Understand Your Software...

Understand for FORTRAN

User Guide and Reference Manual

Version 1.4

Scientific Toolworks, Inc.

Scientific Toolworks, Inc.
1579 Broad Brook Road
South Royalton, VT 05068

Part Number: USTAND-F-UG-166 (8/7/02)

Contents

Chapter 1 Introduction

What is *Understand for FORTRAN*? 1–2
FORTRAN Versions Supported 1–2
Use Server Mode to Integrate with Your IDE. 1–2
For Those Who Don't Like to Read Manuals 1–3
Understand's Parts and Terminology 1–4
Parts 1–4
Terminology 1–5
Right-Click Menus Are Everywhere 1–6
Quickly Find Things in Your Source 1–8
Project Window 1–8
Locator Window. 1–9
Using Find in Files to Explore 1–10
Source and Graphical Views 1–11
The Information Browser Shows It All 1–12
Source Editor Provides Information Wherever You Look 1–14
Hierarchy Views Show Multi-Level Relationships 1–15
Structure Views Quickly Show Structure and Relations. 1–17
ASCII and HTML Reports 1–20

Chapter 2 Analyzing Your Source Code

About *Understand for FORTRAN* Projects 2–2
The *Understand for FORTRAN* Project Database. 2–2
Creating a New Project 2–3
Specifying the FORTRAN Version. 2–3
Adding Source Files to a Project 2–4
Filename Changes within a Project 2–6
Right-click Menus 2–6
Project Configuration Settings. 2–8
Options Tab 2–8
Include Dirs Tab 2–9
Macros Tab. 2–10
Format Tab. 2–11
Include Replace Tab 2–12
Extensions Tab. 2–13
Display Tab. 2–14
Saving the Project Configuration 2–15
Analyzing the Code 2–16

Quick Project Updating 2–17
Output Reports 2–18

Chapter 3 Exploring Your Code

PLEASE RIGHT CLICK 3–2
Various Windows Explained... 3–3
Document Area 3–3
Project Area 3–3
Filter Area 3–4
Information Browser 3–4
Locator Window 3–5
Find in Files 3–6
Hierarchy Browser 3–7
Declaration Browser 3–7
Information Browser 3–9
Drilling Down A Relationship 3–10
Drilling Down Efficiently 3–11
Viewing Metrics 3–11
Saving and Printing Information Browser Info 3–12
Visiting Source Code 3–13
Right Click Menu Source Visiting 3–14
One Click Visiting of References 3–14
Entity History 3–14
Graphical View Browsers 3–15
General Rules for Using the Graphical Browsers 3–15
Reuse Checkbox 3–16
Graphical Browser History 3–17
Graphic Hierarchical Views Available 3–18
Graphic Declaration Views Available 3–19
Controlling Graphics Layout 3–21
Scale Menu 3–21
Text Menu 3–22
Intrinsic Menu 3–22
Name Menu 3–23
Layout Menu 3–23
Level Menu 3–24
Unresolved Menu 3–24
Parameters Menu 3–25
Called by Menu 3–26
Invocations Menu 3–26
Duplicate Subtrees Menu 3–26
Printing Graphical and Source Views 3–27
Shrink to Fit Printing 3–27
Poster Printing 3–27

Configuring Poster Printouts 3–28
Printing on UNIX Machines 3–28
Source File Printing 3–30

Chapter 4 Editing Your Source Code

Source Code Editor 4–2
Status Line 4–2
Status Icons 4–3
Colorizing Source Views 4–4
String Searching 4–5
String and Replace 4–5
Go To Line... 4–6
Line Numbers 4–6
Selecting and Copy Text 4–6
Key Mappings 4–7
Other Features 4–10
Bracket Matching 4–10
Printing Source Files 4–10
File Display Options 4–11

Chapter 5 Searching Your Source

Searching - an Overview 5–2
Locator Window - Find or Browse Entities 5–3
Right-Click Menus 5–4
Column Headers 5–4
Sorting Columns 5–4
Resizing Columns 5–5
Long versus Short Names 5–5
Hiding and Re-ordering Columns 5–5
Filtering 5–5
Filter By Selection 5–6
Filter Dialog Window 5–7
Wildcards Without Regular Expressions 5–8
Regular Expressions 5–8
Removing Filters 5–9
Using Find in Files 5–10
Controlling Searches 5–11
Search Results 5–13

Chapter 6 Text and HTML Reports

An Overview of Report Categories 6–2
Augment with the PERL or C API 6–3
Output Formats 6–3
Report File Naming Conventions 6–3

Searchable Entity Index 6–5
Generating Reports from the Command Line 6–5
Cross Reference Reports 6–6
Data Dictionary Report 6–6
Program Unit Cross Reference Report 6–7
Object Cross Reference Report 6–7
Type Cross Reference Report 6–8
Structure Reports 6–9
Declaration Tree 6–9
Invocation Tree Report 6–10
Simple Invocation Tree Report 6–11
Include Report 6–11
Quality Reports 6–12
Program Unit Complexity Report 6–12
FORTRAN Extension Usage Report 6–13
Implicitly Declared Objects Report 6–14
Unused Object Report 6–14
Unused Type Report 6–15
. . . . . .Unused Program Unit Report 6–15
Metrics Reports 6–16
Three Ways to Get Metrics Information 6–16
What Metrics are Available? 6–16
Project Metrics Report 6–17
Program Unit Metrics Report 6–17
File Metrics Report 6–18
Exporting Project Metrics Info 6–18

Chapter 7 Using External Editors and Other Tools

Using an External Editor 7–2
Configuring Tools 7–4
Adding Tools to the Right-Click Menus 7–6
Adding Tools to the Tools Menu 7–8
Adding Tools to the Toolbar 7–9
Running External Commands 7–10

Chapter 8 Server Mode: Controlling from Other Programs

Understand Client 8–2
Getting the Latest Options with -help 8–2
Specifying the Project Database 8–3
Specifying the Entity 8–3
Opening a File 8–4
Action Commands 8–5
Example Client Commands 8–6

Communication Method 8–8
Editor Synchronization Example 8–9

Chapter 9 Command Line Processing

Using undftn 9–2
Permanent Vs. One Time Options 9–2
Getting Help on Command Line Options 9–3
Command Line Options 9–3
Creating a New Project 9–5
Adding Files to a Project 9–5
Creating a Database and Adding Sources in One Step . . 9–6
Creating a List of Files 9–6
Analyzing a Project 9–6
Keeping a Database Updated 9–7
Generating Reports 9–8
General repftn Options 9–8
Report Options 9–9
Report Format Options 9–10
Extensions Used by the -separate Option 9–10
Creating All Reports in an HTML Directory 9–10
Turning Off Some Reports 9–11
Generating All Text Reports 9–11
Generating All HTML Reports 9–11
Generating Metrics Reports 9–11

Appendix A Graphical Notation

Chapter 1 Introduction

This chapter is intended to help you put *Understand for FORTRAN* to good use quickly and easily. The chapter describes the basic views and modes of operation in *Understand for FORTRAN*.

This manual assumes a moderate understanding of the FORTRAN 77, FORTRAN 90, or FORTRAN 95 programming language.

This chapter contains the following sections:

Section	Page
What is Understand for FORTRAN?	1–2
For Those Who Don't Like to Read Manuals	1–3
Understand's Parts and Terminology	1–4
Right-Click Menus Are Everywhere	1–6
Quickly Find Things in Your Source	1–8
Source and Graphical Views	1–11
The Information Browser Shows It All	1–12
Source Editor Provides Information Wherever You Look	1–14
Hierarchy Views Show Multi-Level Relationships	1–15
Structure Views Quickly Show Structure and Relations	1–17
ASCII and HTML Reports	1–20

What is *Understand for FORTRAN*?

*Understand for FORTRAN*is a source code analyzer; it helps programmers understand their FORTRAN software projects.

Understand for FORTRAN analyzes your FORTRAN software to create a repository of the relations and structures contained within it. The repository is then used to learn about the source code.

Understand for FORTRAN helps you quickly answer questions such as:

- What is this entity?
- Where is it changed?
- Where is it referenced?
- Who depends on it?
- What does it depend on?

Understand for FORTRAN answers these questions through interactive entity specific cross reference reports and graphical diagrams quickly showing the relevant information about a given entity.

FORTRAN Versions Supported

Understand for FORTRAN supports FORTRAN 77, FORTRAN 90, and FORTRAN 95, in both free and fixed format. Extensions supported include Harris FORTRAN and DEC FORTRAN.

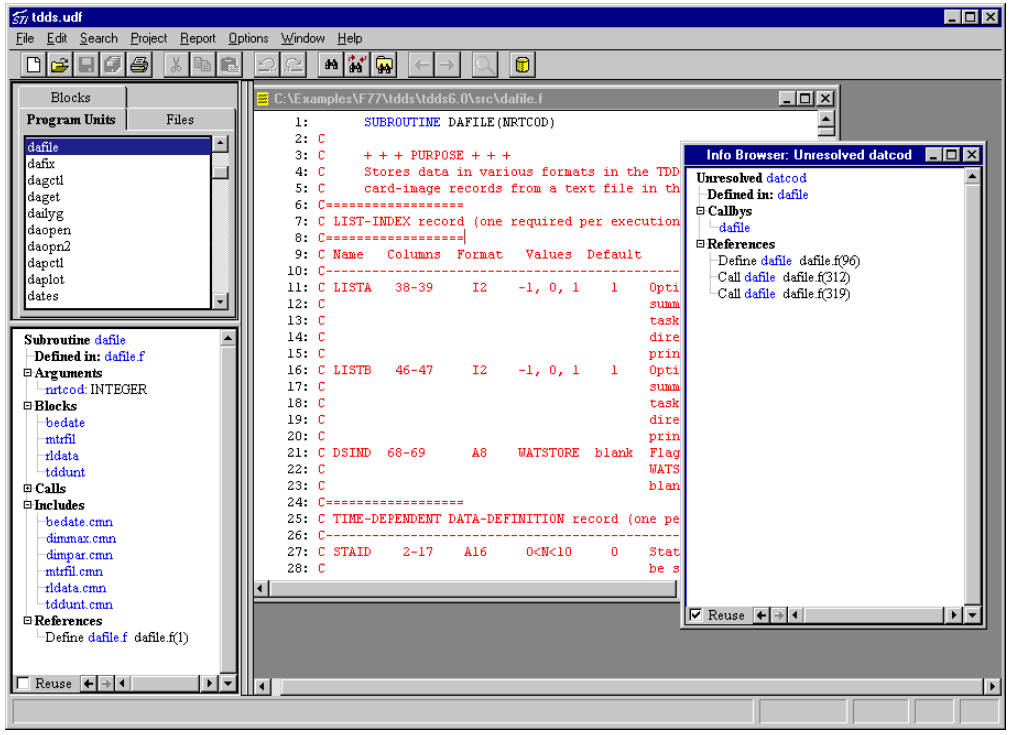
We often expand *Understand for FORTRAN* to support common compiler extensions. If you find that the compiler extensions you are using are not currently supported, contact us at support@scitools.com.

Note: Scientific Toolworks also offers similar tools for C/C++, Java, and Ada.

Use Server Mode to Integrate with Your IDE

Understand for FORTRAN is designed to be used both for standalone browsing/discovery as well as browsing that is controlled from another application.

You can control *Understand for FORTRAN* from any editor or program from which you can launch a program called "understand_f". The client accepts action commands for an entity name, and optional file, line, column specifiers, and more.



Refer to *Understand Client* on page 8–2 for details on using **understand_f** in server mode.

For Those Who Don't Like to Read Manuals

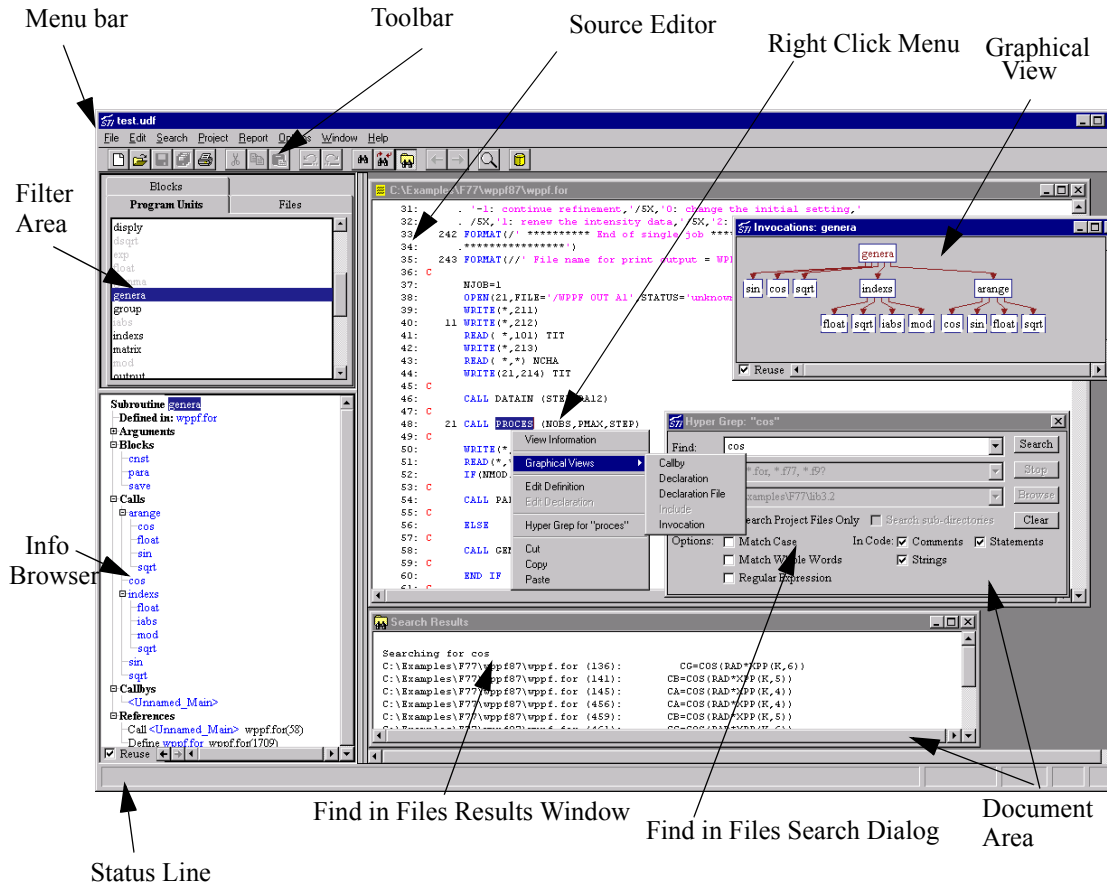
If you are like many engineers at Scientific Toolworks, you like to just dig in and get going with software. We encourage that, or at least we are pragmatic enough to know you will do it anyway! So feel free to use this manual as a safety net, or to find the less obvious features. However, before you depart the manual, read the remaining sections of this chapter for tips on effectively utilizing what *Understand* has to offer.

Understand's Parts and Terminology

Before proceeding to the rest of the manual please take a moment to familiarize yourself with *Understand for FORTRAN*'s parts and terminology. Doing so will make reading the manual more helpful and also put you on the same sheet of music as the technical support team should you have to email or call.

Parts

The following figure shows the main parts of the *Understand for FORTRAN* graphical user interface (GUI):



Terminology

Database: The database is where the results of the source code parsing, as well as project settings, are stored. By default this is a project's ".udf" file.

Entity: An *Understand for FORTRAN* "entity" is anything it has information about. In practice this means anything declared or used in your source code and the files that contain the project. Subroutines, variables, and source files are all examples of entities.

Project: The set of source code you have analyzed and the settings and parameters chosen. A "project file" contains the list of source files and the project settings.

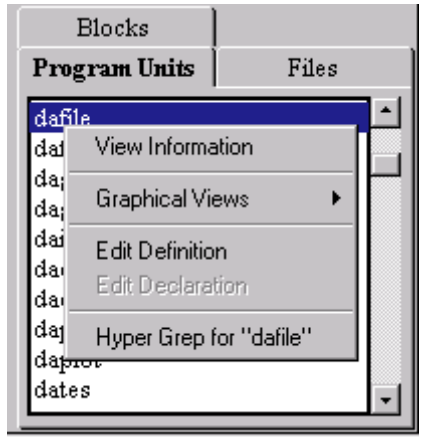
Relationship: A particular way that entities relate to one another. The names of relationships come from the syntax and semantics of FORTRAN. For instance, subroutine entities can have "Call" relationships or "CalledBy" relationships.

Right-Click Menus Are Everywhere

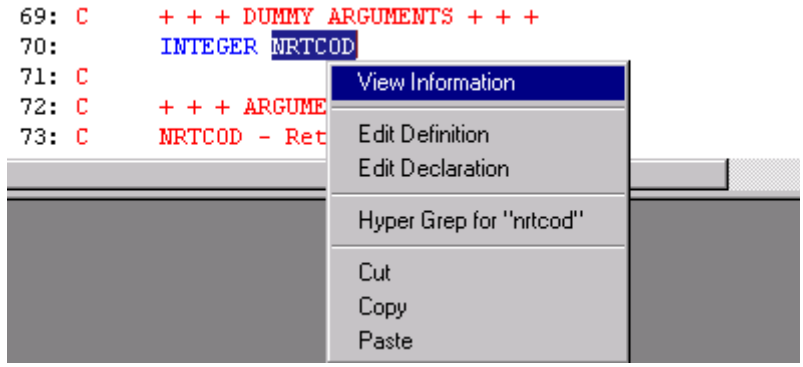
Right-clicking gets you a long way in *Understand for FORTRAN*; almost everywhere you point you can learn more and do more by bringing up menus with your right mouse button.

Tip: Hold down the CTRL key while right-clicking to create new windows rather than re-using existing ones.

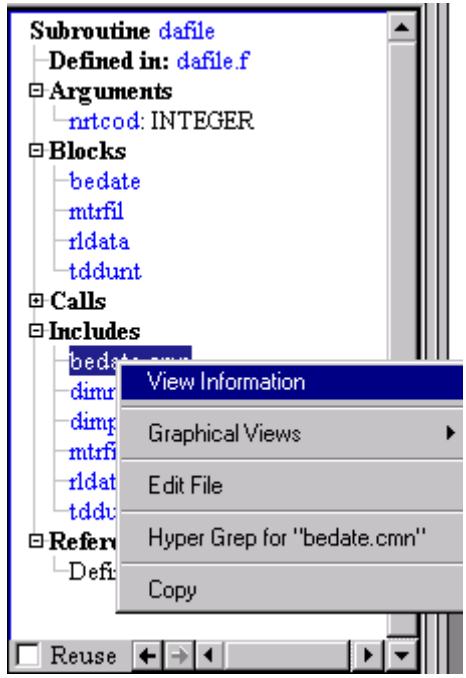
Example: Right-click on an entity in the filter area:



Example: Right-click on an entity in the Source Editor:



Example: Right-click on an entity in the Info Browser.



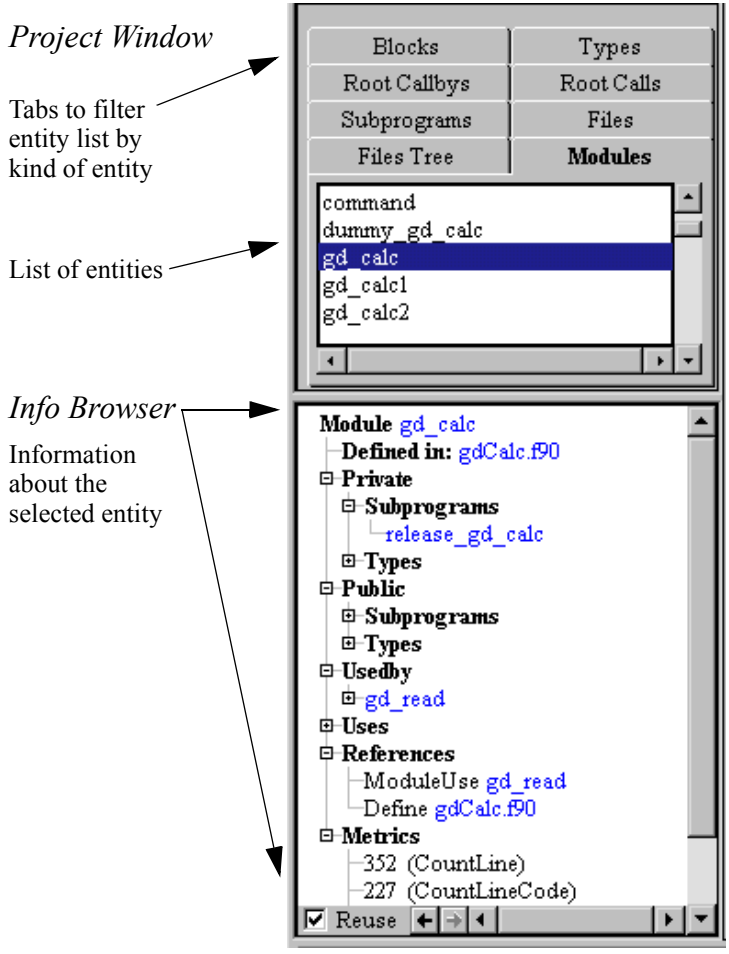
Remember to right-click, anytime, anywhere, on any entity to get more information about that entity.

Quickly Find Things in Your Source

Understand for FORTRAN provides a number of ways to quickly locate items of interest in your source code. The windows you use include the Project Window, the Locator Window, and the Find in Files dialog.

Project Window

The *Project Window* helps you quickly find things in your code by separating that database into lists of **Files**, **Subprograms, Modules, Blocks** and **Types**, as well as providing a directory browser of the source to your project. It also includes an *Information Browser* that automatically shows all known information about the selected entity.

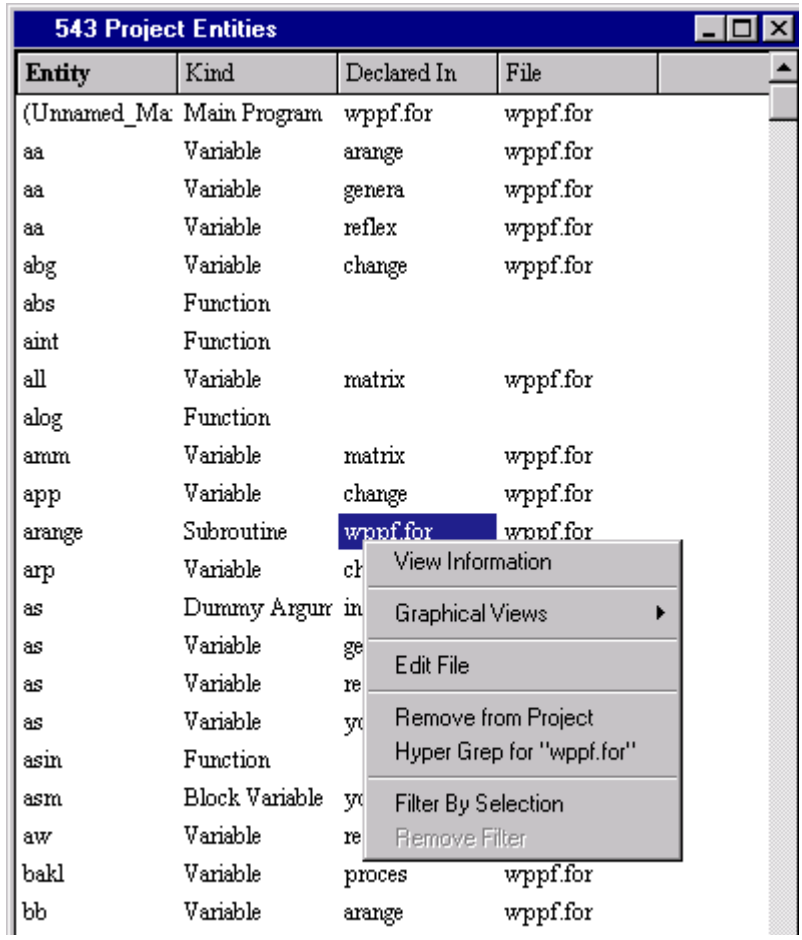


Locator Window

The *Project Window* provides a quick way to find major items that were declared and used in your project. However, some items such as local parameters, variables, and unresolved variables (used but not declared in the processed source) are filtered from the *Project Window*. To search or browse the entire database for your project, use the *Locator Window*.

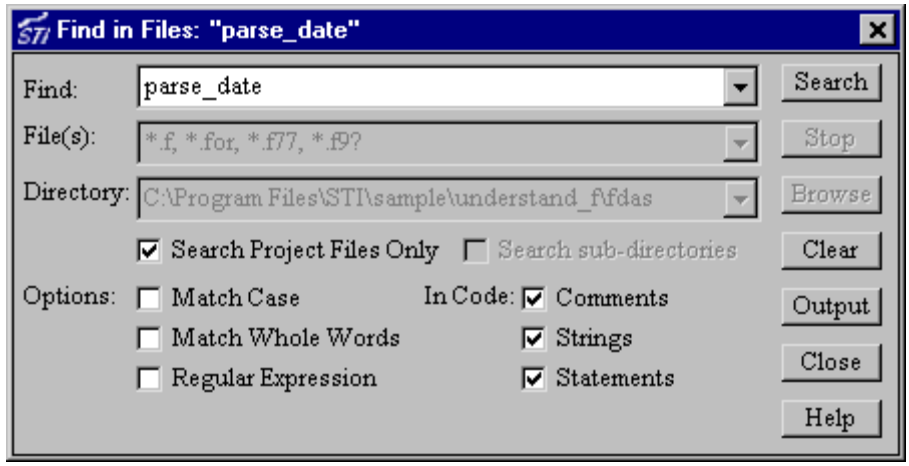
Browse all the database entities in the *Locator Window* by selecting **Search->Browse Entities** or search for entities matching a particular text or regex string using **Search->Locate Entities**. Entities listed within the *Locator Window* can also be sorted or filtered further. For more details, refer to *Locator Window - Find or Browse Entities* on page 5–3.

As in any other window, the right-click menu is also active as shown below.



Using Find in Files to Explore

Similar to the UNIX command *grep*, you may search a selection of files for the occurrence of a string. The **Find in Files** function is available from all windows. Select **Find in Files** either from the **Search** pull-down on the menu bar or from a right click menu.



When you click the **Search** button, a list of all occurrences matching the specified string or regular expression is displayed in the *Search Results* window. Double click on any of the search results to display the *Source View* where the string occurs.

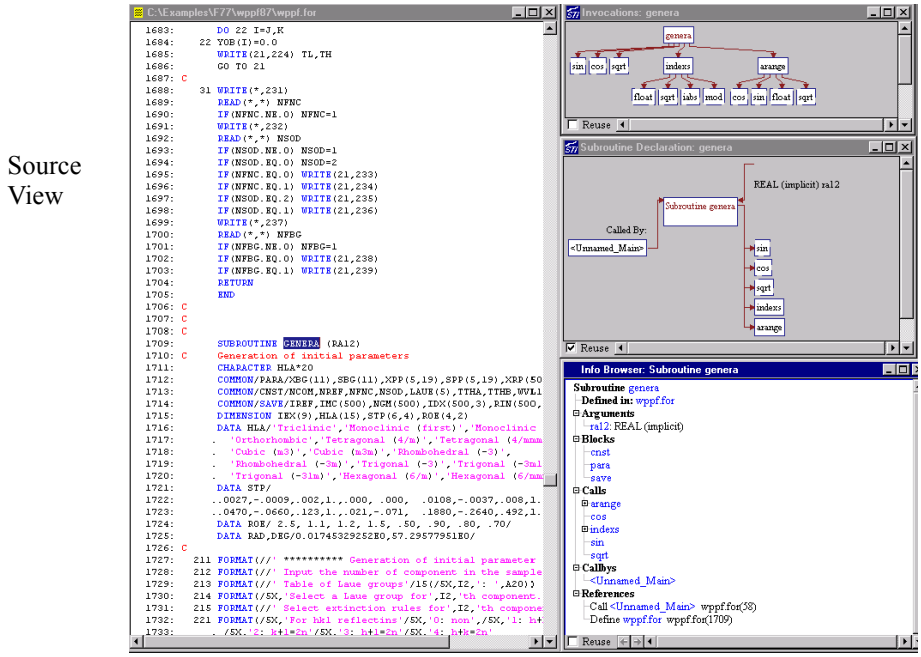
Refer to *Using Find in Files* on page 5–10 for more information on using Find in Files.

Source and Graphical Views

Understand for FORTRAN analyzes your FORTRAN software and creates a database containing information about the entities and the relations between entities. The database can then be browsed using various “view” windows. The views are divided into four kinds:

- **Information** views show all that *Understand for FORTRAN* knows about a given entity.
- **Hierarchy** views show relations between entities. Each view follows a relation (for instance “Calls”) from the starting entity (that you inquired about) through its children and successors.
- **Structure** views quickly show the structure of any entity that adds to the structure of your software (for instance a compilation unit, package, function, procedure, task).
- **Source** views show your source in a hyperlinked fashion. They provide information when looking at the source code, or track source code when looking at other information.

Examples of each type are shown in the following figure:

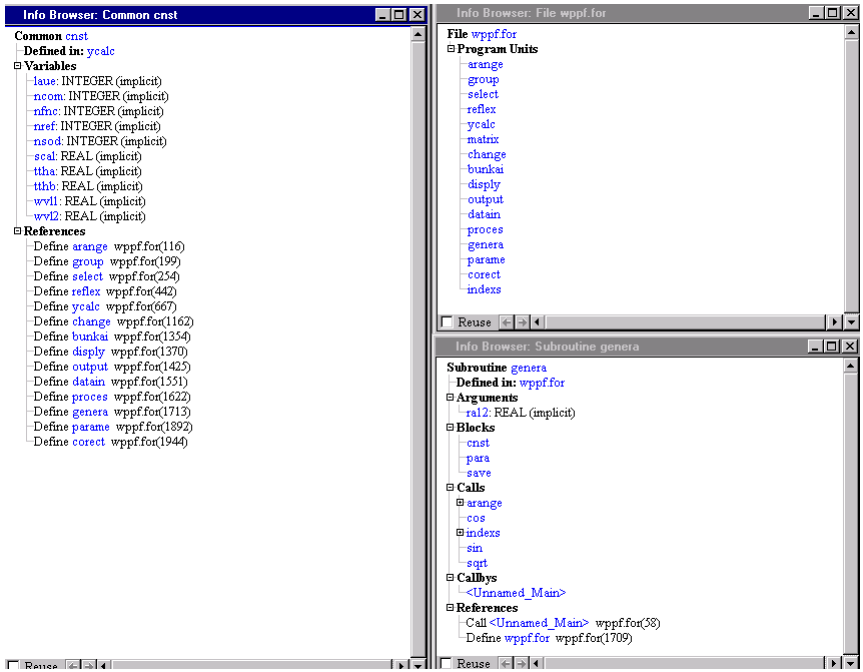


The Information Browser Shows It All

Just about everything *Understand for FORTRAN* knows about your code is shown in the Information Browser (IB). The IB is used for every type of entity available.

