKPT
RXLKMAIN
RXLKOTUP
RXLKPREP
RXLKPROC
RXLKPURG
RXLKPWUP
RXLKRVUP
RXLKSAVE
RXLKSECU
RXLKSTAT
RXLKTIME

Select modules required for processing by left clicking the mouse or combination of shift left click to select a block of modules, or control left click to select additional individual modules. By default he dialogue box shows only those modules that have not got up to date dataflow files. To see all modules regardless of their dataflow state, select the "Show All Modules" box before performing the selection.

To select all visible functions check the Select All Functions box.

Then press the **Generate Dataflow** button to invoke the process.

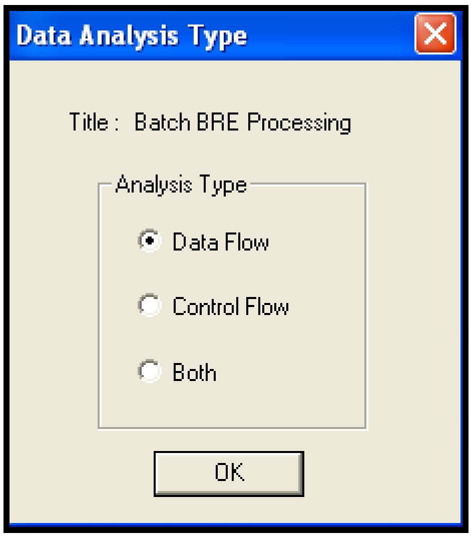
23.9 Batch Reset Source File

This resets either all or the selected modules to the original listing, the same as Reset Source File see above. Selecting this option will produce a dialogue box similar to the Batch Generate Dataflow.

23.10 Batch BRE Process

Having performed the Global Scan as part of the Batch Data Tracker or Easy Data Tracker process, this process will perform a bi-directional code slice on all hits of data in every affected module. This is then used to more accurately determine the amount of effort required to make a system change, it will also highlight all code that is executed which relates to the matched data elements.

The user is prompted to select which type of bi-directional code slice to perform, using the following dialogue box:

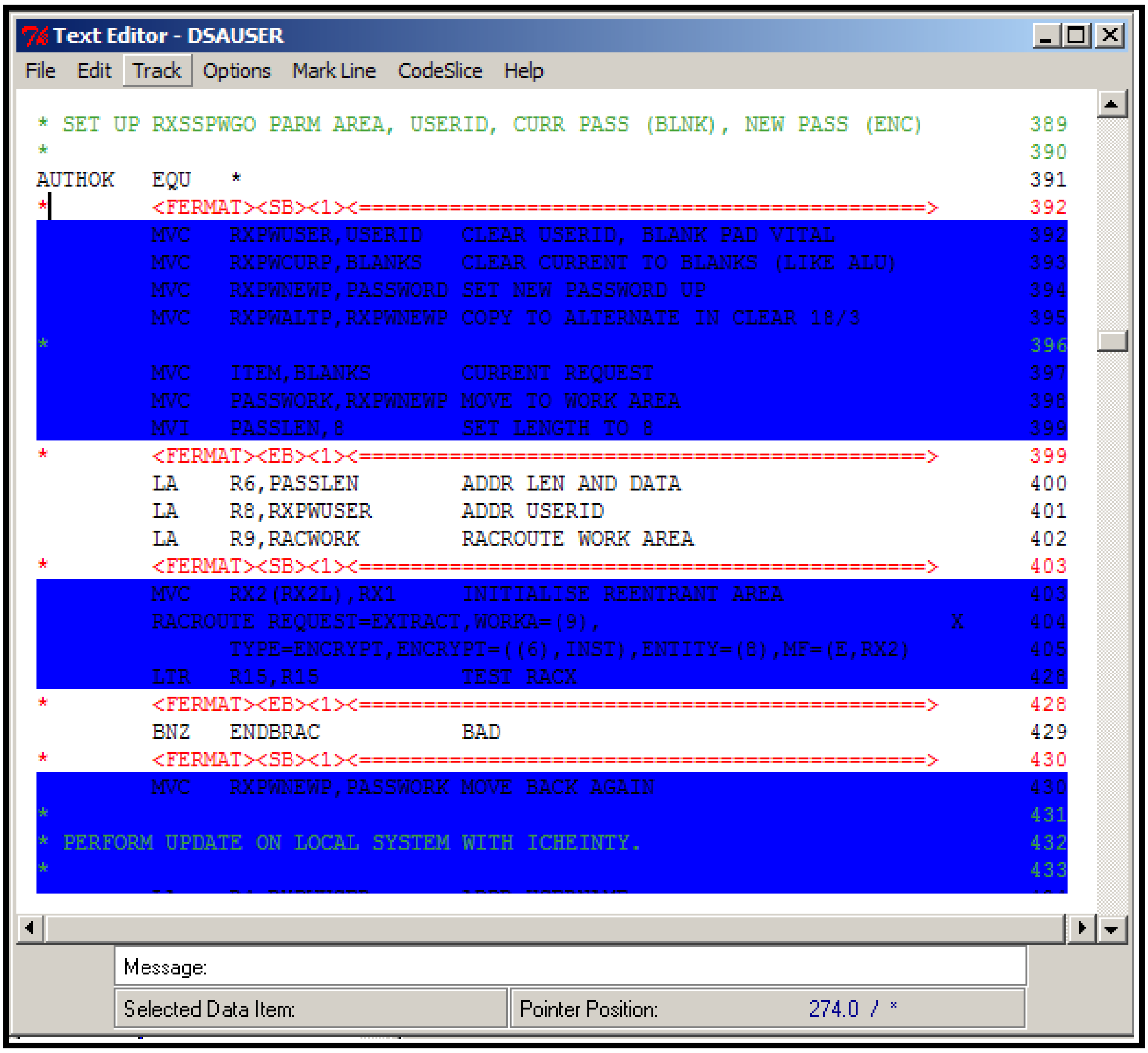


Perform the Batch Data Tracker exercise we preformed previously, again set the search criteria for a comment containing new password,
Apply Seek Table
Export Results and close the Configure Seek Tables screen.
Perform the Global Scan as before.
Select the batch BRE function by Business Rule Extraction|Batch BRE Processing

This process will perform a bi-directional code slice (either dataflow, control flow or both) for each hit in the assembler module. To see the code that is affected do the following.

Select the module DSAUSER in the function catalogue and double click to make it the current active module.
Open the Text Editor
Click in the text and press shift F8 to move to the blocks of code identified by the BRE processor.
101 lines are now highlighted from the 3 original hits in the code.
Look at modules RXLKPWUP and ZICH0001 to see the results of the BRE processing on these modules.

The following is displayed in the Text Editor – blue indicates the highlighted instructions.



```
* SET UP RXSSPWGO PARM AREA, USERID, CURR PASS (BLNK), NEW PASS (ENC)
*
AUTHOK   EQU   *
*        <FERMAT><SB><1><=========================================>
         MVC   RXPWUSER,USERID   CLEAR USERID, BLANK PAD VITAL
         MVC   RXPWCURP,BLANKS   CLEAR CURRENT TO BLANKS (LIKE ALU)
         MVC   RXPWNEWP,PASSWORD SET NEW PASSWORD UP
         MVC   RXPWALTP,RXPWNEWP COPY TO ALTERNATE IN CLEAR 18/3
*
         MVC   ITEM,BLANKS       CURRENT REQUEST
         MVC   PASSWORK,RXPWNEWP MOVE TO WORK AREA
         MVI   PASSLEN,8         SET LENGTH TO 8
*        <FERMAT><EB><1><=========================================>
         LA    R6,PASSLEN        ADDR LEN AND DATA
         LA    R8,RXPWUSER       ADDR USERID
         LA    R9,RACWORK        RACROUTE WORK AREA
*        <FERMAT><SB><1><=========================================>
         MVC   RX2(RX2L),RX1     INITIALISE REENTRANT AREA
         RACROUTE REQUEST=EXTRACT,WORKA=(9),                          X
               TYPE=ENCRYPT,ENCRYPT=((6),INST),ENTITY=(8),MF=(E,RX2)
         LTR   R15,R15           TEST RACK
*        <FERMAT><EB><1><=========================================>
         BNZ   ENDBRAC           BAD
*        <FERMAT><SB><1><=========================================>
         MVC   RXPWNEWP,PASSWORK MOVE BACK AGAIN
*
* PERFORM UPDATE ON LOCAL SYSTEM WITH ICHEINTY.
*
```

Message:

Selected Data Item: Pointer Position: 274.0 / *

23.11 BRE Assessment Report

This report can be executed following the Batch BRE process section 23.10, this will provide 2 reports which compliment the Impact Assessment Report and detail not only the number of data hits in Modules but also the number of BRE hits found as a result of the bi-directional code slicing performed in the Batch BRE process

23.12 AutoGen Business Rules

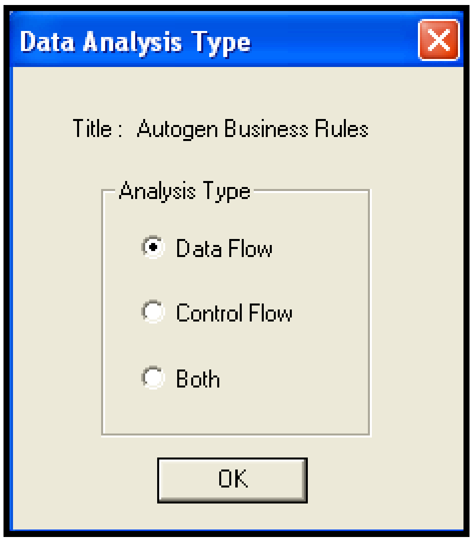
A very powerful feature of the Workbench is the ability to perform backward code slices on Data items to identify the code executed that generates the current value of the data element.

During the Generate Dataflow process the workbench generates the Dataflow and Control Flow analysis tables. It also generates an .rpt file for each module processed which is stored in the Dataflow Folder of the Level2 of the projects data structure. This file contains a number of elements required in software re-engineering projects it also holds details of all data variables both Csects and Dsects that are updated in the Module.

The AutoGen Business Rules process will perform backward code slices on the selected modules for every data element that is updated and will store these business rules in the Business Rule Repository for subsequent analysis.

Selecting this option will produce the select module dialogue box as Batch Generate Dataflow or Batch Reset Source File.

The user will be prompted to select which type of backward code slice to perform, using the following dialogue box.

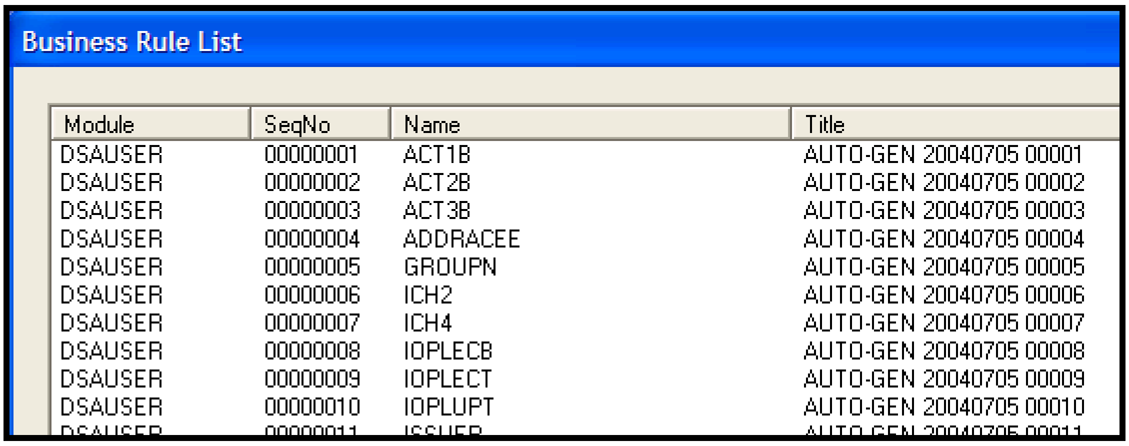


Title : Autogen Business Rules

Analysis Type: [✓ Data Flow] Control Flow Both

23.13 View Business Rule List

This option will display all the business rules in the Business Rule Repository for the entire project in the following dialogue box.



Business Rule List

Module	SeqNo	Name	Title
DSAUSER	00000001	ACT1B	AUTO-GEN 20040705 00001
DSAUSER	00000002	ACT2B	AUTO-GEN 20040705 00002
DSAUSER	00000003	ACT3B	AUTO-GEN 20040705 00003
DSAUSER	00000004	ADDRACEE	AUTO-GEN 20040705 00004
DSAUSER	00000005	GROUPN	AUTO-GEN 20040705 00005
DSAUSER	00000006	ICH2	AUTO-GEN 20040705 00006
DSAUSER	00000007	ICH4	AUTO-GEN 20040705 00007
DSAUSER	00000008	IOPLECB	AUTO-GEN 20040705 00008
DSAUSER	00000009	IOPLECT	AUTO-GEN 20040705 00009
DSAUSER	00000010	IOPLUPT	AUTO-GEN 20040705 00010
DSAUSER	00000011	ISSUER	AUTO-GEN 20040705 00011

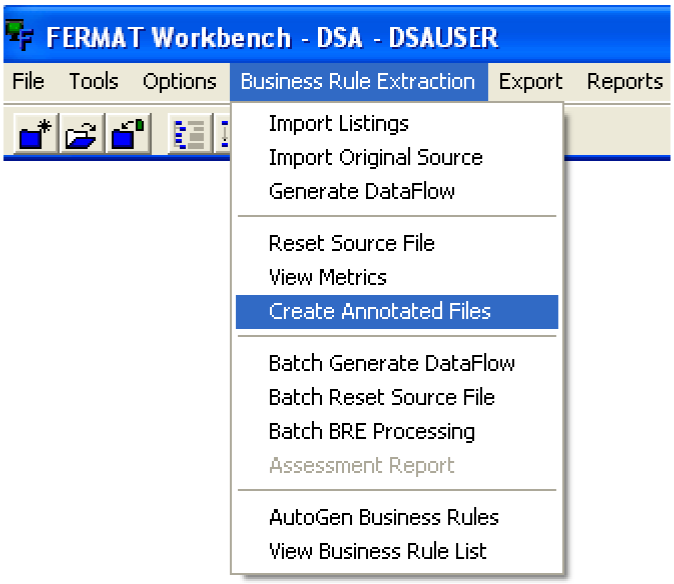
The contents of the columns can be sorted by clicking on the title bar e.g. Name.

Note AUTO-GEN at the start of the title indicates that the rules were generated by the AutoGen Business Rule Process.

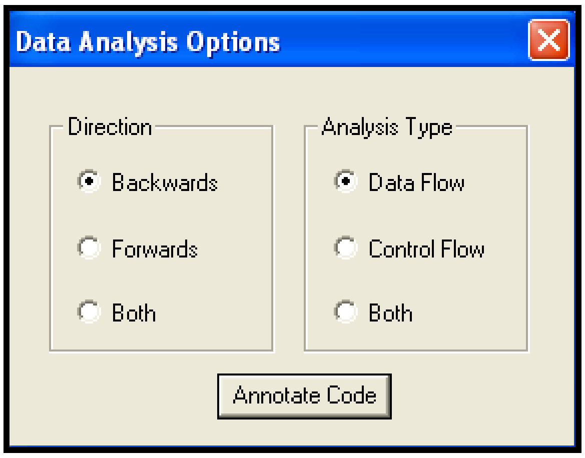
23.14 Business Rules Extraction (Code Slicing) within the Text Editor

Within the text editor the programmer or analyst can perform a code slice to identify code that is executed in generating a data element or register at this point in the module or code that is affected by this data element as the module is executed.

The user selects the target for the code slice operation by right clicking on the variable, register or label in the Text Editor, then selecting Set Code Slice Criteria from the context menu displayed. The user must then close the Text Editor and select Create Annotated Files form the Business Rule Extraction main menu option, as below.



The user is then prompted for which type of code slice to perform, using the following dialogue box.



Direction: [✓ Backwards] Forwards Both
Analysis Type: [✓ Data Flow] Control Flow Both

A detailed description of the differences in these options is explained in section 34 Understanding Data Flows and Control Flows.

Select the module DSAUSER from the Function Catalogue
Reset the source file using Business Rules Extraction | Reset Source File from the main menu bar (to make sure we have a clean source module as we have used this module before for examples in this guide).
Open the Text Editor
Go to line 452
Place the cursor over the data element NDATE
Right Click Mouse, from the context menu select the last one **Set Code Slice Criteria <NDATE>**
Close the Text Editor
Execute Create Annotated Files from Business Rules Extraction | Create Annotated files
Choose a backwards "both" slice then press Annotate Code – this will perform a backwards code slice and mark the identified items as Marked Block 1 in the colour specified.
Perform the same again but this time select a forwards "both" slice and annotate the code – this will perform a forwards code slice highlighting the identified code as Marked Block 2 in the colour specified.
(Note if the colours have not been specified – customising the text editor you will be prompted for the colour when you open the Text Editor)
Open the Text Editor
Click in the Text and Press Shift F8 to go to the identified code, Shift F8 will move to next blocks etc.
Try this on the other hits identified in modules RXLKPWUP and ZICH0001
Note code slicing is not performed on data definition lines such DS.

23.15 Documenting Business Rules

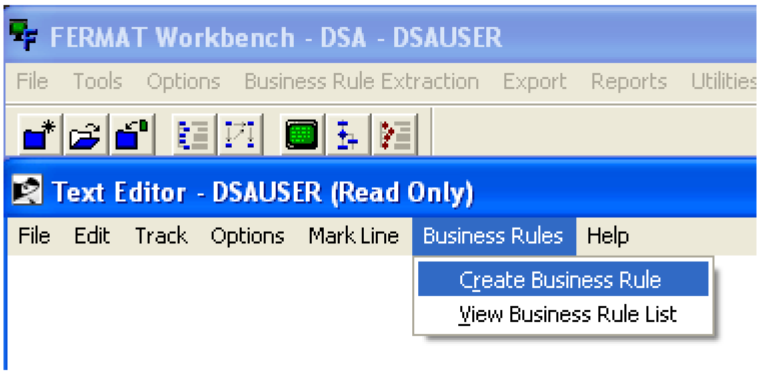
Using the above techniques to identify the lines of code that affect data elements, the programmer is now able to document this in business terms and store this in the Business Rule Repository within the Workbench. When generating the code-slices the Workbench records the Module, data-name and line number used to create the code-slice and the lines affected in a control file xref_bre.txt in the level 2 Dataflow directory. Data for a specific module is cleared from this file when a Reset Source is processed for the module.

This information is stored together with the programmer input describing the business rule in the Business Rule Repository, and is used to re-highlight (annotated) lines of code contributing to a business rule when required in the text editor.

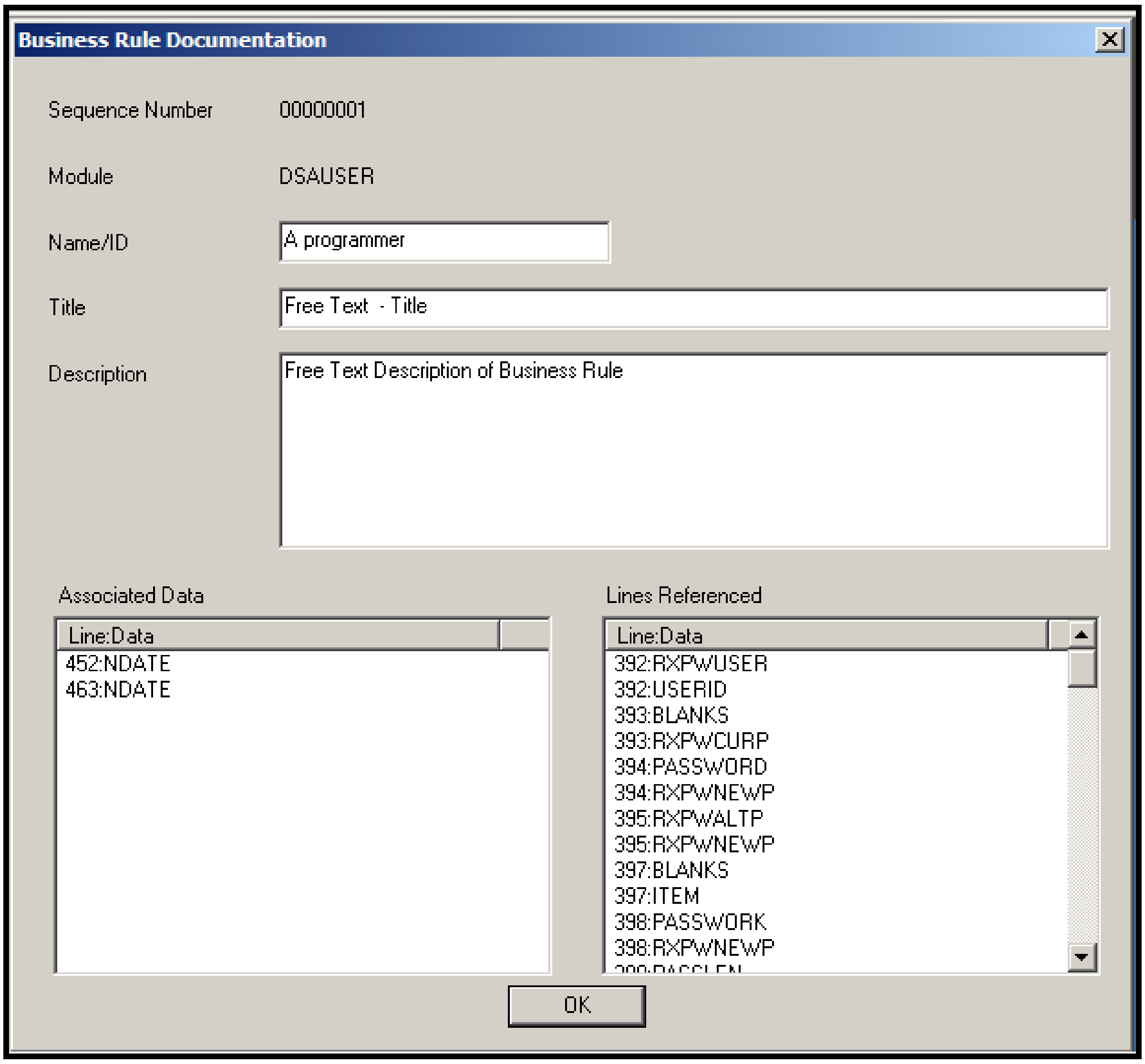
23.16 To Enter a Business Rule

To create a business rule first run the Batch BRE process as shown in section 23.10, the BRE assessment report will show whether the current module is affected by this. If the current module has no hits as a result of the Batch BRE process a warning will be displayed rather than the dialogue detailed below.

With the module open in the Text Editor select Business Rules from the menu bar, the following options appear in the drop down box.


FERMAT Workbench - DSA - DSAUSER
File Tools Options Business Rule Extraction Export Reports Utilities
Text Editor - DSAUSER (Read Only)
File Edit Track Options Mark Line Business Rules Help
Create Business Rule
View Business Rule List

Select **Create Business Rule** - the following screen will be displayed:



Business Rule Documentation

Sequence Number	00000001
Module	DSAUSER
Name/ID	A programmer
Title	Free Text - Title
Description	Free Text Description of Business Rule

Associated Data

Line:Data
452:NDATE
463:NDATE

Lines Referenced

Line:Data
392:RXPWUSER
392:USERID
393:BLANKS
393:RXPWCURP
394:PASSWORD
394:RXPWNEWP
395:RXPWALTP
395:RXPWNEWP
397:BLANKS
397:ITEM
398:PASSWORK
398:RXPWNEWP

The **Sequence number** and **module name** is auto populated by the Workbench; the Name/ID field is the user logon name but can be over written.

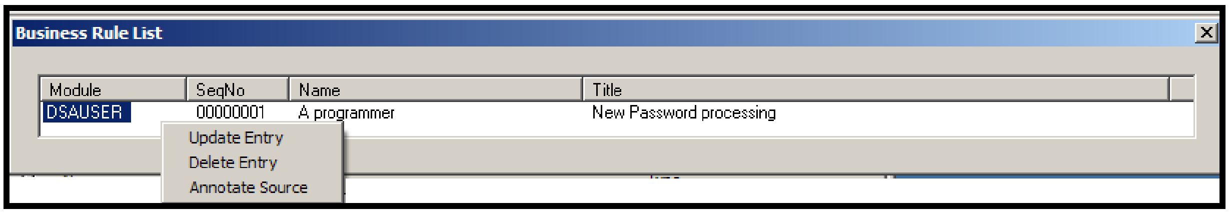
The **Title** and **Description** boxes are free text fields for the programmer to input a description of the high-lighted code in business terms. Pressing OK will file the information in the Business rule repository.

Associated Data details the line number and the data-name used to produce the code-slice.

Lines Referenced details the lines affected by the code-slice(s).

23.17 To View Existing Business Rules

In the Text Editor select Business Rules from the Menu Bar (as above), then select View Business Rule List, this will then show any business rules already stored for this Module, as below.



Module	SeqNo	Name	Title
DSAUSER	00000001	A programmer	New Password processing

Selecting the business rule required and right-click will produce the drop down list above.

- Update Entry: allows the user to view and/or update the stored business rule.
- Delete Entry: will delete the business rule from the repository.
- Annotate Source: will use the information stored in the business rule to highlight the lines of code in the source code that relate to the business rule in the Text Editor, the module must be reopened in the text editor to view these highlights.

To View all the business rules for a project select Business Rule Extraction from the Workbench Main Menu Bar, then select View Business Rule List. This will produce a screen similar to the above but for all stored business rules and provide the options of Update, Delete and Annotate Source.

24. Export Function

The Export function on the Main Workbench title bar is used to export the contents of the internal repository to other systems.

The **Function XML Files** will produce a file in XML format detailing the contents of the project and the relationships between the various entities. This file is placed in the Export Folder of the project directory.

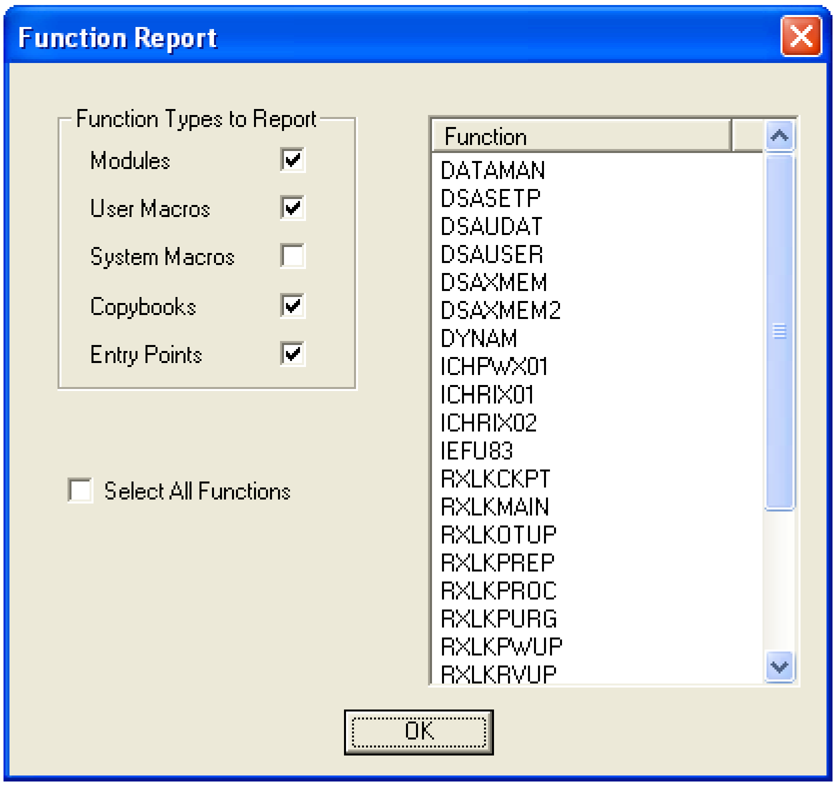
The **Business Rules files** will export the contents of the business rule repository in XML format detailing the Module name the title of the business rule, description and the reference information. This file is placed in the Export Folder of the project directory.

25. Reports

The Reports option on the main title of the Workbench allows for standard reports to be generated.

25.1 Function Catalogue Details Report

This report is used to provide a hard copy of the Function Catalogue details; it will produce the following dialogue Box.



Modules ☑
User Macros ☑
System Macros ☐
Copybooks ☑
Entry Points ☑

☐ Select All Functions

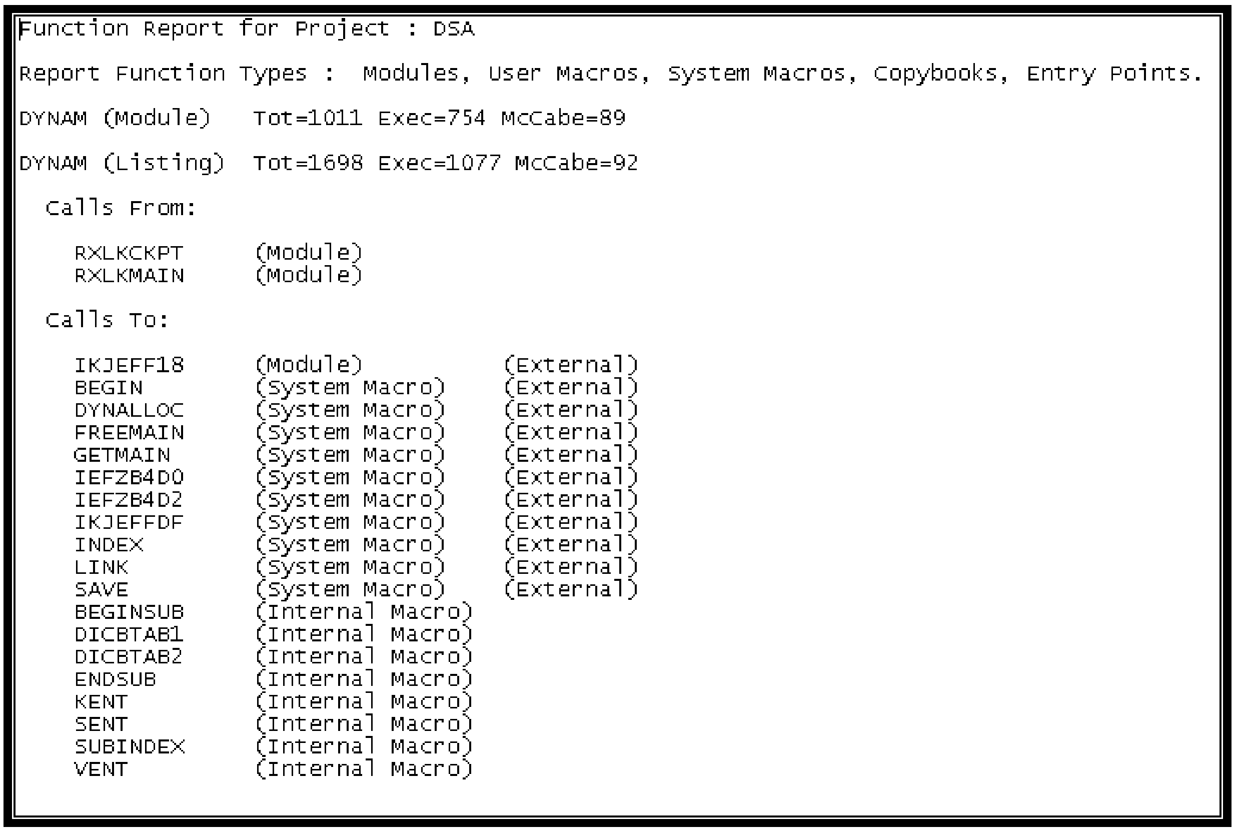
Function
DATAMAN
DSASETP
DSAUDAT
DSAUSER
DSAXMEM
DSAXMEM2
DYNAM
ICHPWX01
ICHRIX01
ICHRIX02
IEFU83
RXLKCKPT
RXLKMAIN
RXLKOTUP
RXLKPREP
RXLKPROC
RXLKPURG
RXLKPWUP
RXLKRVUP

The user can choose to what level of detail the report is to produce by selecting any of the Function Types.

The User can produce the report for all the modules in the project or just for selected modules.

Modules are selected using left Click or with Shift left Click to select a block or Control left click to select additional individual modules.

The report will generate a text file in the Level1\inventory\ called FunctionReport01.txt and will launch notepad (or the associated text file viewer) to display the report.



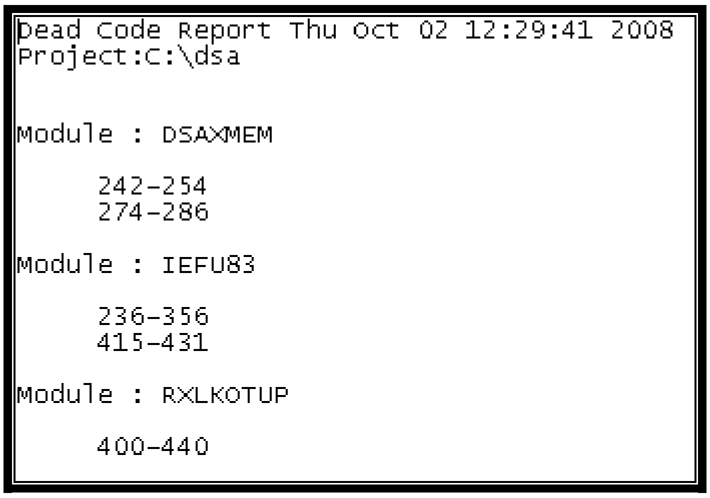
```
Function Report for Project : DSA

Report Function Types :  Modules, User Macros, System Macros, Copybooks, Entry Points.

DYNAM (Module)    Tot=1011 Exec=754 McCabe=89

DYNAM (Listing)  Tot=1698 Exec=1077 McCabe=92

  Calls From:

    RXLKCKPT     (Module)
    RXLKMAIN     (Module)

  Calls To:

    IKJEFF18     (Module)            (External)
    BEGIN        (System Macro)      (External)
    DYNALLOC     (System Macro)      (External)
    FREEMAIN     (System Macro)      (External)
    GETMAIN      (System Macro)      (External)
    IEFZB4D0     (System Macro)      (External)
    IEFZB4D2     (System Macro)      (External)
    IKJEFFDF     (System Macro)      (External)
    INDEX        (System Macro)      (External)
    LINK         (System Macro)      (External)
    SAVE         (System Macro)      (External)
    BEGINSUB     (Internal Macro)
    DICBTAB1     (Internal Macro)
    DICBTAB2     (Internal Macro)
    ENDSUB       (Internal Macro)
    KENT         (Internal Macro)
    SENT         (Internal Macro)
    SUBINDEX     (Internal Macro)
    VENT         (Internal Macro)
```

Example Function Catalogue Details Report.

25.2 Dead Code Details Report

This report will generate a text file stored in the Level1\inventory folder called DeadCodeReport.txt and launch notepad to display it.

The report details any dead code candidates found in the project following the Generate Dataflow process. The report details module name and the lines of code affected.



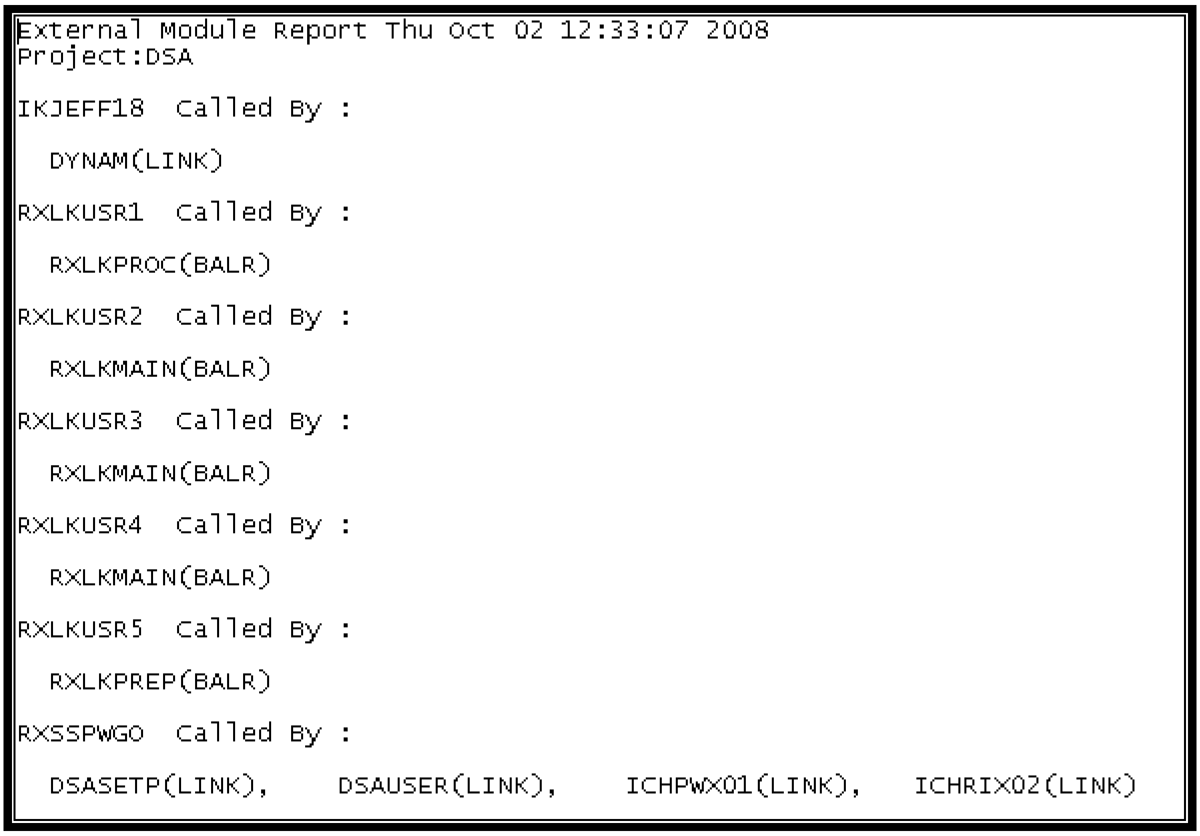
```
Dead Code Report Thu Oct 02 12:29:41 2008
Project:C:\dsa


Module : DSAXMEM

     242-254
     274-286

Module : IEFU83

     236-356
     415-431

Module : RXLKOTUP

     400-440
```

Example of Dead Code Analysis Report.

25.3 External Module Details Report

This report will generate a text file stored in the Level1\inventory folder called ExternalModuleReport03.txt and launch notepad to display it.

The report details any external modules called from the project. The report details the called module name and the calling module.



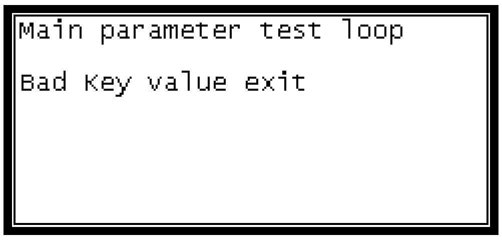
```
External Module Report Thu Oct 02 12:33:07 2008
Project:DSA

IKJEFF18  Called By :

  DYNAM(LINK)

RXLKUSR1  Called By :

  RXLKPROC(BALR)

RXLKUSR2  Called By :

  RXLKMAIN(BALR)

RXLKUSR3  Called By :

  RXLKMAIN(BALR)

RXLKUSR4  Called By :

  RXLKMAIN(BALR)

RXLKUSR5  Called By :

  RXLKPREP(BALR)

RXSSPWGO  Called By :

  DSASETP(LINK),    DSAUSER(LINK),    ICHPWX01(LINK),    ICHRIX02(LINK)
```

Example of External Module details Report.

25.4 Annotation Report

This report will generate a text file stored in the Level1\inventory folder called Annotation_*ModuleName*.txt and launch notepad to display it. If there are no annotations in the current module a warning is displayed.

The report details all the user annotation text entries for the current module.

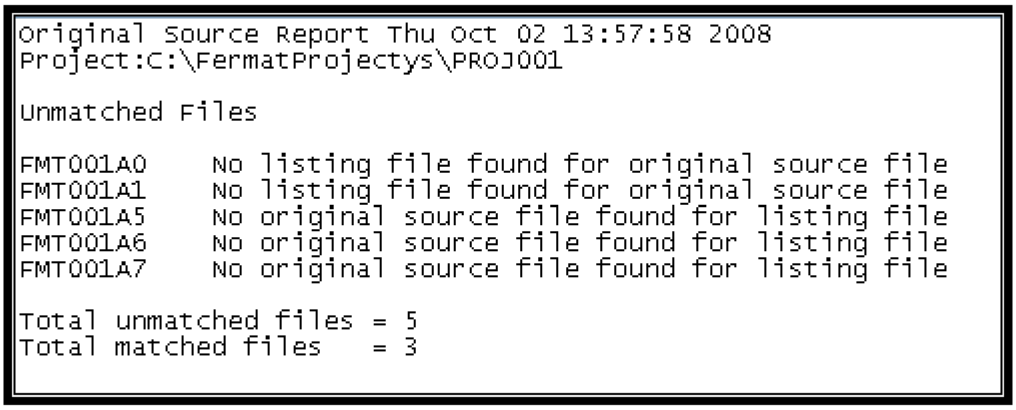


```
Main parameter test loop

Bad Key value exit
```

Example Annotation Report.

25.5 Original Source Report

This report will generate a text file stored in the Level1\inventory folder called OriginalSourceReport.txt and launch notepad to display it. The report details discrepancies between imported listings and their original source.

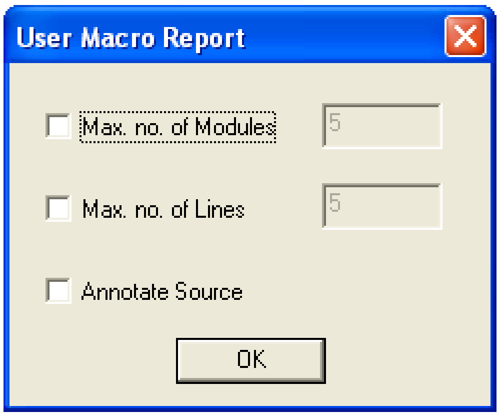


```
Original Source Report Thu Oct 02 13:57:58 2008
Project:C:\FermatProjectys\PROJ001

Unmatched Files

FMT001A0    No listing file found for original source file
FMT001A1    No listing file found for original source file
FMT001A5    No original source file found for listing file
FMT001A6    No original source file found for listing file
FMT001A7    No original source file found for listing file

Total unmatched files = 5
Total matched files   = 3
```

Example Original Source Report.