The IB configures itself to show the different kinds of information *Understand for FORTRAN* knows about entities such as source files, include files, functions, subroutines, blocks, types, and variables. Information that is hierarchical in nature can be drilled down either manually or with one click to expand the entire tree.

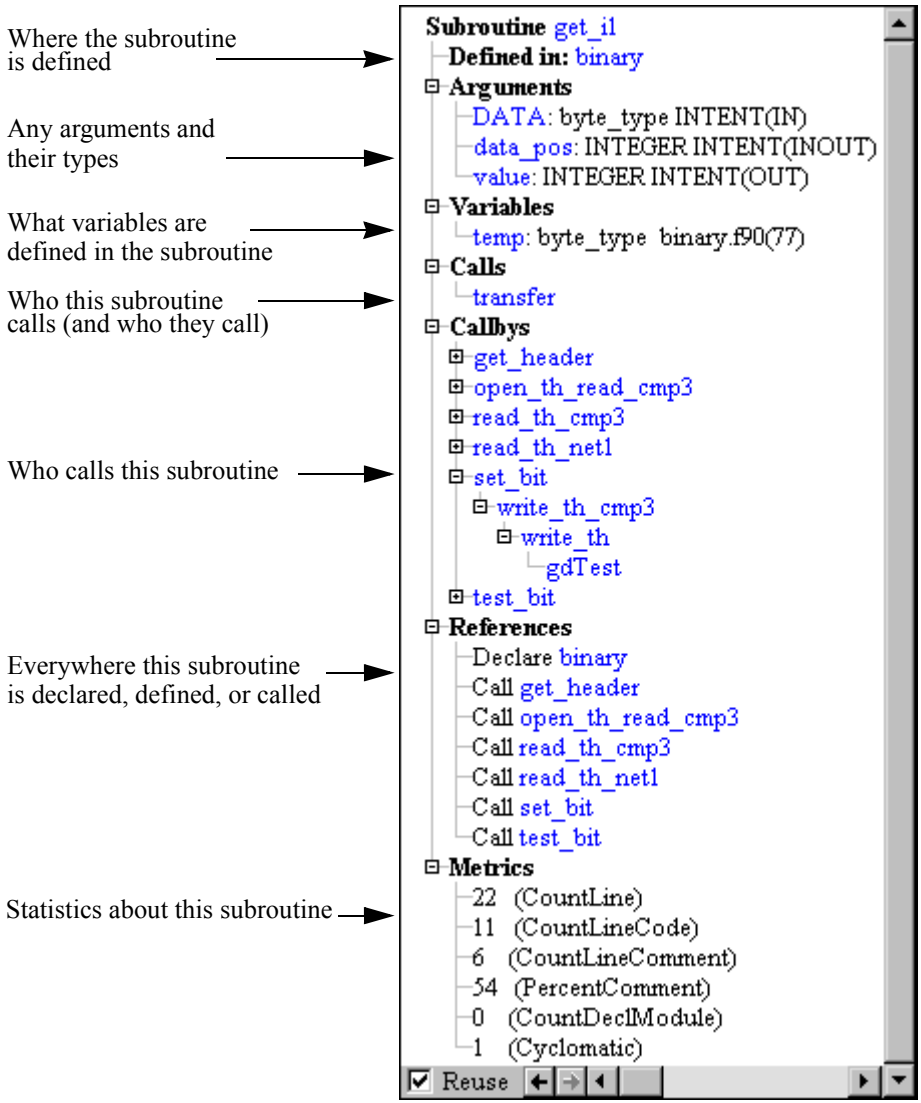
Below are three different *Information Browser* windows, one each for a block, file, and subroutine:



Note that the Information Browser shows different things depending on the type of entity selected.

A detailed view of the Information Browser for a subroutine follows:

Information Browser showing information about a subroutine.

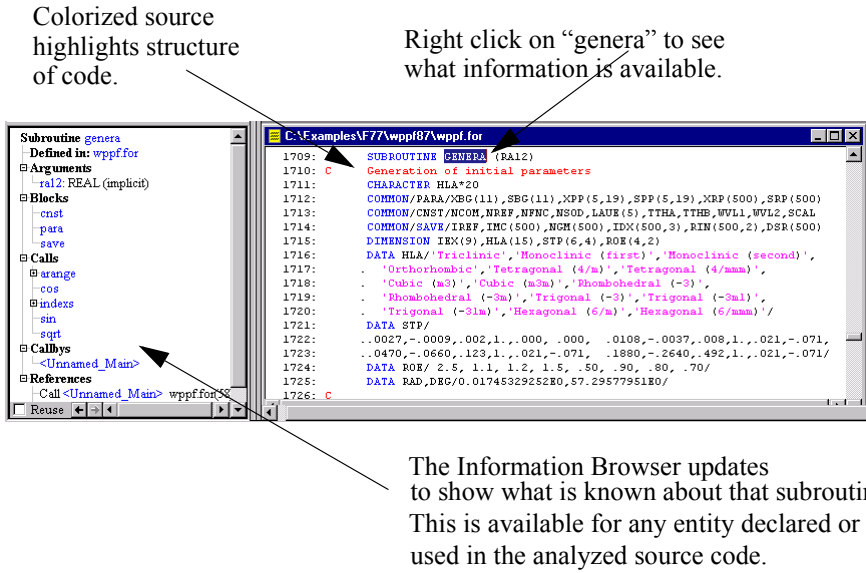


Source Editor Provides Information Wherever You Look

Understand for FORTRAN has a source editor that not only lets you edit your source code, it colorizes the source code and also tells you about the code you are editing.

Source can be visited by double-clicking almost anywhere else in the tool. You can move forward or backward through such “visits” by using the **Window->Next** and **Window->Previous** commands.

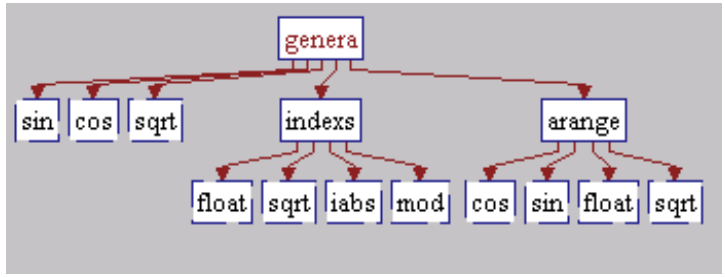
As with any other place in *Understand for FORTRAN*, a right-click menu is available throughout the editor. To learn about something just right-click on it to see what information is available.



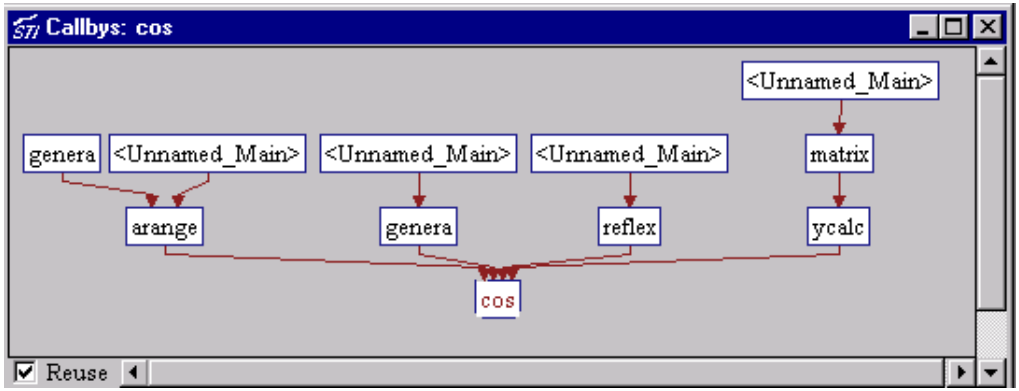
Hierarchy Views Show Multi-Level Relationships

Hierarchy views show the relationships between entities. Here are examples of the types of hierarchy views that *Understand for FORTRAN* offers.

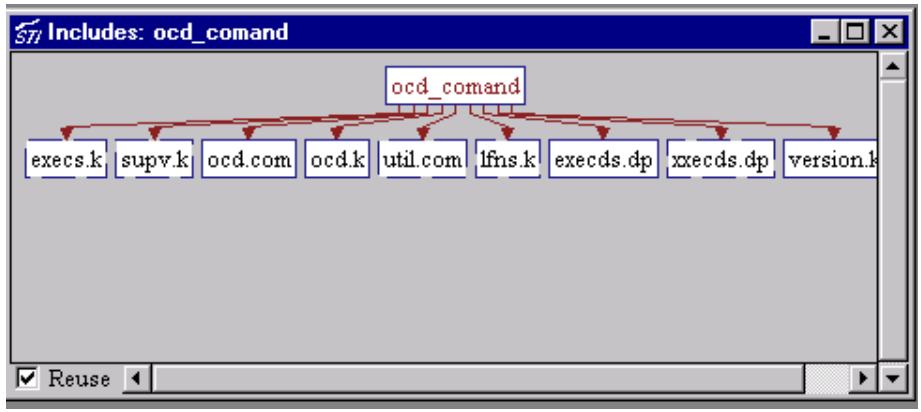
- **Invocation** - Shows the entire chain of invocations emanating from this function. Each line between entities is read as "entity invokes entity". In this example, *genera* invokes *indexs* which invokes *float* (and others).



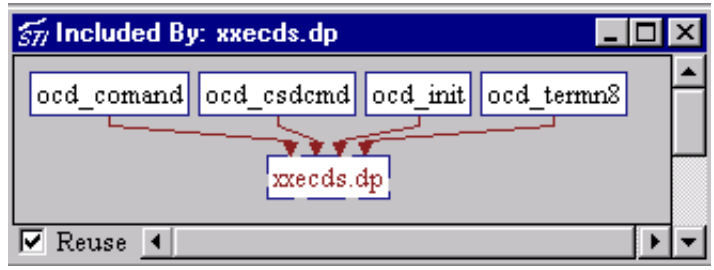
- **Call By** - Shows who calls a function, and who calls each parent. Each line connecting an entity is read as "entity is called by entity". In this example, *cos* is called by *ycalc,* which is called by *matrix* which is called by <*Unnamed Main*>. Note that this view is read from the bottom up or right to left.



- **Include** - Shows the include hierarchy of a function. A connecting line is read as “function includes filename.” In this example, *ocd_command* includes *execs.k, supv.k*, and *others*.



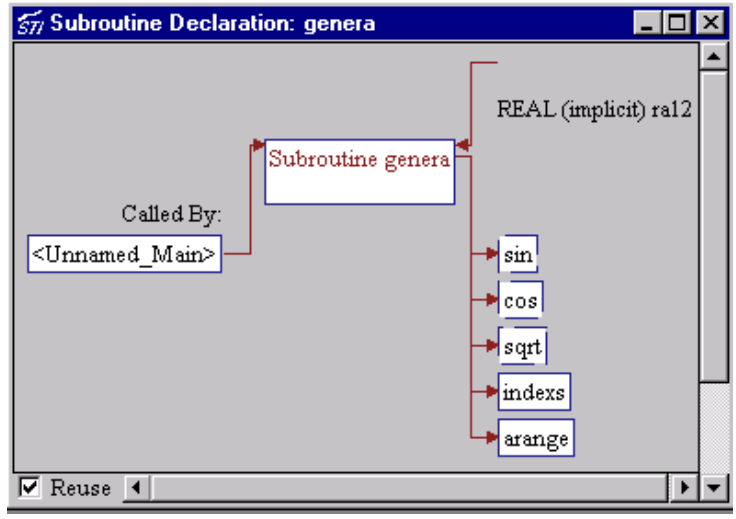
- **Include By** - Shows what functions include a given file. A connecting line is read as “filename is included by function”. In this example, *xxecds.dp* is included by *ocd_command*, *ocd_csdcmd, ocd_init,* and *ocd_termn8*. Note that this view is read from the bottom up or right to left.



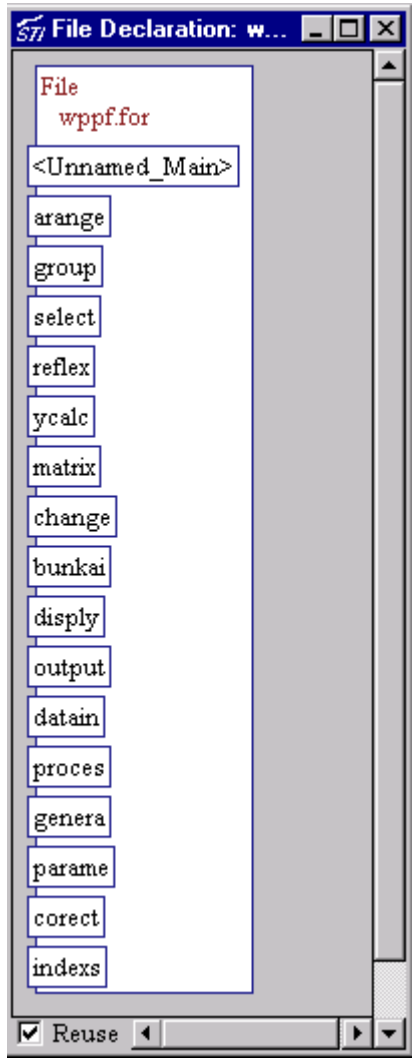
Structure Views Quickly Show Structure and Relations

Understand for FORTRAN structure views are designed to present essential information about an entity in a small and concise manner. The structure diagram is derived from the ggraphs presented by Booch and Buhr in their respective books “Software Engineering with Ada” and “System Design in Ada.” Where needed, the symbols and annotations have been extended or altered to represent the structures and relations used in FORTRAN programs.

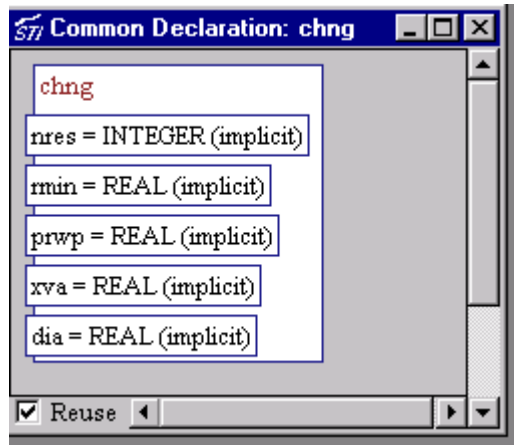
- **Subroutine Declaration** - One place to see all of the parameters, return type, declared entities and types, as well as who this subroutine calls and who uses this subroutine.



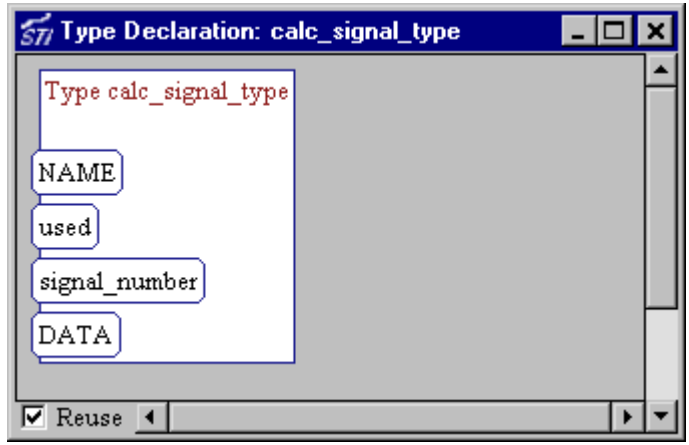
- **File Declaration** - Abstractly shows the subroutines, types, and entities declared in the FORTRAN code file.



- **Common Block Declaration** - Shows variables composing a common block.

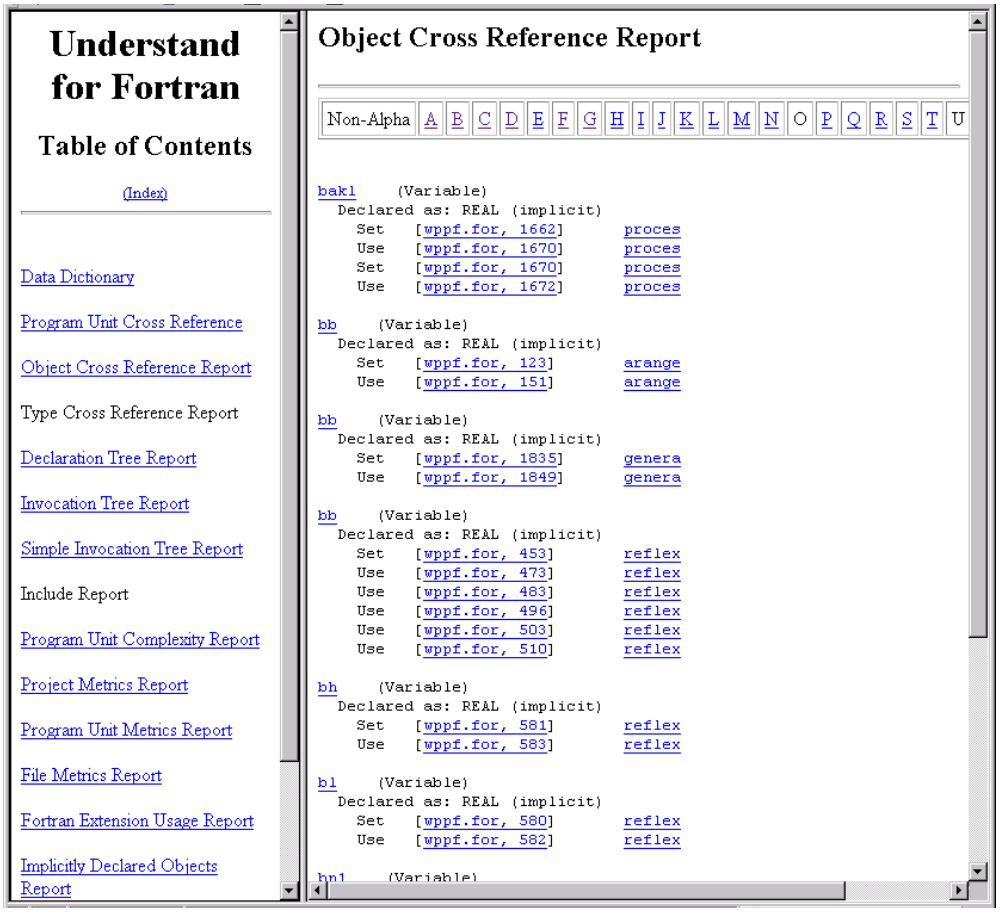


- **Type Declaration** - Shows types and their components.



ASCII and HTML Reports

Views in *Understand for FORTRAN* provide information about individual entities. The reports bundle information about all entities in ASCII or HTML format.



The HTML and ASCII reports also show information not available interactively, such as project metrics and quality reports. The reports are suitable for printing or browsing with a web browser. See *Text and HTML Reports* on page 6–1 for more information.

Chapter 2 Analyzing Your Source Code

This chapter shows how to create new *Understand for FORTRAN* project files, and how to analyze your source code. It also describes how to generate HTML and text reports for your project.

This manual assumes a moderate understanding of the FORTRAN programming language.

This chapter contains the following sections:

Section	Page
About Understand for FORTRAN Projects	2–2
Creating a New Project	2–3
Adding Source Files to a Project	2–4
Project Configuration Settings	2–8
Saving the Project Configuration	2–15
Analyzing the Code	2–16
Output Reports	2–18

About *Understand for FORTRAN* Projects

Understand for FORTRAN is like a compiler, except it creates information, not executable code.

In order for *Understand for FORTRAN* to analyze your source code it needs much of the same information that your compiler needs. Specifically it needs to know:

- What source files to analyze.
- Where to find include files referred to in the source code.

If you developed the program or have been working with it for some time, this information is probably obvious to you. However, if you inherited this source code from another programmer, team, or company, you will probably have to examine the project building files (e.g. makefile) in order to come up with the information needed for accurate parsing of the code.

The easiest way to analyze your code is to use *Understand for FORTRAN*'s GUI to build and parse a project.

You can also use command line tools for batch processing of source files and generation of information reports. Refer to *Command Line Processing* on page 9–1 for details on the command line tools.

The *Understand for FORTRAN* Project Database

The *Understand for FORTRAN* project database is stored in a proprietary binary format. The file format uses a network/object format that is optimized for storing *Understand for FORTRAN* information.

Understand for FORTRAN databases have a file extension of **.udf**.

The project file permits multiple simultaneous read accesses, but it does not (yet) support multi-user write access.

Occasionally, adding a new feature to *Understand for FORTRAN* requires a change to the database format. Such changes are noted in the Change Log. After you install a build that modifies the database format, existing projects are automatically reparsed when you open them in *Understand for FORTRAN*.

Creating a New Project

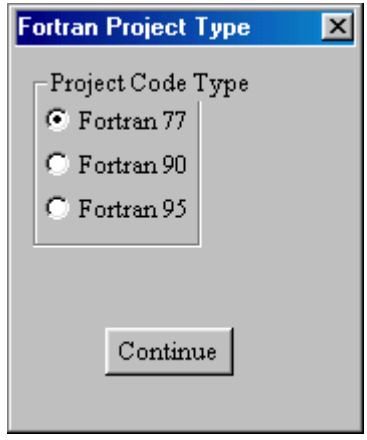
The easiest way to analyze your source is to use *Understand for FORTRAN*'s GUI to create a project and specify what source files to parse. The GUI then parses the code and creates an *Understand for FORTRAN* database that you can browse. This database can be refreshed incrementally from within the GUI, or updated using command line tools.

To create a new project:

1. Select **File->New Project.**
2. Browse to where you wish to store the project database and type the file name for the project. The **.udf** file extension will be added automatically.
3. Click Open to create an empty project database file.

Specifying the FORTRAN Version

In order for *Understand for FORTRAN* to properly analyze your code it needs to know what version of FORTRAN the code is and whether it is free or fixed format.



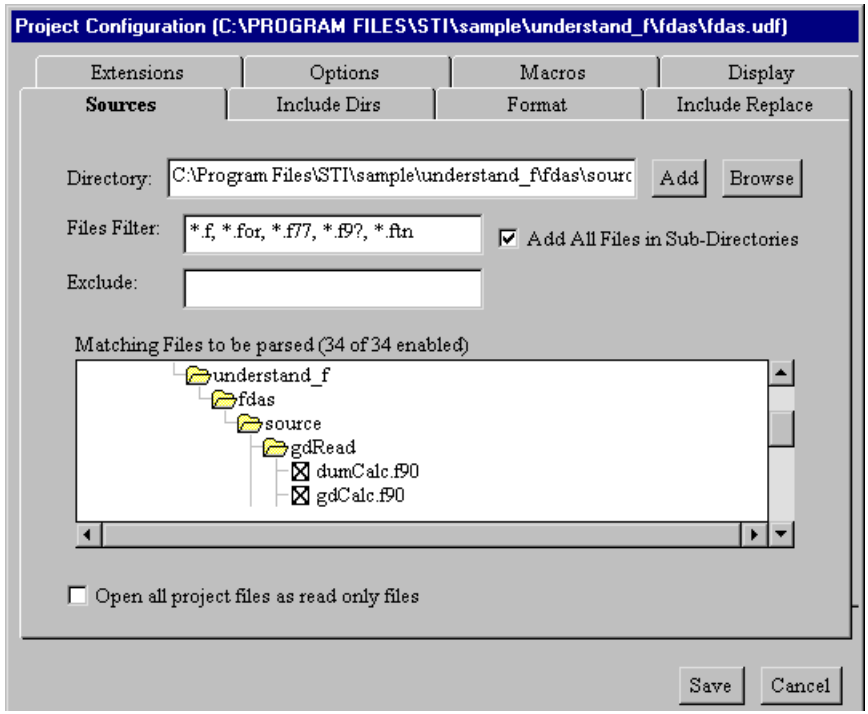
The choices for language are FORTRAN 77, FORTRAN 90, or FORTRAN 95. If you have a mix of code then choose the newest language variant (i.e. F77 and F95 code - choose F95).

Tip: The code type cannot be changed later (other than by creating a new project), so choose wisely.

Adding Source Files to a Project

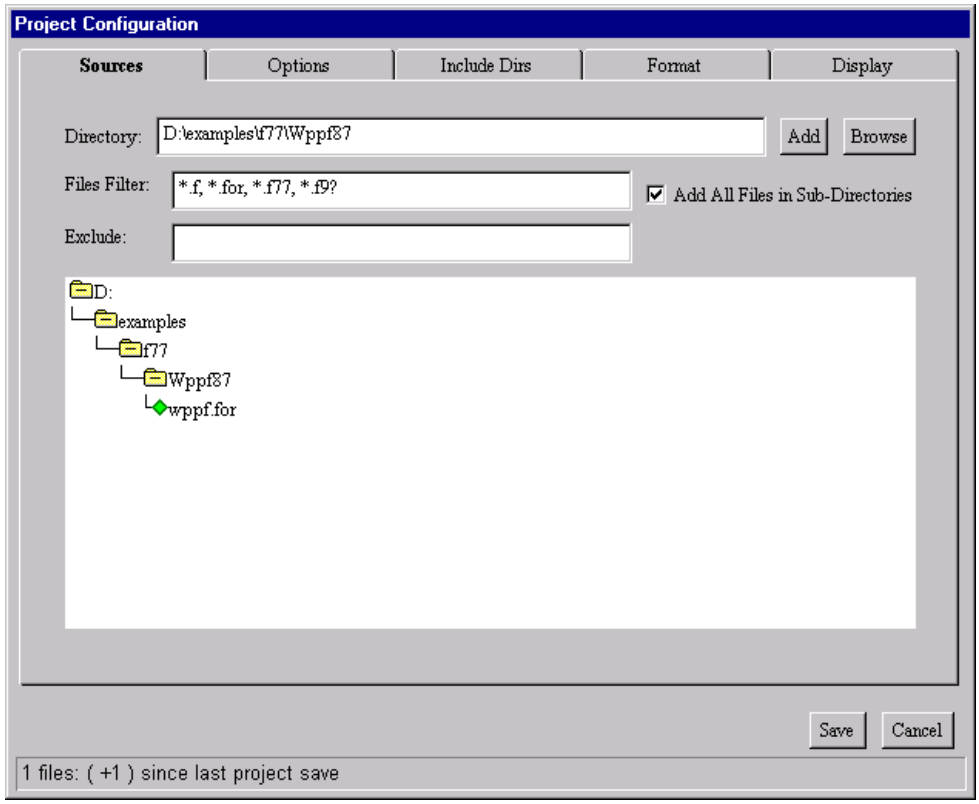
When you create a new project, the **Project Configuration** dialog pops up automatically with the **Sources** tab shown. Use it to add entire directories of source code with one click of the button. You can also turn parsing on/off for specific files or entire directories.

You can open this dialog for an existing project at any time by choosing the **Project->Configure** menu item.



To add source files to the project:

1. Type the full directory path, or click **Browse** and locate the directory that contains the source files. Click the **Open** button to load the directory path into the Directory field.
2. Modify the file filter to match your source files. By default, all .f, .for, .f77, .f9?, and .ftn files will be added to the project. Simple wild cards (* and ?) are supported in the file filter. So “f9?” matches “f95” and “f90”.
3. If you want to exclude certain files, you can enter a filter for that purpose. For example, temp*.* excludes all files that begin with “temp”.



4 To select and add multiple subdirectories to a project configuration, check the **Add All Files in Sub Directories** box. This causes all source files matching the filter in all subdirectories of the specified path to be added to the project.

5 On UNIX, you can choose whether symbolic links should be followed when adding files.

6 After you have specified the path and file filter, click the **Add** button or press Enter to add the source files in that directory to the project.

Tip: You may browse and add files from multiple directory trees.

7 If you do not want to make any changes to the source code with *Understand for FORTRAN*, check the **Open all project files as read only files** box.

If you are using Microsoft Windows, you may drag and drop a folder, a file, or a selection of files, from another window into the Project Configuration dialog to add it to the project. To drag and drop a folder and all its subdirectories into the project, be sure the

Add All Files in Sub Directories box is checked. When dragging an individual file into the project, that file will be added to the project whether it matches the file filter or not.

Once the directory paths and files are specified and added to the project, the directory tree and project files are shown.

Filename Changes within a Project

If a file in the project is deleted, moved, or re-named, the file display in the **Sources** tab of the Project Configuration dialog (open by choosing **Project->Configure**) shows the missing project file with a red "X". If the file was renamed and not yet added to the project, the new file will be displayed but will not be included in the project definition until explicitly added.

The following icons are used to identify files in the Sources tab:

- ☐ File matches the filter but is not selected for parsing.
- ☒ File exists and is selected for parsing as part of the project source.
- ☒ File is selected for parsing, but is not found at the specified location.
- File is part of an MSVC project file (cannot be removed except on MSVC tab).
- File is part of an MSVC project file, but is not found at the specified location.

Click a box to toggle the source file in or out of the project configuration. Status information about the project files are displayed in the status bar at the bottom.

Right-click Menus

You can use right-click menus in the **Sources** tab of the Project Configuration dialog. The right-click menu for a file allows you to open the source file in a separate window.

The right-click menu for a folder provides the following options:

- **Close / Open**
  Toggles the selected folder between closed and open. You can also simply left-click on a folder to close or open it.
- **Remove Directory**
  Removes this directory and all the files it contains from the project configuration.
- **Convert to Relative Path / Convert to Absolute Path**
  Toggles a directory between an absolute and relative file path. Relative file paths allow *Understand for FORTRAN* to find source files if you relocate the project directory tree. For example, suppose a project is located at c:\myProject and the source files

are in c:\myProject\bin\src. With relative paths, you can move the project to h:\projects\abc and source files to h:\projects\abc\bin\src. Projects can contain a mix of absolute and relative paths. Projects need to be reanalyzed after switching between relative and absolute directory references.

- **Select -> Add All**
  Changes the status of all files to "selected" (a box with an X).
- **Select -> Clear All**
  Changes the status of all files to "unselected" (an empty box).

Tip: To add just a few source files from a large directory, first add the entire directory to the project. Then use the right-click menu to remove all files from the project, and manually select the required files to add them back to the project.

Project Configuration Settings

The tabs of the Project Configuration dialog allow you to specify various project settings to be used during analysis. You can open this dialog by choosing the **Project->Configure** menu item.