25.6 User Macro Report

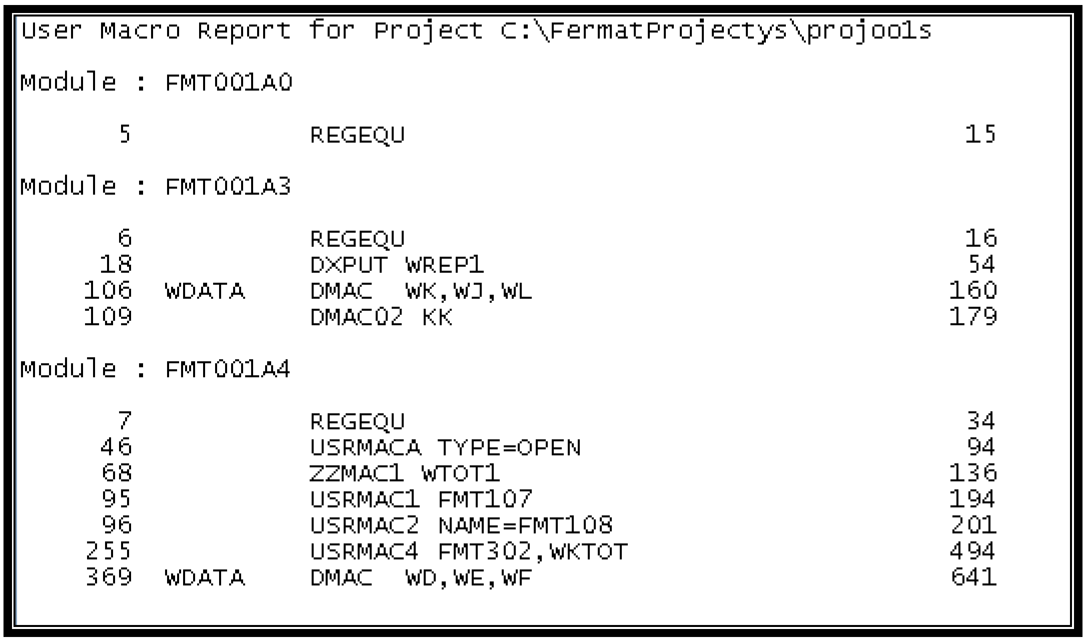
Selecting this option will display the dialogue shown below:



This dialogue allows the following options to clarify the output of the report:

Option	**Description**
Max Modules check box	Ticking this enables the filter for number of modules. A user macro will only be reported as invoked, from the first n modules
Max Modules	Defines n above.
Max Lines check box	Ticking this enables the filter for number of invocations of a macro. A user macro will only be reported as invoked n times in a given module.
Max Lines	Defines n above.
Annotate Source	Ticking this adds a highlight in the text editor for each user macro invocation.

Pressing OK on this dialogue will generate a text file stored in the Level1\inventory folder called UserMacroReport.txt and launch notepad to display it. The report shows user macros invoked from each module in the project and details module name and the source line where it is invoked.

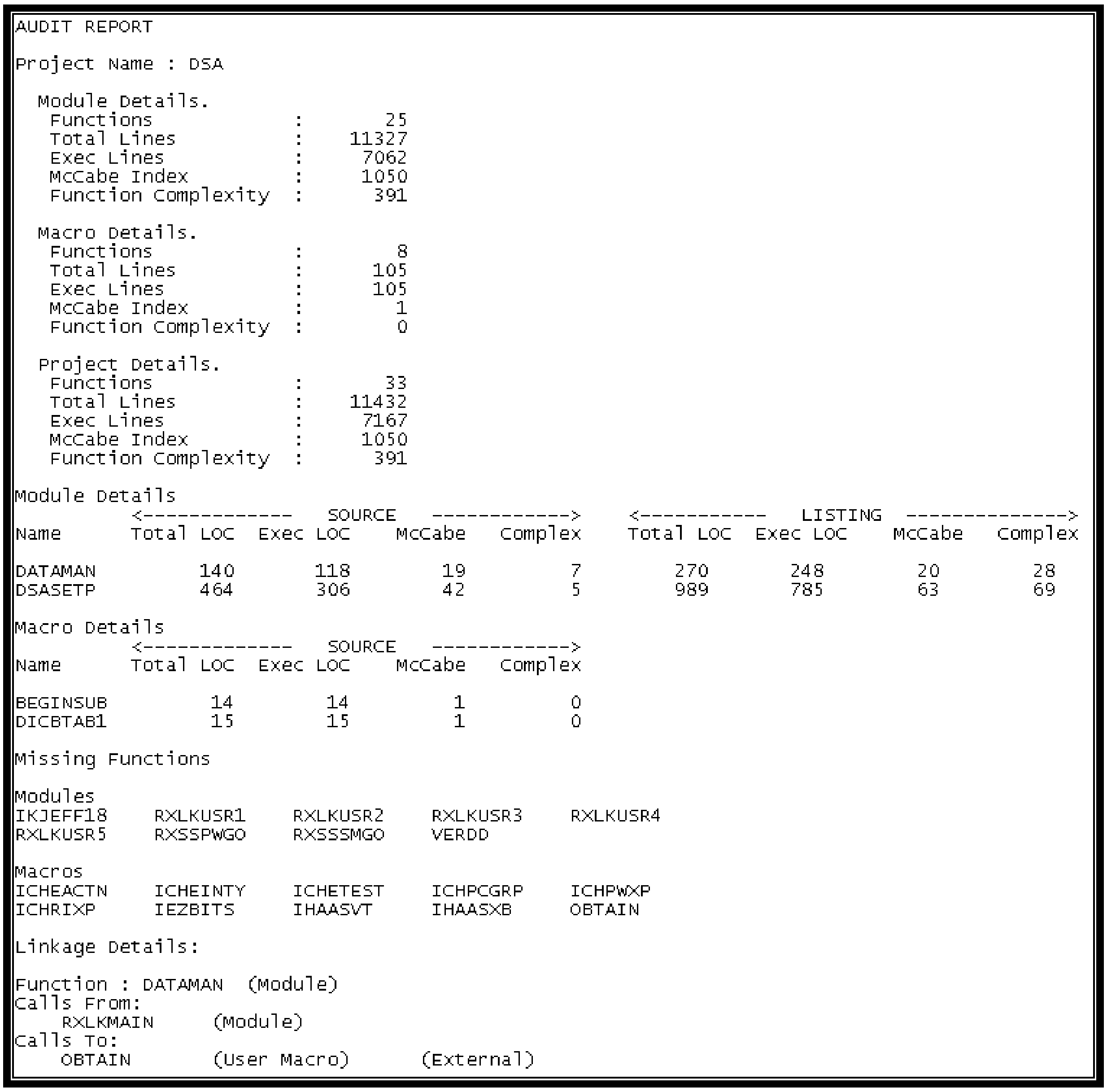


```
User Macro Report for Project C:\FermatProjectys\projoo1s

Module : FMT001A0

      5           REGEQU                                   15

Module : FMT001A3

      6           REGEQU                                   16
     18           DXPUT  WREP1                             54
    106  WDATA    DMAC   WK,WJ,WL                         160
    109           DMAC02 KK                               179

Module : FMT001A4

      7           REGEQU                                   34
     46           USRMACA TYPE=OPEN                        94
     68           ZZMAC1  WTOT1                           136
     95           USRMAC1 FMT107                          194
     96           USRMAC2 NAME=FMT108                     201
    255           USRMAC4 FMT302,WKTOT                    494
    369  WDATA    DMAC   WD,WE,WF                         641
```

Example User Macro Report.

25.7 Audit Report

This report will generate a text file stored in the Level1\inventory folder called AuditReport.txt and launch notepad to display it, it will also generate a csv file with the same information called AuditReport.csv that can be opened with Excel. The report provides an overview of the project. It details lines of code and complexity for each module and totalled for the project along with a list of missing functions and macros and detailed linkage information.

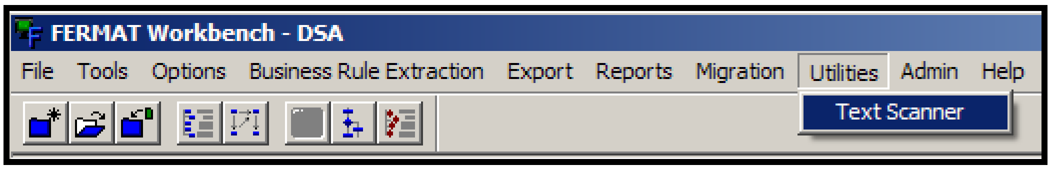


```
AUDIT REPORT

Project Name : DSA

  Module Details.
   Functions             :       25
   Total Lines           :    11327
   Exec Lines            :     7062
   McCabe Index          :     1050
   Function Complexity   :      391

  Macro Details.
   Functions             :        8
   Total Lines           :      105
   Exec Lines            :      105
   McCabe Index          :        1
   Function Complexity   :        0

  Project Details.
   Functions             :       33
   Total Lines           :    11432
   Exec Lines            :     7167
   McCabe Index          :     1050
   Function Complexity   :      391

Module Details
          <-------------   SOURCE   ------------>    <-----------   LISTING   -------------->
Name      Total LOC  Exec LOC    McCabe   Complex    Total LOC  Exec LOC    McCabe   Complex

DATAMAN         140       118        19         7          270       248        20        28
DSASETP         464       306        42         5          989       785        63        69

Macro Details
          <-------------   SOURCE   ------------>
Name      Total LOC  Exec LOC    McCabe   Complex

BEGINSUB         14        14         1         0
DICBTAB1         15        15         1         0

Missing Functions

Modules
IKJEFF18    RXLKUSR1    RXLKUSR2    RXLKUSR3    RXLKUSR4
RXLKUSR5    RXSSPWGO    RXSSSMGO    VERDD

Macros
ICHEACTN    ICHEINTY    ICHETEST    ICHPCGRP    ICHPWXP
ICHRIXP     IEZBITS     IHAASVT     IHAASXB     OBTAIN

Linkage Details:

Function : DATAMAN  (Module)
Calls From:
    RXLKMAIN     (Module)
Calls To:
    OBTAIN       (User Macro)      (External)
```

Example Audit Report.

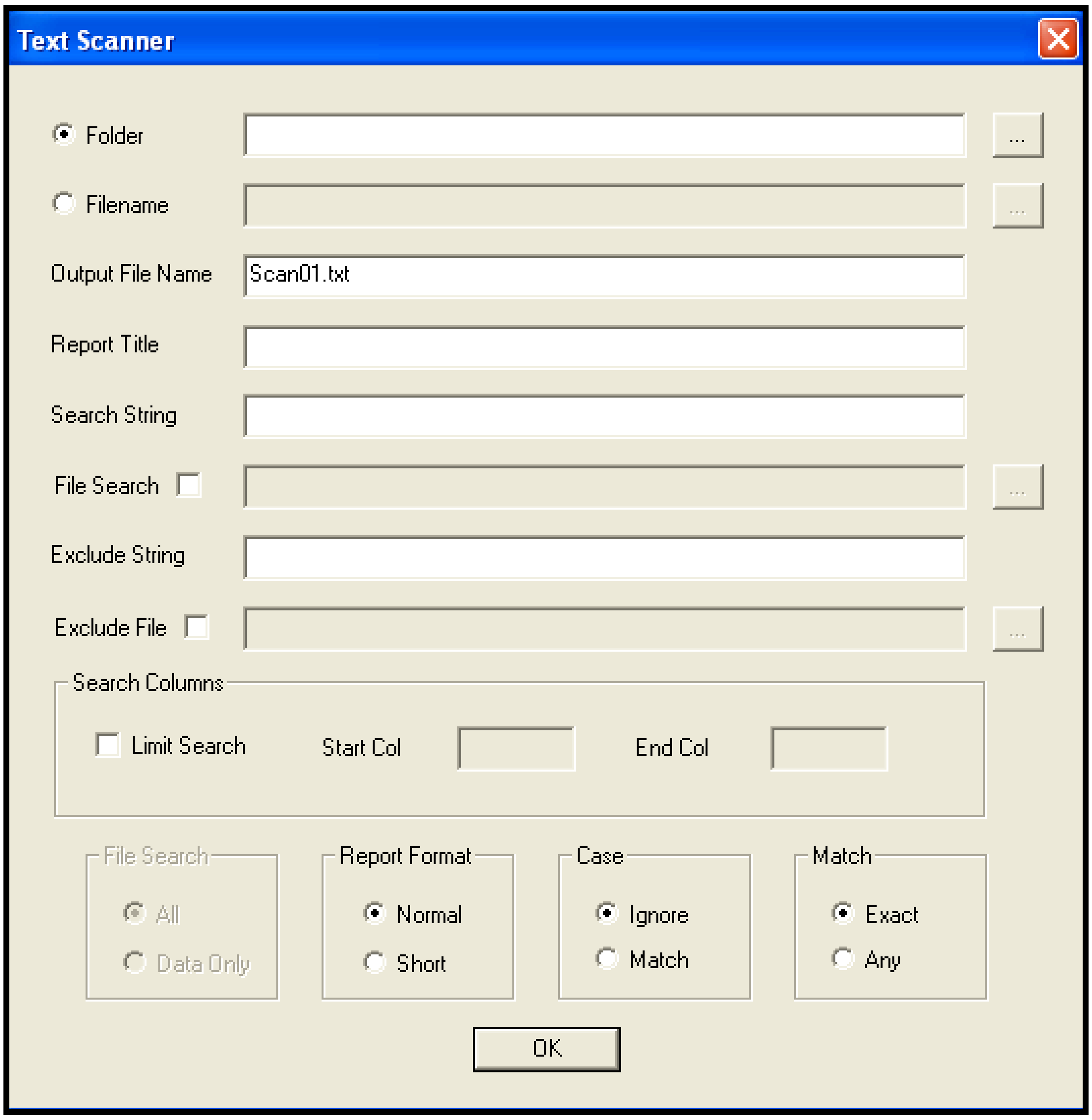
26. Utilities – Text Scanner

The Text Scanner is selected by selecting Utilities fro the main Workbench Menu bar, then selecting Text Scanner, note process is only available when a project is loaded.



The Text Scanner is a “free text” search facility that can be used to search any file on the system for one or more search strings and produce a report of the matched items which is written to the Level1\Inventory folder of the current open project.

Selecting Text Scanner will display the TextScan dialogue box; as below



Text Scanner

(● Folder) Folder:
Filename:
Output File Name: Scan01.txt
Report Title:
Search String:
File Search ☐
Exclude String:
Exclude File ☐

Search Columns
☐ Limit Search — Start Col: — End Col:

File Search: [✓ All] Data Only
Report Format: [✓ Normal] Short
Case: [✓ Ignore] Match
Match: [✓ Exact] Any

Values from any previous use of the Text Scanner will be displayed.

The user can either select a folder or a single file to search, select the appropriate radio button, then either enter to location of the file or select the browse button to navigate to the file or folder.

The user must specify the name of the output file (default is Scan01.txt), the output file will be written to *project name*/Level1/Inventory Folder, and will overwrite any existing report with the same name. The text scanner will launch this file with Notepad after the search if any matches are detected.

The Report Title is optional, if entered it will be written once at the beginning of the output .txt file.

User can then enter a search string or by selecting the File Search check box or enter the name of the file containing multiple search criteria or navigate to the file using the browse button.

The user can specifically limit the search to certain columns for each line search by entering the start/end columns to search within.

If the user has specified a filename to search opposed to a folder, the Search File Options All or Data Only are available. If the file being searched is the output from a previous search, the user can select Data Only to only search the original data, i.e. ignoring the file name and line number information. Note when this is selected the Search Columns information if entered will apply to the original data. If All is chosen the entire line contents will be search and the columns information would apply to the whole line.

The Exclude String and Exclude File are similar to the Search String and Search File as above; the Exclude is used to exclude specific matches from the results. The scan process will first perform the search operations to create the matched items file, the exclude operation then scans this file and removes any items that meet the exclude criteria.

Report Format either Normal or Short (see Report Examples later in this section)

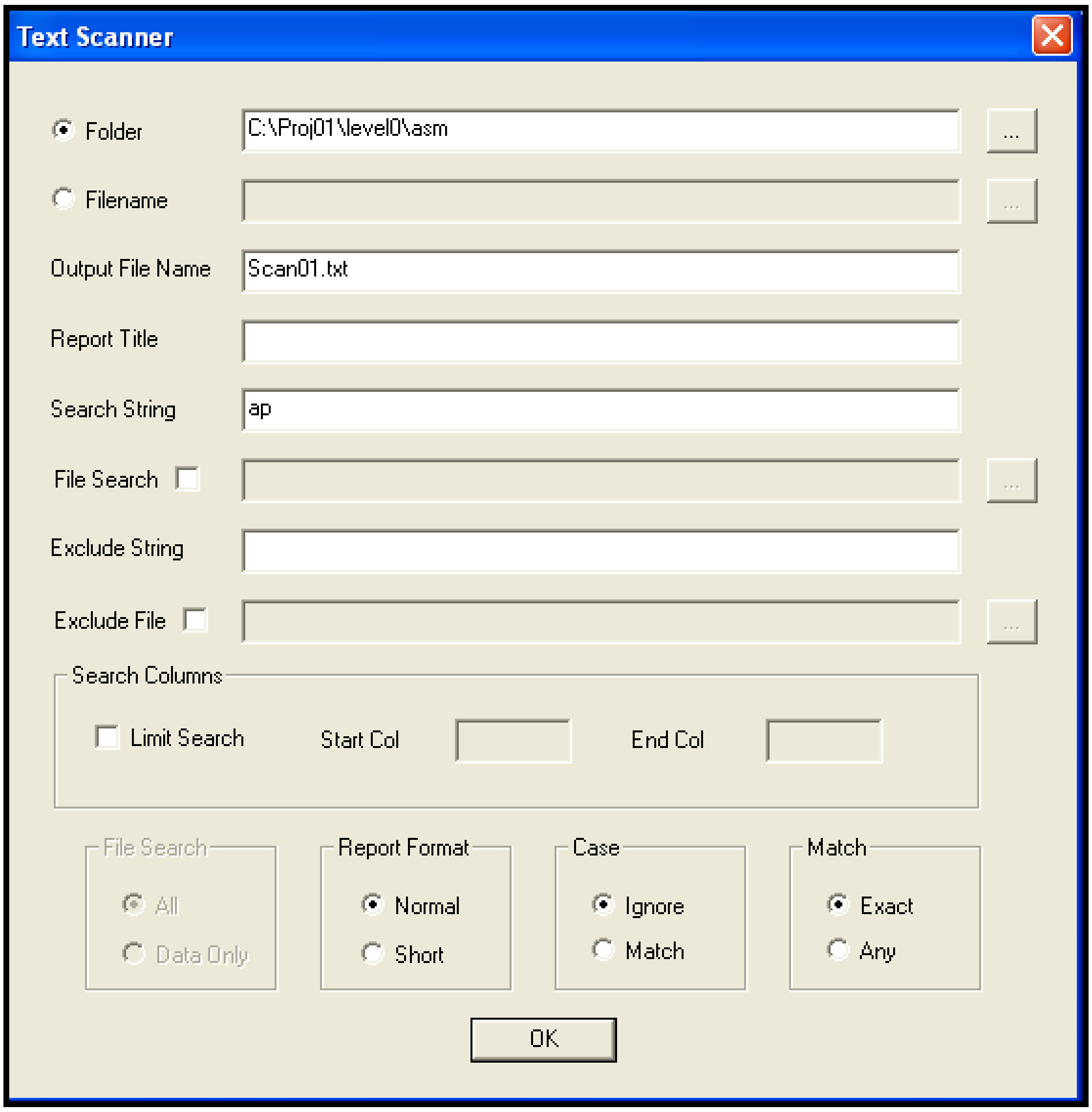
The Short Format should be used when the user may want to perform subsequent searches on the output file to refine the results or to create a subproject from, so that the original location of the data can be identified i.e. Module name and line number.

The case options are either match the case of the string or ignore the case for purposes of matching.

The Match options Exact or Any. For an exact match, the characters before and after the matched string must not be alphanumeric (i.e. a-z, A-Z or 0-9).