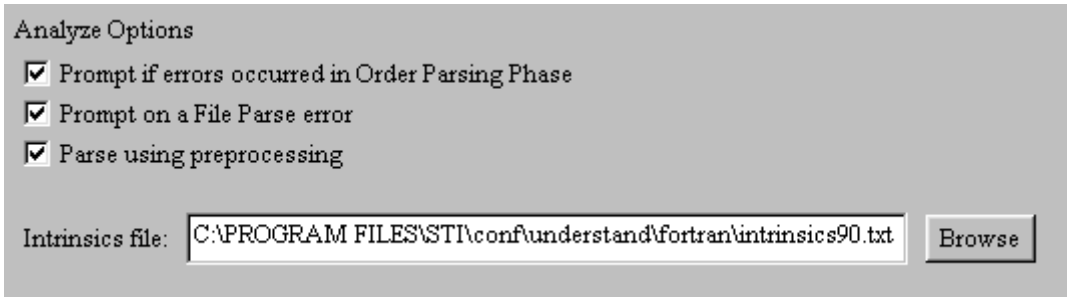
The Project Configuration dialog contains the following tabs:

- **Sources** tab - Specify the source files to be analyzed. For details, see *Adding Source Files to a Project* on page 2–4.
- **Options** tab - Specify additional information to customize the analysis, including information about the source code, how to handle errors found during parsing, and the intrinsics file. For details, see *Options Tab* on page 2–8.
- **Include Dirs** tab - Specify the include paths to search. Any include files that you also want to be analyzed must also be specified in the **Sources**. For details, see *Include Dirs Tab* on page 2–9.
- **Macros** tab - Define macros to be used when analyzing the project. For details, see *Macros Tab* on page 2–10.
- **Format** tab - Specify the file format used, either free or fixed format and the column position to truncate if fixed format. For details, see *Macros Tab* on page 2–10.
- **Include Replace** tab - Specify any path locations to substitute in the project. For details, see *Include Replace Tab* on page 2–12.
- **Extensions** tab - Add support for various FORTRAN extensions. For details, see *Extensions Tab* on page 2–13.
- **Display** tab - Specify how to display entity names. For details, see *Display Tab* on page 2–14.

Options Tab

The **Options** tab in the Project Configuration dialog (which you open with **Project->Configure**) provides a variety of analysis options, including information about the source code, how to handle errors found during parsing, and the intrinsics file.

The **Options** tab contains the following fields:



- **Prompt if errors occurred in Order Parsing Phase**
  Prompt on a File Parse Error
  By default, parsing errors cause a prompt asking how to handle that error. When prompted during analysis, you may choose to ignore that error or all future errors. Turn this option off to disable this prompting feature. If you turned it off during analysis, but later want to turn error prompting back on, check it here.
- **Parse Using Preprocessing**
  Use this option to disable or enable preprocessor support.
- **Intrinsics file**
  Type or browse for a file that contains intrinsic functions you want to be parsed.

Include Dirs Tab

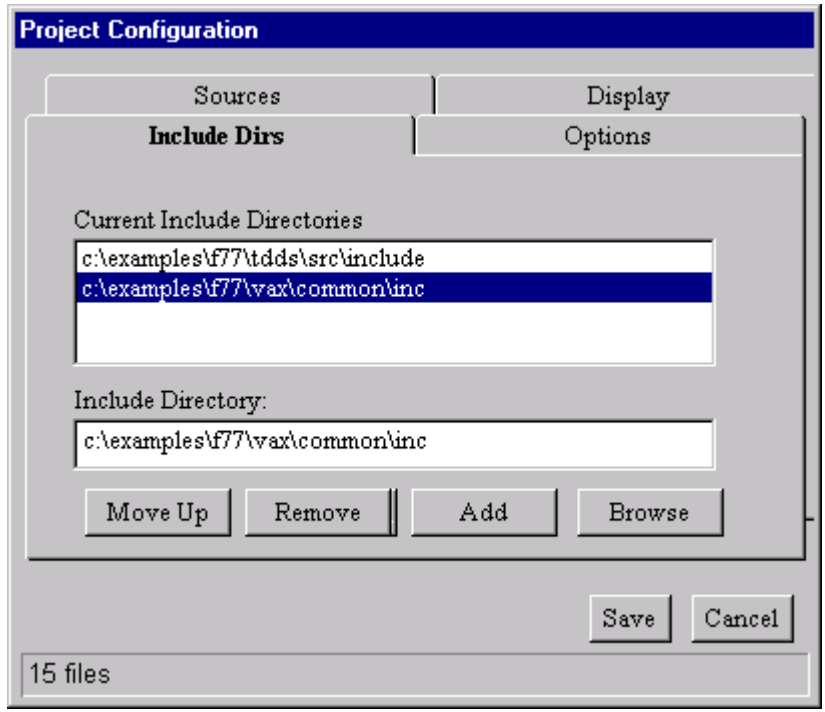
The **Include Dirs** tab in the Project Configuration dialog (which you open with **Project->Configure**) allows you to specify include paths. You can specify multiple directory paths to search for include files used in the project.

Include paths are not recursively searched; that is, any subdirectories will not be searched for include files unless that subdirectory is explicitly specified in the list of include directories.

To add a directory, click **Browse**, select the directory, and click **OK**. Then, click **Add** to add the selected directory to the list.

During analysis, the include directories will be searched in the order that they appear in the dialog. You can click **Move Up** or **Move Down** to change the order in which directories will be searched.

The following figure shows the **Include Dirs** tab.



Macros Tab

C source code is often sprinkled with pre-processor directives providing instructions and options to the C compiler. Directives such as the following affect what the software does and how it should be parsed:

```
#define INSTRUMENT_CODE
#ifdef INSTRUMENT_CODE
... statements ...
#endif
```

The **Macros** tab in the Project Configuration dialog (which you open with **Project->Configure**) allows you to add support for such C preprocessor directives in fixed-format FORTRAN code. The #if, #ifdef, and #else directives are supported.

For *Understand for FORTRAN* to successfully analyze your software it needs to know what macro definitions should be set.

To define macros, select the **Macros** tab of the *Project Configuration* dialog.

The Macros tab lists macros and their optional definitions. Each macro may be edited or deleted. To define a preprocessor value in the **Macros** tab, type the macro and any value for the macro and click **Add**.

Note that a macro must have a name, but that the definition is optional. Macros that are defined but have no definition value are commonly used in conjunction with *#ifdef* pre-processor statements to see if macros are defined.

To change the definition of an existing macro without changing the name, select the macro, modify the definition, and press **Add**.

To use an existing macro as the basis for a new one, select the macro, edit the definition and the name, and press **Add**. This creates a new macro.

You can set a macro on the undftn command line with the -define name[=value] option. You can turn off all preprocessor handling with the -preprocessor off option.

Format Tab

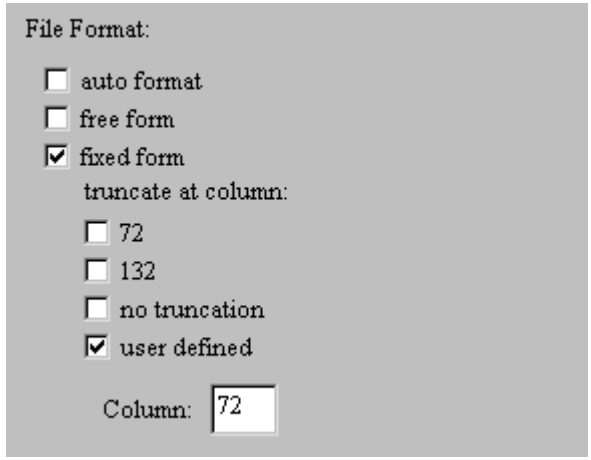
The **Format** tab in the Project Configuration dialog (which you open with **Project->Configure**) allows you to specify the file format used (fixed or free).

Some older FORTRAN variants and all new variants permit *free form* statements, which may cross lines). Fixed form statements are terminated by a line end or column number.

The default is “auto format”, which automatically detects the parsing format (fixed or free) on a file-by-file basis. This allows you to mix free and fixed format. Auto format also determines the correct truncation point for fixed format files.

Choose “fixed” or “free” in the Project Configuration dialog only if all your source files have the same format.

If you choose fixed form, you may also choose what column terminates statements. Common columns 72 and 132 are available or you may specify a column or no truncation.

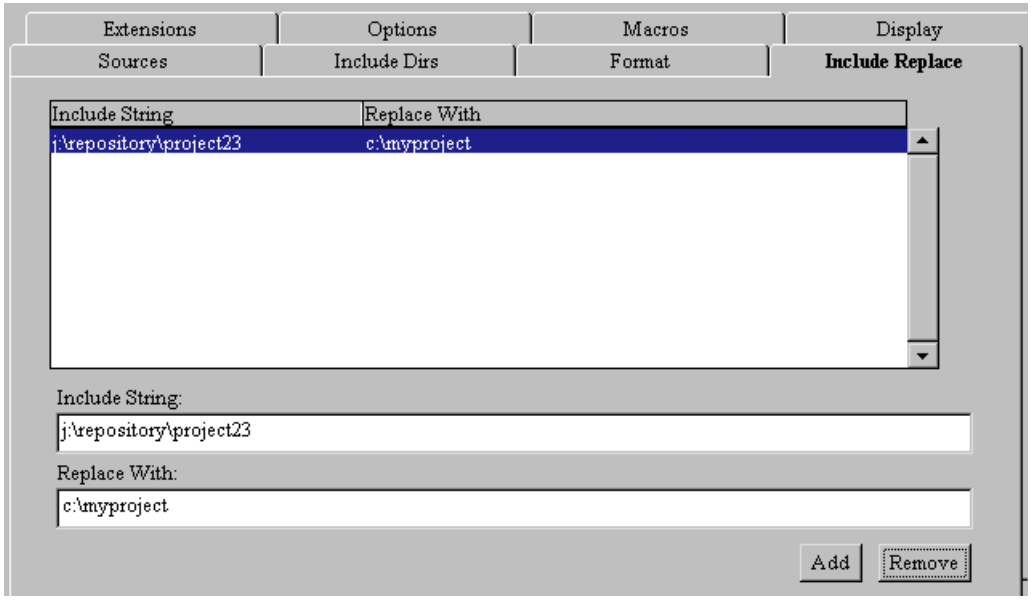


You may use the -list format option on the undftn command line to get a list of what files have been assigned what format.

Include Replace Tab

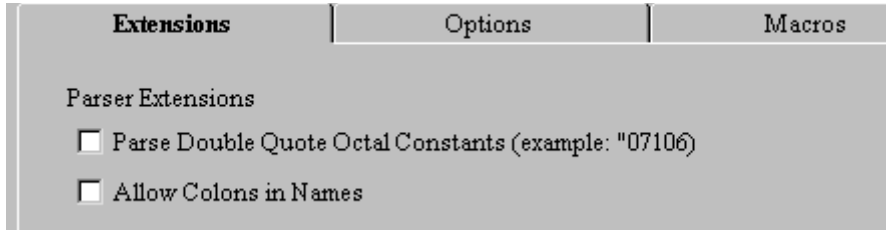
The **Include Replace** tab in the Project Configuration dialog (which you open with **Project->Configure**) allows you to substitute different include paths.

For example, if you have transferred your project to a different location, you can type the old location in the **Include String** field and the new location in the **Replace With** field. Then, click **Add**.



Extensions Tab

The **Extensions** tab in the Project Configuration dialog (which you open with **Project->Configure**) allows you to add support for various FORTRAN extensions.



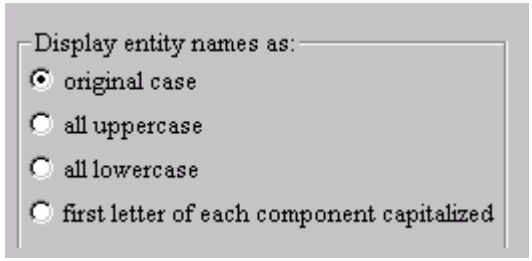
- **Parse Double Quote Octal Constants:** Check this box if a double quote mark (") should be treated as the start of a DEC-style octal constant. For example, "100000. If this box is not checked (the default), a double quote mark begins a string literal. On the undftn command line, you can enable this option by using the -ext_doublequote_octalconstant option.

- **Allow Colons in Names:** Check this box to allow colons (:) used in identifiers in F77 code. Enabling this option could cause problems in F77 code that does not use this extension, so the default is off. On the undftn command line, you can allow colons in names by using the -ext_colon_in_names option.

If used in the source files, these FORTRAN extensions are reported in the FORTRAN Extensions report.

Display Tab

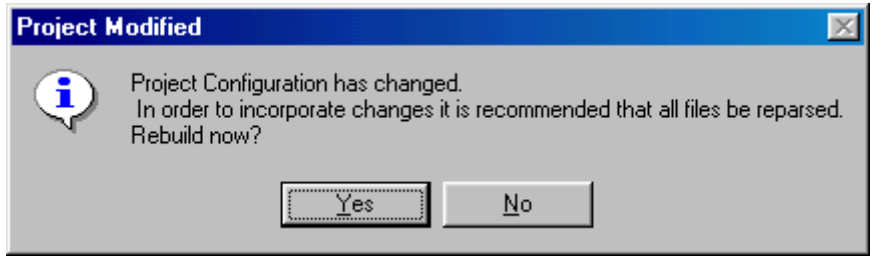
The **Display** tab in the Project Configuration dialog (which you open with **Project->Configure**) controls how entity names are formatted and presented throughout the tool.



Saving the Project Configuration

After you have changed the project configuration, click the **Save** button and the configuration will be saved. **Cancel** closes the dialog without saving your changes.

Whenever the files in the project configuration are modified, including at the time of project creation, a dialog alerting you to the change in configuration appears.



Choose “Yes” and *Understand for FORTRAN* then begins parsing (that is, analyzing) the code.

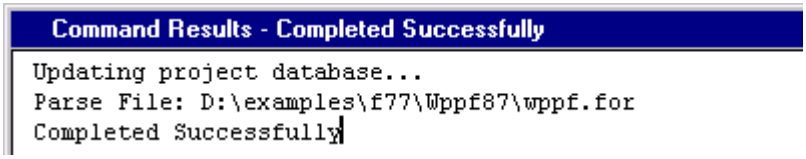
Analyzing the Code

Once you have configured the project, *Understand for FORTRAN* can parse (that is, analyze) the project. During analysis, the source files are examined and a data is stored in the *Understand for FORTRAN* database.

When you save or modify the project configuration, a prompt to analyze the project appears automatically. You can also analyze the project by choosing either **Project->Analyze Changed Files** or **Project->Analyze All Files** from the menu bar.

- **Analyze Changed Files** - This analyzes all files that have been changed and all files that depend on those changed since the last analysis. This is also referred to as "incremental analysis."
- **Analyze All Files** - This forces a full analysis of all project files, whether they have changed since the last analysis or not.

For either command, the status is reported on the Status Line and the *Command Results* window appears with a log of the results.



There are two phases in the analysis process: determining compilation order, and then parsing. You do not need to specify the parse order. *Understand for FORTRAN* figures out the parsing order automatically. It also detects if needed source files are missing and reports this. You can optionally stop at errors, or continue, ignoring the errors.

When the analysis is complete, the source code for any errors or messages, may be examined by double-clicking on the message in the *Command Results* window.

To save the *Command Results* log to an ASCII file, select the *Command Result*s window and choose **File->Save As**. Specify the location and name of the file you want to save.

After parsing, the *Understand for FORTRAN* database contains lots of data to browse.

Tip: A configured project may be analyzed in batch mode using the command line program *"undftn"*. Refer to *Using undftn* on page 9–2 for details on using *"undftn"*.

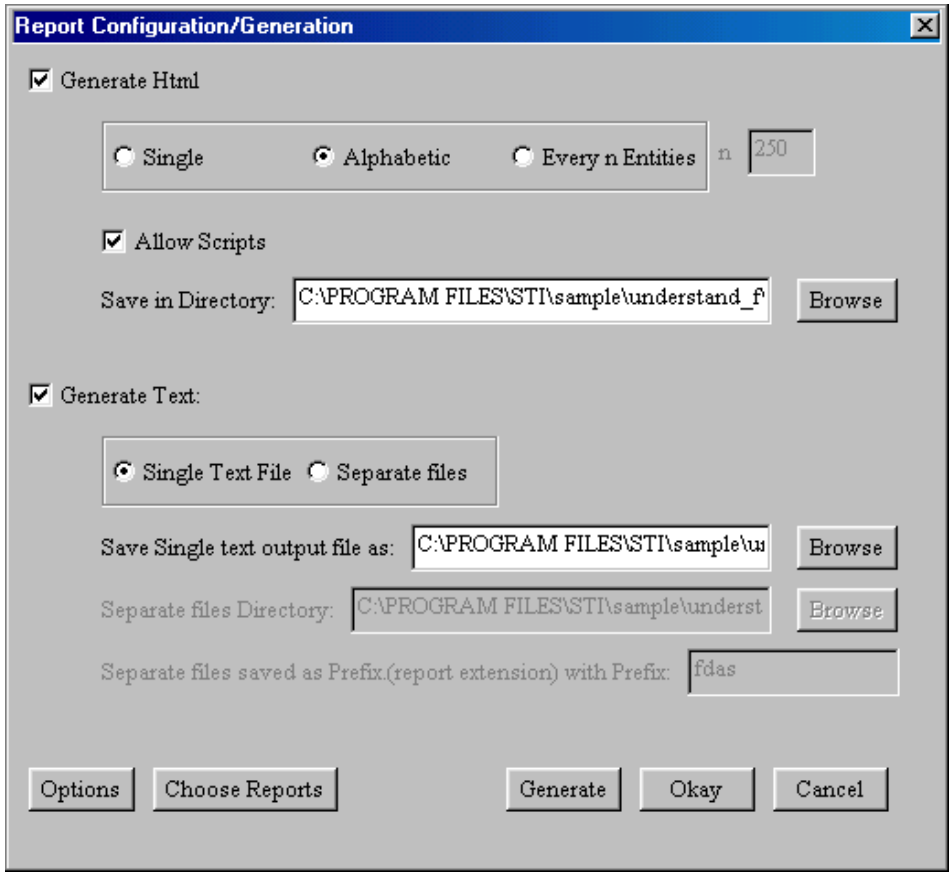
Quick Project Updating

After editing a file, or changing parameters that might affect parsed information, click the “Reparse” icon (shown below) to quickly update the database for any files that have changed. This causes only the Parse Order phase of the analysis to be performed.

To perform both phases of the analysis, you can right-click on the icon and choose **Force Rebuild of Database**.

Output Reports

Choose **Project->Reports Generate** from the menu bar to begin generating reports. You will see the Report Configuration/Generation dialog.



HTML or text files may be generated. Specify the directory location where the generated reports are to be saved.

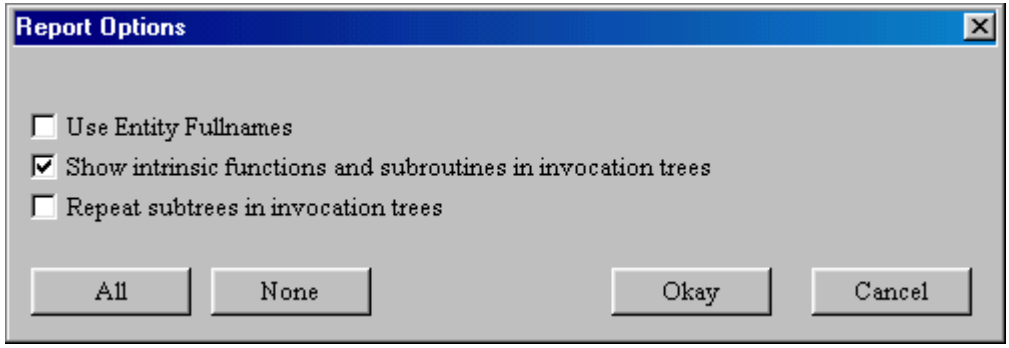
When generating HTML reports, you may generate multiple HTML files for each report type. Choose *Alphabetic* to generate multiple HTML files per report which are split up alphabetically by the first letter of the entity name. Choose *Every n Entities* to generate multiple HTML files per report which are split up every “n” number of entities. It is recommended that you split up the HTML files for

large projects as a single HTML report file may be too large for the HTML browser. The “home” file is index.html. By default, a single HTML file per report will be generated for each letter of the alphabet.

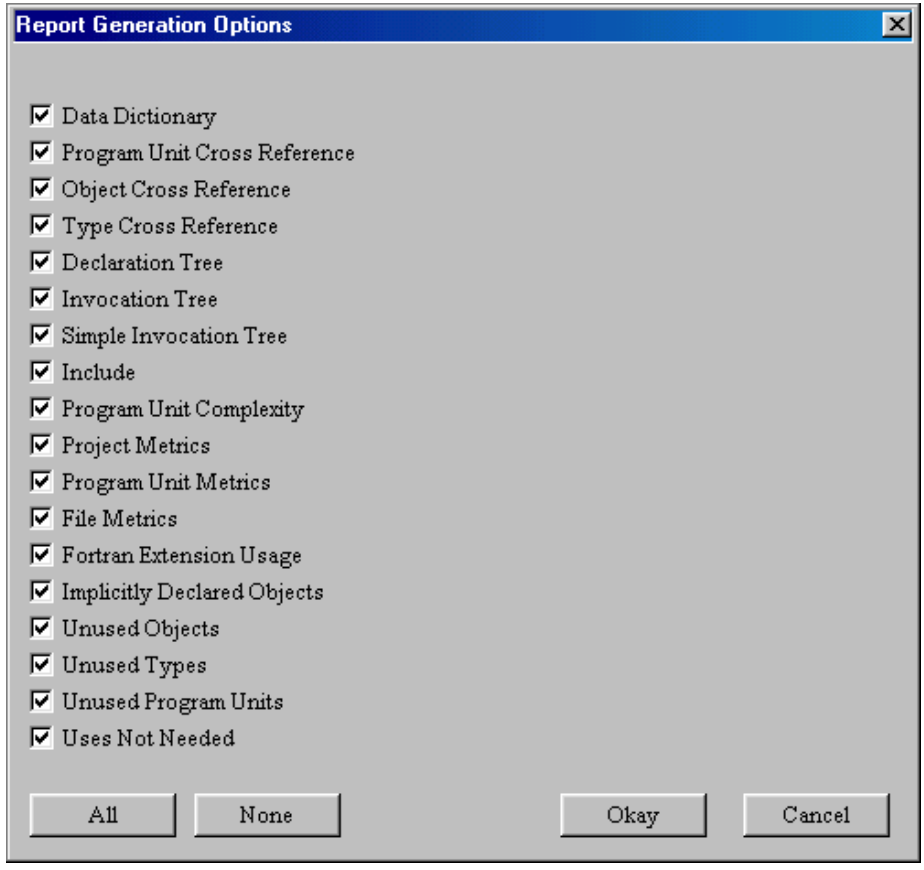
When generating text files, you may generate one text file of the specified name (by choosing File). Alternately, you may generate multiple text files (by choosing Separate) and specify a common filename prefix. The file extensions of each text file will denote the separate reports. Specify the desired directory location with the file prefix.

Note: Reports can also be generated from the command line program “repftn”. Refer to *Generating Reports* on page 9–8 for more details about “repftn”.

In the Report Configuration/Generation dialog, click **Options** to open a dialog that allows you to set various report options, including whether to show entity fullnames in the reports instead of entity shortnames, which is the default.



In the Report Configuration/Generation dialog, click **Choose Reports** to open a dialog that allows you to select the reports to be generated.



Once you have specified the types of reports to be generated, click **Generate** in the Report Configuration/Generation dialog to generate the selected reports.

As with the analysis, report generation status is written to the Status Line and to the Command Results window.

To view the HTML or text reports, choose **Project->Reports View->HTML** or **Project->Reports View->Text**. Refer to *Text and HTML Reports* on page 6–1 for more information on the reports generated.

Chapter 3 Exploring Your Code

This chapter covers the basic windows in *Understand for FORTRAN* 's and their options in detail. It also covers operations within the view windows: the Filter Area, the Information Browser, and the graphical Hierarchical and Declaration view windows.

Details on the use and operation of the **Source Editor** is contained in the following chapter *Editing Your Source Code* on page 4–1.

Details on the use and operation of the Locator Window and Find in Files for searching for and locating entities are provided in *Searching Your Source* on page 5–1.

The symbols used to describe entities are shown graphically, along with a textual set of rules for knowing a symbol's meaning are described in *Graphical Notation* on page A–1.

This chapter assumes a moderate understanding of the FORTRAN programming language and an understanding of using menus under Windows or X-Windows.

This chapter contains the following sections:

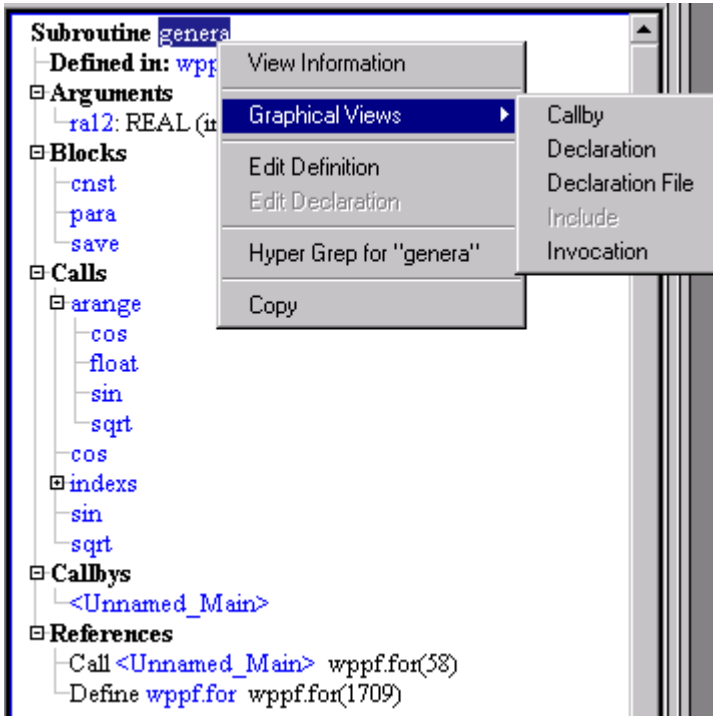
Section	Page
PLEASE RIGHT CLICK	3–2
Various Windows Explained...	3–3
Information Browser	3–9
Graphical View Browsers	3–15
Controlling Graphics Layout	3–21
Printing Graphical and Source Views	3–27

PLEASE RIGHT CLICK

Sorry for shouting (by using all caps above). In order to make the *Understand for FORTRAN* interface as quick, tight and elegant as possible, we have hidden a lot of power beneath your mouse buttons.

The general rule is that Anywhere you look you can right-click to do or learn something. A second general rule is that right-click reuses windows where it can and **CTRL + Right-Click brings up new windows.**

So please right-click. There will be no more reminders.



Check out all the stuff you can learn or do by right clicking!

Right Click almost anywhere brings up an information window.

CTRL Right Click brings up the same menu but actions happen in a new window.

Various Windows Explained...

Understand's GUI is broken into two main areas, the *Document Area* and the *Project Area*. The Project Area is further divided into two areas, the *Filter Area* and the *Information Browser*.

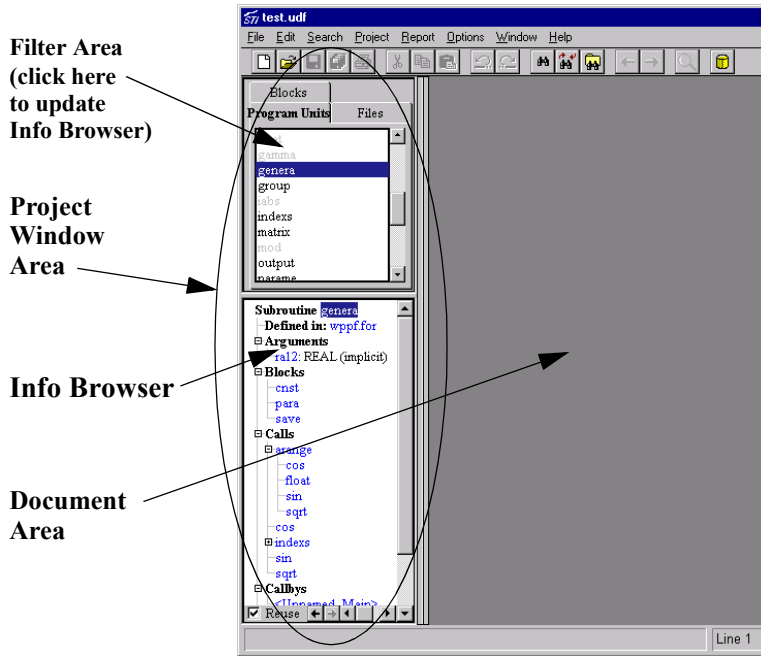
Document Area

The *Document Area* is where source files, graphical views and a variety of other browsers and windows are displayed, including:

- *Graphical Hierarchy Browsers*
- *Graphical Declaration Browsers*
- *Source Code Editor*
- additional *Info Browsers*
- *Locator Window*
- Search Results
- Command Results

Project Area

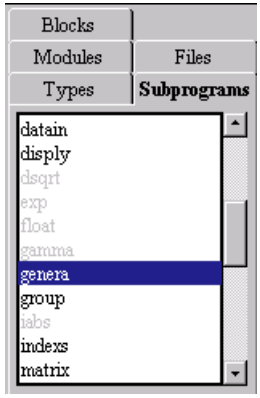
The *Project Area*, which can be hidden or shown, provides a quick way to find entities and information about entities.



The Project Area is further divided into two areas, the *Filter Area* and the *Information Browser*.

Filter Area

The *Filter Area* provides a quick list of entities of each kind shown in the selected *Filter Area* Tab. Options are **Types, Subprograms, Modules, Files**, and **Blocks**. In each of these, any entity that has been declared (or used) in the source code can quickly be found.



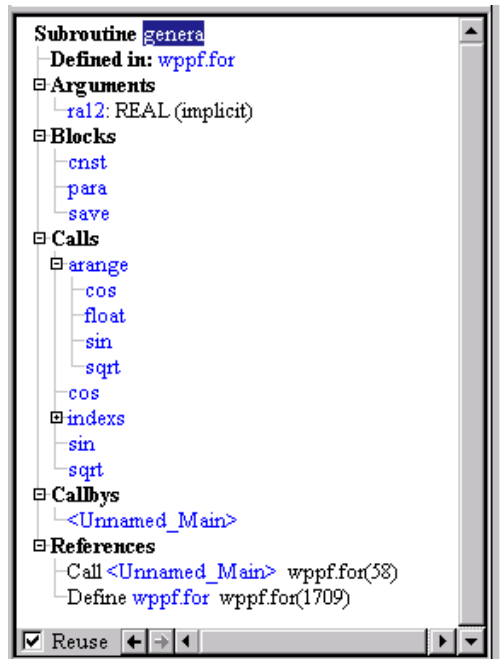
Because there can be a lot of tabs, you also have the option of showing the Filter Area options as a Pull-down menu:

The default uses tabs. You can change this setting in the **General** tab of the **Options->Preferences** dialog.

Information Browser

Any time you left click on an item in the *Filter Area* the *Information Browser* updates to show everything that *Understand for FORTRAN* knows about that entity. The *Information Browser* shows this data as a tree which whose branches can be expanded individually or all at once.

There is always an *Information Browser* in the *Project Window*. In addition, by holding down the CTRL key while right clicking and choosing **View Information**, new *Information Browser* windows can be displayed in the *Document Area*.

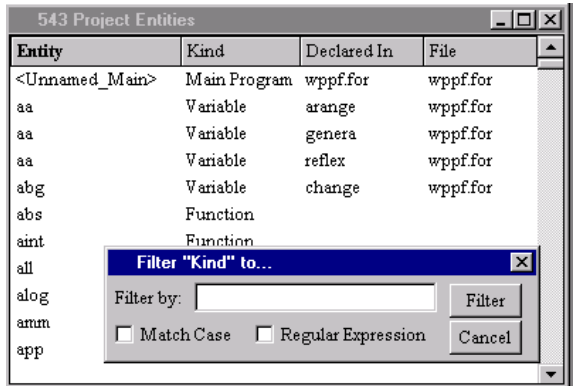


All information in an *Information Browser* window can be saved to a text file, or copied and pasted via standard Windows or X11 copying functions.

For more details on using the Information Browser, refer to *Information Browser* on page 3–9.

Locator Window

Not all entities fall into one of the tab categories shown in the *Filter Area*. You can find and learn more about any entity by using the *Locator Window*, which provides a filterable list of entities in the database. You may filter by name, by entity type, or by a kind of relationship (e.g. invocation, with, callby, etc...).



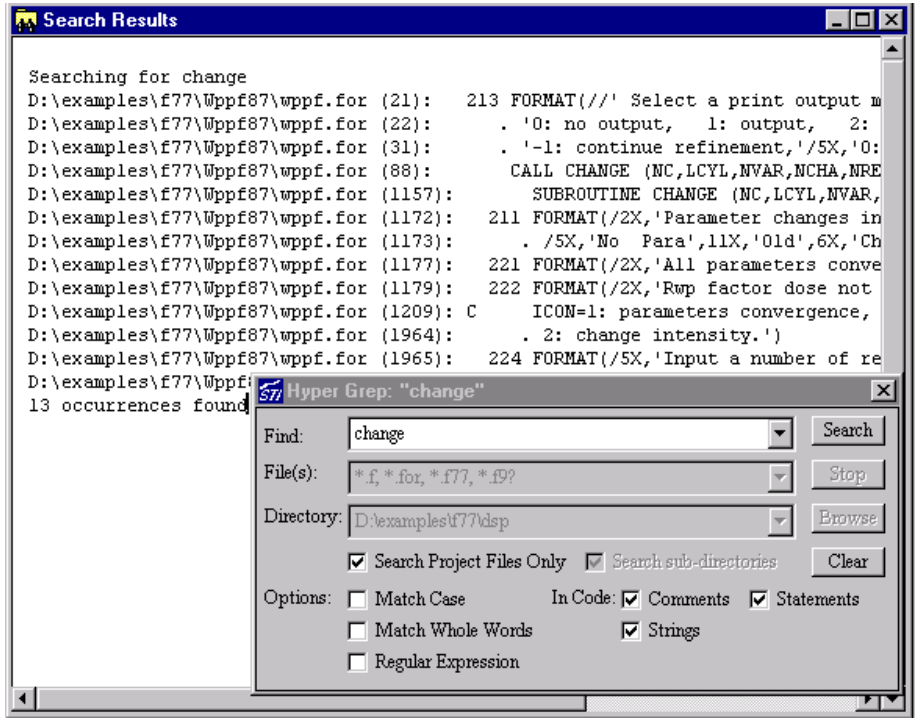
To open the *Locator Window*, choose the **Search->Locate Entities** or **Search->Browse Entities** menu item on the main menu bar.

The entities listed in the *Locator Window* may be filtered by entity name, by entity kind, by declared-in entity name, or by file name.

For more details on using the *Locator Window*, refer to *Locator Window - Find or Browse Entities* on page 5–3.

Find in Files

You may search the project files or another selection of files for the occurrence of a text string or regular expression. Matches are shown in the *Search Results* window and can be visited in the source code by double-clicking on any line.

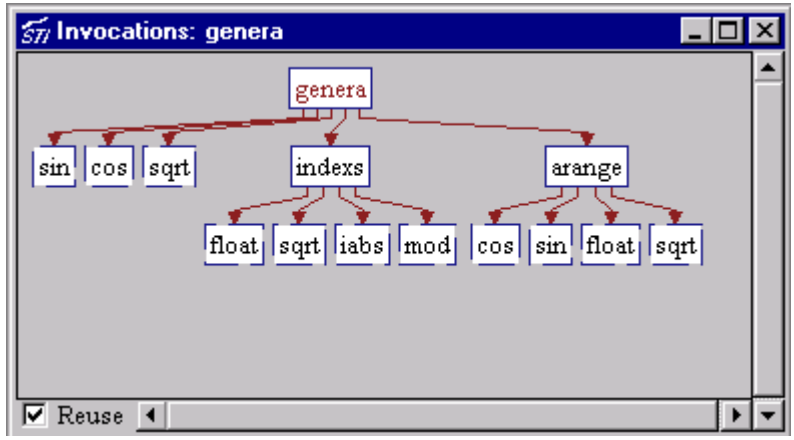


For more details on using Find in Files, refer to *Using Find in Files* on page 5–10.

Hierarchy Browser

The *Hierarchy Browser* shows multiple level relationships between entities. All relationships are multi-level and are shown to the top or bottom of their respective tree unless a level option is set in the preferences.

Following is an invocation view for a subroutine.



Understand for FORTRAN offers hierarchical information about the following types of relationships:

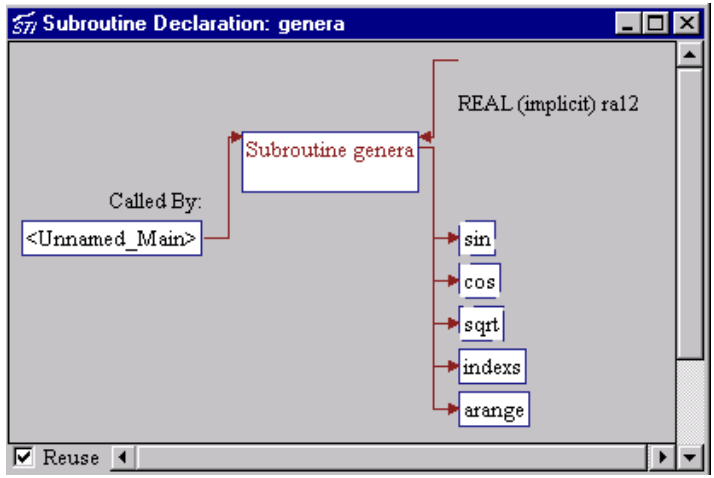
- **Callby** - view of who calls a given entity.
- **Invocation** - who this entity calls.
- **Include** - shows who this subprogram includes.
- **IncludeBy** - shows who includes this file.

Declaration Browser

Declaration views offer a one glance way to see important structure and relational information about a given entity. *Understand for FORTRAN* offers these declaration views:

- **Subprogram** - Shows the parameters, invocations, and callbys of a given subprogram.
- **Block Declaration** - Shows what a block is composed of.

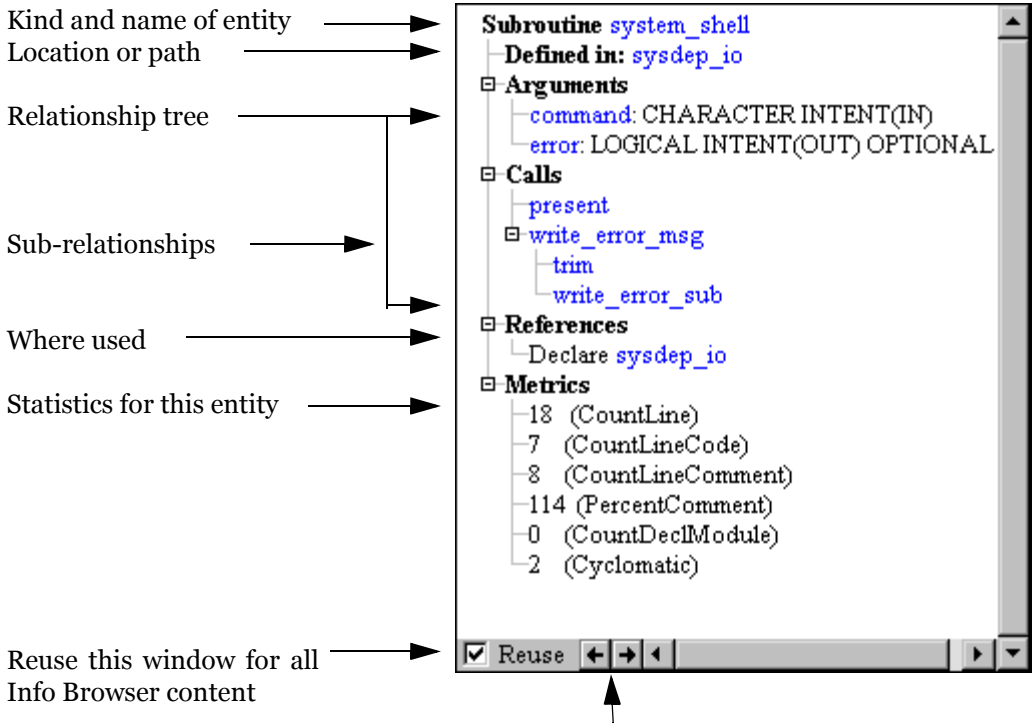
Following is an example of a subprogram declaration:



Information Browser

Everything *Understand for FORTRAN* knows about an entity can be learned using the *Information Browser*. The information is shown in a tree. The tree can be expanded selectively or in bulk. Each branch of a tree follows an aspect of the entity. Each terminating item (leaf) of a tree provides some information about that entity.

As you drill down you can change which entity you are learning about. Each time you change the entity, it is remembered in the Information Browser history for quick backtracking.

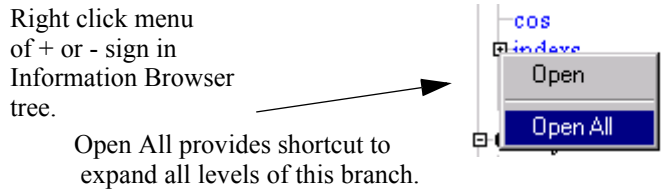


Drilling Down A Relationship

Drilling down the tree works as expected (mostly). To expand a tree, click on the + sign. To close the tree click on the - sign.

There are a few tricks however...

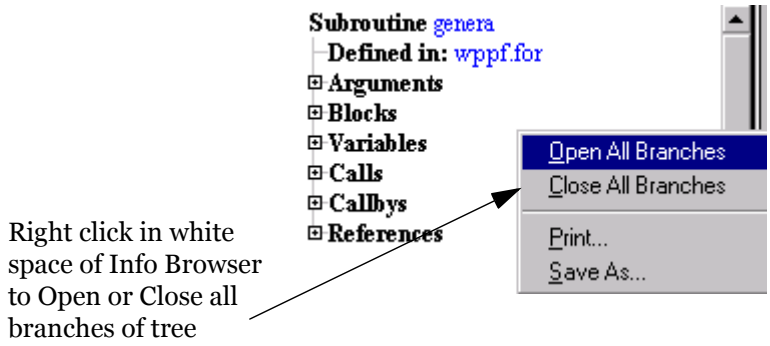
Right-clicking a + or - sign in the tree brings up a collapse/expand menu:



If the tree node is currently closed, you will be presented with the option to **Open** or **Open All**. If the tree node is currently open, you will be presented with the options to Close, Close All, or Open All.

Open All and Close All operate on all levels of the selected branch.

To open or close the entire tree, right-click in the white space of the Information Browser and choose **Open All Branches** or **Close All Branches**.



See *Saving and Printing Information Browser Info* on page 3–12 for details on the other options in this right-click menu.

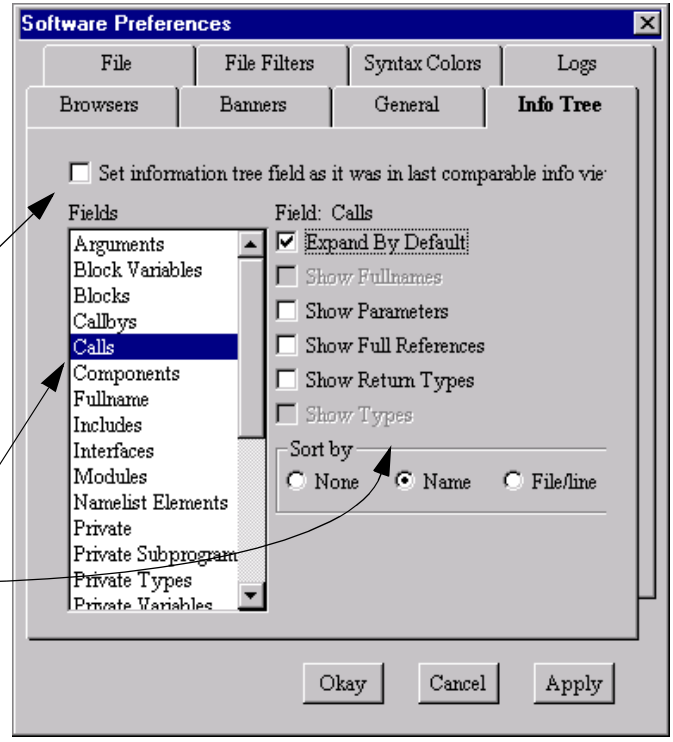
Drilling Down Efficiently

Relationships that you always use can be set to “pre-expand”. For instance, “+Calls” can be set to always expand. Another mode is for the Information Browser to “remember” how you had a tree expanded the last time you looked at a particular kind of information. Set these options from the **Options->Preferences** Dialog, **Info Tree** tab:

Remember expand/ close settings from last view

OR

Specify preferences for each kind of relationship



Viewing Metrics

The last node on the Information Browser tree is **Metrics**. This branch shows the metrics available for the current entity.

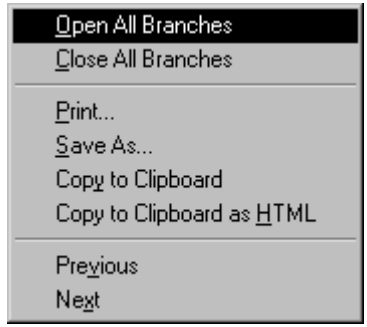
If you are changing the entity frequently in a large project, closing this node until you want to view the metrics may improve the speed of Information Browser updates.

See *Metrics Reports* on page 6–16 for details on metrics.

Saving and Printing Information Browser Info

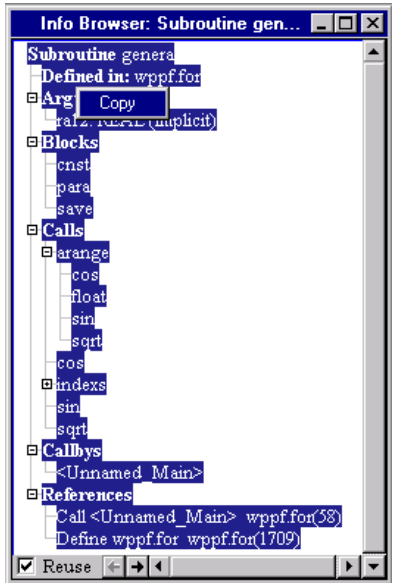
All the text shown in the *Information Browser* can be printed, saved as a text file, or copied to the clipboard for pasting into another application as text or HTML. Only the currently expanded branches are shown in the output. When saving or pasting in text format, the branches of the tree are represented by indents in the text.

Right-click on white space in the Information Browser to see this menu. (If you see a menu that begins with "View Information," right-click to the right of a bold heading in the tree.)



- **Open All Branches** and **Close All Branches** - See *Drilling Down Efficiently* on page 3–11.
- **Print** - Choosing this item opens a standard printing dialog. Clicking OK print the current contents of the Information Browser. The printed tree is opened or closed to the same level as shown on your screen. The printed output is formatted like the Information Browser display (including color on a color printer).
- **Save As** - Choosing this item opens a file dialog. Choose the location and name of a file to contain the Information Browser information as text. Branches of the tree are represented by indents in the text.
- **Copy to Clipboard** - Choosing this item copies the highlighted text in the Information Browser (or all the text if nothing is selected) to the clipboard. You can paste the text into any application.
- **Copyto Clipboard as HTML** - Choosing this item copies the highlighted portion of the Information Browser (or the complete Information Browser if nothing is selected) to the clipboard as formatted HTML. You can paste the HTML into applications such as a web page editing tool or Microsoft Word.

Marked Info Browser

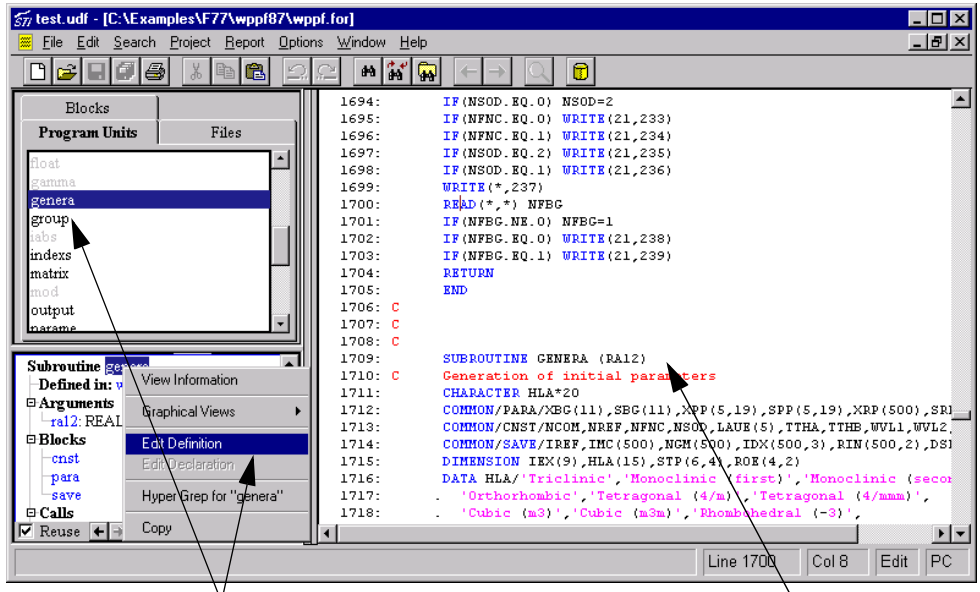


Resulting Text Copied

```
Subroutine genera
  Defined in: wppf.for
  Arguments
    ra12: REAL (implicit)
  Blocks
    cnst
    para
    save
  Calls
    arange
      cos
      float
      sin
      sqrt
    cos
    indexs
    sin
    sqrt
  Callbys
    <Unnamed_Main>
  References
    Call <Unnamed_Main>  wppf.for(58)
    Define wppf.for  wppf.for(1709)
```

Visiting Source Code

In general, if you double click on an entity in an informational window (*Information Browser* or *Filter Area*) the FORTRAN declaration of that entity will be loaded into the *Document Area*.



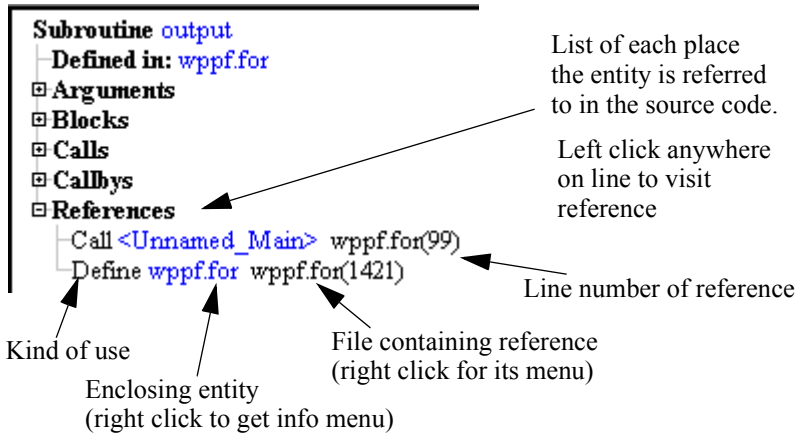
Double click on genera, or right click on it and choose “Edit Definition” to visit source code.

Right Click Menu Source Visiting

Another way to visit source, this time from any entity you see in *Understand for FORTRAN*, is the right-click menu. Any entity's Right Click Menu will contain a menu item for visiting each of its declaration locations:

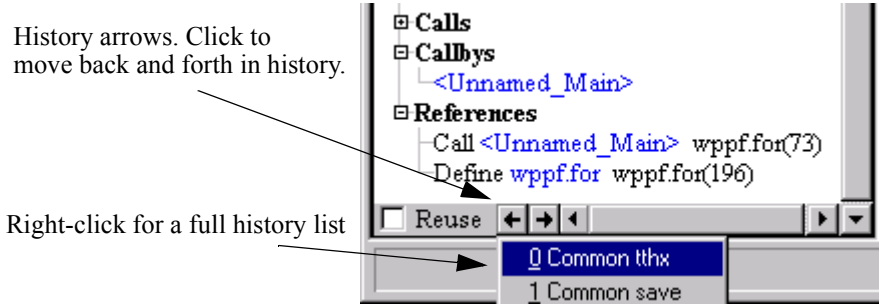
One Click Visiting of References

The portion of the *Information Browser* labeled "References" lists everywhere the entity is referred to in the analyzed source code:



Entity History

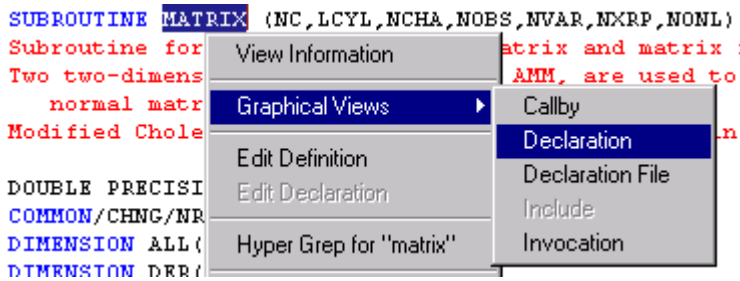
As you explore your code, you can go a lot of places quickly. Often you want to backtrack to explore a new path. To help you do this, each *Information Browser*, *Hierarchy*, *Declaration*, and *Source* window contain a full history of what they have done. The *Information Browser* history can be found in the bottom left corner:



Use the right and left arrows to move back and forward in the history list. You can right-click on an arrow to see the full history list (in that direction) and select an entity from the list to jump to.

Graphical View Browsers

The right-click menu of an entity with a structure or hierarchy (not variables or parameters) offers a choice called “Graphical Views”:



The Graphical Views menu adapts based on what kind of entity has been right clicked on. A greyed-out item refers to information normally available for that kind of entity but not applicable to this particular entity.

General Rules for Using the Graphical Browsers

There are some general rules that can be used for browsing any type of graphical view.

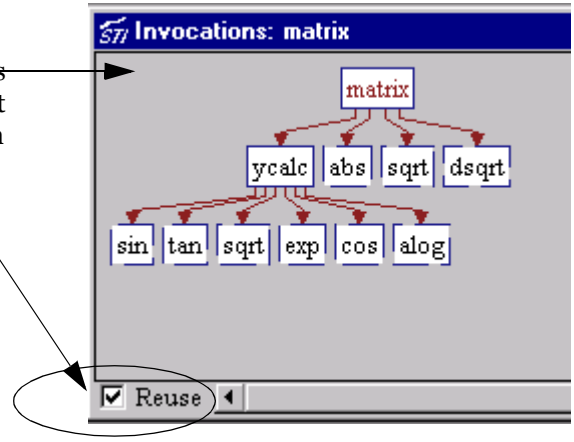
- Anywhere you see an entity, you can right click on it to learn more.
- CTRL-RIGHT-CLICK does the same as a normal right-click action but displays the information in a new window.
- Reuse is turned on by default and is very helpful for quickly seeing the same kind of information about different entities.
- Layout is done automatically, there is no need to move lines or boxes around for a better view. Options are available for changing the layout decisions automatically
- Everything you see can be printed as you see it. Printing may be done to one page (squeezing the picture) or across multiple pages (poster style).

Reuse Checkbox

By default, “Reuse” is on in the graphical view. When set, the same view window displays all graphical views. If you uncheck this field, a new window is opened for the next view you select. A maximum of one view can have the Reuse button checked at any time.

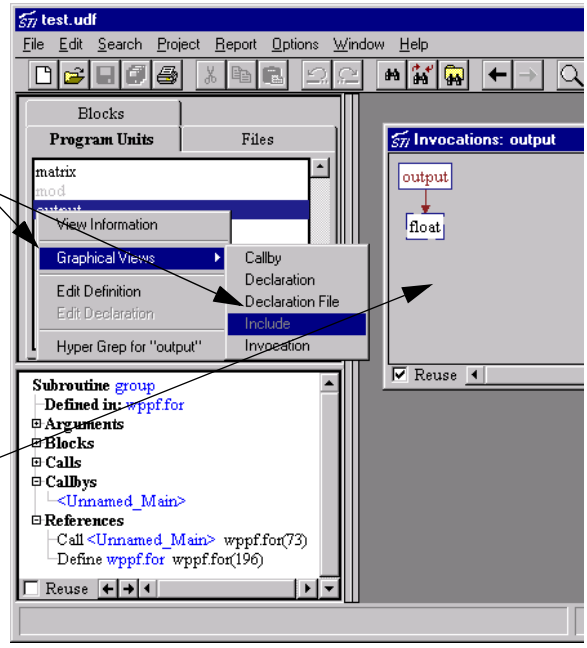
Reuse CheckBox

If checked, then this window will present similar information about each entity that is left clicked on.



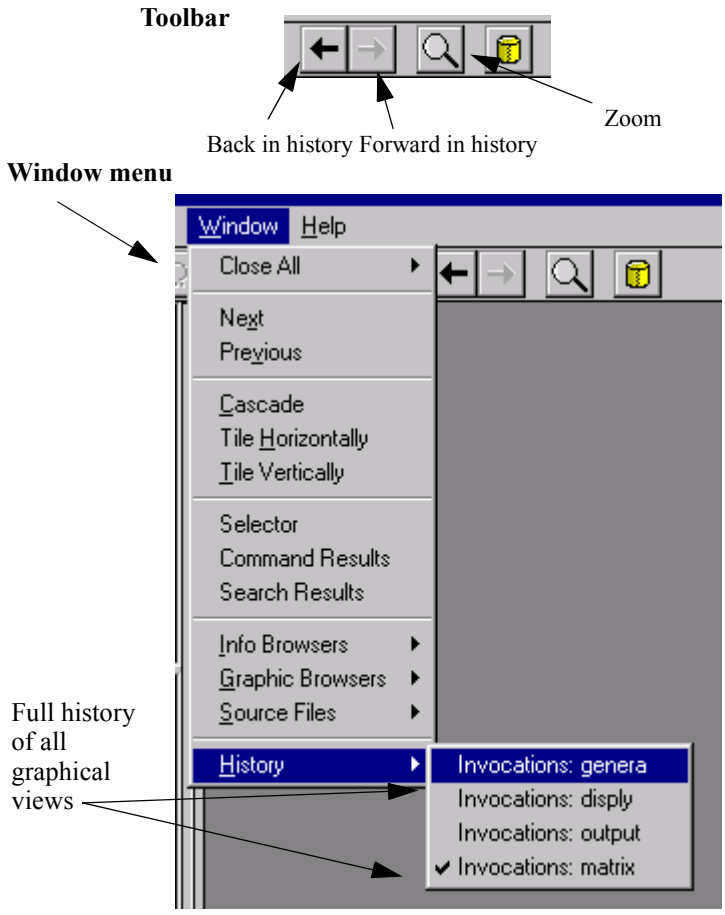
Right click here and choose **Graphical Views** and then the desired view.

The *Declaration Window* will then “sync” to present information about what was clicked on.

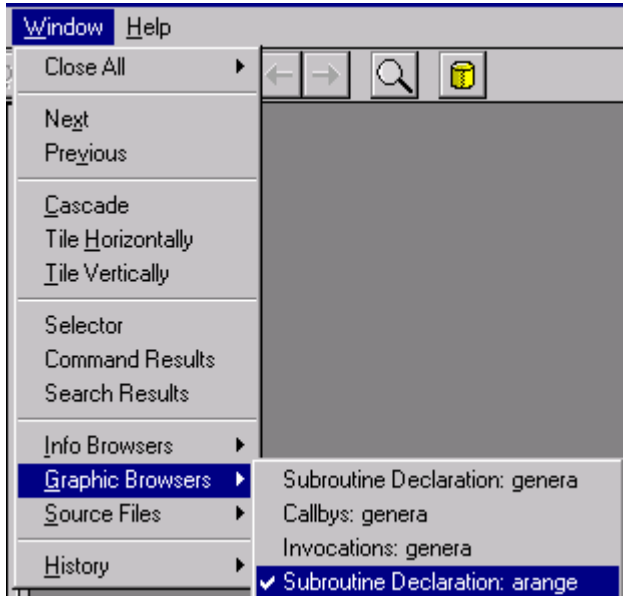


Graphical Browser History

Each graphical view window keeps a history of the views it has presented. The history buttons on the toolbar (left and right arrows) and the **Window->History** menu item can be used to move back and forth in this history. This is useful for quick backtracking.



You can also choose from a list of all open graphic view windows by using the **Window->Graphic Browsers** menu:

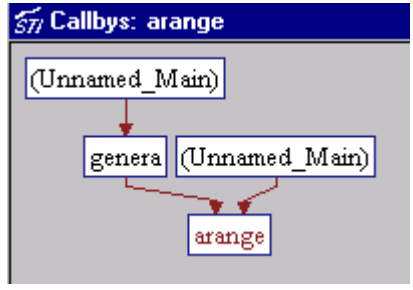


Graphic Hierarchical Views Available

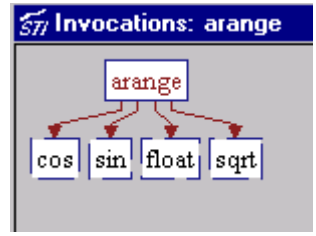
Hierarchical views show multiple level relationships between entities. Graphical *Hierarchical* views available are:

- **Callby** - view of who calls a given entity
- **Invocation** - who this entity calls

Callby shows who calls function *arange*



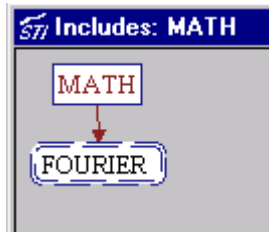
Invocation shows who function *arange* calls



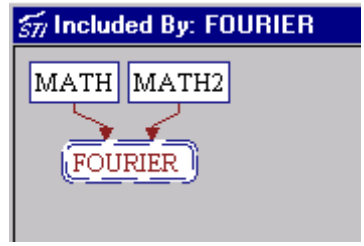
- **Include** - shows who this subprogram includes.