Dialogue Examples

Search Folder



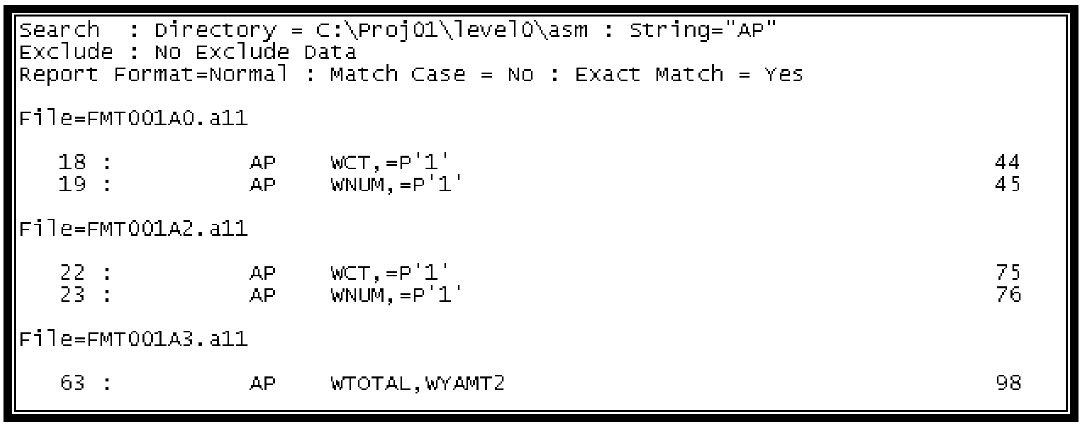
Text Scanner

(● Folder) C:\Proj01\level0\asm
( ) Filename
Output File Name: Scan01.txt
Report Title:
Search String: ap
File Search ☐
Exclude String:
Exclude File ☐

Search Columns
☐ Limit Search   Start Col   End Col

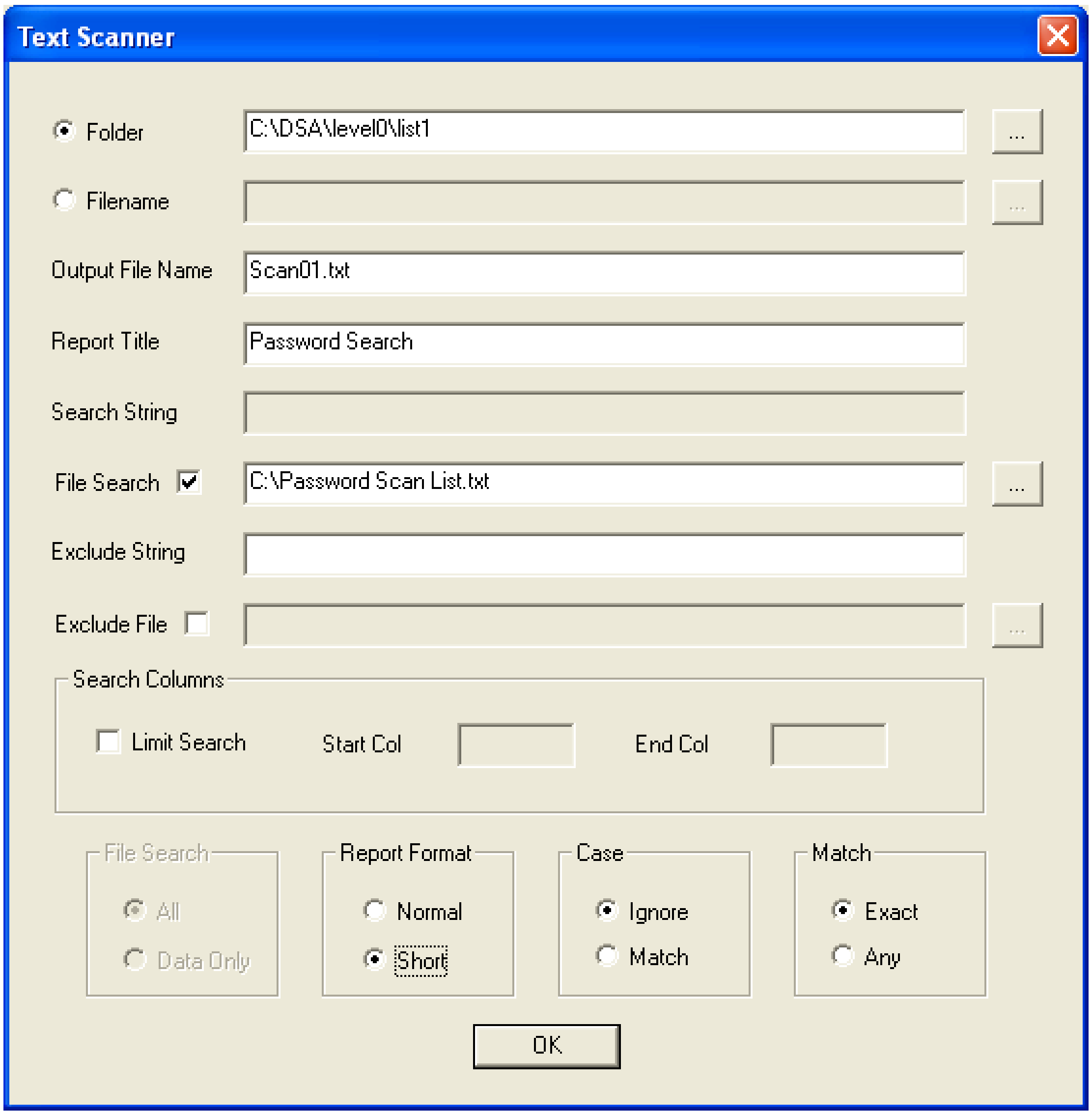
File Search: [✓ All] Data Only
Report Format: [✓ Normal] Short
Case: [✓ Ignore] Match
Match: [✓ Exact] Any

The above example will search all files in the folder C:\Proj01\Level0\asm for the string “ap” and put the normal formatted output file in C:\Proj01\level1\Inventory\Scan01.txt with no report title.



```
Search   : Directory = C:\Proj01\level0\asm : String="AP"
Exclude  : No Exclude Data
Report Format=Normal  : Match Case = No  : Exact Match = Yes

File=FMT001A0.a11

   18  :          AP     WCT,=P'1'                                       44
   19  :          AP     WNUM,=P'1'                                      45

File=FMT001A2.a11

   22  :          AP     WCT,=P'1'                                       75
   23  :          AP     WNUM,=P'1'                                      76

File=FMT001A3.a11

   63  :          AP     WTOTAL,WYAMT2                                   98
```

Search File; Multiple Search Strings in File


Text Scanner

[✓ Folder] C:\DSA\level0\list1

Filename

Output File Name: Scan01.txt

Report Title: Password Search

Search String

File Search ✓ C:\Password Scan List.txt

Exclude String

Exclude File ☐

Search Columns: ☐ Limit Search, Start Col, End Col

File Search: [✓ All] Data Only

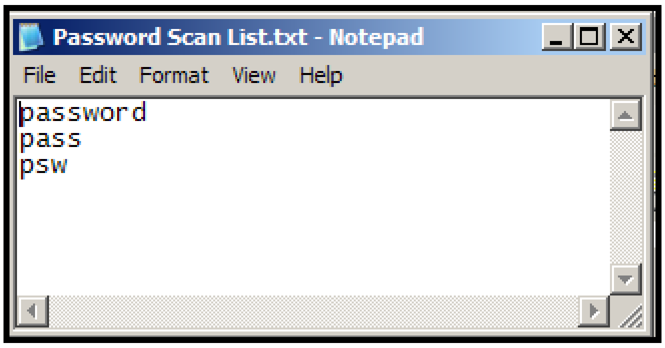
Report Format: Normal [✓ Short]

Case: [✓ Ignore] Match

Match: [✓ Exact] Any

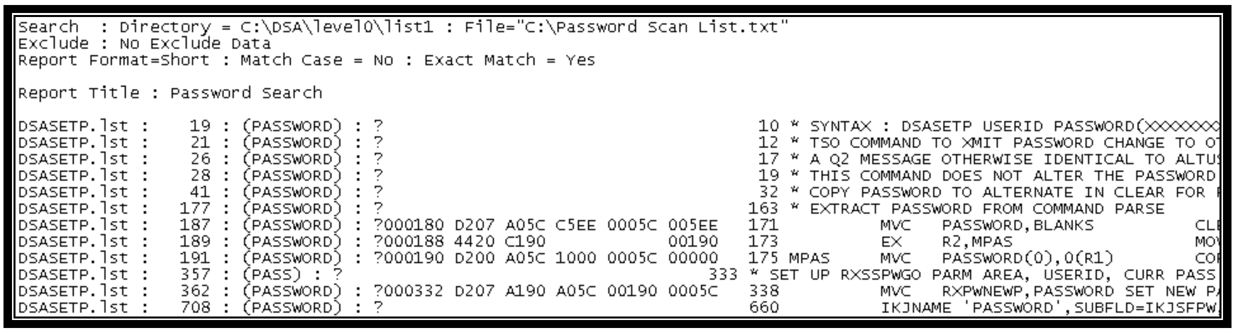
This scan will search all the files in the folder C;\DSA\level0\list1 for strings that match those in the file C:\Password Scan List.txt. The report will have the title “Password Search” and will be in the short form.

The Password Scan List.txt contains the followings search strings.



```
password
pass
psw
```

And will search the files for exact matches of “password”, “pass” or “psw” ignoring the case.



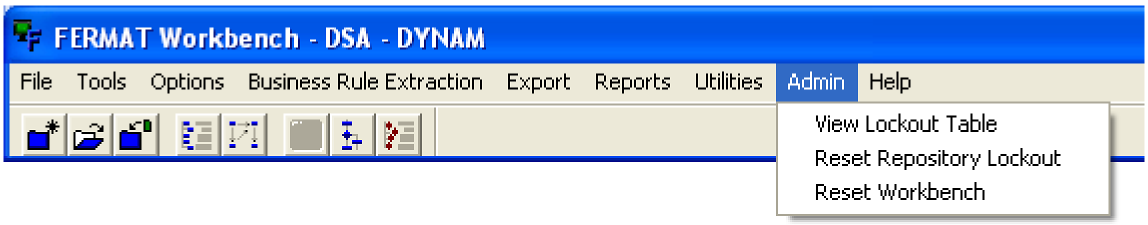
```
Search  : Directory = C:\DSA\level0\list1 : File="C:\Password Scan List.txt"
Exclude : No Exclude Data
Report Format=Short : Match Case = No : Exact Match = Yes

Report Title : Password Search

DSASETP.lst :    19 : (PASSWORD) : ?                                     10 * SYNTAX : DSASETP USERID PASSWORD(XXXXXXX
DSASETP.lst :    21 : (PASSWORD) : ?                                     12 * TSO COMMAND TO XMIT PASSWORD CHANGE TO O
DSASETP.lst :    26 : (PASSWORD) : ?                                     17 * A Q2 MESSAGE OTHERWISE IDENTICAL TO ALTUS
DSASETP.lst :    28 : (PASSWORD) : ?                                     19 * THIS COMMAND DOES NOT ALTER THE PASSWORD
DSASETP.lst :    41 : (PASSWORD) : ?                                     32 * COPY PASSWORD TO ALTERNATE IN CLEAR FOR
DSASETP.lst :   177 : (PASSWORD) : ?                                    163 * EXTRACT PASSWORD FROM COMMAND PARSE
DSASETP.lst :   187 : (PASSWORD) : ?000180 D207 A05C C5EE 0005C 005EE   171          MVC   PASSWORD,BLANKS          CL
DSASETP.lst :   189 : (PASSWORD) : ?000188 4420 C190            00190   173          EX    R2,MPAS                  MO
DSASETP.lst :   191 : (PASSWORD) : ?000190 D200 A05C 1000 0005C 00000   175 MPAS     MVC   PASSWORD(0),0(R1)        CO
DSASETP.lst :   357 : (PASS) : ?                                    333 * SET UP RXSSPWGO PARM AREA, USERID, CURR PASS
DSASETP.lst :   362 : (PASSWORD) : ?000332 D207 A190 A05C 00190 0005C   338          MVC   RXPWNEWP,PASSWORD SET NEW P
DSASETP.lst :   708 : (PASSWORD) : ?                                    660          IKJNAME 'PASSWORD',SUBFLD=IKJSFPW,
```

Note the title “Password Search” at the beginning of the report. On each matched line in the report it gives the file name and the line number and what string it was matched against.

Software Migrations Ltd

27. Admin Utilities

The Admin Utilities are invoked by selecting Admin Option from the main Workbench menu bar; as below



If the Workbench Lock Table for the current project is empty i.e. no modules are locked by the Text Editor for update, then the View Lockout Table will be greyed out.

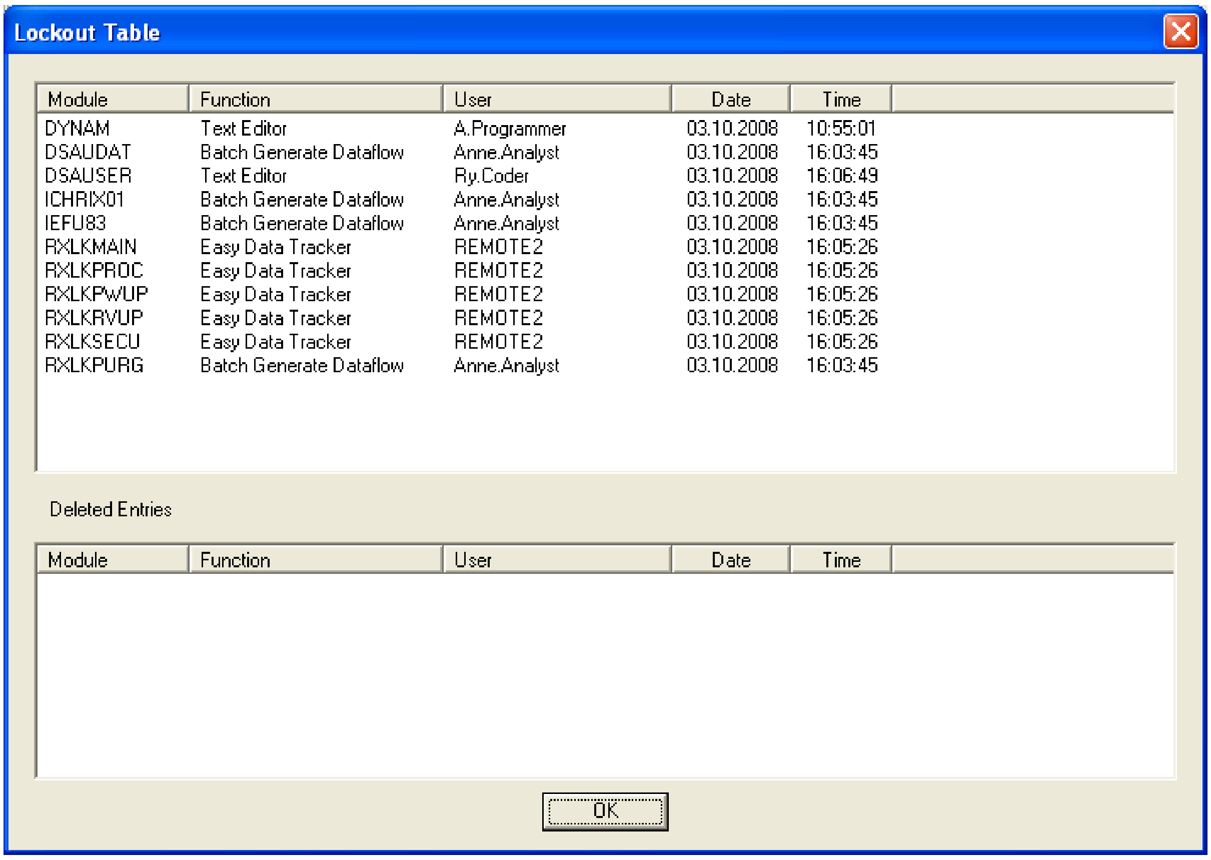
The Projects lock table is used to ensure that no more than one user can check out a module for updating at the same time. A user would check out a module for updating if he wishes to create annotations or mark lines or blocks of code and save them in the repository.

The lock table is checked and set by other processes within the Workbench so that these processes can complete and will not be affected by the locked out modules.

- Import Listings or Modules
- Export Files
- Creating Sub projects
- Add Functions
- Delete Functions
- Data Scanner from Batch Data Tracker
- Global Scan from the Batch Data Tracker
- Easy Data Tracker
- Code Slicing
- Generate Data Flow
- Auto Gen Business rules
- Reset Source Files
- Batch BRE
- Create business Rule

Within these processes the modules to be processed are checked against the lock table, if they are present in the Lock Table a message is displayed and the process halted. If not present all the modules to be processed will be placed in the lock to prevent others checking out the module in update mode until the process has finished, at which time the locks will be deleted.

Selecting View Lockout Table if available will present the user with the following dialogue box.



Module	Function	User	Date	Time
DYNAM	Text Editor	A.Programmer	03.10.2008	10:55:01
DSAUDAT	Batch Generate Dataflow	Anne.Analyst	03.10.2008	16:03:45
DSAUSER	Text Editor	Ry.Coder	03.10.2008	16:06:49
ICHRIX01	Batch Generate Dataflow	Anne.Analyst	03.10.2008	16:03:45
IEFU83	Batch Generate Dataflow	Anne.Analyst	03.10.2008	16:03:45
RXLKMAIN	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKPROC	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKPWUP	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKRVUP	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKSECU	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKPURG	Batch Generate Dataflow	Anne.Analyst	03.10.2008	16:03:45

Deleted Entries

Module	Function	User	Date	Time

The Table displays the contents of the project lock table, providing details of the module name, the user id who has locked the locked the module, it also displays the date and time the module was locked out..

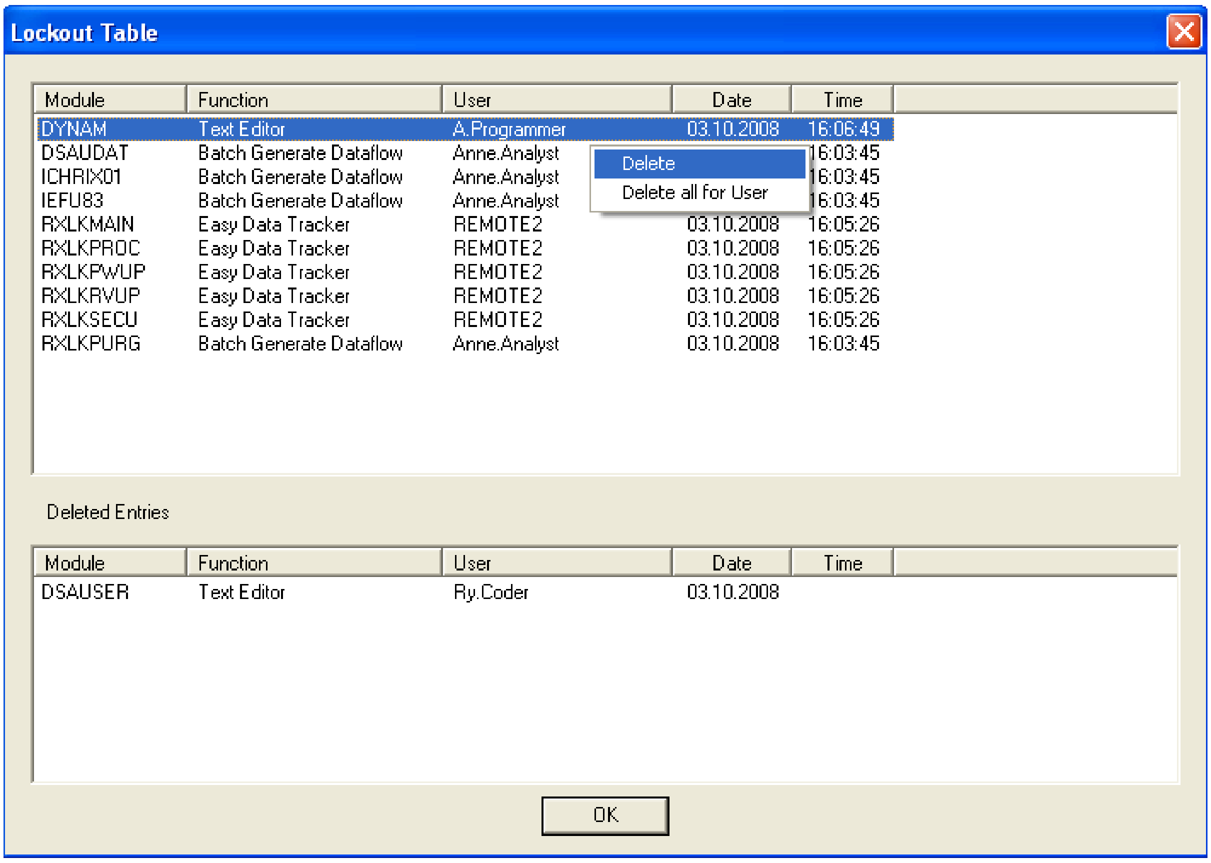
As a normal user no other functionality is available within the Admin menu, if selected a warning message will be displayed and no processing will occur.

If you are assigned administrator rights, which is based on your logon id, the following functions are available:

- Removing individual records from the lock table
- Removing all records for a user
- Resetting the entire lock table for a project
- Finally resetting the Workbench.

Please note care should be exercised in the use of the administration functions. These facilities are provided to ensure continued operation in the event of a user abnormally terminating the Workbench or a Workbench process, or a server failure.

In the View Lockout Table, the administrator can select records to be removed from the project lock table, by right clicking on the module line and selecting delete as below.