- **IncludeBy** - shows who includes this file.

Include shows who this subprogram includes



Includeby shows who includes this file

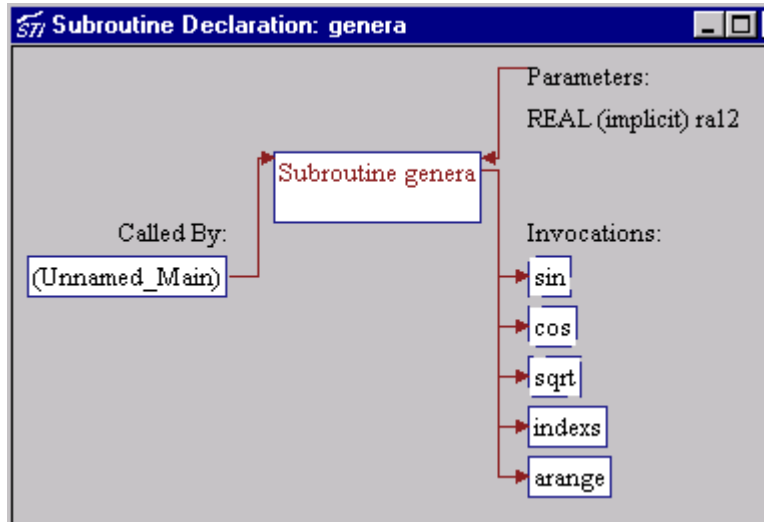


Graphic Declaration Views Available

Declaration views offer a one glance way to see important structure and relational information about a given entity. *Understand for FORTRAN* offers these graphical *Declaration* views:

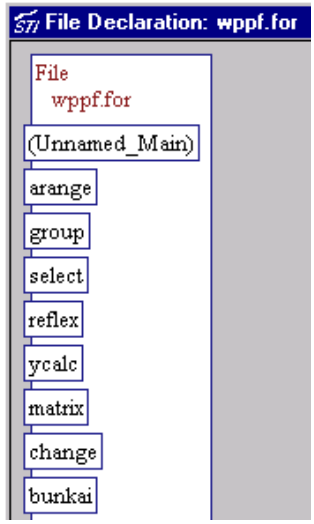
- **Subprogram Declaration** - Shows the parameters, invocations, and callbys of a given subprogram.

Subprogram Declaration

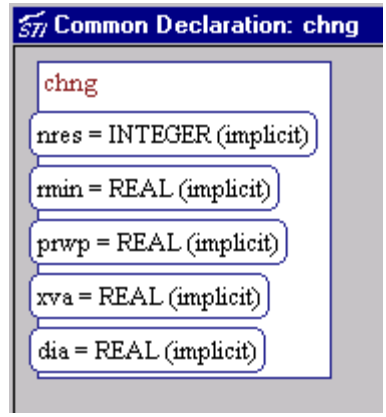


- **Common Block Declaration** - shows what a block is composed of.

File Declaration



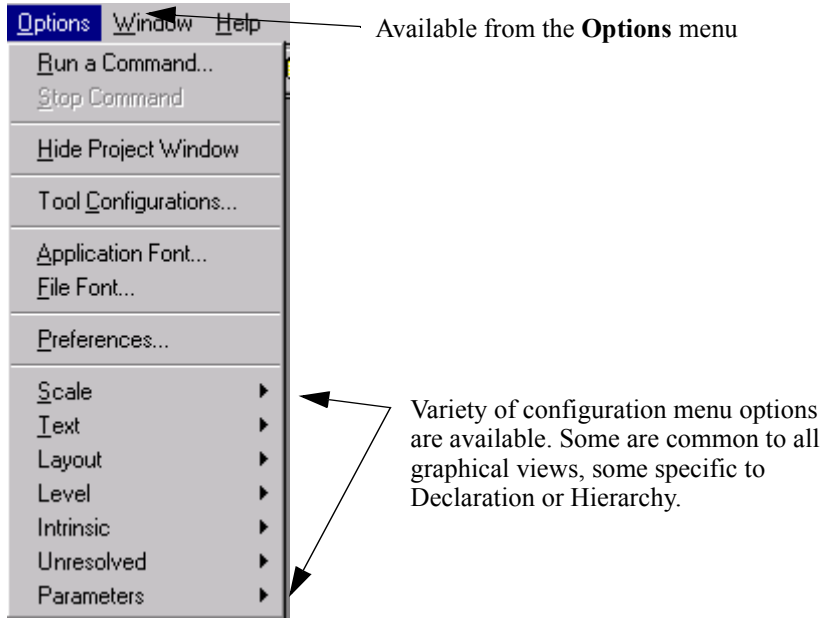
Block Declaration



Controlling Graphics Layout

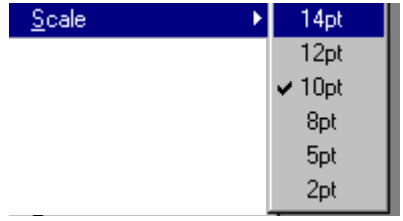
The two types of graphical view windows, ***Hierarchy*** and ***Declaration*** have a variety of configuration options which may be set via the **Options** menu of the menu bar. These options control the layout and drawing of the graphic views and vary based on the type of view.

In addition, certain toolbar actions are in effect when a Graphical View is in focus.



Scale Menu

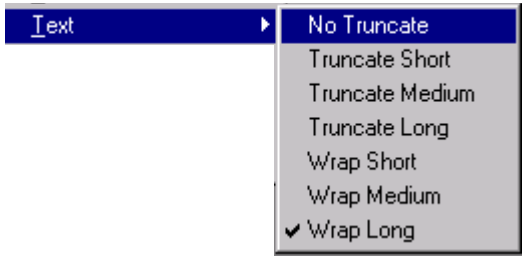
The **Scale** menu allows you to choose the size of the text used. It is available for both declaration and hierarchy views. All picture sizes and layouts vary with text point size. The currently selected size is indicated by a check mark.



Other point sizes can be added by customizing configuration files found in the *Understand for FORTRAN* installation directory. Contact *support@scitools.com* for information on how to do this.

Text Menu

The **Text** menu sets the way entity names are trimmed or altered to accommodate the layout of graphics. It is available for both declaration and hierarchy views. Names may be truncated to a certain length or wrapped at a certain length.



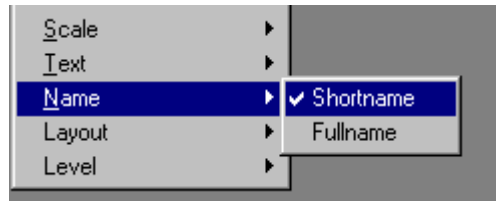
- **No Truncation -** Uses the name as defined in the source code. The default.
- **Truncate Short** - Cuts off names at 10 characters.
- **Truncate Medium** - Cuts off names at 20 characters.
- **Truncate Long** - Cuts off names at 30 characters.
- **Wrap Short** - Wraps the name between 8 and 10 characters. Location in that range depends on if a natural wrapping character is found. Natural wrapping characters are . _ - and :
- **Wrap Medium** - Similar to *Wrap Short* except wrapping range is 15-20 characters.
- **Wrap Long** - Similar to *Wrap Short* except wrapping range is 20-30 characters.

Intrinsic Menu

The **Intrinsic** menu controls whether intrinsic functions (e.g. cos, sin) are displayed or hidden.

Name Menu

The **Name** menu controls whether or not fullnames are used in views. It is available for both declaration and hierarchy views.



A fullname includes its parent compilation units, For example:

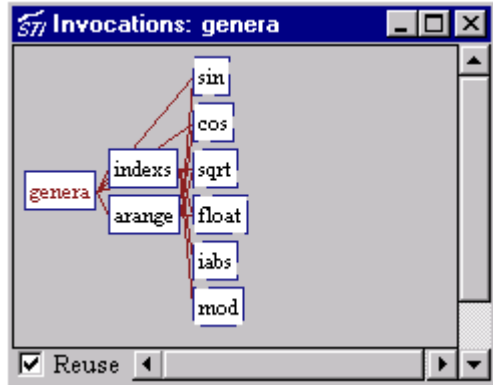
- Text_Io.Put is the fully specified name.
- Put is the Short Name

Longer versus shorter names can alter the layout of pictures substantially.

Layout Menu

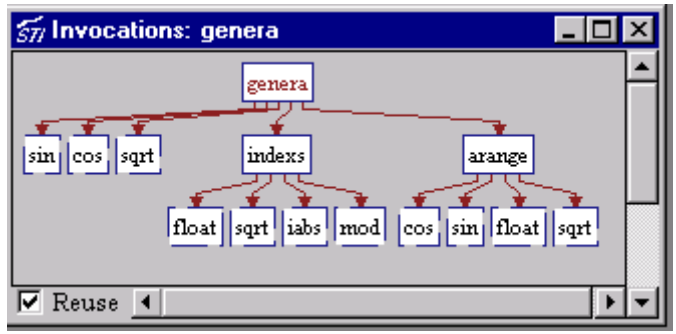
The **Layout** menu controls the layout algorithm for a hierarchical chart. It is available only in hierarchy views .

- **Crossing** - a left-to-right view, minimizing space used but sacrificing some readability by permitting lines between entities to cross.**Horizontal Non-Crossing** - a left-to-right layout,



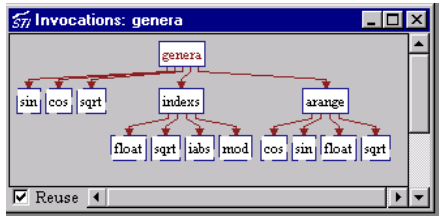
using more space in some situations but enhancing readability by having no crossing lines.

- **Vertical Non-Crossing** - an top-to-bottom layout similar to Horizontal Non-Crossing.

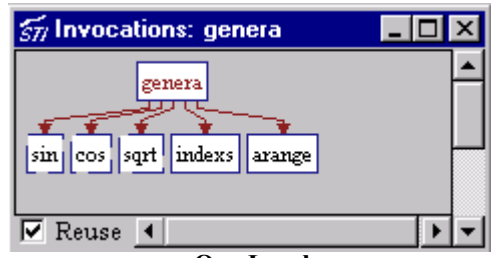


Level Menu

The **Level** menu controls the number of levels to be traversed when laying out a hierarchical view. The default value is “All Levels”. Values of 1 to 5 may be set. It is available only in hierarchy views .



All Levels

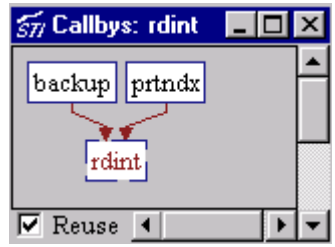


One Level

Unresolved Menu

The **Unresolved** menu controls whether entities that have been used but no declaration was found should be drawn. This option is available on any graphical view (hierarchy and declaration). Unresolved functions and entities are those used in the analyzed source without a definition . Unresolved include files are those included but not found along a declared include path (either a compiler or project include path).

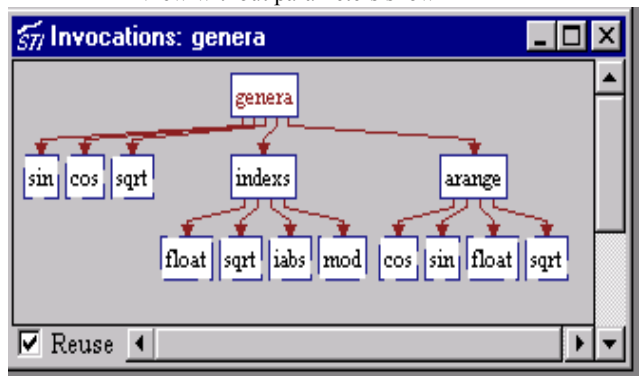
Unresolved entities are drawn as normal but with a dashed border:



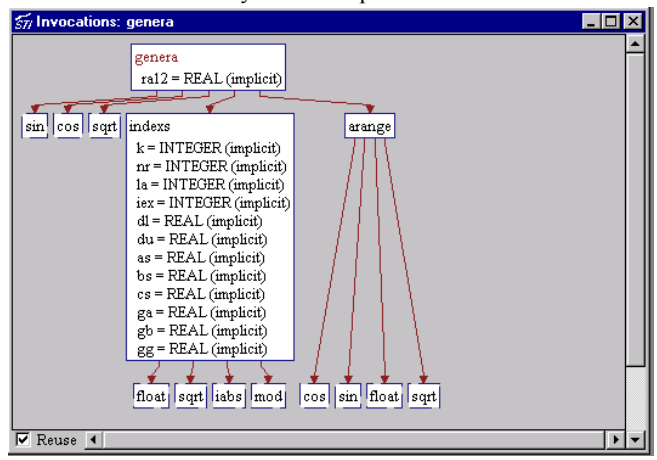
Parameters Menu

The **Parameters** menu controls whether parameters are shown in hierarchical views. Available on any graphical view showing hierarchies of subroutines and functions (invocation and callby). The default is Off, turning this On can make hierarchical pictures much bigger.

View without parameters shown



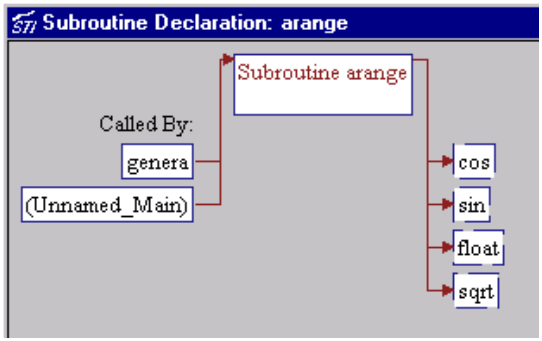
Same entity view with parameters shown



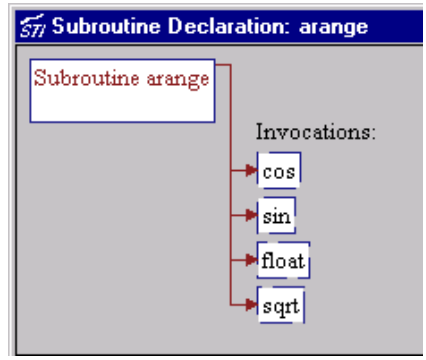
Called by Menu

The **Called by** menu controls whether functions and subroutines that call the current function or subroutine are shown in the *Declaration* view.

View shows CalledBy



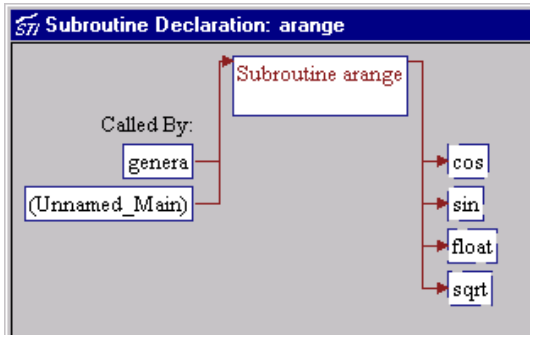
View without Calledby shown



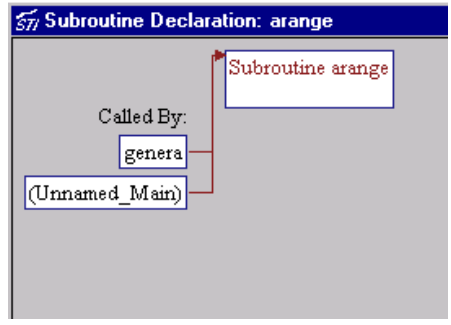
Invocations Menu

The **Invocations** menu controls if functions and subroutines that are called by the current function or subroutine are shown in the *Declaration* view.

View shows Invocations



View without Invocations shown



Duplicate Subtrees Menu

The **Duplicate Subtrees** menu controls whether multiple occurrences of the same sub-tree are shown in hierarchy views. The options are to Hide or Show such subtrees. The default is to show duplicate subtrees. In some applications, hiding duplicate subtrees can dramatically simplify hierarchy views. Duplicate subtrees are not shown if a view has over 1000 nodes.

Printing Graphical and Source Views

Understand for FORTRAN has three printing modes:

- **Shrink to fit** is used by the graphical views (Hierarchy and Declaration) and fits a picture, no matter what its size, onto the current printer page size.
- **Poster Printing** prints graphics using the point size selected in the graphical view. Instead of shrinking to fit on one page, it prints the picture across the number of pages needed.
- **Source File** printing does nothing particularly special and simply queues the file to the printer using 66 lines of source per page.

Shrink to Fit Printing

All graphics views offer the menu option **File->Print Drawing**. This option will print the graphic, making it fit on the given page size. On Windows, the standard Windows printer setup dialog is used to configure page size, printer selection, portrait/landscape and other options.

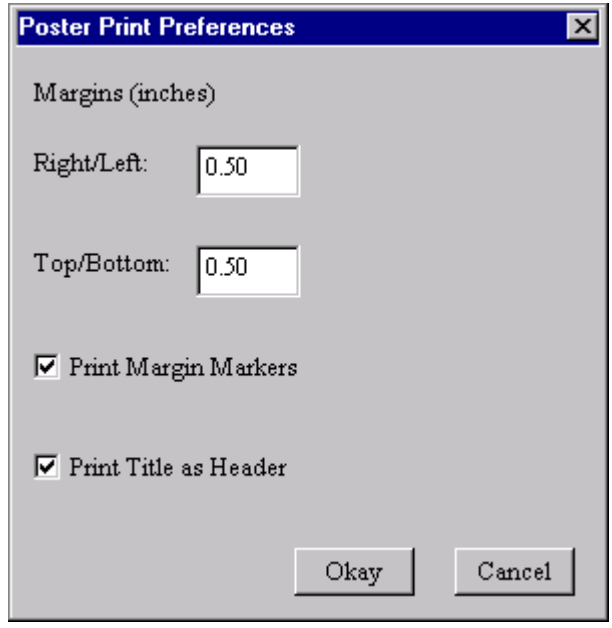
See *Printing on UNIX Machines* on page 3–28 for more information about configuring printouts on UNIX machines.

Poster Printing

All graphics views also offer the menu options **File->Print Poster** and **File->Print Poster Setup**. Choosing **File->Print Poster** sends the current picture to the printer using the current poster configuration. On Windows, the printer setup dialog can then be used to further configure printing. See *Printing on UNIX Machines* on page 3–28 for information about configuring printing on UNIX machines.

Configuring Poster Printouts

Choosing **File->Print Poster Setup** opens this dialog:



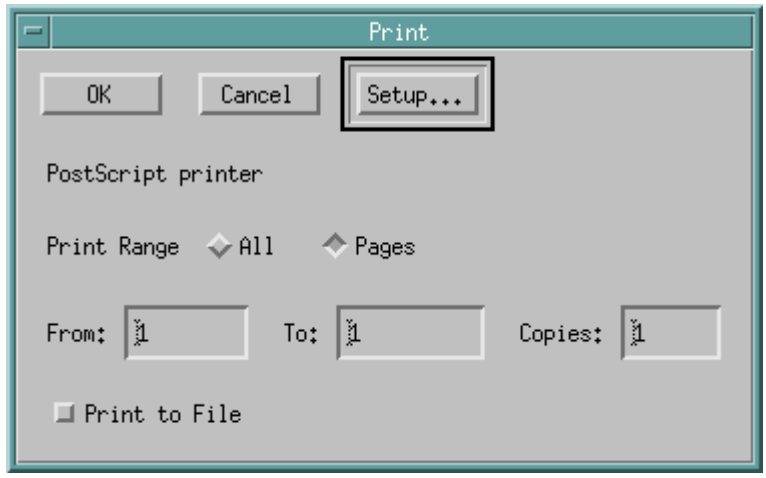
- **Margin (Right/Left and Top/Bottom) -** The margin to leave around the view on each page. A conservative setting (1/2 inch) is used by default. If your printer can print to the paper's edge, you may set these values to zero.
- **Print Margin Markers** - If this box is checked, margin markers, sometimes known as "cut marks," are printed on each page. These indicate where you can trim the pages to create a seamless printout.
- **Print Title as Header -** If this box is checked, the title of the View Window is used as Title. This is the text shown in the upper left corner above the menu bar. In this example, "Callbys: gamma" would be added as a Title to the printout.

Printing on UNIX Machines

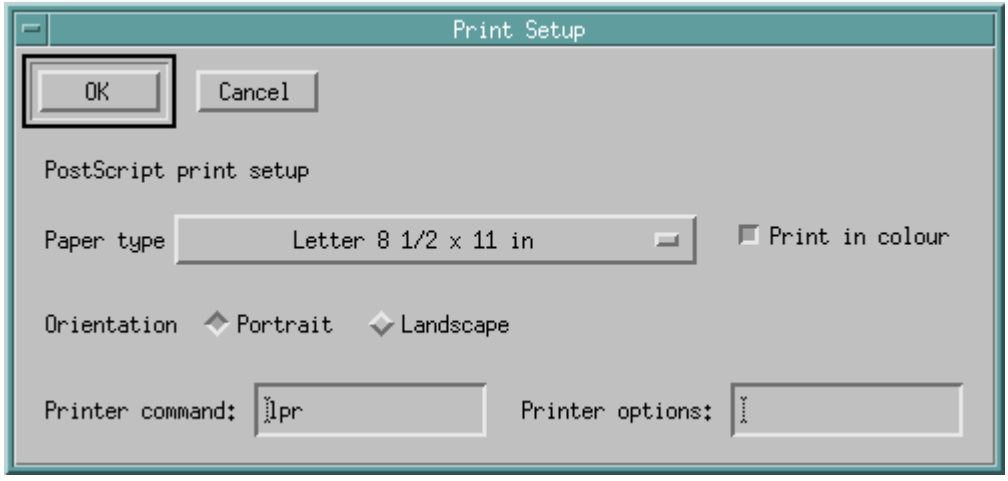
On UNIX machines *Understand for FORTRAN* uses Postscript as its primary output format. The output files created are Level 2, Encapsulated Postscript without a preview image.

All output is to a file, the file can optionally be sent to a queuing command (such as *lp* or *lpr*).

Initially the Print Dialog appears:



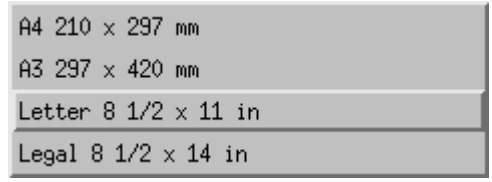
- **Print Range** specify optional range of pages (if more than one page)
- **Copies -** specify number of copies of each printed page
- **Print to File**, if checked then a dialog is displayed so you can specify what file to print to.
- **Setup** - use to further configure printing, as shown in this panel:



- **Printer Command** - Sets the system command used to print. The default is lpr. This command name may be set to any command. No checking is made to ensure it is a valid command or that it is on your executable search path. The name of the Postscript file is passed as its first argument. The

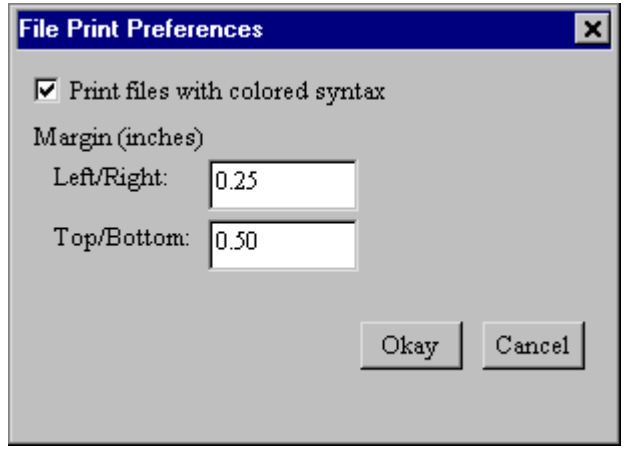
text (if any) contained in the **Printer Options** field is passed as arguments following the Postscript file name.

- **Paper Types Toggle -** Lets you choose what paper size to use. The default is 8 1/2 by 11 (letter).



Source File Printing

By default, files are printed with the syntax colorized as it appears on the screen. To print in black and white, chose **File->Print File Setup** and uncheck the "Print files with colored syntax" option.



Chapter 4 Editing Your Source Code

This chapter covers *Understand for FORTRAN*'s source and text file editor.

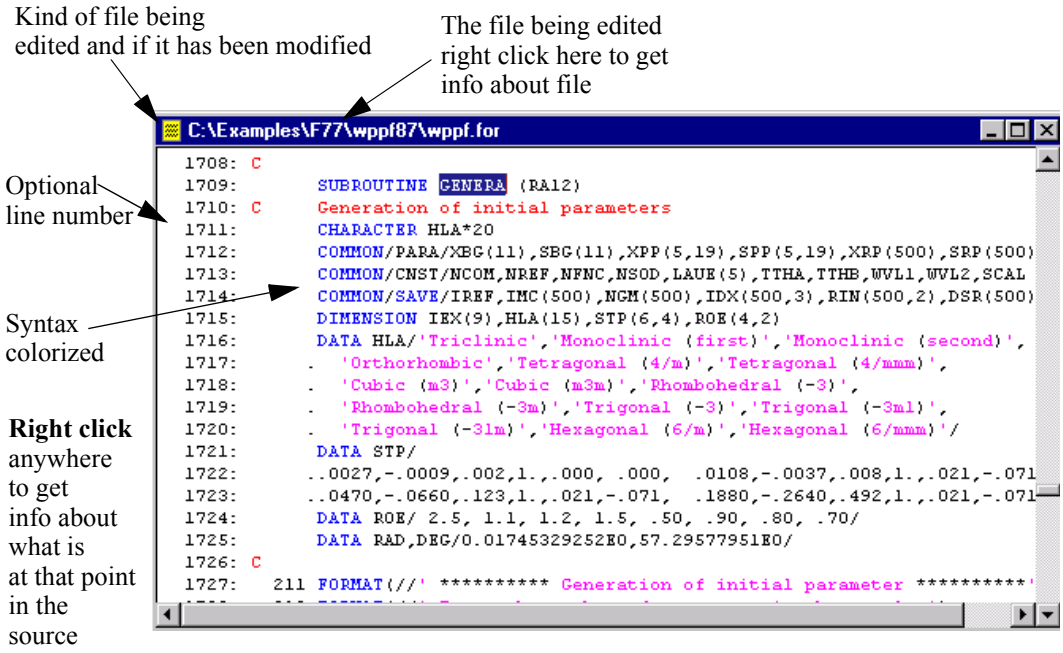
This chapter assumes a moderate understanding of the FORTRAN programming language and an understanding of using menus under Windows or X-Windows.

This chapter contains the following sections:

Section	Page
Source Code Editor	4–2
Colorizing Source Views	4–4
String Searching	4–5
Key Mappings	4–7
Other Features	4–10

Source Code Editor

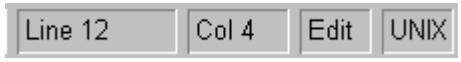
The **Source Code Editor** offers a full featured source code editor, with syntax coloring, and right-click and synchronized access to information most entities in your code.



You can optionally use an editor other than the one provided with *Understand for FORTRAN* for viewing and editing your source code. For example, you can use Microsoft Visual C++ or Emacs as your editor. For details, see *Using an External Editor* on page 7–2.

Status Line

In the bottom right corner of Understand's main window is a status line that gives basic editor status information at a quick glance:



The first two columns are self explanatory, telling the line and column where the cursor is currently.

Column 3 tells the mode of the file. “Edit” means that the file can be edited. “Read” means the file is “Read-Only” and that editing is not active.

The fourth column indicates the kind of file that this is. “UNIX” means that file lines are terminated in UNIX fashion (carriage return only). PC means that file lines are terminated in DOS fashion (carriage return and line feed).

Status Icons

Each file loaded into the editor has a status icon in its upper left window title. This is used to indicate the kind of file and its status:


Yellow icon = project file (has been or will be parsed)

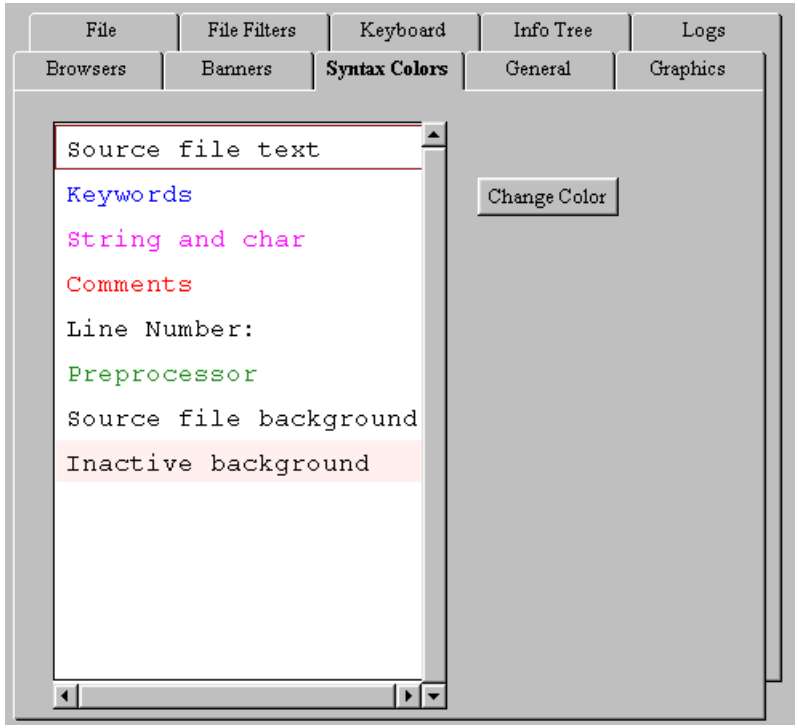

Yellow with M = modified project file (needs to be parsed)


White = file not in the project

Colorizing Source Views

You can customize the colors used in the Source Code Editor in the Software Preferences dialog.

To open this dialog, choose **Options->Preferences** and move to the **Syntax Colors** tab. To change a color, select one of the items in the list and click **Change Color**. Use the color selection dialog to choose a new color for that item.

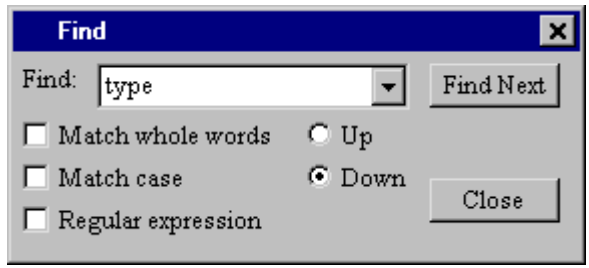


By default, the following color codes are used for the source code:

- **Blue text:** used for language keywords
- **Pink text:** used for characters and character strings
- **Red text:** used for comments
- **Green text:** used for preprocessor statements
- **Black text:** used for all other source text and for line numbers
- **White background:** used for most source text
- **Pink background:** used for inactive lines of code

String Searching

You can search for the occurrence of a string within the source code browser by entering the string in the search box that is brought up by either the CTRL-F keystroke or the **Search->Find** menu item.



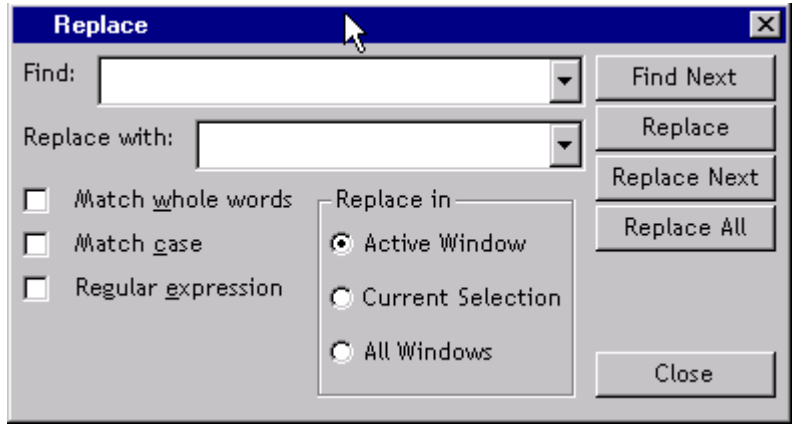
Choose Up or Down to search in either direction from the current location. No change is made to the cursor editing position until you left-click in the file.

You may also start a Find in Files search. Refer to *Using Find in Files* on page 5–10 for more information on using Find in Files.

String and Replace

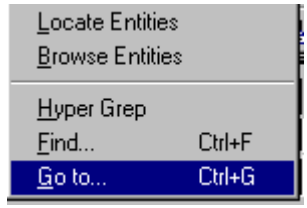
You can also Search and Replace any occurrence of a string within a file, a selection in that file, or among all open files.

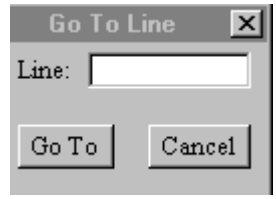
Launch with CTRL-E, or via **Search->Replace**.



Go To Line...

You may also go to a specific line number by choosing **Search->Go To Line** and specifying the line number.





Line Numbers

By default line numbers are shown in files being edited. This can be turned off via the **File** tab of the **Options->Preferences** dialog. Line number color may also be specified in the **Syntax Colors** of the **Options->Preferences** dialog.

Selecting and Copy Text

Text can be selected (marked) then cut or copied into the Windows (or X11) clipboard. Marking text works as for the standard for the window system in use. For Windows dragging while holding down the left mouse marks text. Alternately one can hold down the Shift key and move the cursor (via arrows or other cursor movement keys). Once text is marked the commands that can be done are shown under the “Edit” menu or on the right click menu occurring when right-clicking on the marked text itself.

You may also copy the selected text by using **Edit->Copy Selection** on the Browser’s toolbar.

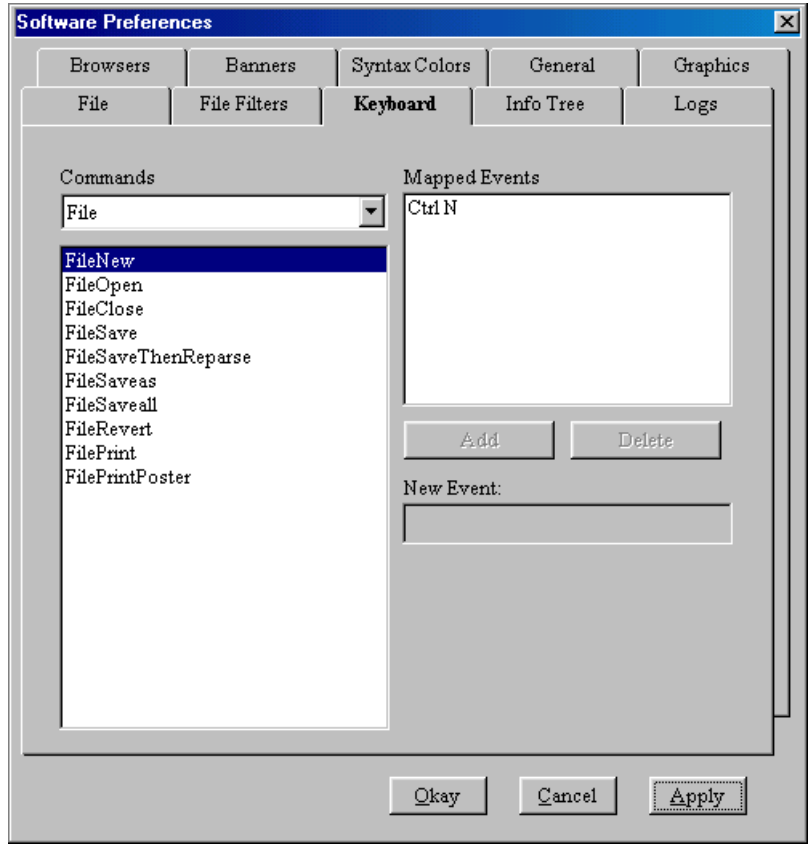
Key Mappings

The functions of keys in *Understand for FORTRAN* can be customized. The default keyboard mappings affect all areas of the software, including the source editor.

For a complete list of the current key mappings, choose the **Help->Keyboard Mappings** menu item.

To change or add key bindings:

1 Choose the **Options->Preferences** menu item and go to the **Keyboard** tab.



2 Select a menu or category of commands from the **Commands** list. The categories include various menus and submenus from the menu bar, right-click commands for entities, and printing and non-printing ASCII characters.

3 Select a command name from the list.

4 Press the keystroke you want to map to the selected command. The current function for that keystroke (if any) is shown below the **New Event** field.

5 Click **Add** to map that keystroke to the selected command.

6 To remove a keystroke, select it in the **Mapped Events** list and click **Delete**.

7 Click **Okay** when you have finished making changes.

The default keyboard mappings include the following keystrokes that can be used in the Source Code Editor.

Keystroke	Action
Ctrl-c	Copy the selection to the system's clipboard. Marking with left mouse has this effect on X-Windows systems.
Ctrl-e	Open the Search/Replace dialog.
Ctrl-g	Open the Go To Line dialog.
Ctrl-f	Open the Find dialog (used to find text in the current file).
Ctrl-Shift-F	Find Next occurrence of a previous find.
Ctrl-m	Match brackets (see *Bracket Matching* on page 4–10).
Ctrl-n	Open a new empty file.
Ctrl-o	Open an existing file (opens file selection dialog).
Ctrl-r	Rebuild the current database by parsing only changed files (and those who depend on the changed files) Equivalent to clicking the Reparse icon of the toolbar.
Ctrl-s	Save file (if modified). Uses current name for the file. Brings up Save As dialog if file is not yet named.
Ctrl-tab	Move between open windows in Windows Multiple Document Interface (MDI) mode.
Ctrl-v	Paste clipboard text at cursor position. If text is currently marked it will replace that marked text with contents of the clipboard.
Ctrl-x	Cut selection into the clipboard.
Ctrl-y	Redo last event that was reversed by Undo

Keystroke	Action
Ctrl-z	Undo last editing event
Ctrl-Insert	Same as Ctrl-c. (copy to clipboard)
Shift-Insert	Same as Ctrl-v (Paste clipboard)
Shift-Delete	Same as Ctrl-x (Cut to clipboard)
Alt-Backspace	Same as Ctrl-z (Undo)
Delete	Delete current selection. Or delete (moving forward) character at the current cursor position
End	Move cursor to end of current line
Ctrl-End	Move cursor to end of file
Home	Move cursor to beginning of first word on line. If no words on line then beginning of line.
Ctrl-Home	Move cursor to beginning of file.
Page Down	Move cursor down one page.
Ctrl-Page Down	Scroll the horizontal scroll right.
Page Up	Move cursor up one page.
Ctrl-Page Up	Scroll the horizontal scroll left.
Left Arrow	Move cursor left one position.
Ctrl-Left Arrow	Jump to beginning of previous word.
Right Arrow	Move cursor right one position.
Ctrl-Right Arrow	Jump cursor to beginning of next word.
Down Arrow	Move cursor down one position, retaining column position if possible.
Ctrl-Down Arrow	Move vertical scroll bar down one position.
Up Arrow	Move cursor up one line, retaining column position if possible.
Ctrl-Up Arrow	Move vertical scroll bar up one position.

Other Features

The Source Code editor also provides bracket matching, the option to print files, and several options for displaying files.

Bracket Matching

A handy feature of the *Understand* editor is syntax bracket matching. Use this feature to find the matching ending bracket of syntactically used braces, parenthesis and brackets. Symbols matched are (,),{,},[,]. Matching isn't done inside comments.

There are two modes. The first is the use of the keystroke CTRL-M to "match" a brace, parenthesis, or bracket that the cursor is over. CTRL-M isn't active unless over a symbol that it can match. CTRL-M then jumps the editor to the matching end or beginning brace. Another CTRL-M takes you back where you started. Try it - most programmers who try it come to depend on it.

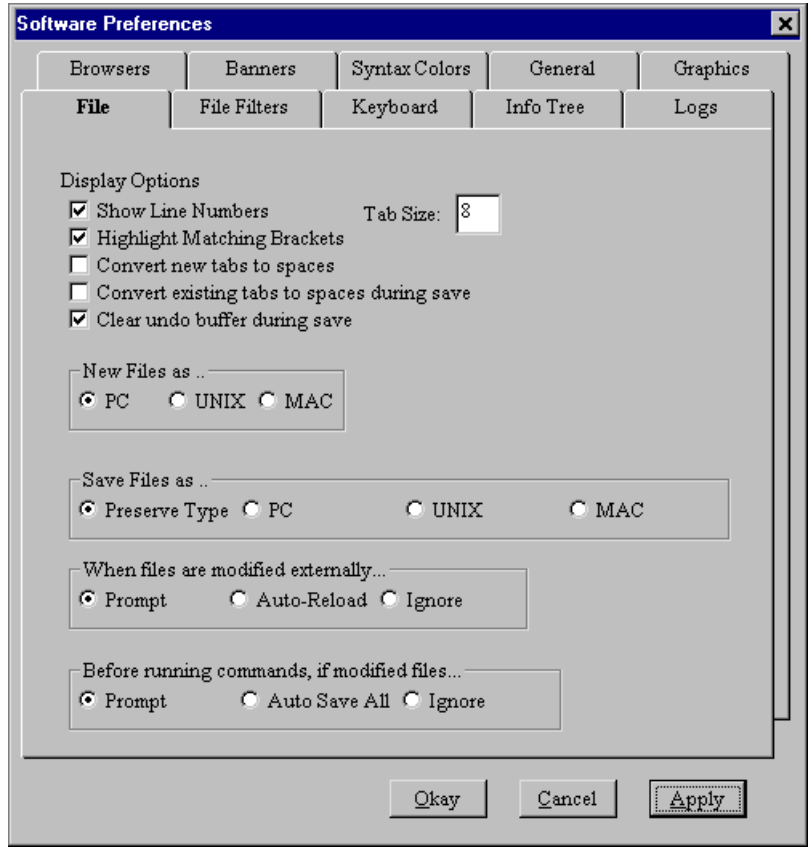
Another variant of bracket matching is that when enabled each time you type one of the bracketing symbols the editor will quickly highlight the entire region being bracketed. This too is handy for quickly and efficiently entering syntactically correct code that does what you hope it does.

Printing Source Files

The menu option **File->Print** will send the currently viewed source file to the printer. The printout will use 66 lines per page. As with other printing, the Windows driver setup is used on Windows and see *Printing Graphical and Source Views* on page 3–27 for details of printing on UNIX machines.

File Display Options

In addition to Line Numbers many things can be controlled from the **File** tab of the **Options->Preferences** dialog:



- **Show Line Numbers** - check (the default) to turn on line numbers in the source view
- **Highlight Matching Brackets** - check to enable matching of brackets. See description *Bracket Matching* on page 4–10.
- **Convert new tabs to spaces** - check to have any new tabs replaced with spaces while editing.
- **Convert existing tabs to spaces during save** - check to have any existing tabs replaced with spaces when the file is saved.
- **Clear undo buffer during save** - check (the default) this box to discard the history of changes when saving the file. If this box is checked, you cannot use Edit->Undo to remove changes after saving the file.

- **Tab size** - the number of blank spaces to use when showing tabs or replacing with spaces.
- **New Files as...**. - Sets the type of file to create when making a new file.
  - **PC (Microsoft Windows)** line-endings are terminated with a combination of a carriage return (\r) and a newline (\n), also referred to as CR/LF.
  - **UNIX** line-endings are terminated with a newline (\n), also referred to as a linefeed (LF).
  - **Macintosh** line-endings are terminated with a single carriage return (CR).
- **Save Files as** - whether to force a given file type or to preserve the type the file have before editing. The default is to preserve the type.
- **When files are modified externally....** - If an open file is changed through some other program, *Understand* will detect this. Choose **Prompt** if you want to be notified and asked to load that changed version. **Auto-Reload** does this without prompting. **Ignore** is dangerous and not recommended.
- **Before running commands, if modified files...** - If a file has been modified, this setting controls the action performed when you run a command. The options are to **Prompt** you for whether to save, to **Automatically save all** modified files, or to **Ignore** modified files.

Chapter 5 Searching Your Source

This chapter covers how to use *Understand for FORTRAN*'s Find in Files and Locator Window features to locate thing in your source code.

This chapter assumes a moderate understanding of the FORTRAN programming language and an understanding of using menus under Windows or X-Windows.

This chapter contains the following sections:

Section	Page
Searching - an Overview	5–2
Locator Window - Find or Browse Entities	5–3
Using Find in Files	5–10

Searching - an Overview

Finding things in large bodies of source code can be difficult, tedious, and error prone.

Understand for FORTRAN offers three solutions for finding things:

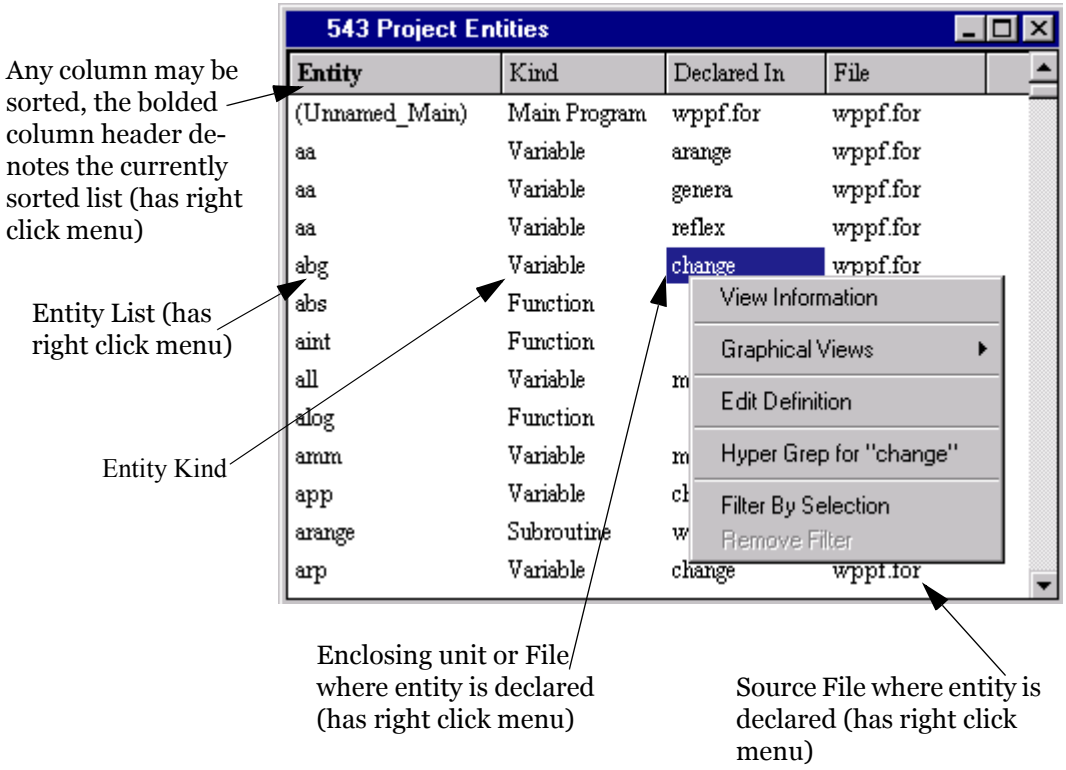
- Single file searching in the Editor. See *String Searching* on page 4–5.
- Project wide, entity only, searching using the Locator Window. This kind of search finds only entities (not strings, or comments, or non-syntactically declared or used items). See *Locator Window - Find or Browse Entities* on page 5–3.
- Project wide, text-based searching using Find in Files. See *Using Find in Files* on page 5–10.

Each of these searching methods has advantages and disadvantages. Together they provide powerful ways to easily find what you need to find to better understand and modify your code.

Locator Window - Find or Browse Entities

The *Locator Window* is used to find entities that have been declared or used in the analyzed source code. It is launched from the **Search** menu.

- **Search->Locate Entities** displays the *Locator Window* and also pops up a *Filter* dialog where you can specify a text string or regular expression in which you wish to filter the entity list.
- **Search->Browse** Entities displays the *Locator Window* without the *Filter* dialog.



As in other windows in *Understand for FORTRAN*, when right clicking on an entity anywhere in the *Locator Window*, a menu of additional information available for the item appears.

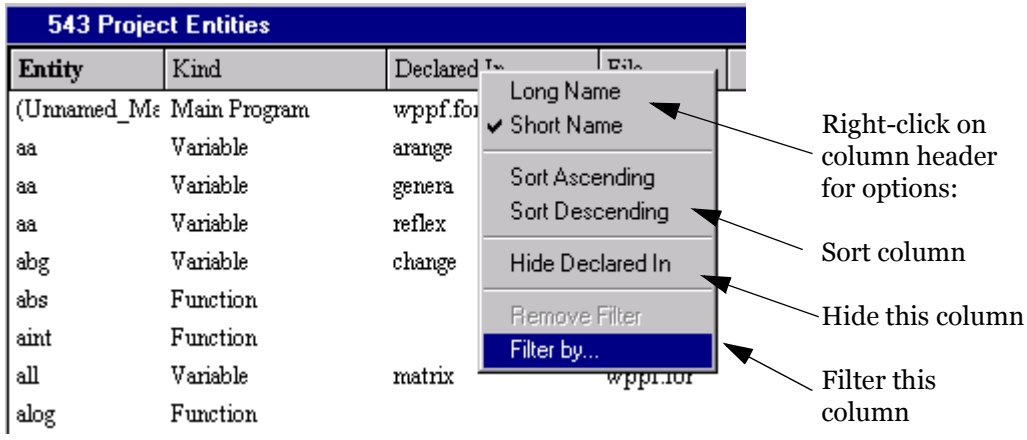
Right-Click Menus

Right-click menus are available in many views in *Understand for FORTRAN*, including the *Locator Window*. By right-clicking on an item in the *Locator Window*, a menu of additional information available for the item will appear.

Right-click menus are also available for items in other views, including the Hierarchy, Declaration, and Source Editor views.

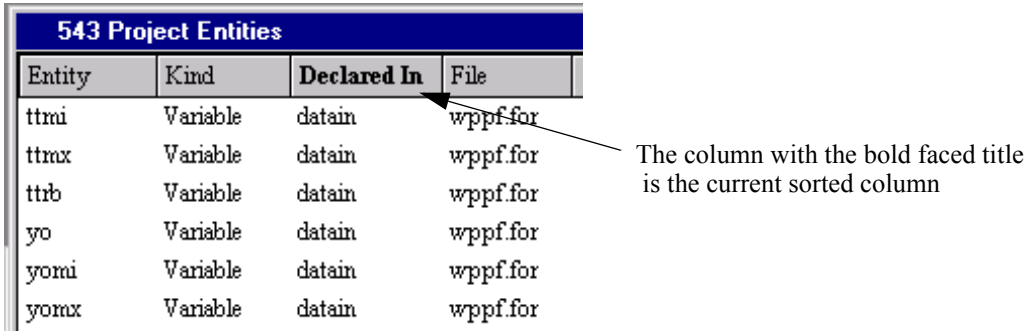
Column Headers

Column headers are tools in the *Locator Window*. Left-click them to sort, right-click them to filter or change their configuration.



Sorting Columns

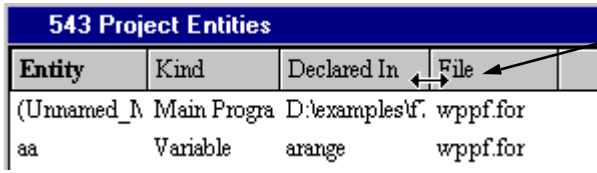
Any column in the entity list may be sorted if desired. Right-click on any column header for sorting options. Left-click on the column header to toggle between sorting in ascending order and descending order. The column header of the sorted column will appear in bold type. The default sorting order is in ascending order of entity names.



Resizing Columns

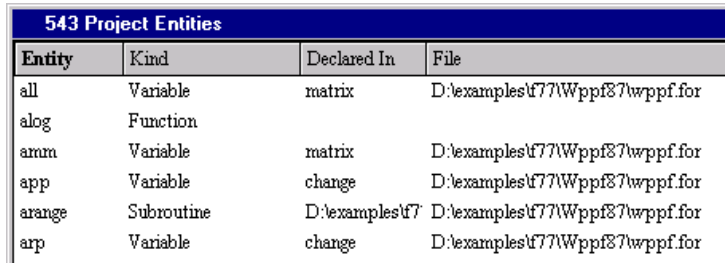
Column widths can be sized to adjust how much of each column is visible. Place the cursor on a column header divider between two columns. Double-click on the column header divider while the double-headed arrow is displayed and the field will be expanded to the maximum size needed for viewing all items in the list.

Column width can also be adjusted by clicking and dragging the double-headed arrow to the desired column width.



Long versus Short Names

In the *Entity*, *Declared In* and *File* columns, you can also specify viewing of short or long names. Long names include the name of the compilation unit for entity and function names or the full path of the file. Following is an example that shows long entity names.



Hiding and Re-ordering Columns

In the *Kind*, *Declared In*, and *File* columns, you can also choose to Hide one or more of those columns from view. Once a column is hidden, choose Show Column to redisplay it. You can also use the Hide/Show features to re-order the columns if desired.

Filtering

The right-click menus on cells in each column also have an option to set a filter limiting the entities shown by the *Locator Window*. The filter can be entered manually or automatically based on what was right clicked on.

For example, you may filter by the *Kind* "Parameter" by right-clicking on any parameter listed in the *Kind* column and selecting **Filter By Selection** from the menu. This filters the list of entities to contain only those entities which are parameters. Any filter options selected in the main filter area of the Locator Window will remain in effect.

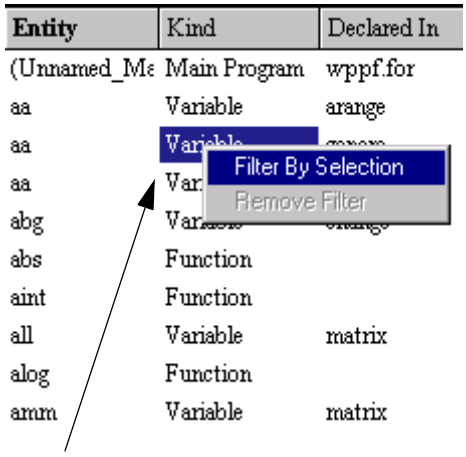
Only one **Filter** may be in effect at a time. Further **Filters** will replace any previous Filter. **Filter** can be performed on any item in the *Kind*, *Declared In*, or *File* Columns. Choosing **Remove Filter** can be selected from any column and removes the selected filtering previously placed on any other column.

Filter By Selection

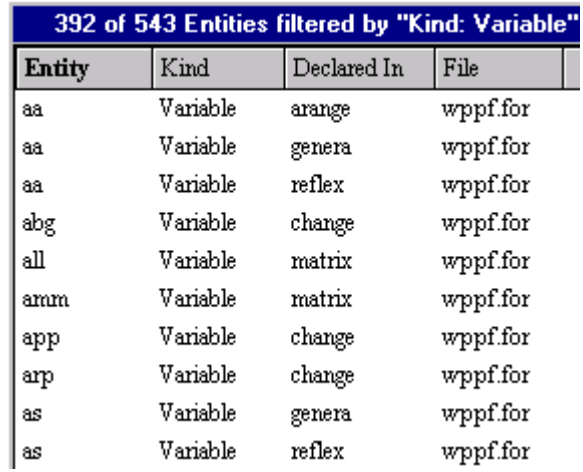
Filters may be set automatically by right clicking on any cell and choosing "Filter by Selection". For example, you may filter by the *Kind* "Variable" by right-clicking on any variable listed in the *Kind* column and selecting **Filter By Selection** from the menu. This filters the list of entities to contain only those entities which are variables.

Only one **Filter By Selection** can be applied at a time. Further actions of **Filter By Selection** will replace any previous Filtering By Selection. Choosing **Remove Filter** can be selected from any column and removes the selected filtering previously placed on any other column.

Following is an example showing **Filter By Selection** for an entity *Kind*



Right click and choose **Filter By Selection** in *Kind* column on cell with "variable" in it.

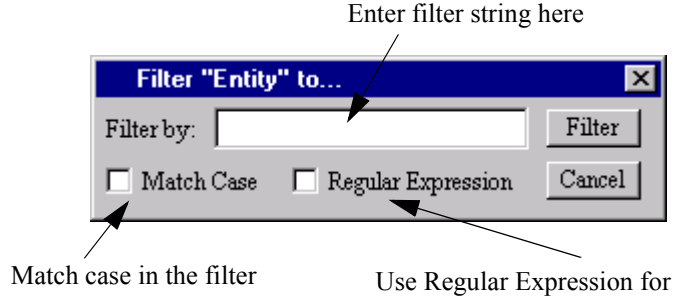


Now only variables are shown (still sorted by entity name).

Note that the new filter is described in the title bar

Filter Dialog Window

Filtering from the right-click menu is easy and quick. However it doesn't provide as much control as manually specifying a filter. To manually specify a filter, right click on the column header of the column you wish to filter on and choose "Filter By". This brings up the Filter By dialog.



The Filter dialog is used to limit the list of entities to just those of interest to you. The *Filter* dialog filters the Entity List based on textual filtering of the selected column.

You can open the Filter dialog by right clicking a column header and choosing **Filter by**. Choosing **Search->Locate Entities** from the main menu bar opens the Filter dialog for the Entity column.

- **Match Case -** Check this box to indicate case-sensitive matching is to be used in finding items in the entity list.
- **Regular Expression** - Check this box to select the use of *regular expressions* as the filtering mechanism. Enter the regular expression in the "Filter by:" field.

Wildcards Without Regular Expressions

The "Filter By" text field can be used to specify simple text patterns to match, or more complicated ones using Regular Expressions.

When "Regular Expression" is **not** checked, a simple subset of regular expressions can be used. These are * or ?, where * matches any string of any length and ? matches a single character.

- * matches any string
- B* matches any string beginning with uppercase B
- ??ext_io matches any entity name having 8 letters and ending in *ext_io*

Regular Expressions

When "Regular Expression" is checked, regular expressions are used. Regular expressions are a powerful and precise way to filter and manipulate text. You cannot use the Match Case option if you are using regular expressions.

The following table lists some special characters used in regular expressions.

Symbol	Description	Example
^	Match at the beginning of a line only.	^*word* Finds lines with w in the first column.
$	Match at end of a line only.	*word$* Finds lines that end with "word" (no white space follows word).
\<	Match at beginning of word only.	\<*word* Finds wordless and wordly but not fullword or awordinthemiddle.
\>	Match at end of word only.	\<*word* Finds keyword and sword but not wordless or awordinthemiddle.

Symbol	Description	Example
.	A period matches any single character.	*w.rd* Finds lines containing word, ward, w3rd, torward, and so on, anywhere on the line.
*	Asterisk matches zero or more occurrences of the previous character or expression.	*word** Finds word, wor, work, and so on.
+	Match one or more occurrences of the previous character or expression.	*wor+d* Finds word, worrd, worrrd, and so on.
?	Match zero or one occurrences of the previous character or expression.	*wor?d* Finds word and wod.
[ ]	Match any one of the characters in brackets but no others.	*[AZ]* Finds any line that contains A or Z. *[Kk][eE][Nn]* Finds any variation of case when spelling "Ken" or "KEn" or "keN".
[^]	Match any character except those inside the brackets.	*[^AZ]* Finds any line that does not contain the letters A or Z.
[-]	Match a range of characters.	*[A..Z]* Finds any line containing letters A through Z on them but not lower case letters
|	A vertical bar acts as an OR to combine two alternatives into a single expression.	*word | let+er* Finds word, leter, letter, lettter, and so on.
\	Make a regular-expression symbol a literal character.	**/$* Allows searching for *. This example finds all lines ending in */

A full explanation of regular expressions is beyond the scope of this manual. UNIX users may refer to the manual page for *regex* using the command "*man -k regex*". For a comprehensive explanation of *regex* expressions we refer you to the book "*Mastering Regular Expressions*", published by O'Reilly and Associates (www.ora.com/catalog/regex or 1-800-889-8969).

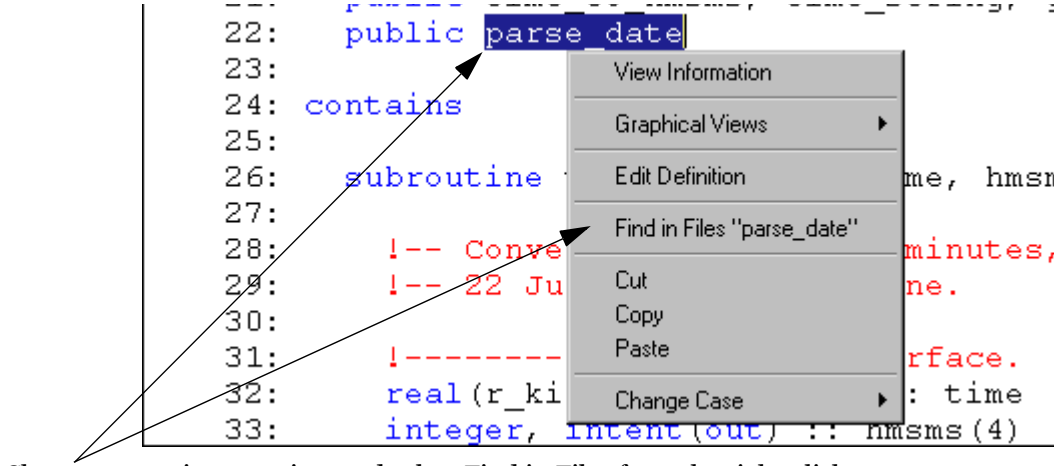
Removing Filters

A filter may be removed by right clicking on any column header and choosing "Remove Filter". This removes the current filter and causes the *Locator Window* to show all entities in the database.