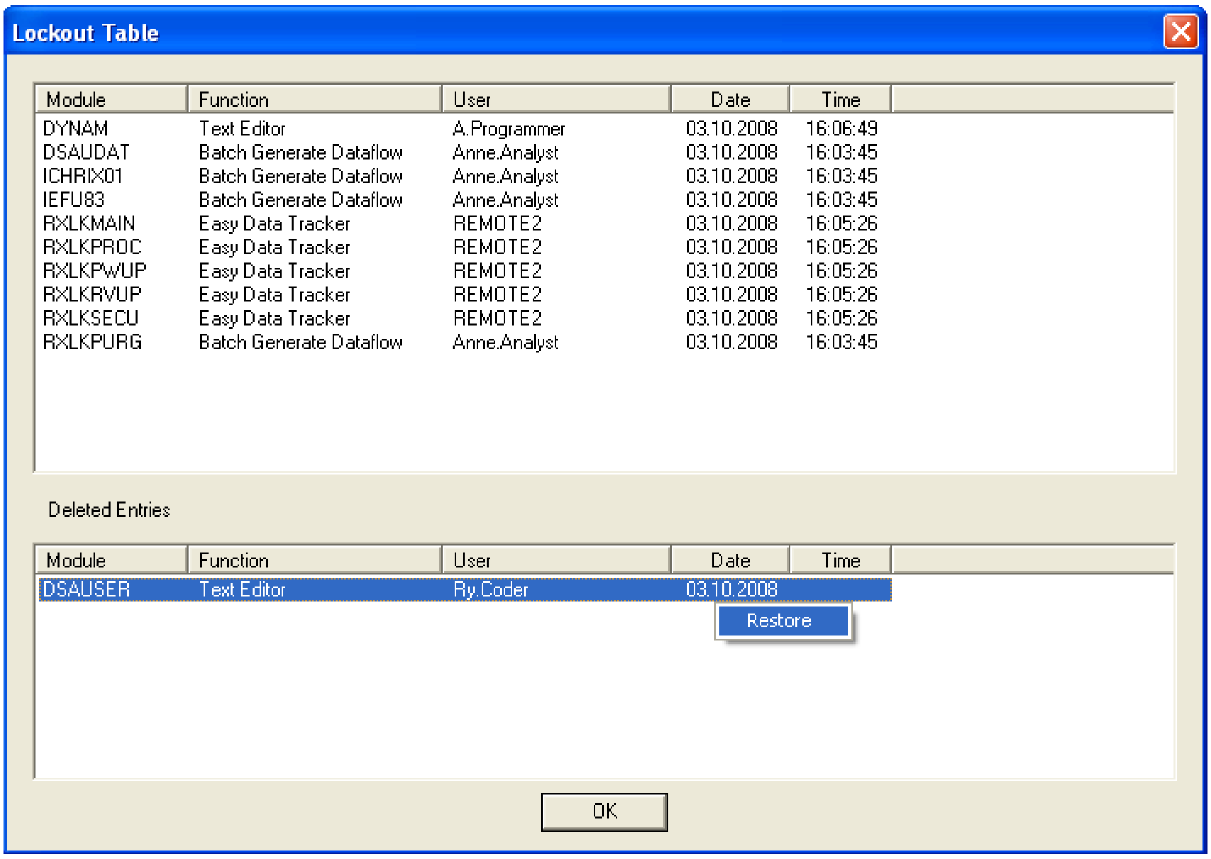
Module	Function	User	Date	Time
DYNAM	Text Editor	A.Programmer	03.10.2008	16:06:49
DSAUDAT	Batch Generate Dataflow	Anne.Analyst		16:03:45
ICHRIX01	Batch Generate Dataflow	Anne.Analyst		16:03:45
IEFU83	Batch Generate Dataflow	Anne.Analyst		16:03:45
RXLKMAIN	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKPROC	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKPWUP	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKRVUP	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKSECU	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKPURG	Batch Generate Dataflow	Anne.Analyst	03.10.2008	16:03:45

Module	Function	User	Date	Time
DSAUSER	Text Editor	Ry.Coder	03.10.2008	

In the above example the Module DATAMAN has previously been selected for deletion and placed in the lower panel. The module DYNAM has also been selected and will be placed in the panel deleted entries.

Selecting "Delete all for User" will place every locked module for the user on the highlighted line in the Deleted Entries panel.

Entries can be removed from the list of entries to delete by right clicking on the module name and selecting restore, as below.



Module	Function	User	Date	Time
DYNAM	Text Editor	A.Programmer	03.10.2008	16:06:49
DSAUDAT	Batch Generate Dataflow	Anne.Analyst	03.10.2008	16:03:45
ICHRIX01	Batch Generate Dataflow	Anne.Analyst	03.10.2008	16:03:45
IEFU83	Batch Generate Dataflow	Anne.Analyst	03.10.2008	16:03:45
RXLKMAIN	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKPROC	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKPWUP	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKRVUP	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKSECU	Easy Data Tracker	REMOTE2	03.10.2008	16:05:26
RXLKPURG	Batch Generate Dataflow	Anne.Analyst	03.10.2008	16:03:45

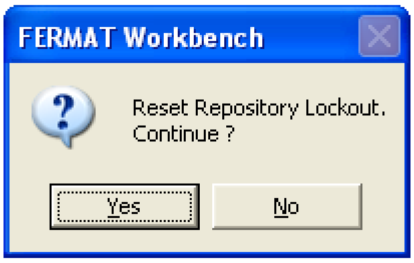
Deleted Entries

Module	Function	User	Date	Time
DSAUSER	Text Editor	Ry.Coder	03.10.2008	

In the above example the module DATAMAN will be removed from the Deleted Entries panel and restored to the lock table.

When the administrator has completed the list of lock table entries to be removed pressing the OK button will execute the command.

The Administrator can reset all the locks for a project by selecting the Reset Repository Lockout, which will display a confirmation dialogue box as below the administrator must select the YES button to proceed with the process.

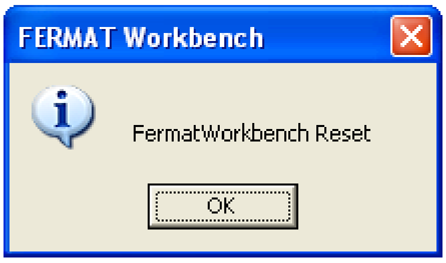


Reset Repository Lockout.
Continue ?

The Reset Workbench option is available to reset the Workbench's temporary control files, in the event of unplanned or abnormal termination of the Workbench, which would leave these temporary in an unknown state.

The symptoms encountered where it would be necessary to reset the workbench, is where no users can start a new function in the Workbench, the Workbench appears to hang. Resetting Workbench will reset the temporary control files and users can continue as before.

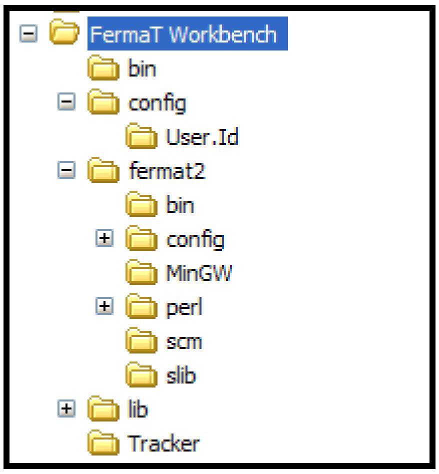
When executed by the administrator the Reset Workbench will display the following confirmation dialogue box.



FermatWorkbench Reset

28. File Structure Workbench Software

This section describes the files and their functions included in the Workbench Software, this detail is for information only, as a user of the Workbench you should not amend or change any of the files:



FermaT Workbench - is the directory where the Workbench Software was installed.

bin - is the folder where all the workbench files for execution are contained.

config – is the folder which will contain subfolders for each user of the workbench e.g. User.Id. The **User.Id** - folder contains transient information on the status of the workbench such as size and position information of the Workbench tools, so that these are reapplied automatically when each tool is started. The User.Id folder also maintains the save user settings for the Text Editor.

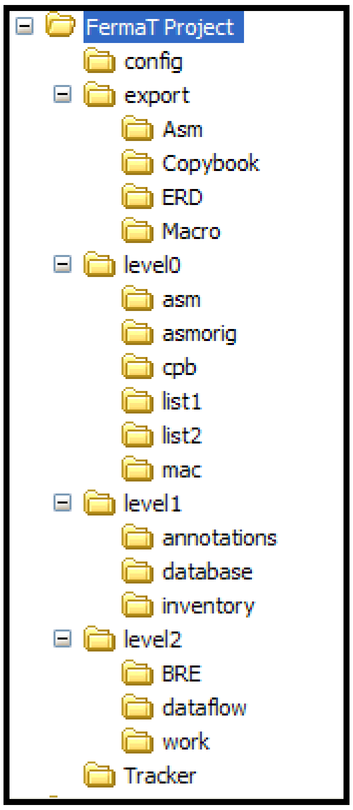
fermat2 – This top level folder contains the software required to run the detailed analysis of the assembler applications it contains the binary executables and perl scripts used by the system including the perl and scheme runtime libraries.

lib – contains the TCL executable programs for the Workbench GUI interface.

Tracker – contains the default values for the Access interface.

29. File Structure Workbench Project

A Fermat Project is a logical container on the system where the Assembler system is imported to. When a new project is created the Workbench creates the following files and automatically populates them with the imported data. The files are then used to store information as the Workbench is executed. The following is a description of the files and their uses within the Workbench.



Fermat Project – This top level is the name of the project e.g. Billing System
config – This folder contains the customisable files used by the workbench to meet customer specific requirements such as macro usage and linkage parameter see Section 26 for further details on how to customise the Workbench.
export – when the export files option is executed the workbench will copy the assembler source to the Asm folder, copybooks to the Copybook folder, macros to the Macro folder and places an XML file in the ERD folder which describes the relationships between the functions in the project.
level0 – is a top level folder which contains the in code that has been imported to the systems, the following details the subfolders:
asm – contains the extracted source (if import was from listing) or the original source imported.
asmorig – contains the original source imported when the import included both listings and source to support full life cycle see Section 27 for further details.
cpb – contains extracted copybooks (if import was from listing) or the original copybook source imported.

list1 – contains the original listing imported to the system

list2 – contains a working copy of the listing imported to the system. It is used to mark lines of code during the code slicing, BRE and Business Rules processes. The reset source function copies form list1 to list2.

mac – contains the extracted macros (if import was from listing) or the original macro source imported.

level1 – is a top level folder containing information generated by the workbench, the following details the subfolders:

annotations – this file contains the detailed information for any user annotations created by the user for this project database – this folder contains a number of files which comprise the main repository of the workbench. It also holds information on the statistics and metrics generated by the workbench as part of the import process.

Database – contains the files comprising the workbench repository.

Inventory – contains intermediate files for generating the inventory reports together log and error files generated by the import process.

UserName – a folder is created for each user to contain the intermediate and temporary files for controlling such things as flowchart highlighting and function and data catalogue details.

Level2 – is a top level folder containing information generated by the fermat2 function of the workbench, the following details the subfolders:

BRE – the folder contains details of the business rules identified and stored by users in the workbench.

dataflow – this file contains the dataflow files created by the generate dataflow functions and is used to identify code slices and auto-generate business rules.

work – is a debug file for any modules which fail to transform successfully using the generate dataflow process, and is used by SML to isolate problems.

Tracker – contains information used and generated by the Batch and Easy Data Tracker process.

30. Customising the Workbench

The Fermat Workbench uses several tables to improve the quality of the information produced from the scan process. Some of these tables can be customised by the user. If these tables are not customised, then the information displayed by the tools will not be complete.

For example.

If the code uses a macro (USRMAC1) to call external modules, then if the tables are not customised, the system will not record the name of the external macro, so the Function Call Graph will not show the link between the two modules. If USRMAC1 is added to the User Linkage table, then the scan process will extract the names of the external modules and show them on the various tools.

If the code uses a macro (ROUTINE) to define a sub-routine header, if the tables are not customised, the system will not be able to identify any sub-routines. If ROUTINE is added to the Routine table, then the scan process will identify all sub-routines and they will be shown on the Function Catalogue.

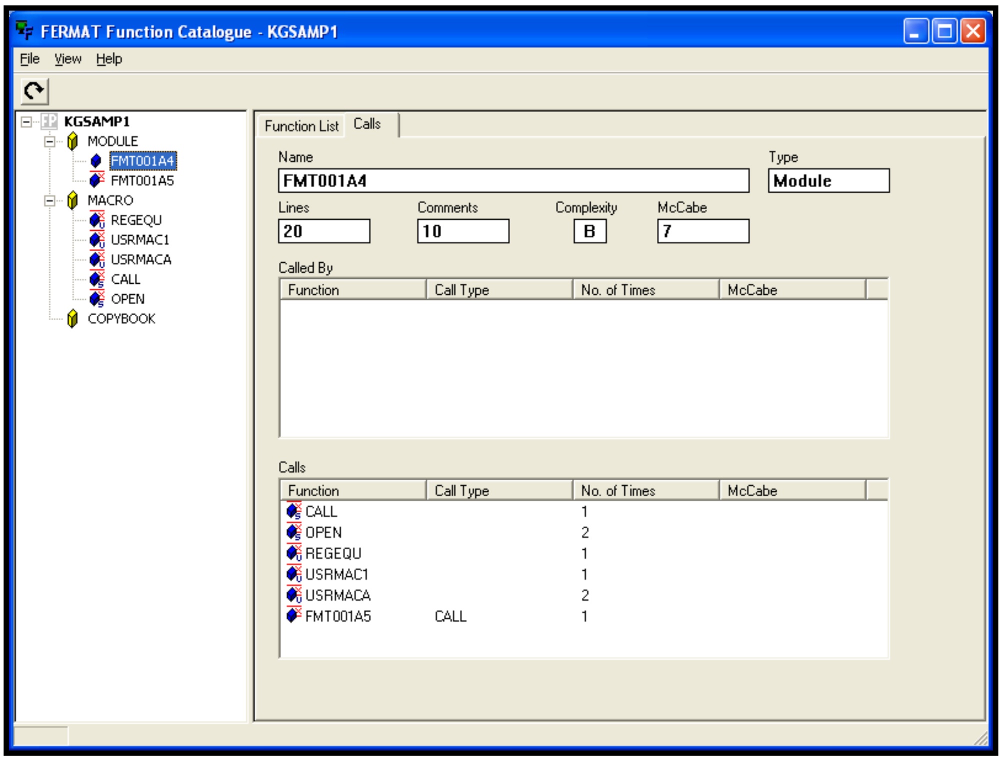
Sample program

```
*********************************************************************
*  FMT001A4  SAMPLE                                                 *
*********************************************************************
*        PRINT NOGEN
         REGEQU
FMT001A4 XNTR BASE=(R3,R12)
         STAKINIT
         PERFORM FILEOPEN
         PERFORM INITIAL             PROGRAM LINKAGE
         XEND
         EJECT
FILEOPEN ROUTINE
         USRMACA TYPE=INITIAL
         USRMACA TYPE=OPEN
         OPEN   (DDIN,(INPUT))
         OPEN   (RDSOUT,(OUTPUT))
         EXIT
         EJECT
INITIAL  ROUTINE
         CALL   FMT001A5,MODSTORE     STANDARD
         USRMAC1 FMT001A6
         EXIT
         EJECT
*
EOF1     BR    R14
*
         SHOWPATH
WSTACK   DS    16F
```

```
         LTORG
         END
```

This module contains two calls to external modules, one to FMT001A5 using the standard CALL macro and the other using a user Macro USRMAC1. The module also contains two sub-routines with the macro ROUTINE used as a routine header.

With no customisation the Function Catalogue displays



Name: FMT001A4
Type: Module
Lines: 20
Comments: 10
Complexity: B
McCabe: 7

Called By

Function	Call Type	No. of Times	McCabe

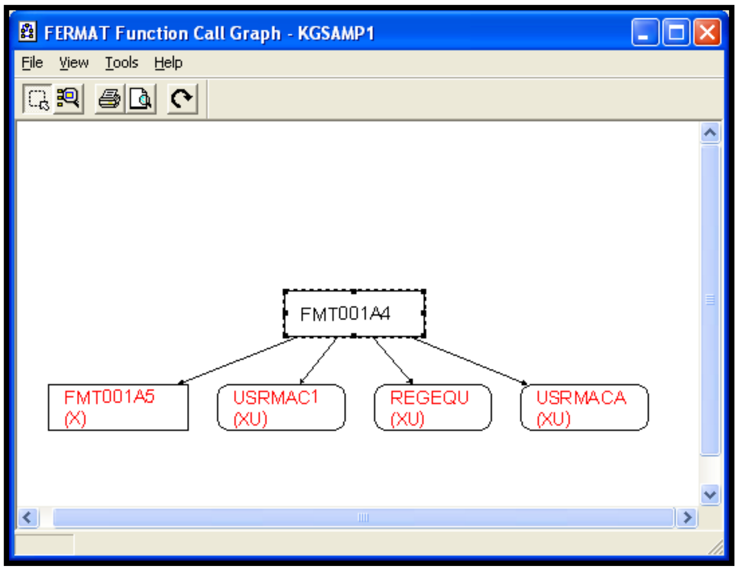
Calls

Function	Call Type	No. of Times	McCabe
CALL		1	
OPEN		2	
REGEQU		1	
USRMAC1		1	
USRMACA		2	
FMT001A5	CALL	1	

Only FMT001A5 identified as an external module.

No sub-routines identified in module FMT001A4.

And the Function Call Graph displays



Only the link to external module FMT001A5 had been identified.

The system is customised by adding USRMAC1 to the User Linkage table and ROUTINE to the routine table.

User Linkage Table Entry

1,USRMAC1,1

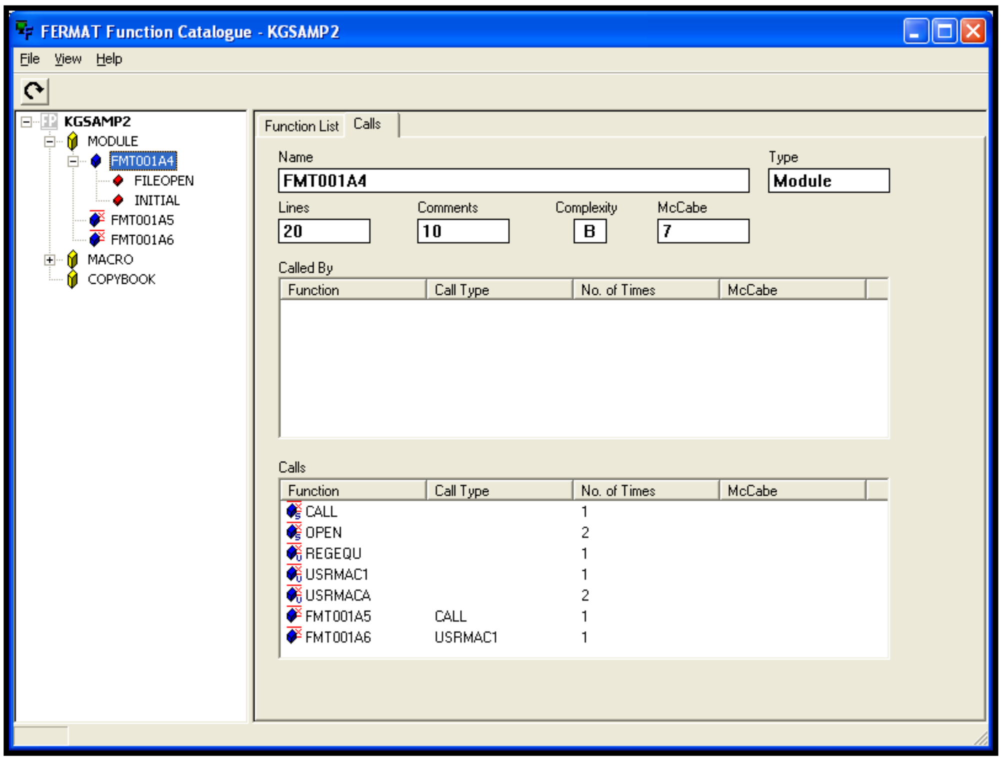
Routine Table Entry

routine

Software Migrations Ltd

The system now displays the following.

Function Catalogue



Name	Type
FMT001A4	Module

Lines	Comments	Complexity	McCabe
20	10	B	7

Called By

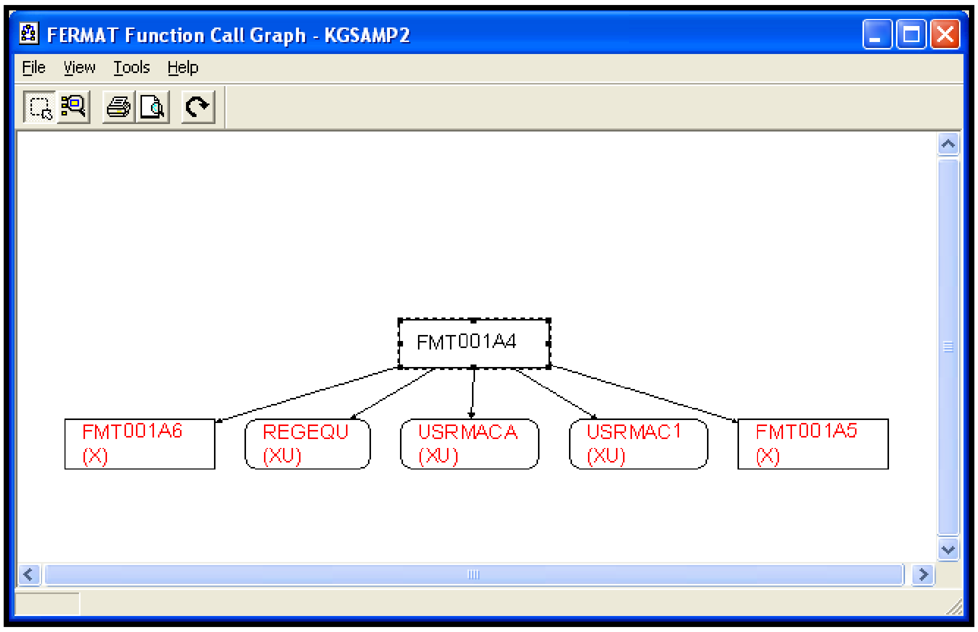
Function	Call Type	No. of Times	McCabe

Calls

Function	Call Type	No. of Times	McCabe
CALL		1	
OPEN		2	
REGEQU		1	
USRMAC1		1	
USRMACA		2	
FMT001A5	CALL	1	
FMT001A6	USRMAC1	1	

Note that the call to the external module FMT001A6 has now been recognised and the two sub-routines, FILEOPEN and INITIAL, in module FMT001A4.

Function Call Graph



Both calls to external modules now shown.

All the Fermat Workbench tables are held in the project Config folder.

Customisation must be preformed after defining the project and before Importing any files.

The following tables may be customised.

complex.tab

entry.tab

macros.tab

routine.tab

udata.tab

ulinkage.tab

uopcodes.tab

UserMacro.tab

ustmac.tsb

version.tab.

30.1 complex.tab

This table holds the default setting for the complexity code, which gives an indication of the complexity of the program.

The complexity code is derived from the value of the McCabe Index.

McCabe = count of branches + count of calls to sub-routines + 1

The current default settings are

A	5
B	10
C	50
D	100
E	200
F	350
G	600
H	800
I	> 800

Thus the bigger the letter the more complex the program.

30.2 entry.tab

This table contains instructions/macros which define entry points to the program. This table is used to identify calls from another program. Without customization, some calls will be shown as external calls.

Table entries are defined as follows:-

CUENTER [or CUENTER,0]	name from label
SASSAVE,n	name from nth positional parameter
BEGIN,ENTRY=	name from keyword parameter

Note data entry starts in column 2.

30.3 macros.tab

This table contains information used by the flowchart generator.

The following user changes may be added

XCALL – macros which call an external program.
ICALL – macros which call an internal sub-routine.
END – macros which end a sub-routine.
EXIT - macros which exit the program.
IGNORE – macros/instructions for the flowchart process to ignore.

30.4 routine.tab

This table contains information to define sub-routine headers.

Without customization, sub-routines will not be identified.

Entries are defined as follows :-

ROUTINE [or ROUTINE,0]	name from label
SASSCALL,n	name from nth positional parameter
SRCALL,ROUTINE=	name from keyword parameter

Note data entry starts in column 2.

30.5 udata.tab

This table contains macros which should be tracked through the Text Editor Data Tracker. The Text Editor Data Tracker has a facility to track through data items by name. By default all macros will be ignored. If a macro uses data names as a parameter, then they may be included in this table so that the Data Tracker will include them in the search.

30.6 ulinkage.tab

This table contains information to define macros which call other external modules or internal sub-routines. Without customization, then none of these calls will be identified.

Entries are defined as follows:-

type,data where type=1 for call to external module and type=2 for call to sub-routine.

1,SACALL,n	name from nth positional parameter
1,BEGIN,MOD=	name from keyword parameter
2,SUB,1	sub-routine call

30.7 uopcodes.tab

This table contains information for instructions/macros to be ignored by the scan process. They will be classed as standard instructions and no statistics will be gathered.

30.8 UserMacro.tab

When the scan process is examining a file to determine if it is a valid Assembler file and if the module consists of only user macros, then the file would not normally be recognized as an Assembler program. The user macro name should be added to this table so that the file will be recognized and imported into the project.

30.9 ustmac.tab

This table contains information of macros which define the start point of a program and contain the name of the program. Normally this name is taken from the START or CSECT instruction. Without customization, the start name will default to CSECT001.

Entries are defined as follows:-

CUSTART [or CUSTART,0]	name from label
SASSART,n	name from nth positional parameter
BEGIN,NAME=	name from keyword parameter

Note data entry starts in column 2.

30.10 version.tab

Some systems use a source version control as a prefix.

Some source names (i.e. filenames) are in the format of VGEF01nn where nn represents the source version.

Internal calls are made to VGEF01. In order to match up the call (VGEF01) to the actual file (VGEF01nn) the significant characters are specified in the version table.

e.g. £ENTER,6

30.11 ImportRules.tab

Data items for a module are defined by sections, one for the CSECT and one for each DSECT. The names for these sections are used to generate the names for the copybooks used in Cobol migration. Each name consists of six characters plus a 2 char suffix.

The six character name is derived from the input name during the import process.

These names must be unique.

Where input names are longer than six characters, then a rule must be defined to show which characters are to be ignored.

Entries are defined as

CpbDrop=a,x,y

a	characters to select programs names beginning with these characters.
	* to define default entry
x	position of 1st character to drop
y	position of 2nd character to drop
	(Optional, only required when names are 8 characters)

Examples

CpbDrop,AB,5,6

CpbDrop,*,2,3

Programs starting with AB will drop characters 5 & 6, all other programs will drop characters 2 & 3.

CpbDrop,*,2

All programs will drop character 2

30.12 utable.tab

This table contains definitions of look-up tables which should be treated by the call-graph as jump-tables to external modules. Without customization, the call-graph has insufficient information to determine how to process a look-up table to create the links to called programs.

Entries are defined as follows:-

type,module,data,param,end

type	This is always 1
module	module name containing the look up table, or first characters followed by an * (wild card) for all modules starting with characters.
data	table name

param	positional param number or keyword=name (where to find the name of the called module)
end	indicator for end of table.

Example:
1,MOD1,CALL_TAB,4,X'FF'

Defines a look-up table called CALL_TAB in module MOD1. The fourth entry in each row defines the called module, and the terminator for the table contains X’FF’

31. Using the SML Workbench in the Application Life Cycle

The SML Workbench is designed to assist organisations to comprehend their Assembler systems and improve programmer productivity for maintenance and enhancements. This is achieved by providing complete transparency of the code in use; detailing relationships between the various entities such as modules, macros and copybooks; providing detailed system analysis to allow the identification of dead code; to enable code slicing and to extract and document business rules hidden in the legacy assembler code.

31.1 Objective

The use of the Workbench is designed to raise the knowledge and capabilities of all programming staff; to share knowledge of the system and to enable staff unfamiliar with code to get “up to speed” sooner.

The Workbench automatically documents the system and provides a number of tools within the Text Editor to annotate the code (mark-up lines and blocks of code) in different colours to aid comprehension and the ability to add programmer annotations that describe the code and its functions. In order to promote knowledge sharing amongst all staff.

The goal is to keep these annotations in step with the source of the system, so that they are available and not have to be re-analysed and annotated each time.

This document describes the recommended approach to achieving this whilst maximising the benefits provided by the Workbench.

31.2 Issues

In order to allow detailed analysis of the assembler modules including dataflow and control dependency analysis, which enables the dead code analysis, code slicing and business rules extraction and documentation the Workbench uses full assembler listing with all macros fully expanded as input; from this we extract the source for use in the workbench. If the assembler source is conditional or is put through a pre-processor before the assembly then the source that is extracted from the listing input will not be the same as the actual source held on the main system libraries.

31.3 Solution

The Workbench will use 2 inputs for each module. The first, the listing as previously described, the second a copy of the actual source as stored on the mainframe.

All processing within the workbench will be performed on the listing and the source we extract from the listing.

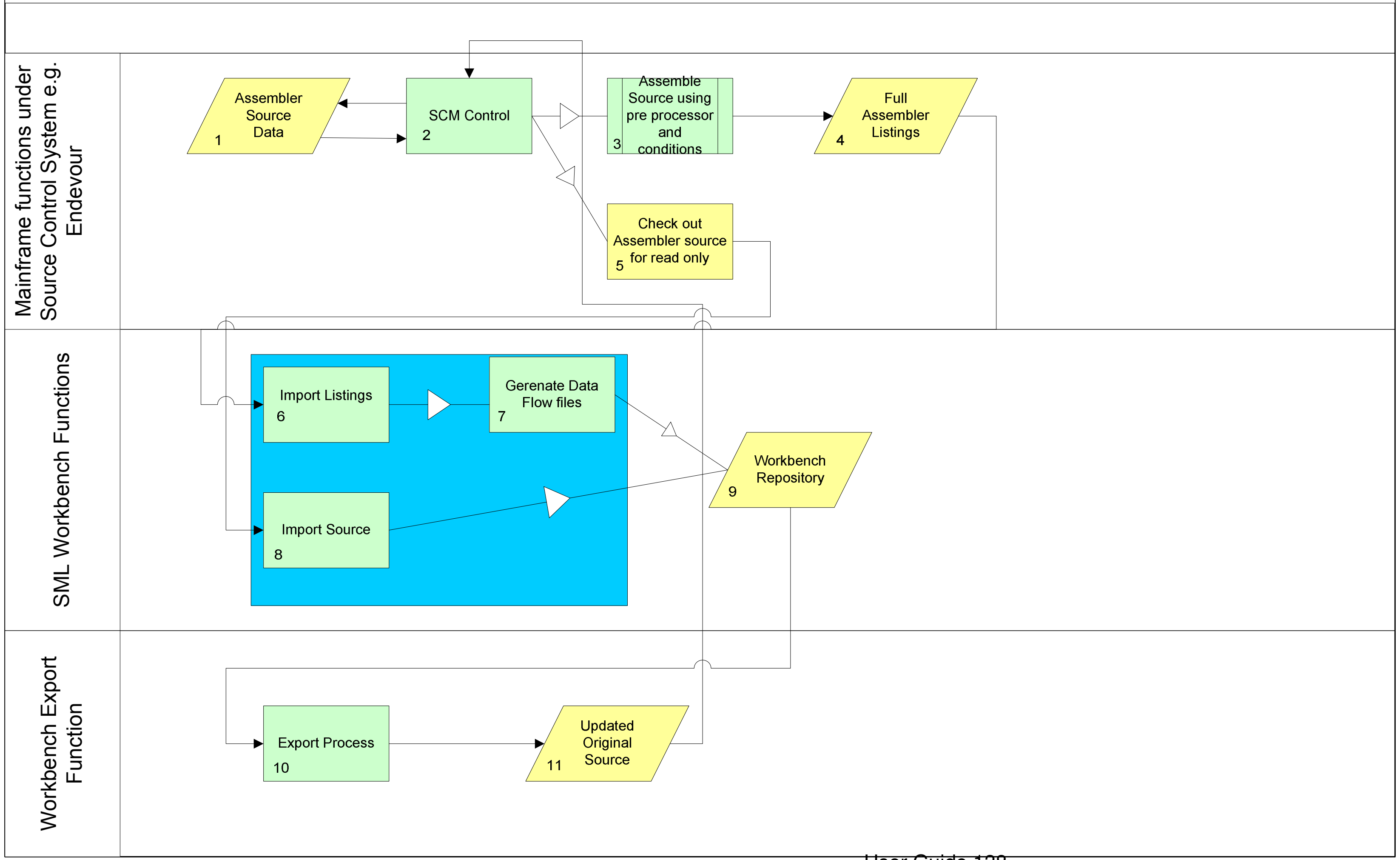
When a module has been annotated it can be exported. This process will merge the Fermat Comments generated in the annotation process with the original source module input.

The exported file can then be used as original source to make enhancements etc.

31.4 Life Cycle Example

Figure 1 - diagram depicting the Life Cycle.

Assembler Source Life Cycle


Mainframe functions under Source Control System e.g. Endevour
SML Workbench Functions
Workbench Export Function
Assembler Source Data 1
SCM Control 2
Assemble Source using pre processor and conditions 3
Full Assembler Listings 4
Check out Assembler source for read only 5
Import Listings 6
Gerenate Data Flow files 7
Import Source 8
Workbench Repository 9
Export Process 10
Updated Original Source 11

Software Migrations Ltd

It is assumed that the assembler source is maintained on the Mainframe system by a Source Control Management System (SCMS) such as Panvelet or Endeavour.

Narrative:

1 - The Assembler source as filed on the Mainframe System.

2 – The SCSM.

3 – Check out the source for read access from the SCSM and assembler as normal, using any conditions or pre-processor.

4 – Generated Full Assembler Listings.

5 – Check out the source for read access.

In the Workbench.

6 – Import the Full Assembler Listings into the Workbench

7 – Generate Dataflow files for detailed analysis within the Workbench

8 – Import the Source from 3 above.

9 – Data held in Workbench repository and available for use in the work bench.

The programmers and analysts can now analyze any modules using the Workbench for marking lines, marking blocks and creating user annotations to aid comprehension.

10 – The Workbench Export process will merge the generated Fermat comments with the original source.

11 – The Original source should be changed to Update on the SCSM and the newly commented source should be copied to the Mainframe Libraries.

Note.

Amendments to the code can be carried out in the Workbench or using the customers preferred editor.

31.5 Benefits

The Workbench and the annotations generated from detailed analysis are encompassed in the overall software life cycle. Therefore the analysis carried out on a module by an individual programmer or analyst is immediately available to all staff, thus promoting knowledge sharing and raising all staff comprehension of the system.

32. Batch Import Processing

The FermaT Workbench provides a facility to allow unattended importing of new and updated source, listings, macros and copybooks into the project repository. This facility works both in normal FermaT Workbench projects, and also in projects which have been marked as Read-only.

It is useful where scheduled updating of the FermaT Repository is required on a regular basis. If updated source, listings, macros and copybooks are made available in a set of directories from an installation's source control package, regular batch runs (e.g. nightly) will allow the Fermat project to be kept in step with the source control version.

In order to use the Batch processing facility, a file called startup.txt must be located in the software/config directory which defines the processing options required.

This file should contain the following parameters:

Project=	Location of Project to update
RefreshListings=	Location of Listings to be imported
RefreshSource=	Location of source to be imported
RefreshMacro=	Location of Macros to be imported
RefreshCopybook=	Location of Copybooks to be imported
ExportFolder=	Location of Function XML files to be exported (Note that this will export the entire contents of the project)
Datascan=yes	Run data scanner after import
Dataflow=yes	Will run the Dataflow creation only for new and updated listings which have been imported into the project during this session
Close=yes	Close workbench down at completion of batch processing

Note that at the completion of the batch process, the workbench will delete the startup.txt file from the software/config directory.

Example Startup.txt file:

```
Project=E:\FermatProjects\global
RefreshListings=E:\Import\RefreshListings
RefreshSource=E:\Import\RefreshSource
RefreshMacro=E:\Import\RefreshMacro
RefreshCopybook=E:\Import\RefreshCopybook
Datascan=yes
```

```
Dataflow=yes
ExportFolder=E:\Export
Close=yes
```

Place note: the final line in the startup.txt file must be a complete line. I.e. it must finish with a newline character.

When the workbench is started (this could either be by timer, or once the transfer of changed files has completed), it will immediately perform the commands as requested by the Startup,txt file.

The batch processing produces a report file in project\level1\inventory directory. This file is called ‘BatchProcess_yymmddhhmm.txt’

where ‘yymmddhhmm’ is the current timestamp

The report file shows what processing was performed during the batch run.

Example output:

```
Listings=E:\Import\REFRESHLISTINGS
Source=E:\Import\REFRESHSOURCE
Macro=E:\Import\REFRESHMACRO
Copybook=E:\Import\REFRESHCOPYBOOK
Datascan=YES
Dataflow=YES
Export=YES
Close=YES

--------------------------

09.10.07 12:49 Import Files
09.10.07 12:51 Generate Dataflow
09.10.07 12:52 Performing Datascan
09.10.07 12:55 Done Datascanning
09.10.07 12:58 Export XML Files
09.10.07 12:58 Batch Processing Complete
```

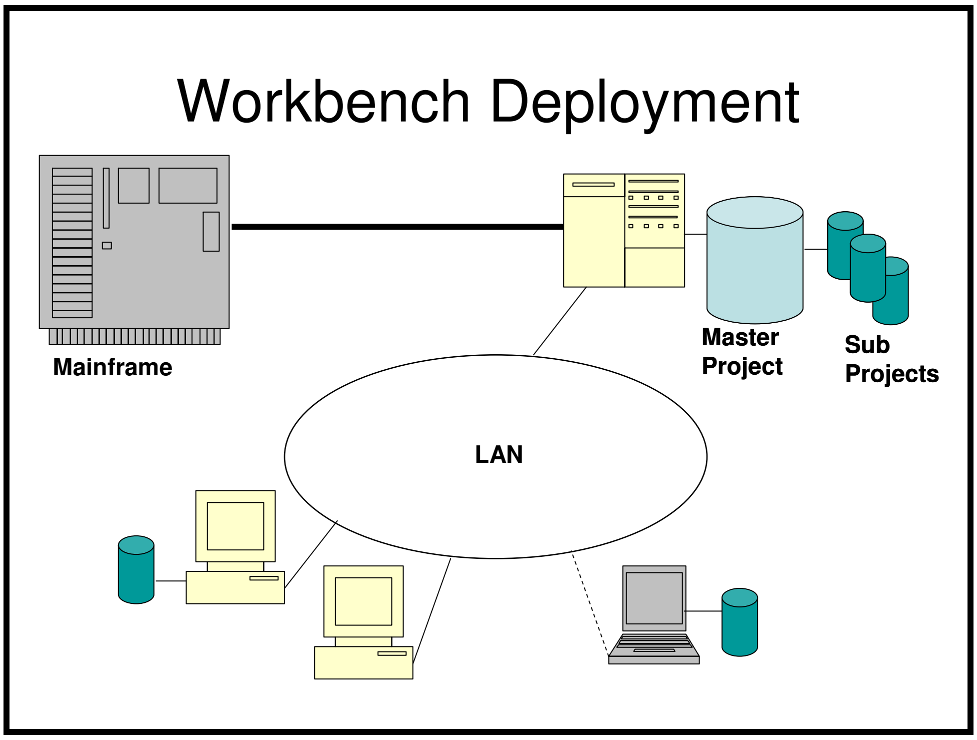
33. Workbench Deployment

The FermaT Workbench can be deployed in a number of ways, from a single user running on a PC or Laptop to deployment at a corporate level in a LAN environment or using Windows Terminal Services.

The simplest deployment is that of a single user PC or Laptop, where the Assembler system is imported directly and processed as either a single project or as a master project and several sub-projects.

In this deployment method all functions of the Workbench are available including integration into the Application Life Cycle.

The second option is to set the workbench up in a LAN environment. As depicted below:



In this deployment we have a master Workstation which is connected to the Mainframe, for integration into the Application Life Cycle, where new copies of the modules are downloaded then imported into the master and managed as sub-projects on the system.

In this deployment, users should copy the projects (master or sub-projects) to their local machine for processing. Whilst it is possible to logically connect to a project on a remote workstation the performance of some project wide processes such as executing the Data Scanner will have performance degradation as the required data files need to be transferred over the LAN for execution.

Care should also be taken over where managed sub-project reside, as depending on the size of the Workbench Projects large amounts of data may require transferring over the LAN, which would have impact on other users of the LAN infrastructure.

The third option for deployment of the Workbench is by deployment via Microsoft Terminal Services.

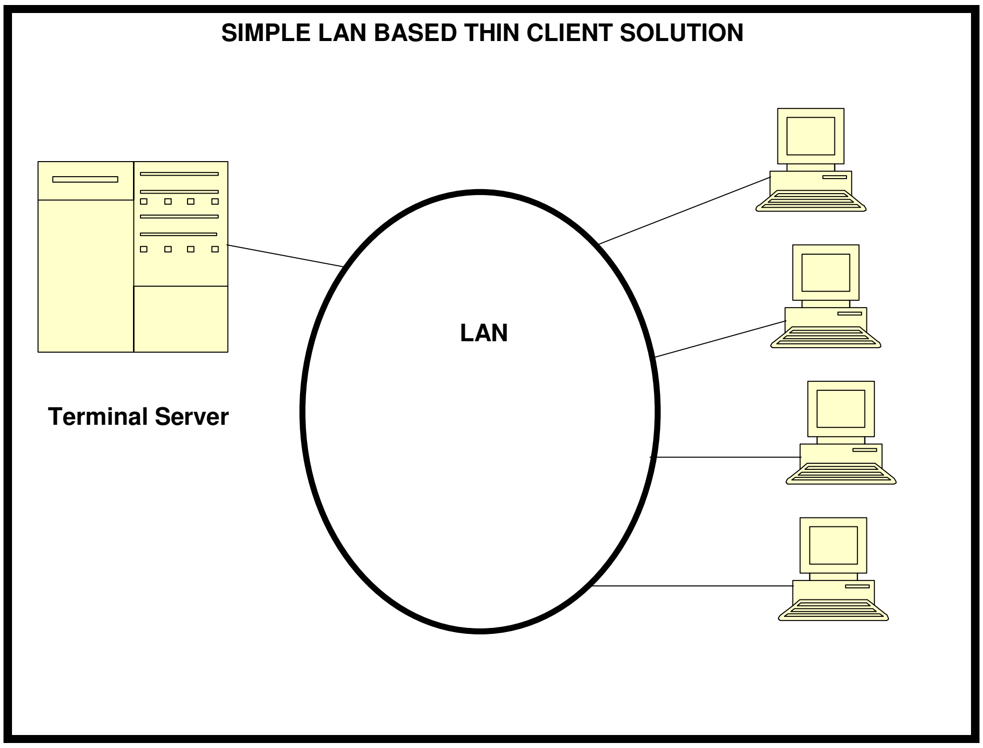
These environments work by having multiple windows sessions running on a server and only sending screen images, key strokes and mouse clicks over the network (also referred to as thin client computing).