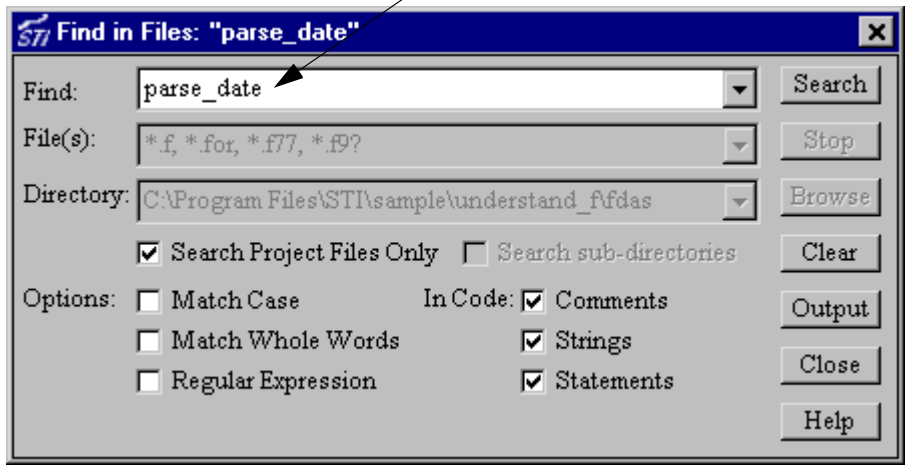
Using Find in Files

The Find in Files dialog allows you to search multiple files for the occurrence of a string. In previous versions, this feature was called Hyper Grep for its similarity to the UNIX command *grep*. The **Find in Files** function is available from all windows.

To open this dialog, choose **Search->Find in Files** from the menu bar, or choose **Find in Files** from any right-click menu.



Choose any entity, or string, and select Find in Files from the right-click menu. The Find in Files dialog is loaded with the selected string.

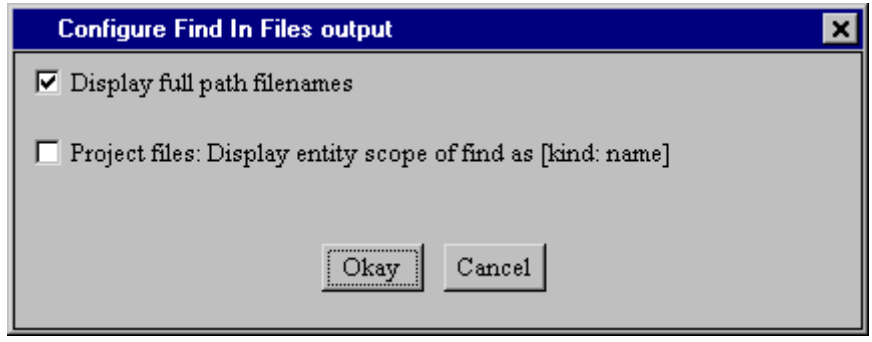


Controlling Searches

In the Find in Files dialog, specify the string to search for and what files to search. The Find in Files dialog also provides the following types of files to control the search:

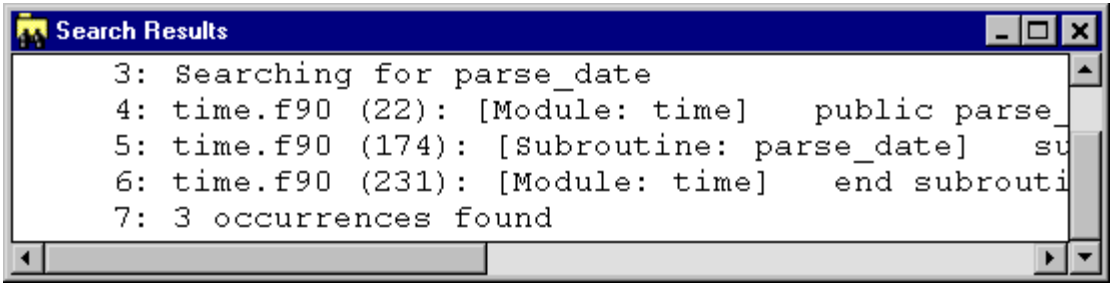
- **Files and Directory:** The search may be conducted on all files that have been included for analysis or on a subset of files that you specify. Multiple files may be specified by using wildcards or by specifying a list of files separating each file with a "," comma.
- **Options:** You can also choose to consider case sensitivity when matching, require matching on whole words, or use regular expressions for matching. For details on searching without regular expressions, see *Wildcards Without Regular Expressions* on page 5–8. For information on using regular expressions, see *Regular Expressions* on page 5–8.
- **In Code:** You can include or exclude searching of strings found in comments, strings, or code statements.

In addition, you can click the **Output** button to open the Configure Find in Files Output dialog:



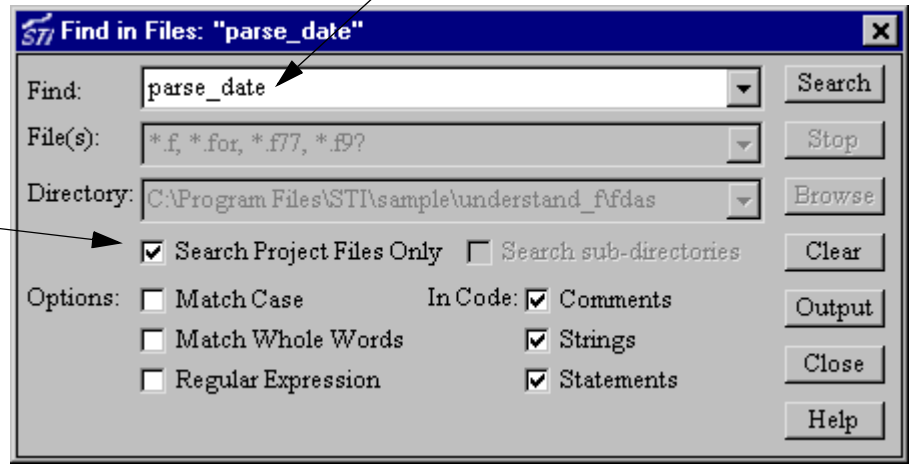
- **Display full path filenames:** If this box is checked, each item in the results shows the full file path. If your files are in a single directory or have unique names, you may want to remove the checkmark from this box to minimize the need for horizontal scrolling of the results.

- **Display entity scope of find as [kind: name]:** If you put a checkmark in this box, the results show the type and name of the entity that contains each result.



Press the *Search* button after specifying the search criteria and a list of all occurrences of the string will be displayed in the *Search Results* window.

Find string: may include wildcards.



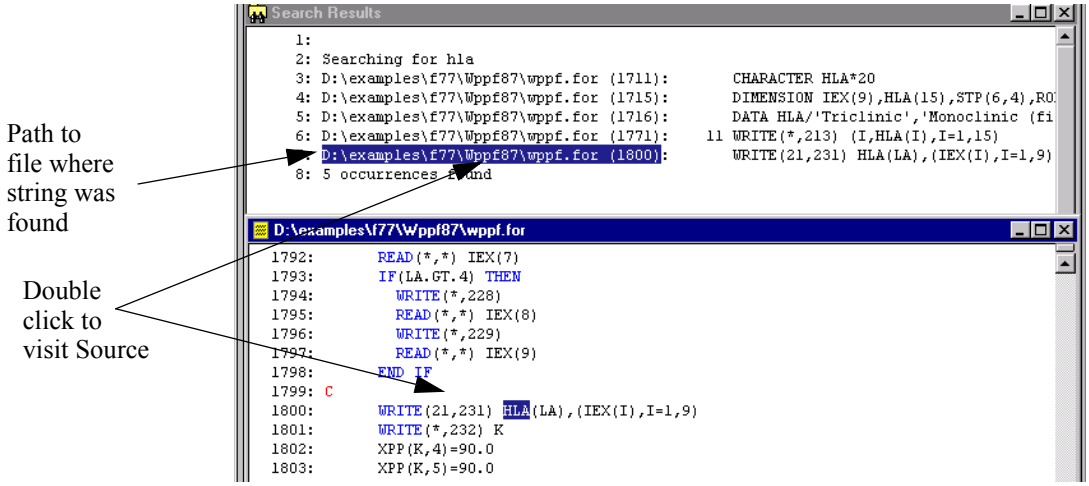
Search all project files if checked.

If not, search directory tree for files that match the pattern.

Search Results

The Find in Files *Search Results* window shows each occurrence of the string as it was found in the source line. Double click on any of the matching occurrences in the list, and the *Source Window* is loaded showing and highlighting that match.

The following shows a completed search of all project files for the string “hla” with the Source Window loaded with the selected matching occurrence.



Chapter 6 Text and HTML Reports

This chapter describes how to create and view reports and the types of reports available.

This chapter contains the following sections:

Section	Page
An Overview of Report Categories	6–2
Cross Reference Reports	6–6
Structure Reports	6–9
Quality Reports	6–12
Metrics Reports	6–16

An Overview of Report Categories

Understand for FORTRAN generates a variety of reports. The reports fall into these categories:

- **Cross-Reference** reports show information similar to that in the *Info Browser*, except that all entities are shown together in alphabetic order. For descriptions, see *Cross Reference Reports* on page 6–6.
- **Structure** reports show the structure of the analyzed program. For descriptions, see *Structure Reports* on page 6–9.
- **Quality** reports show areas where code might need to be examined. For descriptions, see *Quality Reports* on page 6–12.
- **Metrics** reports show basic metric information such as number of lines of code and comments. For descriptions, see *Metrics Reports* on page 6–16.

The following table shows the type and page number for each report.

Report Type	Report Name and Page
Cross-Reference	*Data Dictionary Report* on page 6–6
Cross-Reference	*Program Unit Cross Reference Report* on page 6–7
Cross-Reference	*Object Cross Reference Report* on page 6–7
Cross-Reference	*Type Cross Reference Report* on page 6–8
Structure	*Declaration Tree* on page 6–9
Structure	*Invocation Tree Report* on page 6–10
Structure	*Simple Invocation Tree Report* on page 6–11
Structure	*Include Report* on page 6–11
Quality	*Program Unit Complexity Report* on page 6–12
Quality	*FORTRAN Extension Usage Report* on page 6–13
Quality	*Implicitly Declared Objects Report* on page 6–14
Quality	*Unused Object Report* on page 6–14
Quality	*Unused Type Report* on page 6–15
Quality	*Unused Program Unit Report* on page 6–15
Quality	*Metrics Reports* on page 6–16
Metrics	*Project Metrics Report* on page 6–17
Metrics	*Program Unit Metrics Report* on page 6–17
Metrics	*File Metrics Report* on page 6–18

Augment with the PERL or C API

The reports included with *Understand for FORTRAN* have evolved over many years to accommodate common customer requests. However, we recognize that not all needs can be covered.

To help you develop custom reports we include both PERL and C interfaces to *Understand for FORTRAN* databases.

For details on the PERL interface see our web site:
http://www.scitools.com/perl.shtml

For details on the C API see its manual at
http://www.scitools.com/manuals/latest/

Both pages have a number of example programs and scripts.

Output Formats

Understand for FORTRAN reports are generated either as Text or as HTML files. On Windows, the ASCII text follows the DOS text file format (carriage return and line feed at the end of each line). On UNIX, text files are created according to the UNIX convention (lines end with a carriage return).

HTML reports are generated as HTML 3.0 format files. The generated HTML is not complex, the only HTML 3.0 (versus HTML 2.0) feature used is frames. Netscape 2.0 and higher, and Internet Explorer 3.0 and higher can display the files.

If you have specified multiple HTML files be generated per report in the Report Configuration dialog, the top of the HTML report will show the index of the report sections by displaying the first one or two characters character of the entity name in that section. These are also links to those files, so clicking on the desired index entry will take you directly to that page.

Report File Naming Conventions

File names of reports generated vary based on the type and format of the report generated.

For text files, a single text file containing all selected reports may be generated or separate files for each type of report may be generated. A single text file is named <*project_name*>.txt. For separate text files, the root of the file name is <*project_name*> with varying suffixes added to the file name to distinguish the type of report. The following table shows the file names used for each text report.

For HTML files, a single HTML file containing all selected reports may be generated or separate files for each type of report may be generated. HTML files are further broken down into separate files,

either alphabetically by entity name or in groups of N number of entities. An index file is also generated and contains links to all the other HTML reports generated. The report index file is named index.html.

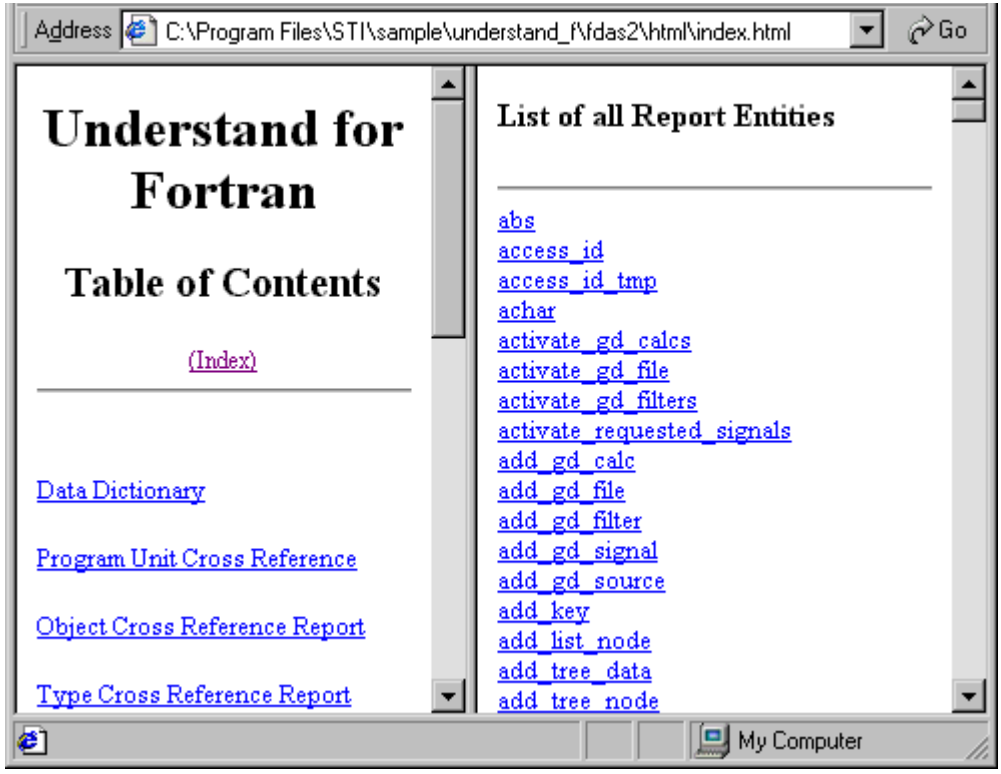
The following table shows the file names used for each separate HTML and text report. In the HTML Reports column, *n* is a letter from A-Z if "Alphabetic" was selected. If "Every n entities" was selected, *n* is a number beginning with zero.

Report Name	Alpha or Numeric HTML Reports	Text Reports
Data Dictionary	dictionary_*n*.html	*project*.dic
Program Unit Cross Reference	progunit_xref_*n*.html	*project*.pux
Object Cross Reference	object_xref_*n*.html	*project*.obx
Type Cross Reference	type_xref_*n*.html	*project*.tyx
Declaration Tree	decltree_*n*.html	*project*.dct
Invocation Tree	invocation_*n*.html	*project*.inv
Simple Invocation Tree	simpleinvtree_*n*.html	*project*.sit
Include Report	include_*n*.html	*project*.inc
Program Unit Complexity	progunitcomp_metrics_*n*.html	*project*.cmx
Project Metrics	projmetrics.html	*project*.jme
Program Unit Metrics	progunit_metris_*n*.html	*project*.pmx
File Metrics	file_metrics_*n*.html	*project*.fmx
FORTRAN Extension Usage	extension_usage_*n*.html	*project*.fex
Implicitly Declared Objects	implicitdeclobjs_*n*.html	*project*.imx
Unused Objects	unusedobject_*n*.html	*project*.qno
Unused Types	unusedtype_*n*.html	*project*.qnt
Unused Program Units	unusedprogunit_*n*.html	*project*.qnu

Searchable Entity Index

For HTML reports only, there is a single index file containing an alphabetical list of all entities found in all other generated HTML reports. The entities listed in the index have hyper links to the Data Dictionary report for that entity. The entity index file is named entity_index.html and can be accessed from the "index" link on the main HTML page.

The following figure shows an example of the entity index.



Generating Reports from the Command Line

HTML, text, and project metrics reports may be generated with the command line program *"repftn"*. Refer to *Generating Reports* on page 9–8 for details on using *repftn*.

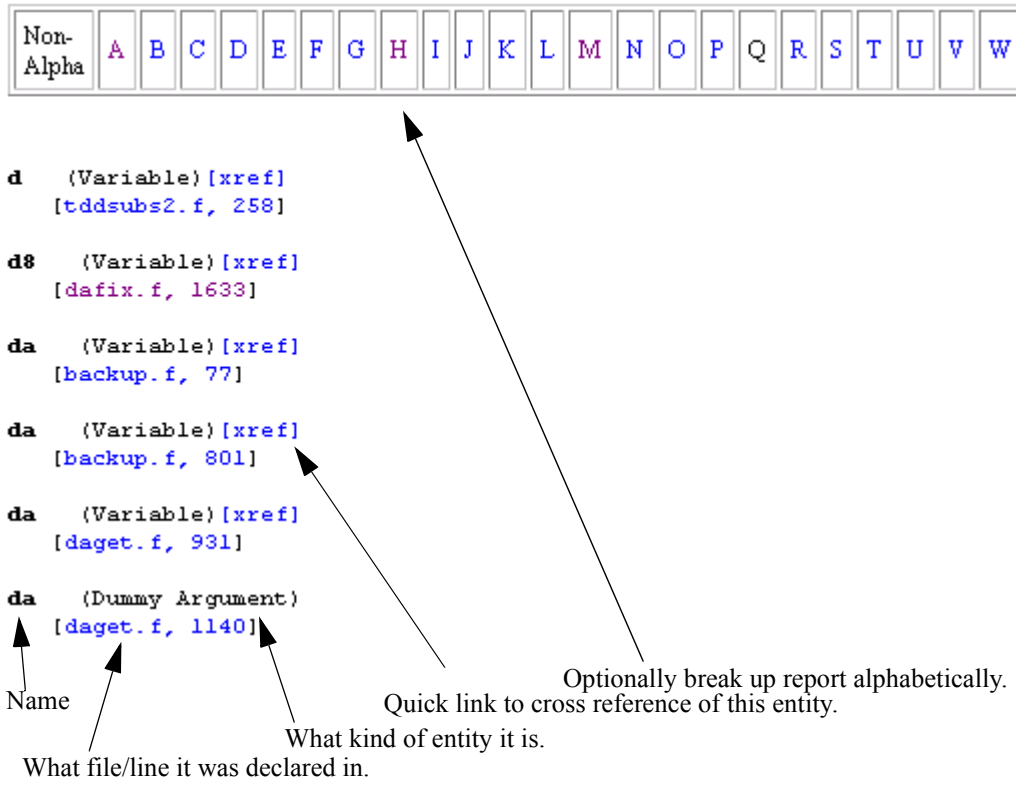
Cross Reference Reports

Cross-Reference reports show information similar to that in the References section of the Info Browser, except that all entities are shown together in alphabetic order. The following table shows the page that describes each type of cross-reference report.

Report Name
Data Dictionary Report on page 6–6
Program Unit Cross Reference Report on page 6–7
Object Cross Reference Report on page 6–7
Type Cross Reference Report on page 6–8

Data Dictionary Report

The *Data Dictionary Report* lists all entities alphabetically. Each listing shows the entity name, what kind of entity it is (*subprogram, type, variable, parameter, function, include file*), along with a links to the location where it is declared in the source code.



Program Unit Cross Reference Report

The *Program Unit Cross Reference Report* lists all subprograms analyzed into the library in alphabetic order along with information about what they return (if anything), what parameters are used, and where they are used by other program units.

The HTML version offers hyperlinks to the Data Dictionary report entry and to the source code where each reference occurs.

Program Unit Cross Reference

Non-Alpha	A	B	C	D	E	F	G	H	I	J	K	L	M

```
allocate     (Subroutine)
    Define    [allocate.f, 1]       allocate.
    Define    [tdds.f, 29]          tdds
    Call    [tdds.f, 108]           tdds
```

Object Cross Reference Report

The Object Cross Reference Report lists all objects (FORTRAN *variables, parameters, macros*) in alphabetic order along with declaration and usage references.

Object Cross Reference Report

Non-Alpha	A	B	C	D	E	F	G	H	I	J	K	L	M

Name, what kind of entity, and what it is declared as.

List of how and where the entity is used.

Enclosing program unit.

```
abmax     (Variable)
  Declared as: DOUBLE
    Define    [dafix.f, 24]          dafix
    Set     [dafix.f, 636]           dafix
    Use     [dafix.f, 640]           dafix
    Set     [dafix.f, 827]           dafix
    Use     [dafix.f, 828]           dafix
    Use     [dafix.f, 828]           dafix
    Use     [dafix.f, 828]           dafix
```

The HTML version of this report includes hyperlinks to the Data Dictionary Report and the source code where the reference occurs.

Type Cross Reference Report

The Type Cross Reference Report lists all declared types in alphabetic order, along with their declaration and usage information. The HTML version of the report offers hyperlinks to the Types data dictionary report entry, as well as the source code where the reference occurs.

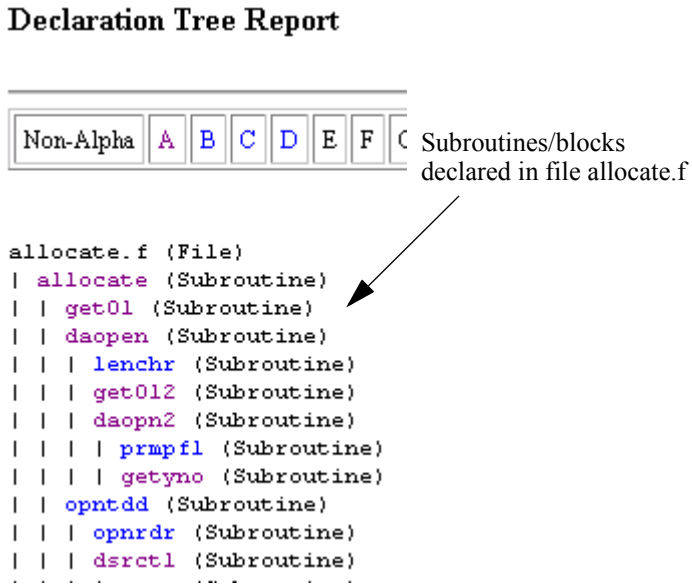
Structure Reports

Structure reports are designed to help you understand the relationships between various entities. The following table shows the page in this chapter that describes each type of structure report.

Report Name and Page
Declaration Tree on page 6–9
Invocation Tree Report on page 6–10
Simple Invocation Tree Report on page 6–11
Include Report on page 6–11

Declaration Tree

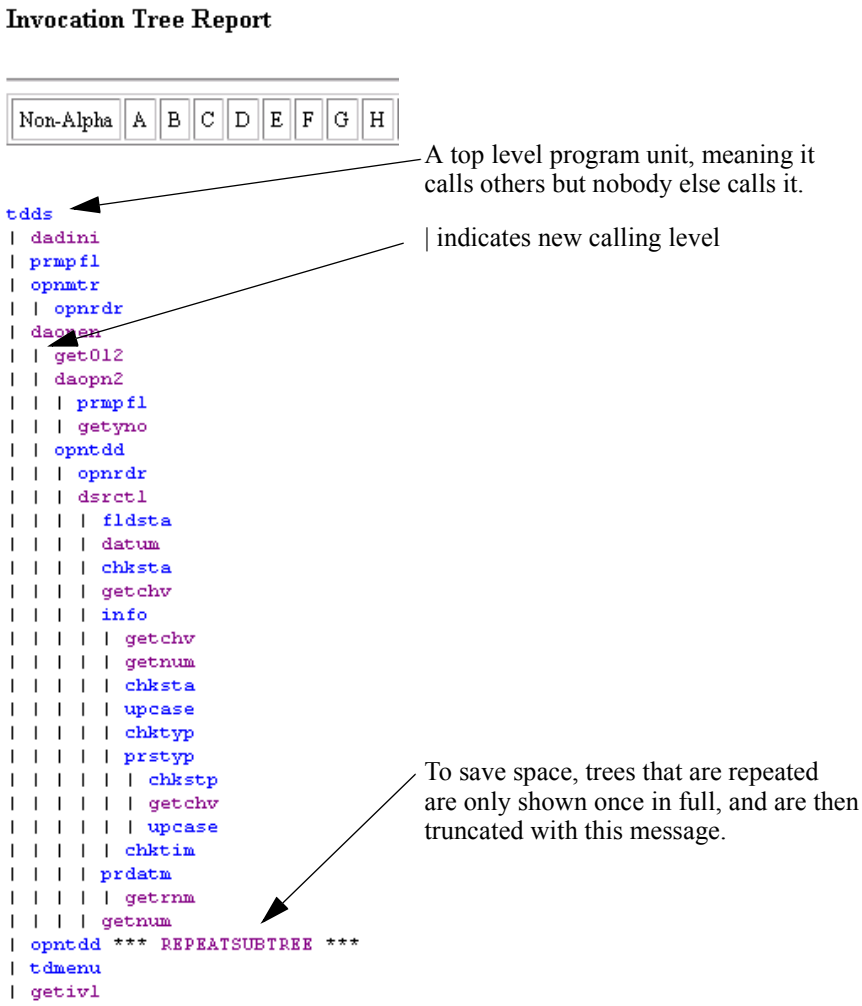
The Declaration Tree report shows a textual representation of an declaration tree for each FORTRAN file.



Invocation Tree Report

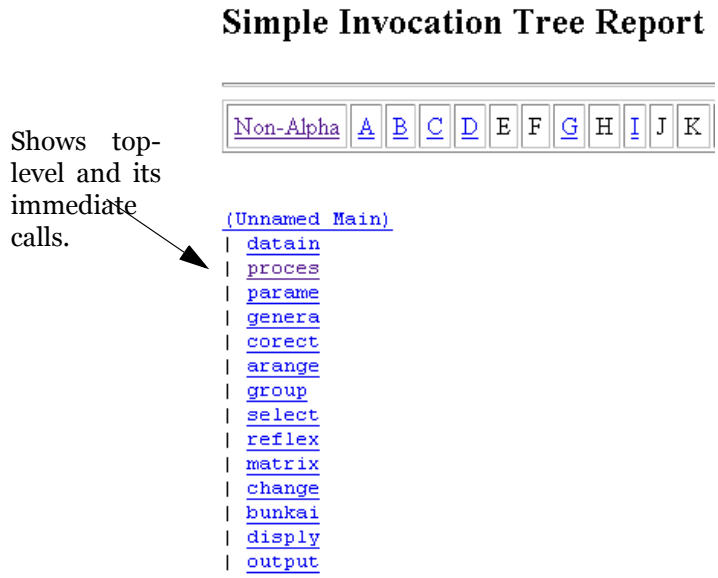
The Invocation Tree Report shows a textual representation of the full invocation tree for each subprogram analyzed. The report shows who each subprogram calls. Levels are indicated by tabs and are lined up with vertical bars. Each nesting level is read as “calls”.

The HTML version offers hyperlinks to the function’s Data Dictionary report entry.



Simple Invocation Tree Report

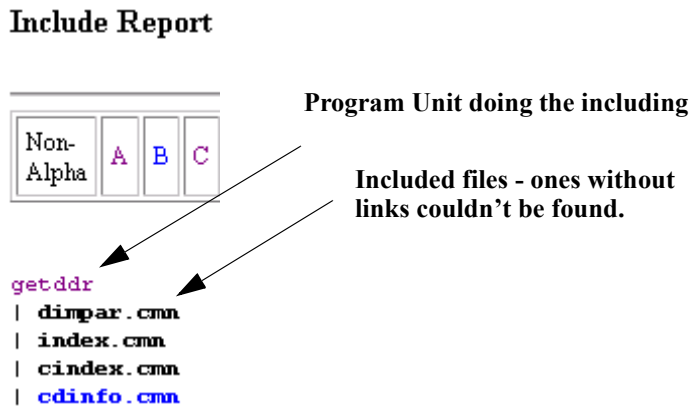
The Simple Invocation Tree Report shows the invocation tree to only one level for each program unit that has been analyzed.



The invocation level is indicated by an indent and a vertical bar and is read as “calls”.

Include Report

The Include Report lists all include files analyzed in the source code in alphabetic order with information about which files include them. The HTML version offers hyperlinks to the source code where each reference occurs.



Quality Reports

Understand for FORTRAN's quality reports are designed to provide information about areas of the analyzed source that might not meet standards or that hold the potential for trouble. They also identify areas where extra programming has been done but not needed. This sometimes identifies areas that aren't yet complete, or that haven't been maintained completely.

The following table shows the page in this chapter that describes each type of quality report.

Report Name and Page
Program Unit Complexity Report on page 6–12
FORTRAN Extension Usage Report on page 6–13
Implicitly Declared Objects Report on page 6–14
Unused Object Report on page 6–14
Unused Type Report on page 6–15
Unused Program Unit Report on page 6–15
Metrics Reports on page 6–16

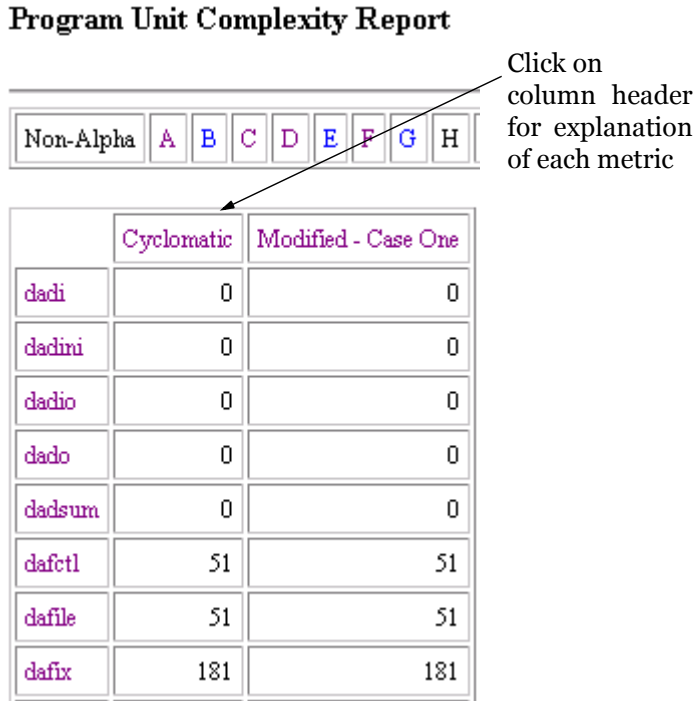
Program Unit Complexity Report

The *Program Unit Complexity Report* lists every subroutine in alphabetic order along with the McCabe (Cyclomatic) complexity value for the code implementing that subroutine.