In order to support multi-user sessions on the same server several enhancements have made to the workbench to support multi-users by making the default actions such as the text-editor read only. If users want to make amendments to modules they must check out the module for update. This locking mechanism is implemented at a project or sub-project level, any processes that require access to checked out files, such as importing new files impact assessments etc, will be suspended and notice given as to who has checked out the file and when. Once all files required for the process are available in the projects the process can then be restarted.

A thin client solution is also of benefit to large organisations who would be working over a LAN environment, as the introduction of the Workbench would have only minimal impact on LAN traffic, no desktop software would be required and the Fermat Workbench software and projects could be centrally managed.

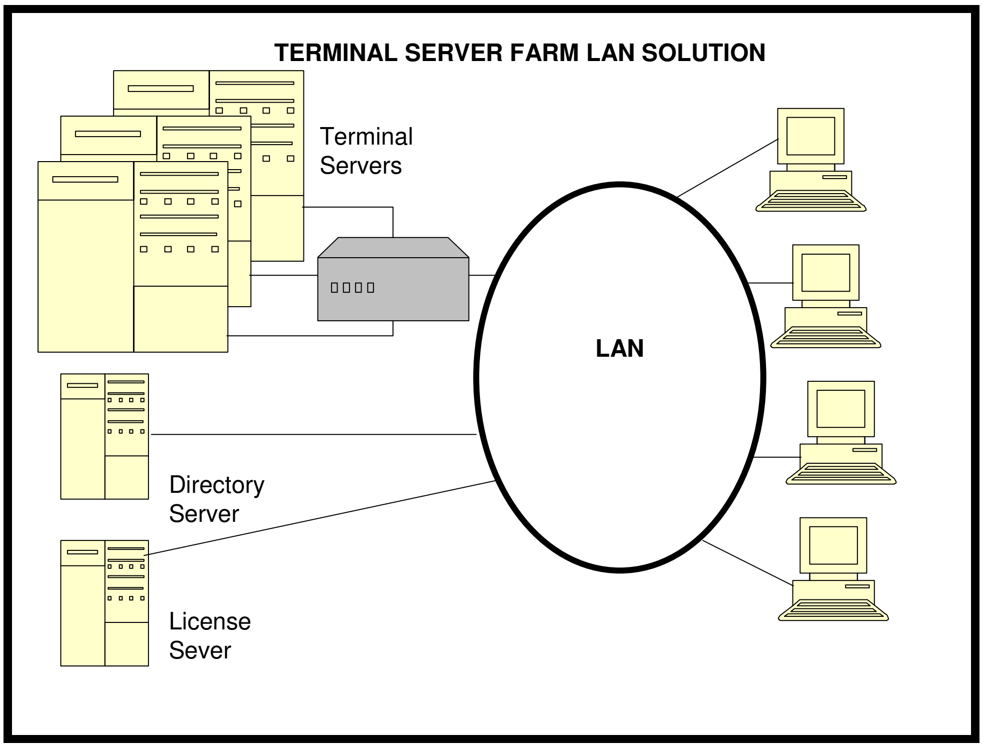
In addition to central management this solution also provides remote desktop support, where support personal can view and take-over sessions from users to resolve problems and guide users through applications.

The following diagram depicts a Terminal Server implementation.



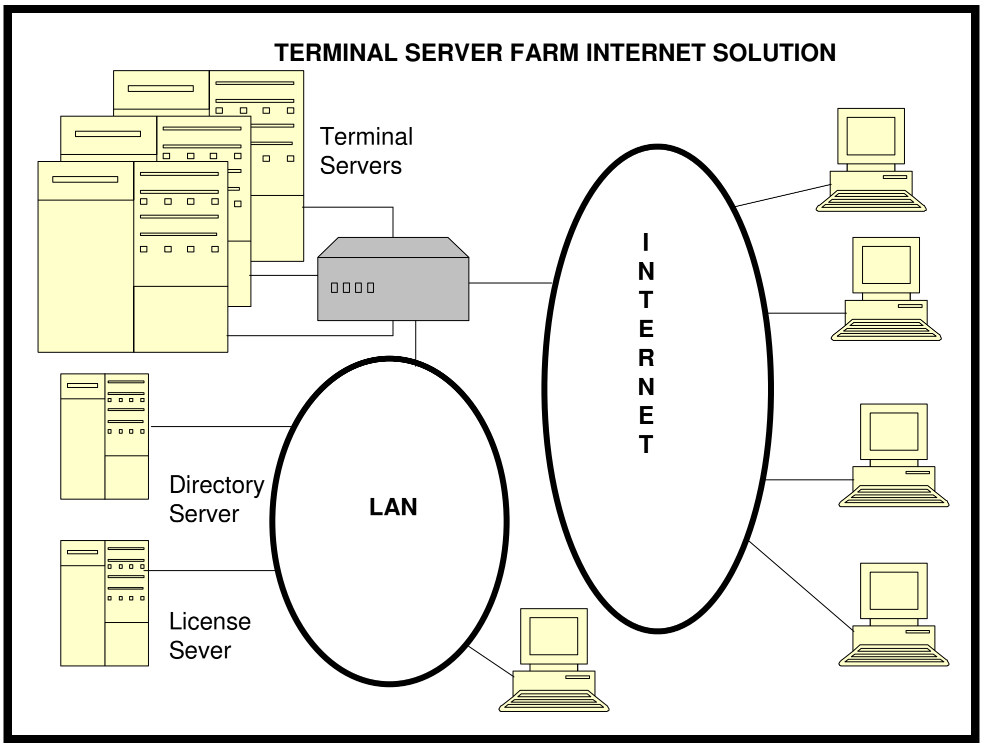
A Terminal Services server is attached to the local LAN which runs the workbench software, local workstations and laptops then connect via the LAN and execute the workbench software, all processing takes place on the server where the data is local and only minimal traffic is placed on LAN.

The following diagram depicts a Terminal Server solution to support more users on a local LAN infrastructure.



In this solution we have a farm of Terminal Services Servers. Each server will have users allocated to it and will run the workbench, the router is used to direct certain users to individual Terminal Servers. The License server is mandatory for Windows 2003 and ensures users are licensed to run the software. Users can be licensed per device or more practically can be based on concurrent users. The directory server is used to ensure each user only runs a single session and also that, in the event of a connection failure will be reattached to the same application once the connection is re-established.

The next diagram depicts a large organisation with a global presence, which have many remote workers.



This solution is similar to the previous, except remote users connect to the terminal servers using a VPN connection over the public internet. Because all processing is performed locally on the Terminal Servers and only minimal traffic is transmitted over the communications channel, user response is acceptable even from remote locations even on dial-up connection speeds.

The key to running such applications on a terminal server is the ratio of the number of users that can be running concurrently on the server. The workbench can be computationally intensive in some operations especially when generating flowcharts and can be resource hungry when performing global operations such as scans or batch processing, however it is the concurrency of such activities which are key.

The exact sizing for a Terminal Services Environment to support the FermaT Workbench is dependant on the size of the Assembler applications imported; the number of sub-projects required and the number of concurrent users.

SML will provide guidance for customers on deployment options and system sizing as part of the project planning for the Workbench implementation.

34. Understanding Dataflows & Controls

34.1 What is a Dataflow.

A dataflow is a link between the update of a variable and a reference to the variable where there is a control flow path between the update and the reference. A backwards dataflow link goes from the reference to the update (i.e. the value of the variable at the reference may depend on the value assigned at the update). Backwards dataflow links are followed during backwards slicing. A forwards dataflow link goes from the update to the reference (i.e. the value assigned to the variable at the update may affect the value seen at the reference).

Forwards dataflow links are followed during forwards slicing.

So a backwards dataflow can start from any reference to a variable in an assembler instruction, and a forwards dataflow can start from any reference to a variable in an assembler instruction. (This does NOT include assembler directives such as USING, DROP etc). A variable may be both referenced and updated in the same instruction.

A backwards dataflow will mark the selected line, any backwards dataflow lines starting at he current line, any of *their* dataflow lines, and so on until no more dependencies are found.

34.2 What is a Control dependency.

A control dependency of a node is any decision node which can "control" whether or not the given node will be executed. To be precise:

(1) From one edge out of the decision node, *every* path to the end of the flow graph passes through the given node; and

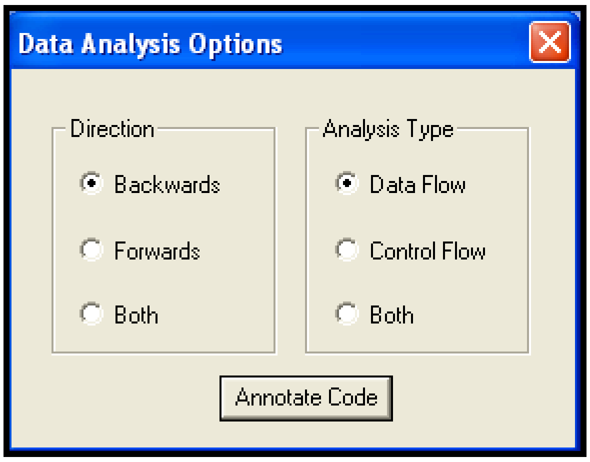
(2) From a different edge out of the decision node, there exists a path to the end of the flow graph which *doesn't* pass through the given node.

A backwards control dependency scan will find all the control dependencies nodes for a given node, plus all *their* control dependencies, and so on.

A forwards control dependency scan can be started from any decision node.

It finds all the nodes which are control dependent on this node, plus nodes which are control dependent on *them* and so on.

Whenever a code slice process is used in the Workbench, such as in the Create Annotate Files, BRE Processing or Auto Gen Business Rules, the user is prompted to select what code slice option is required.



Direction: [✓ Backwards] Forwards Both

Analysis Type: [✓ Data Flow] Control Flow Both

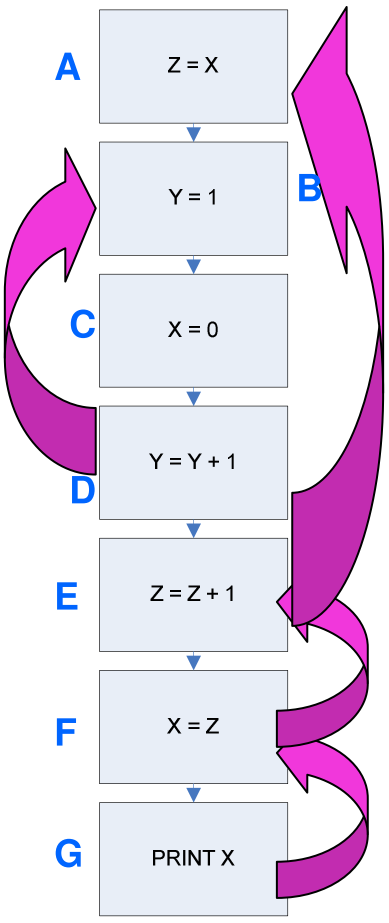
The user can decide the direction of the slice, either backwards, forwards or both. If the "Both" option is selected this is the same as performing first a backwards then a forwards pass on the code and the lines of identified will be marked in separate colours to distinguish between the backward and forward slice.

The user can decide what type of analysis to perform either a dataflow, control dependent or both. When the "Both" option is selected the process will perform a simultaneous dataflow and control dependent, this will show different results than performing separate data flow and control dependent code slices, because a control flow link will lead to some dataflow links, which in turn lead control flow links an so on.

The following examples explain the code slicing operation and the results of performing code slices on various part of the code.

The dataflows are identified by pink arrows in the flowcharts and the control flow by green dotted lines.

34.3 Example 1



```
005         Z=X
010         Y=1
015         X=0
020         Y=Y+1
025         Z=Z+1
030         X=Z
035         PRINT X
```

Perform a backwards dataflow slice on variable **X**

- From Node A will identify nothing
- From Node C will identify nothing (because any previous assignments are lost)
- From Node F will identify Nodes E A
- From Node G will identify Nodes F E A

Perform a forwards dataflow slice on variable **X**

- From Node A will identify Nodes E F G
- From Node E will identify Nodes F G
- From Node F will identify Node G

Perform a backwards dataflow slice on variable **Y**

- From Node B will identify nothing
- From Node D will identify Node B

Perform a forwards dataflow slice on variable **Y**

- From Node B will identify Node D
- From Node D will identify nothing

Perform a backwards dataflow slice on variable **Z**

- From Node A will identify nothing
- From Node C will identify nothing (because any previous assignments are lost)
- From Node F will identify Nodes E A
- From Node G will identify Nodes F E A

Perform a forwards dataflow slice on variable **Z**

- From Node A will identify Nodes E F G
- From Node E will identify Nodes F G
- From Node F will identify Node G

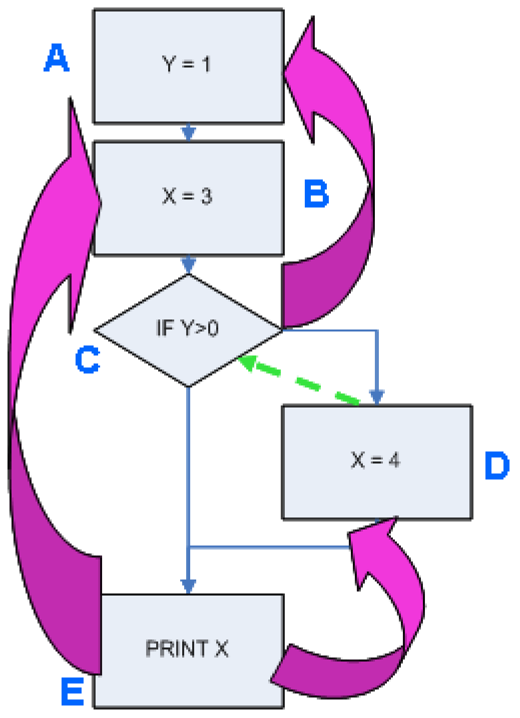
Bi-directional slices simply slice on one direction, then the other. So the results from bi-directional slicing are the set of results from performing a forward slice, then a backwards slice.

Note there are no control dependencies in this example.

	DF	DF	DF
Start Node	B/W	F/W	Both
A – slice on Z		EFG	EFG
B – Slice on Y		D	D
C – slice on X			
D – slice on Y	B		B
E – slice on Z	A	FG	AFG
F – slice on X	EA	G	EAG
G – slice on X	FEA		FEA

Software Migrations Ltd

34.4 Example 2



```
005     Y=1

010     X=3

015     IF Y > 0 THEN X = 4

020     PRINT X
```

Dataflow:
Perform a backwards dataflow code slice on variable X

- From Node E will identify Nodes D B
- From Node D will identify nothing
- From Node B will identify nothing

Perform a forwards dataflow code slice on variable X

- From Node B will identify Node E
- From Node D will identify Node E

Note there is no dataflow link between Nodes B and Node D since Node D overwrites any previous value of X.

	DF	DF	DF
Start Node	B/W	F/W	Both
A		C	C
B		E	E
C	A		A
D		E	E
E	DB		DB

Control Dependency:

A backwards Control Dependency is asking the question “which decision points are critical to the execution of this node”

In this example, the only node where this question of relevant is node D. If I perform a backwards control dependency from node D it will identify node C (because node C could exit directly to Node E bypassing node D completely).

A forwards Control Dependency is asking the question “which nodes are dependant on the execution of this node”

In this example the only node where this question is relevant is node C. If I perform a forwards control dependency from node C, it will identify node D

	CD	CD	CD
Start Node	B/W	F/W	Both
A			
B			
C		D	D
D	C		C
E			

Combined Slice (Dataflow/Control Dependency)

The combined slice combines the characteristics of both the dataflow slice, and the control dependency slice. It and uses the results of the slice in progress to dynamically increase the scope of the slice criteria as it progresses.

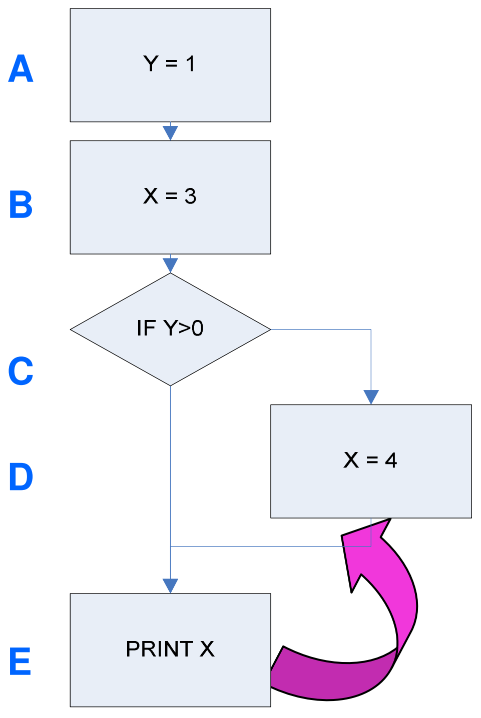
As the user selected slice progresses and starts to identify nodes which comprise part of either the dataflow, or of the control dependency slice, then these new nodes are added to the slicing criteria then the slicing continues in the chosen direction.

As a result of this dynamic expansion of the searching criteria, the combined slice will usually identify more results than the Dataflow and the Control Dependency separately

Worked example of backwards combined slice

Performing a backwards combined slice on X from node E

Current Slicing Criteria	
Dataflow (Field)	Control Dependency (node)
X	E



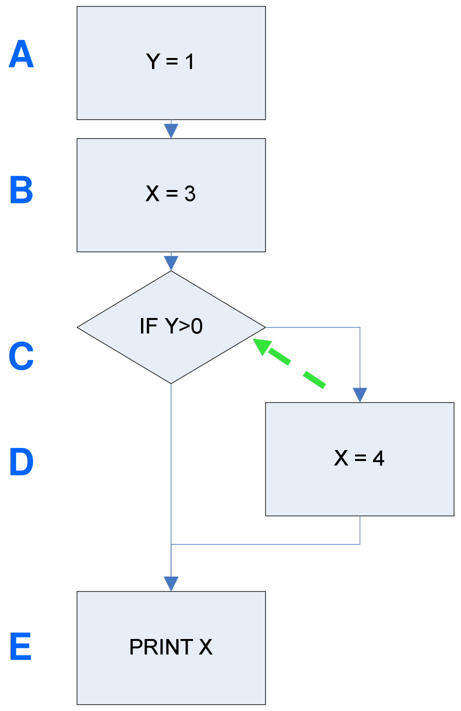
Backwards dataflow from node E, finds node D. (because X is being updated). This field gets added to current dataflow criteria (already present-so no change and this node gets added to the Control Dependency criteria

Current Slicing Criteria	
Dataflow (Field)	Control Dependency (node)
X	E, **D(added)**

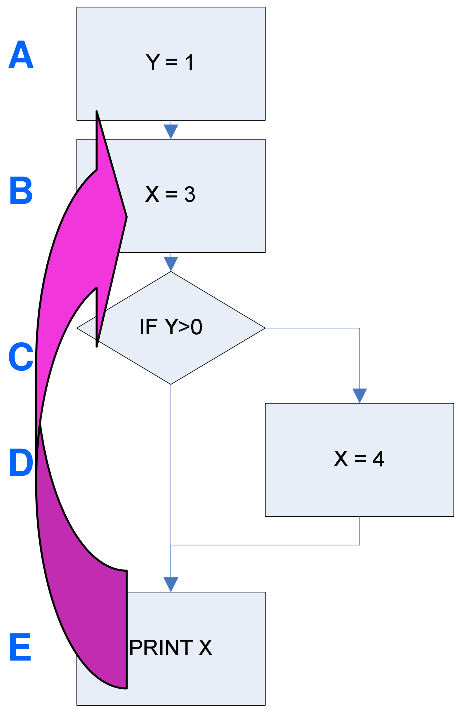
Now slicing backwards, we find a control dependency from node D to node C (critical decision point).

So we add field Y to the data flow criteria (because it's the value in the control dependent node), and we also add node C to the control dependency criteria

Current Slicing Criteria	
Dataflow (Field)	Control Dependency (node)
X, **Y(added)**	E, D, **C(added)**



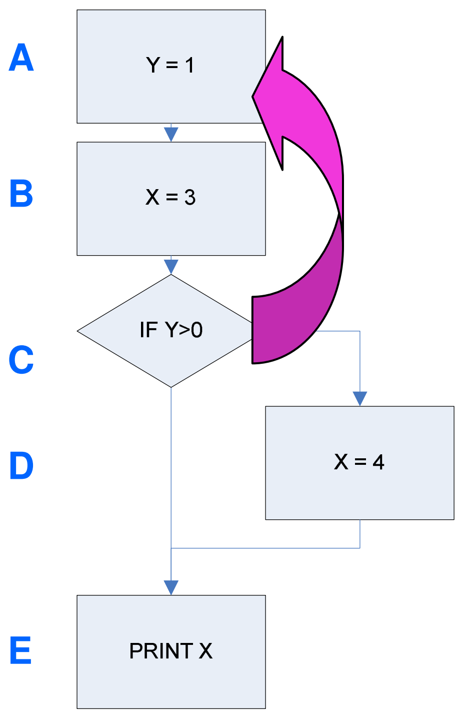
Now slicing back to node B we find that B is part of the dataflow from node E



However since X is already part of the dataflow criteria there is no need to add it. We add node B to the control dependency criteria.

Current Slicing Criteria	
Dataflow (Field)	Control Dependency (node)
X, Y	E, D, C, **B(added)**

Continuing the slice backwards, we now find that node A is part of the dataflow slice of field Y from node C. Since field Y is already part of the dataflow slice criteria, there is no need to add it, and there is no need to add node A to the control dependency criteria (because B will yield the same control dependency results)

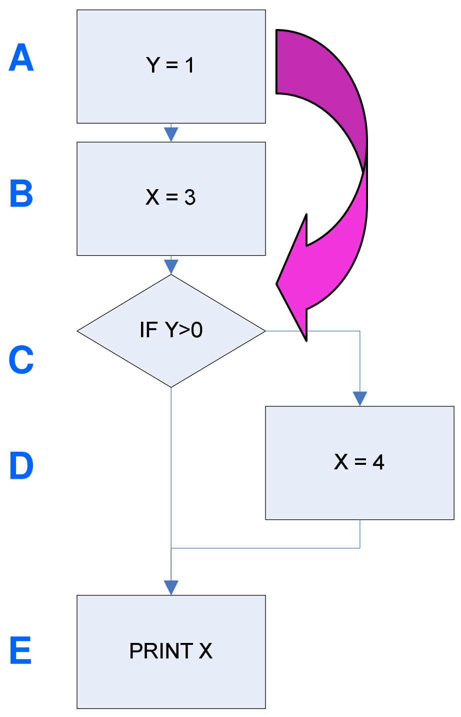


The results of the backwards dataflow/control dependency slice will include nodes D, C, B, and A

Worked example of forwards combined slice

Performing a forwards combined slice on field Y starting at node A

Current Slicing Criteria	
Dataflow (Field)	Control Dependency (node)
Y	A

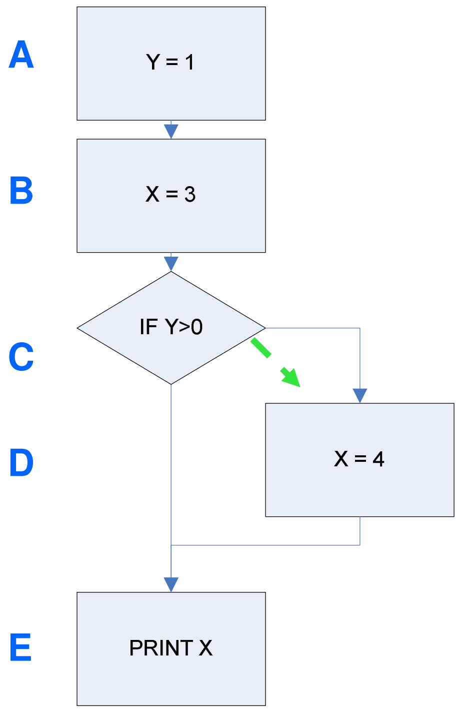


Forwards control flow from node A will yield no results (since this is not a decision point, there can be no dependencies on it), forwards dataflow on field Y will identify node C.

We do not need to add field Y to the dataflow criteria (already there), but we will add not C to the control dependency criteria.

Current Slicing Criteria	
Dataflow (Field)	Control Dependency (node)
Y	A, **C(added)**

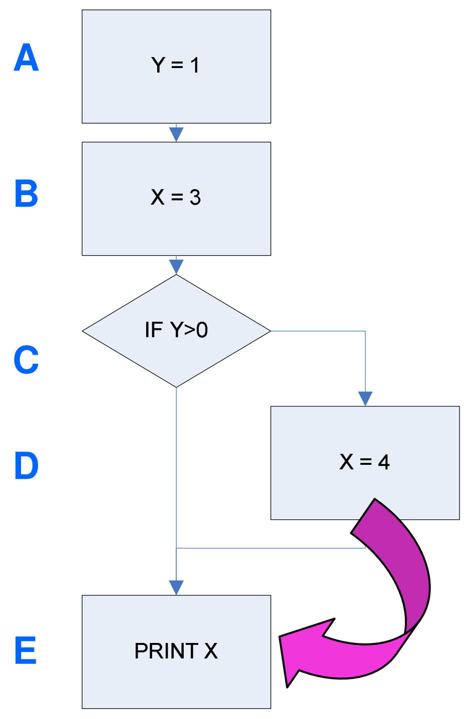
Continuing to slice forward, we identify node D (since this has a dependency on node C).



We therefore add field X to the data flow criteria (because it's the value in the control dependent node). We do not need to add node D to the control dependency criteria because it it not s decision point, it can have no dependencies.

Current Slicing Criteria	
Dataflow (Field)	Control Dependency (node)
Y, **X(added)**	A, C

Continuing to slice forward, we find that node E is part of the dataflow from node D. We don't need to add field X to the dataflow criteria (it's already there), and we don't need to add node E to the control l dependency since it is not a decision point (and cannot therefore have any further dependencies).



The results of the forwards dataflow/control dependency slice will include nodes C, D and E.

This gives the following slicing results for combined slices from each node:

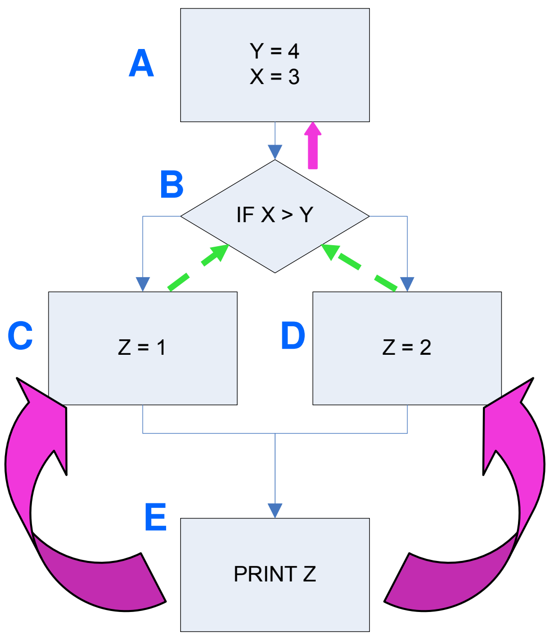
	DF/CD	DF/CD	DF/CD
Start Node	B/W	F/W	Both
A		CDE	CDE
B		E	CDE
C	A	DE	ADE
D	CA	E	CAE
E	DCBD		DCBA

Gathering the results from the dataflow slices, control dependency slices and the combined slices gives the follow full set of results.

	DF	DF	DF	CD	CD	CD	DF/CD	DF/CD	DF/CD
Start Node	B/W	F/W	Both	B/W	F/W	Both	B/W	F/W	Both
A		C	C					CDE	CDE
B		E	E					E	CDE
C	A		A		D	D	A	DE	ADE
D		E	E	C		C	CA	E	CAE
E	DB		DB				DCBD		DCBA

The effect of the criteria expansion in the combined slice can clearly be seen in this table. The combined slice consistently produces more results than the other types of slice.

34.5 Example 3



```
005         Y= 4
010         X = 3
015         IF X>Y THEN
020            Z=2
025         ELSE
030            Z=1
035         END IF
040         PRINTZ
```

Perform a backwards dataflow slice on variable Z
- From Node E will identify Nodes C D
- From Node C will identify Nothing
- From Node D will identify Nothing

Perform a forwards dataflow slice on variable Z
- From Node E will identify nothing
- From Node C will identify Nodes E
- From Node D will identify Nodes E

Perform a backwards control dependency
- From Node E will identify nothing
- From Node C will identify Node B
- From Node D will identify Node B

Perform a forwards control dependency
- From Node B will identify Nodes C D

Note:- There are no other valid control flow dependency starting points

Perform a backwards dataflow and control dependency on variable Z
- From Node E will identify Nodes C D B A
- From Node C will identify Nodes B A
- From Node D will identify Nodes B A

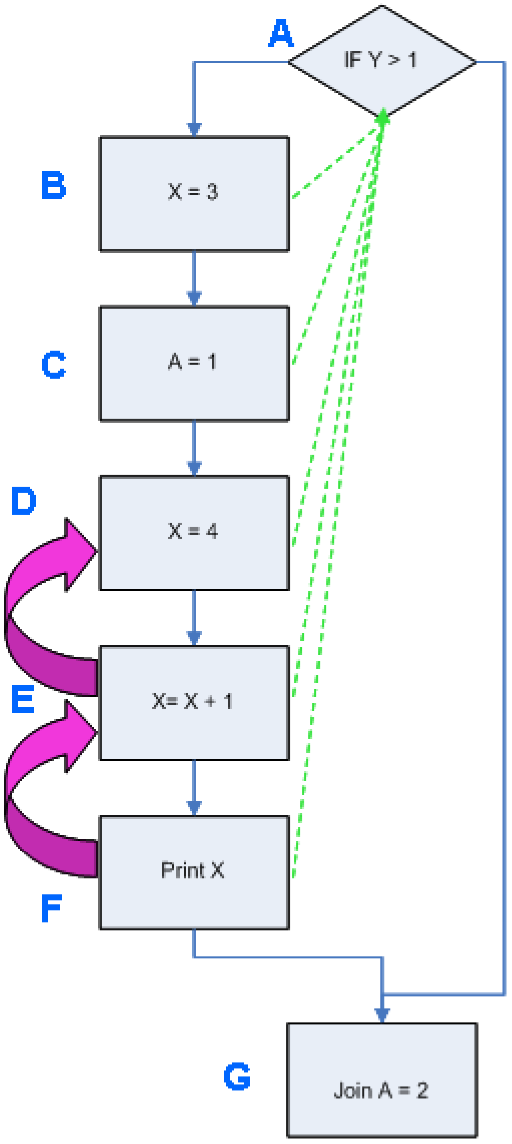
Perform a backwards dataflow and control dependency on variable X
- From Node B will identify Node A

Perform a forwards dataflow and control dependency on variable X
- From Node B will identify Nodes C D E

Perform a forwards dataflow and control dependency on variable Z
- From Node C will identify Nodes E
- From Node D will identify Nodes E

	DF	DF	DF	CD	CD	CD	DF/CD	DF/CD	DF/CD
Start Node	B/W	F/W	Both	B/W	F/W	Both	B/W	F/W	Both
A		B	B					BCDE	BCDE
B	A		A		CD	CD	A	CDE	ACDE
C		E	E	B		B	BA	E	BAE
D		E	E	B		B	BA	E	BAE
E	CD		CD				CDBA		CDBA

34.6 Example 4



```
005         IF Y =1 THEN GOTO JOIN

010         X=3

015         A=1

020         X=4

025         X=X+1

030         PRINT X

035 JOIN    A=2
```

Perform a backwards dataflow slice on variable X

- From Node F will identify Nodes E D
- From Node E will identify Nodes D
- From Node D will identify Nothing
- From Node B will identify Nothing
-

Perform a forwards dataflow slice on variable X

- From Node F will identify nothing
- From Node E will identify Nodes F
- From Node D will identify Nodes E F
- From Node B will identify nothing
-

Perform a backwards control dependency on variable X

- From Nodes B C D E F will identify Node A
- From Node G will identify nothing as the outcome of Node A is irrelevant.

Perform a forwards control dependency

- From Node A will identify Nodes B C D E F

Note:- There are no other valid control flow dependency starting points

Perform a backwards dataflow and control dependency on variable X

- From Node F will identify Nodes A D E
- From Node E will identify Nodes A D
- From Node D will identify Nodes A
- From Node B will identify Nodes A

Perform a forwards dataflow and control dependency on variable X

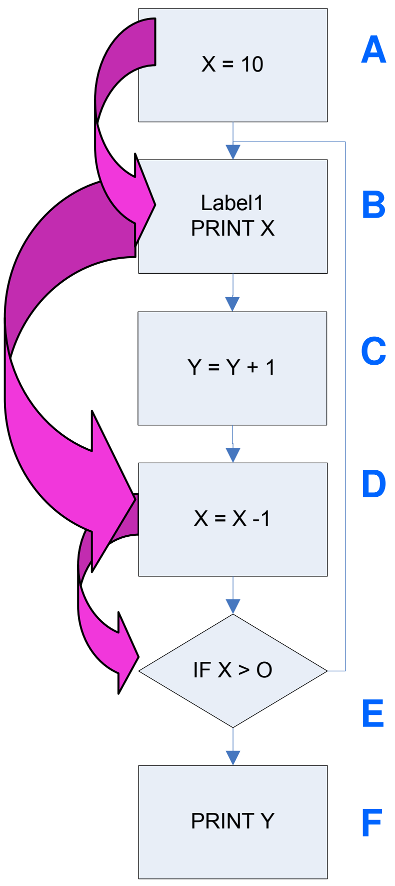
- From Node F will identify nothing
- From Node E will identify Nodes F
- From Node D will identify Nodes E F
- From Node C will identify nothing
- From Node B will identify nothing
- From Node A will identify Nodes B C D E F

Perform a bi-directional dataflow and control dependency on variable X

- From Node F will identify Nodes A E D
- From Node E will identify Nodes A D F
- From Node D will identify Nodes A E F
- From Node C will identify Node A
- From Node B will identify Node A
- From Node A will identify Nodes B C D E F
- From Node G will identify nothing
-

	DF	DF	DF	CD	CD	CD	DF/CD	DF/CD	DF/CD
Start Node	B/W	F/W	Both	B/W	F/W	Both	B/W	F/W	Both
A					BCDEF			BCDEF	BCDEF
B				A		A	A		A
C				A		A	A		A
D		EF	EF	A		A	A	EF	AEF
E	D	F	DF	A		A	DA	F	DAF
F	ED		ED	A		A	EDA		EDA
G									

34.7 Example 5



```
005              X=10
010     LABEL1 PRINT X
015              Y=Y+1
020              X=X-1
025              IF X>0 GOTO LABEL1
030              PRINT Y
```

Fix this stuff………..Perform a backwards dataflow code slice on variable X
- From Node D will identify Nodes C A
- From Node C will identify Node A

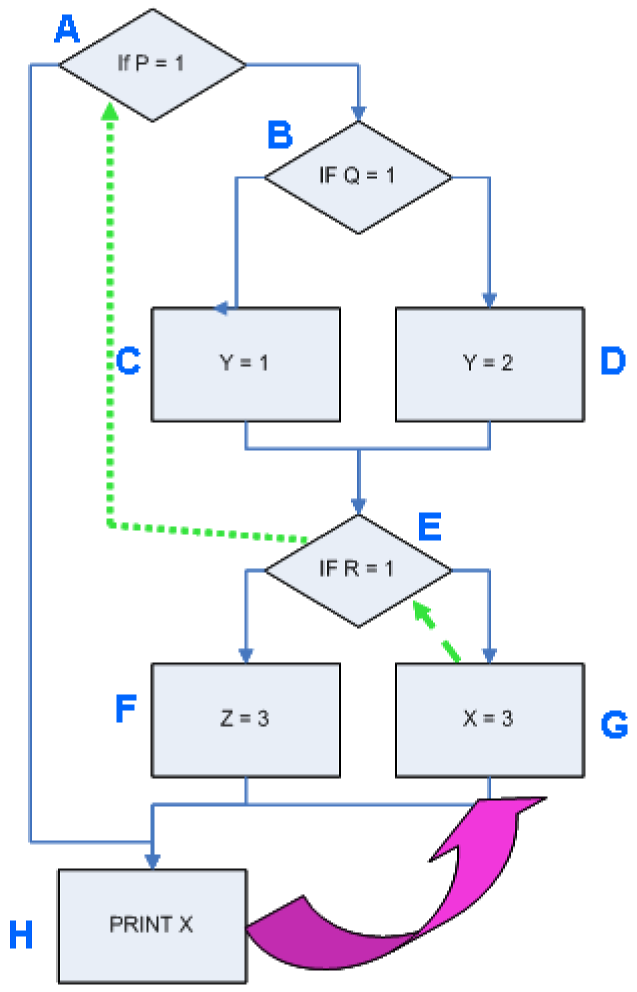
Perform a forwards dataflow code slice on variable X
- From Node A will identify Node C D
- From Node C will identify Node D

Perform a backwards control dependency code slice
- From Node B will identify Node D
- From Node C will identify Node D

Perform a forwards control dependency code slice
- From Node D will identify Nodes B C

	DF	DF	DF	CD	CD	CD	DF/CD	DF/CD	DF/CD
Start Node	B/W	F/W	Both	B/W	F/W	Both	B/W	F/W	Both
A		DE	DE					BDECF	BDECF
B	AD		AD	E		E	AED		AED
C		F	F	E		E	EDA	F	EDAF
D	A	EB	AEB	E		E	AE	EBCF	AEBCF
E	DA		DA		BCD	BCD	DA	BCDF	DABCF
F	C		C				CEDA		CEDA

34.8 Example 6



```
005     IF P=1
010         THEN IF Q=1
015             THEN Y=1
020             ELSE Y=2
025         END IF
030         IF R=1
035             THEN X=3
040             ELSE Z=3
045         END IF
050     END IF
```

Performing a backwards dataflow slice on variable X in Node H will identify Node G

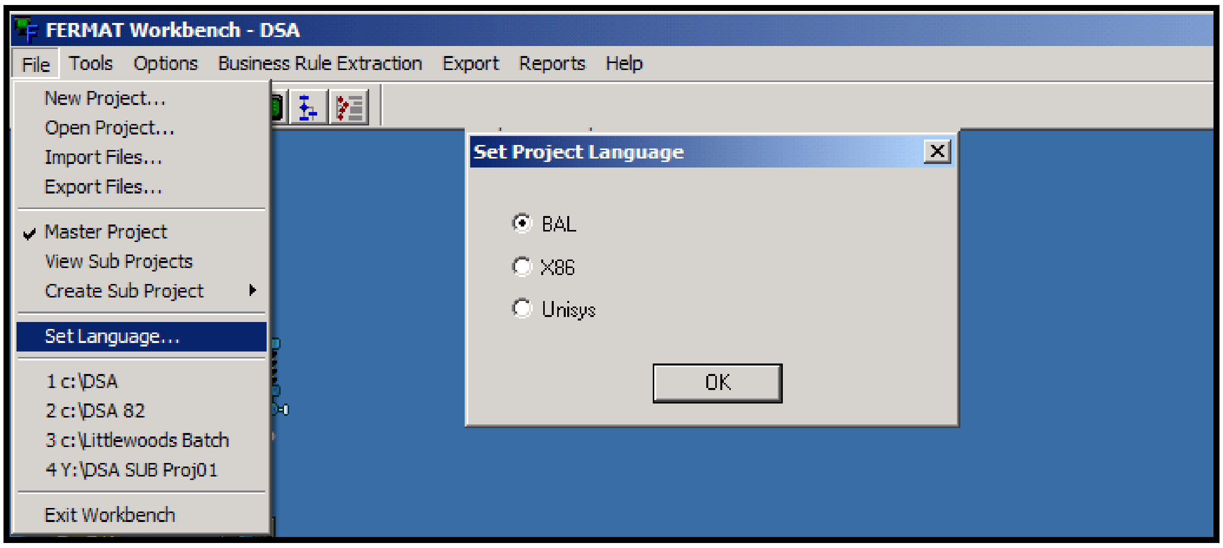
Performing a backwards control dependency on variable X in Node H will identify nothing.

Performing a backwards control dependency on variable X in Node G will identify Node E as there is a direct control flow link and will also identify Node A. The reason for this the decision Node E is control dependant on Node A, therefore both Nodes E and A are identified in the code slice from Node G.

	DF	DF	DF	CD	CD	CD	DF/CD	DF/CD	DF/CD
Start Node	B/ W	F/W	Both	B/W	F/W	Both	B/W	F/W	Both
A					BCDEFG	BCDEFG		BCDEFGH	BCDEFGH
B				A	CD	ACD	A	CD	ACD
C				BA		BA	BA		BA
D				BA		BA	BA		BA
E				A	FG	AFG	A	FGH	AFGH
F				EA		EA	EA		EA
G		H		EA		EA	EA	H	EAH
H	G		G				GEA		GEA

35. Supporting Different Assembler Languages

The FERMAT Workbench can used to import an increasing number of different Assembler languages. This is achieved by Creating a new project using the main menu of the workbench then, by selecting Set Language from the File Menu in the main menu, the user is prompted to select the type of assembler language that is to be used for that project. See below:



In the above example 2 assembler languages are supported BAL (IBM Mainframe Assembler) and x86 Assembler which supports Intel 80186, 80286 and 80836 code.

36. Feedback and Support

Your comments and suggestions are very welcome, as user feedback is the essence of a quality product. If you need any assistance while evaluating the Workbench, or would like to comment on the product, please contact the SML Support team by email:

Support enquiries :	support@smltd.com
All other enquiries :	enquiries@smltd.com