The cyclomatic complexity is the number of independent paths through a module. The higher this metric the more likely a program unit is to be difficult to test and maintain without error.

The Modified column shows the cyclomatic complexity except that each case statement is not counted; the entire switch counts as 1.

The Strict column shows the cyclomatic complexity except && and || also count as 1.

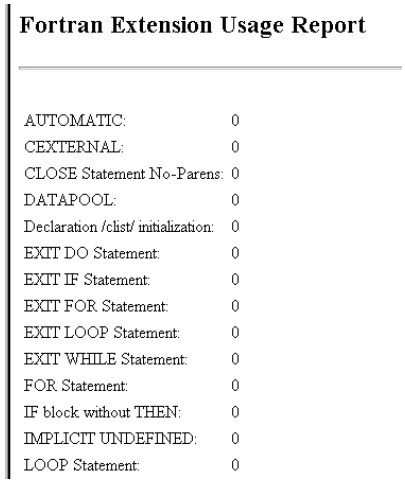


FORTRAN Extension Usage Report

This report lists anywhere your source code has non-standard FORTRAN extensions. The report factors in what variant (F77, F90, F95) you chose on your project configuration.

Extensions are explained in the glossary link of the report.

Following is a snippet from a sample FORTRAN Extension Usage report:



Fortran Extension Usage Report

AUTOMATIC:	0
CEXTERNAL:	0
CLOSE Statement No-Parens:	0
DATAPOOL:	0
Declaration /clist/ initialization:	0
EXIT DO Statement:	0
EXIT IF Statement:	0
EXIT FOR Statement:	0
EXIT LOOP Statement:	0
EXIT WHILE Statement:	0
FOR Statement:	0
IF block without THEN:	0
IMPLICIT UNDEFINED:	0
LOOP Statement:	0

Implicitly Declared Objects Report

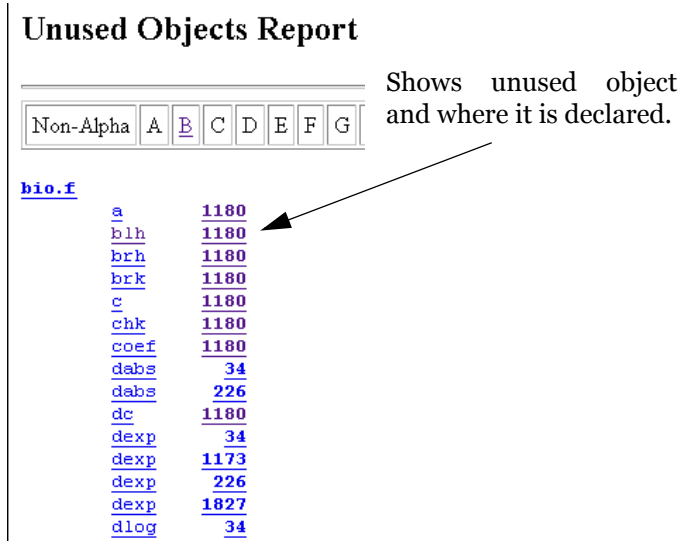
The *Implicitly Declared Objects Report* lists any variables or parameters that were implicitly declared using FORTRAN's implicit declaration mode. Using implicitly declared variables is considered a risky practice, and this report helps you weed out where the practice is occurring in your code.

The HTML version offers hyperlinks to the function's Data Dictionary report entry.

Unused Object Report

The *Unused Object Report* is formatted the same as the *Object Cross Reference Report*. However, only objects that are declared but never used are listed.

The HTML version offers hyperlinks to the function's Data Dictionary report entry and the source where the object is declared.



Unused Type Report

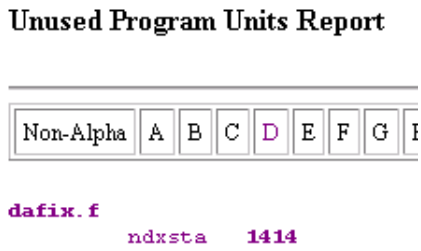
The *Unused Type Report* is formatted the same as the *Type Cross Reference Report*. However, only types that are declared but never used are listed.

The HTML version offers hyperlinks to the function's Data Dictionary report entry and the source where the type is declared.

Unused Program Unit Report

The *Unused Program Unit Report* identifies program units that are declared but never used.

Note that this listing in this report doesn't mean the system doesn't need this program unit. For instance, interrupt handlers that are called by system interrupts are often never "used" within the other source of the program.



Metrics Reports

Metrics reports show basic metric information such as number of lines of code and comments. The following table shows the page in this chapter that describes each type of cross-reference report.

Report Name and Page
Project Metrics Report on page 6–17
Program Unit Metrics Report on page 6–17
File Metrics Report on page 6–18

Reports can be generated in HTML and text formats. Since these reports are often read by those not having *Understand for FORTRAN* the metrics are also defined in the HTML reports.

Three Ways to Get Metrics Information

Understand for FORTRAN provides these ways to get metrics information:

- **Information Browser** - The last node on the Information Browser tree is Metrics. This branch can be expanded to show all the metrics available for the currently selected entity.
- **Reports** - These are described below. It is important to note that not all metrics collected are reported. Some are not shown in the reports to save space or because they do not fall into a particular report category.
- **Export -** All metrics may be exported from the **Project-> Metrics Export** menu. Metrics may be chosen. This output is comma-delimited, which allows you to import it into most spreadsheets and databases.
- **PERL/C API -** a more advanced way to get existing metrics, and also to calculate new metrics, is with the PERL and C API. These provide full access to the *Understand for FORTRAN* database. Examples and documentation can be found at http://www.scitools.com/perl.shtml

What Metrics are Available?

The complete list of metrics available in *Understand for FORTRAN* changes frequently - more frequently than this manual is reprinted.

A complete and accurate list is always available on our web site: http://www.scitools.com/metrics.txt

Project Metrics Report

The *Project Metrics Report* provides metric information about the entire project. The metrics reported are: the total number of files, the total number of functions, and the total number of lines of source code. This information is also reported on the main title page of the HTML report.

Project Metrics Report

Files:	24
Lines:	11483
Blank Lines:	1
Code Lines:	8324
Comment Lines:	3158
Declarative Statements:	1342
Executable Statements:	6820
Percent Comment:	37%

Program Unit Metrics Report

The *Program Unit Metrics Report* provides information on various metrics for each subroutine that has been analyzed.

The following metrics are provided for each subroutine:

- Total number of lines
- Total number of lines of code
- Total number of lines that contain comments
- Total number of blank lines

Note: code+comment+blank != lines
Some lines may contain both code and comments.

Program Unit Metrics Report

Non-Alpha	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

	Lines	Blank Lines	Code Lines	Comment Lines	Execution Statements	Declaration Statements
facscl	60	0	39	21	32	8

File Metrics Report

The *File Metrics Report* provides information similar to that in the *Program Unit Metrics Report*. However, it is organized by file rather than by *program unit*.

Click on each metric column to get a detailed description of it.

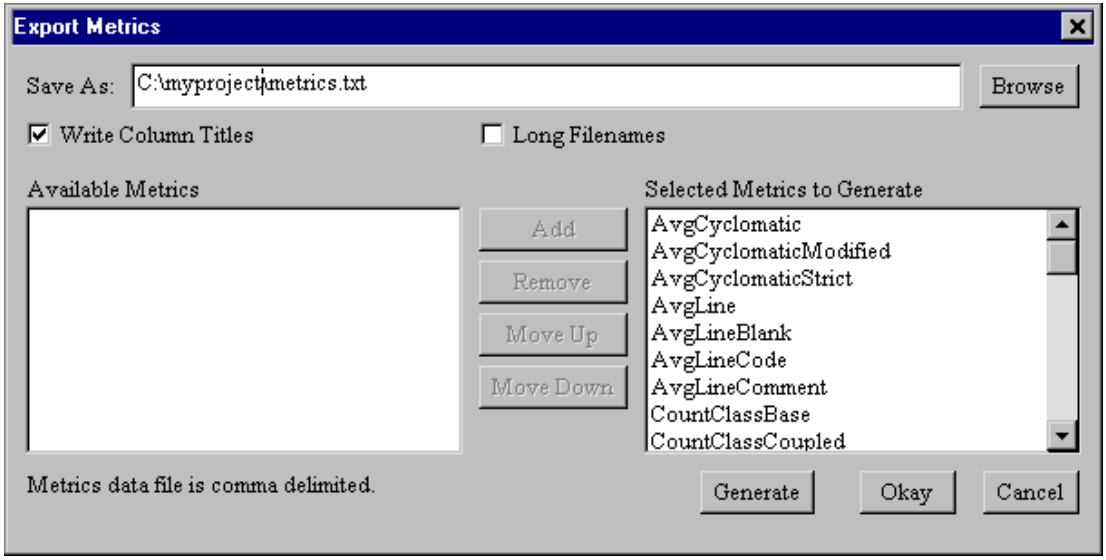
Note: code+comment+blank != lines
Some lines may contain both code and comments.

	Lines	Blank Lines	Code Lines	Comment Lines	Execution Statements	Declaration Statements	Mains / Subprograms	Avg. Complexity	Avg. Complexity Case 1	Percent Comment
allocate.f	65	0	40	25	12	34	1	10	10	62

Exporting Project Metrics Info

The Project Metrics Report provides metric information about the entire project. Project metrics can be saved to a comma-delimited text file which can be used in Excel and other spreadsheet programs.

Choose **Project**->**Metrics Export** to open the Export Metrics dialog.



Specify the file name for the metrics text file and whether column titles and full file paths are to be included in the file. Also select the metrics you want to include in the generated file.

After setting options, click **Generate** to export the file. The following is an excerpt of the comma-delimited metrics file produced.

```
Kind,Name,AvgCyclomatic,AvgCyclomaticModified,AvgCyclomaticStrict,AvgLines,AvgLinesBlank,
Function,alternate,0,0,0,0,0,0,0,0,13,13,17,48,5,43,3
Function,chimaera,0,0,0,0,0,0,0,0,19,19,19,95,6,83,13
Function,complain,0,0,0,0,0,0,0,0,1,1,1,8,0,8,0
Function,complain,0,0,0,0,0,0,0,0,1,1,1,9,0,9,0
Function,egsecute,0,0,0,0,0,0,0,0,5,5,7,17,1,16,0
Function,error,0,0,0,0,0,0,0,0,1,1,1,9,0,9,0
```

Chapter 7 Using External Editors and Other Tools

This chapter will show how to configure and use source code editors and other external tools from within *Understand for FORTRAN*.

This chapter contains the following sections:

Section	Page
Using an External Editor	7–2
Configuring Tools	7–4
Adding Tools to the Right-Click Menus	7–6
Adding Tools to the Tools Menu	7–8
Adding Tools to the Toolbar	7–9
Running External Commands	7–10

Using an External Editor

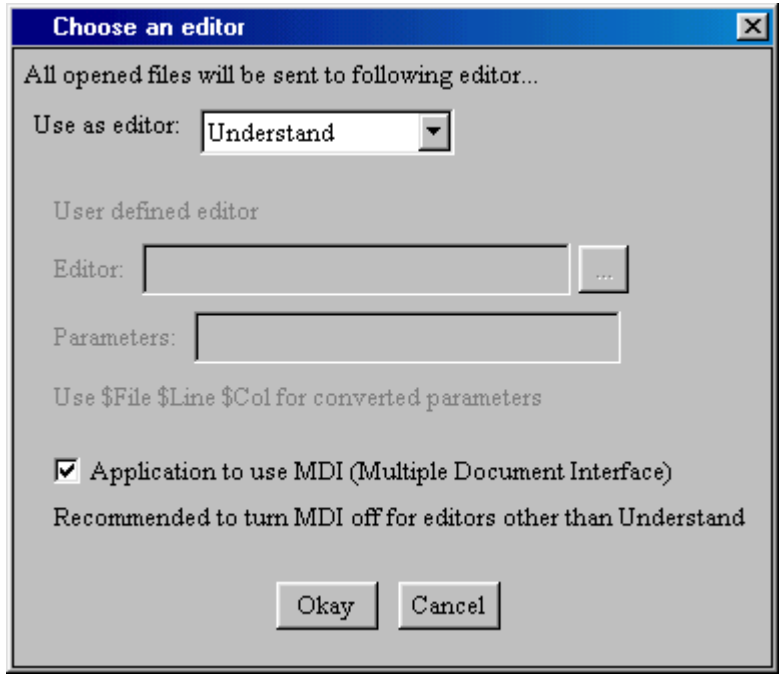
You can use an editor other than the one provided with *Understand for FORTRAN* for viewing and editing your source code. The editor you select is used whenever you open source code. This provides convenient source navigation while using a familiar editor. For example, you can use Microsoft Visual C++ or Emacs as your editor.

You should choose an editor that accepts command line parameters that specify the file to open, and a line and column number to go to.

To change the editor, follow these steps:

1 Choose **Options->Editor Selection**.

2 In the Choose an Editor dialog, select the editor you want to use. The default is "Understand". On Windows, you can select Msdev for the Microsoft Visual C++ editor. Other standard editors may be available in the drop-down list.



3 If your editor is not in the **Use as editor** drop-down list, choose "User defined editor". Then click the "**...**" button and browse for the editor's executable file. In the **Parameters** field, type the command line parameters you want to use when opening the

editor. Use the following $File, $Line, and $Col variables to allow *Understand for FORTRAN* to open source files to the correct location.

For example, for the GVIM editor on UNIX, the **Editor** is "gvim", and the **Parameters** should be as follows (for GVIM 6.0 or later):

```
--servername UNDC --remote +$line $file
```

For the TextPad editor on Windows, the **Editor** is most likely c:\Program Files\textpad4\textpad.exe, and the **Parameters** should be as follows:

```
$file($line,$col)
```

4 On Windows, we recommend that you remove the checkmark from the **Application to use MDI** box if you are using an external editor or using *Understand for FORTRAN* in server mode. If possible, turn off the Multiple Document Interface in your external editor.

If this box is checked, *Understand for FORTRAN*runs in Multiple Document Interface (MDI) mode. The main application window contains windows opened from within *Understand for FORTRAN*.

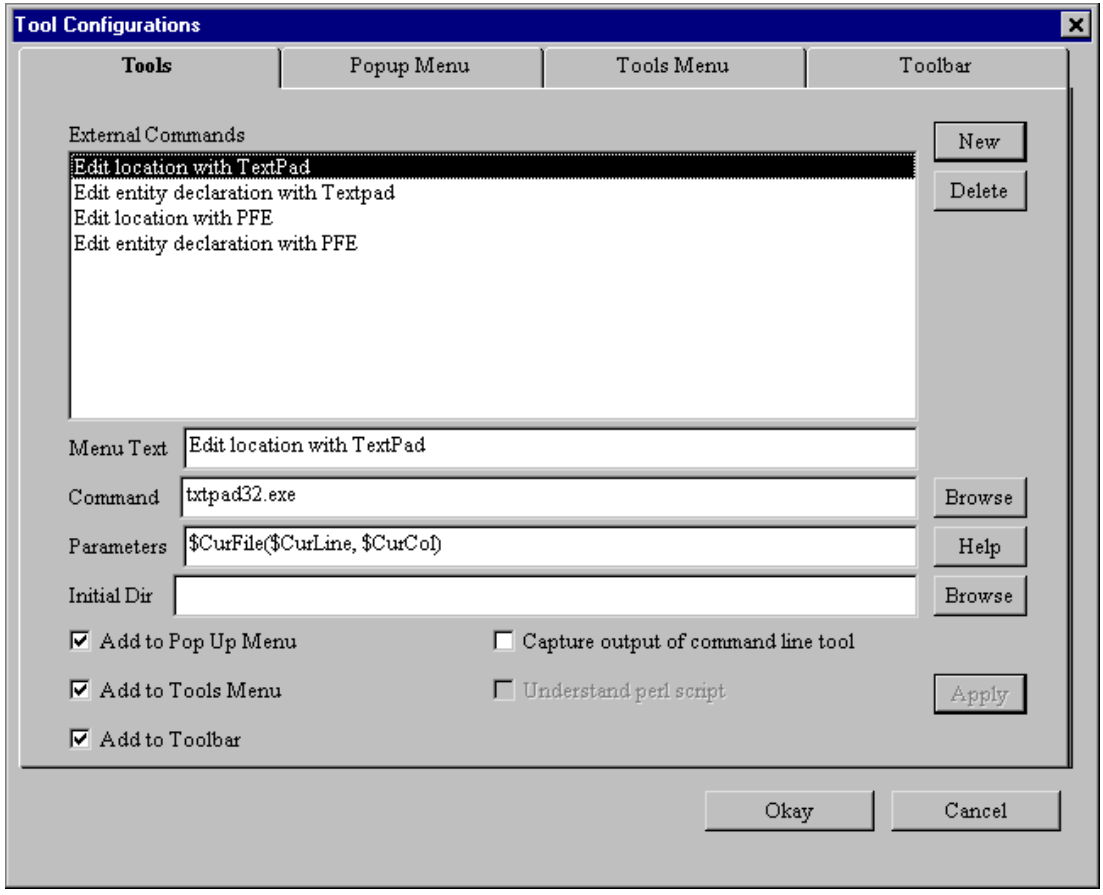
If this box is unchecked, *Understand for FORTRAN* runs in Single Document Interface (SDI) mode. In SDI mode, there is no single application window to contain the *Understand for FORTRAN* windows. Instead, each window is separate. In SDI mode, the Information Browser window has a "Stay on Top" push pin icon you can click to cause its window to stay on top of other windows. This is useful when using the Information Browser in conjunction with other tools as described in Chapter 8, "Server Mode: Controlling from Other Programs".

You will need to close and restart *Understand for FORTRAN* in order for changes to the MDI/SDI setting to take effect.

Configuring Tools

Select **Options->Tool Configurations** from the toolbar menu to configure external tools such as source code editors for use within *Understand for FORTRAN*. External tools configured for use will be available for context-sensitive launching from the right-click popup menus.

Use the **Tools** tab of the *Configurations* dialog to define the application command and parameters to invoke the tool. Also specify the desired name to appear on the right-click menus in *Understand for FORTRAN*. If the tool you use is on your executable search path, simply enter its name. If not, then use the *Browse* button to specify the full path to its executable.



Parameters beginning with $Cur are current position variables that are only available on menus inside a file display. Parameters beginning with $Decl are declaration variables that are only available for entities with an entity declaration.

Variable	Description
$CurProject	Current fullname location of opened project
$CurProjectDir	Directory the opened project is located
$CurProjectName	Current short filename of opened project (not including extension)
$CurReportHtml	Current fullname location of opened project's HTML Report
$CurReportText	Current fullname location of opened project's single file Text Report
$CurCol	Current file's column position
$CurFile	Current file's full path name
$CurFileDir	Current file's directory
$CurFileShortName	Current file's name without full path
$CurFileName	Current file's name not including extension or full path
$CurFileExt	Current file's extension
$CurLine	Current file's line position
$CurSelection	Selected text in the Current window (currently file windows only)
$CurWord	The word/text at the current cursor position in the Current file window
$DeclCol	Column in which the selected entity was declared, defaults to 1
$DeclFile	Full path name of the file in which the selected entity was declared
$DeclLine	Line in which the selected entity was declared, defaults to 1
$PromptForText	"text descriptor" before executing command, prompts user for this needed string parameter

Clicking the **Help** button displays the most up-to-date list of symbols that may be used for parameters.

To automatically add the tool to the right-click pop-up menu, check the “Add to Pop Up Menu” box. To automatically add the tool to the **Options->User Tools** menu, check the “Add to Tools Menu” box. To automatically add an icon for the tool to the toolbar, check the “Add to Toolbar” box.

Select *Apply* when completed to add the new command to the list. Selected commands may also be deleted from the list.

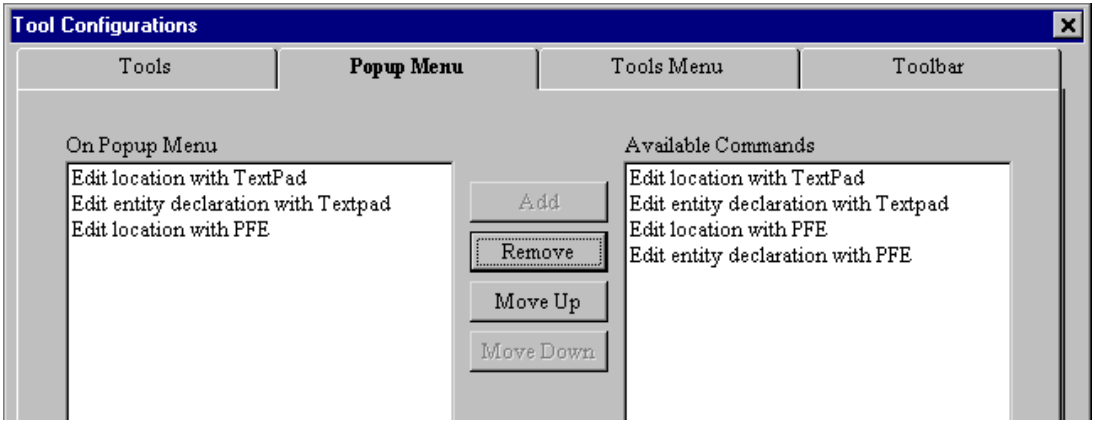
Adding Tools to the Right-Click Menus

Once the command is defined, the **Popup Menu** tab lists the available tool commands which you can selectively include or exclude from the right-click popup menu. The tools will be active or inactive on the right-click menu based on the context of the parameters provided to the tool.

Note: For example, a source editor which specifies $DeclFile as a parameter will be selectable from the right-click menu for any entity where the declaration is known, but will not be active for an undeclared entity.

The selected tools will appear on the right-click menu in the order in which they are defined here. Use the *Move Up* and *Move Down* buttons to order the tools as desired.

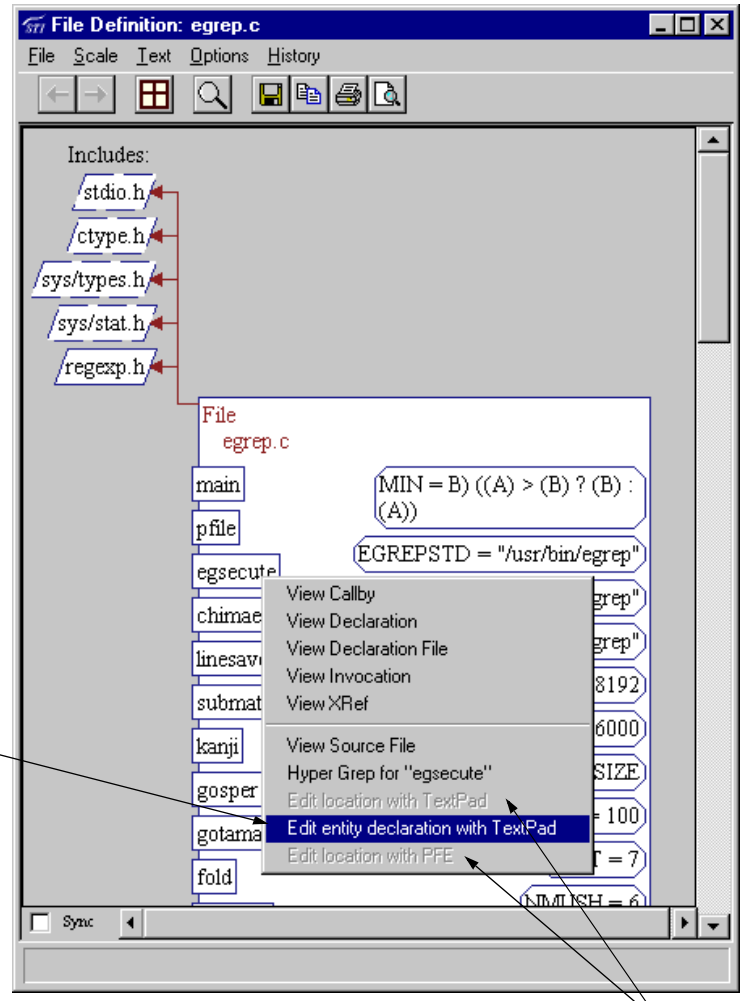
In the example below, three of the four tools defined will appear on the right-click popup menu.



The following figure shows a right-click menu for an entity showing the available external tools. In a declaration view, tools that reference where the entity is declared will be active.

External tools are available on the right-click popup menus.

Tools are available for selection based on context.

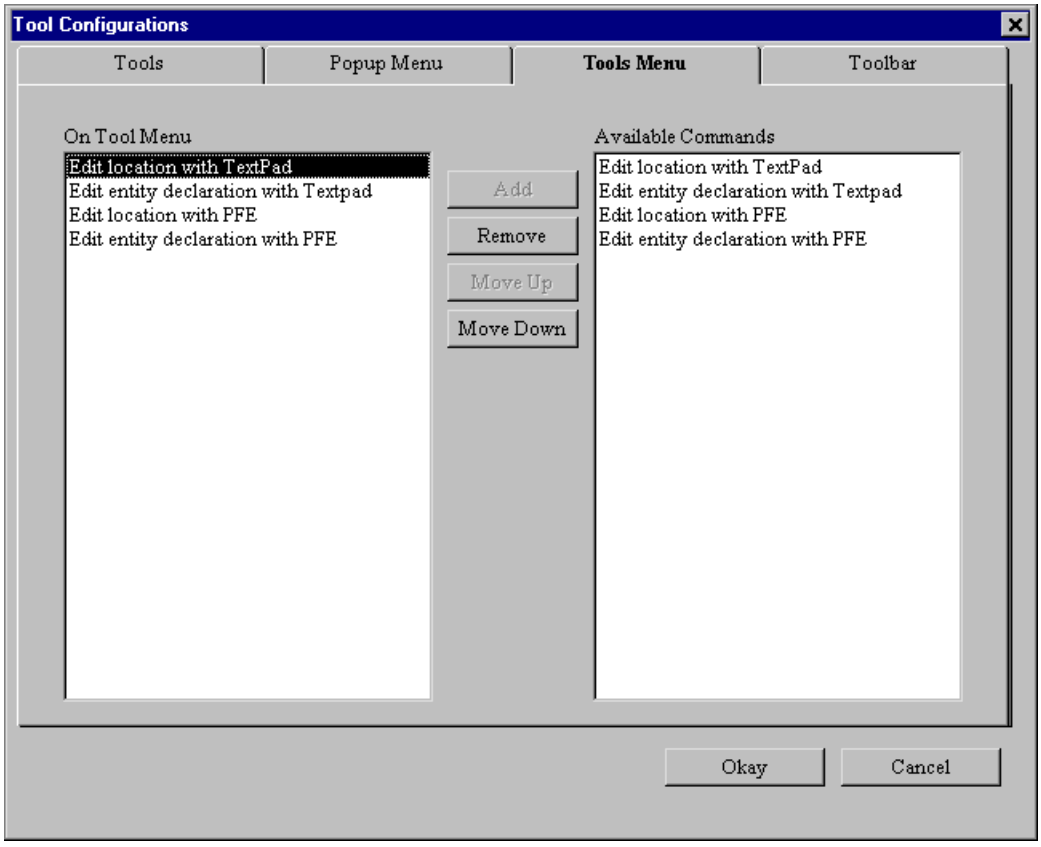


Tools which are not applicable for the given context are grayed out.

Adding Tools to the Tools Menu

To add a tool to the **Options->User Tools** menu, go to the **Tools Menu** tab. You can selectively include or exclude tools from the menu.

The selected tools appear on the menu in the order in which they are listed in the left box. Use the **Move Up** and **Move Down** buttons to order the tools as desired.

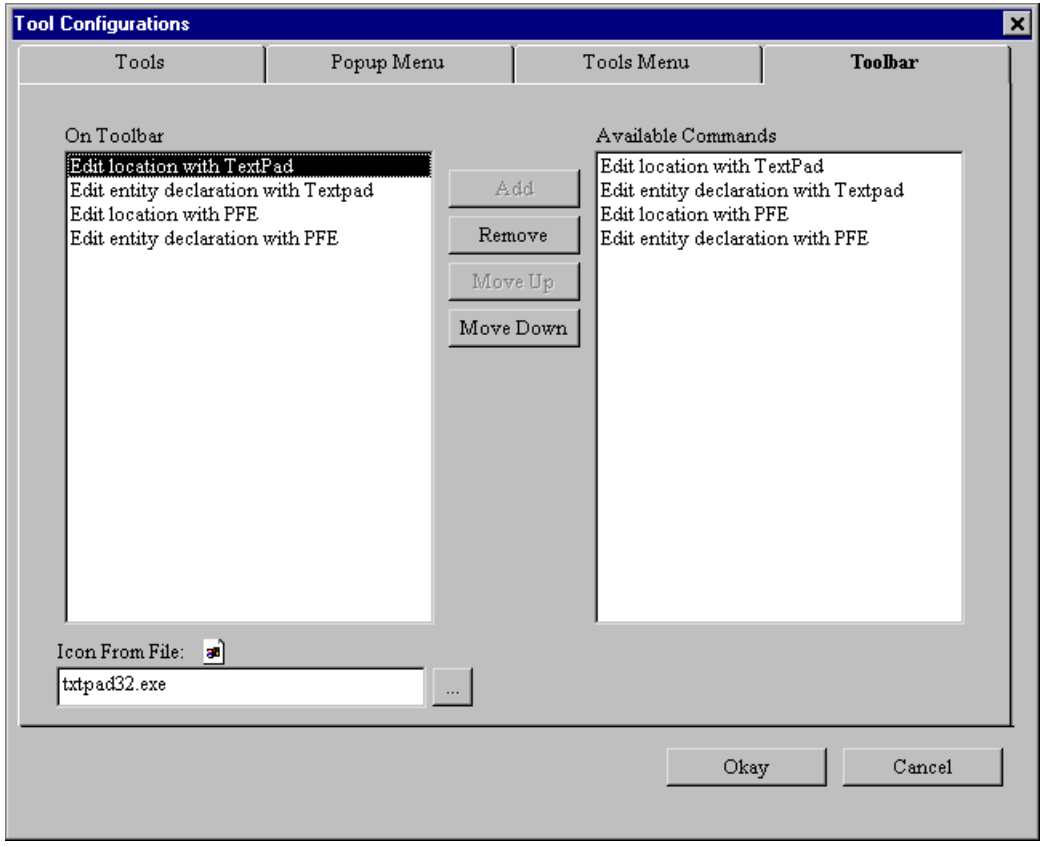


Adding Tools to the Toolbar

To add a tool to the toolbar, go to the **Toolbar** tab. You can selectively include or exclude tools from the toolbar.

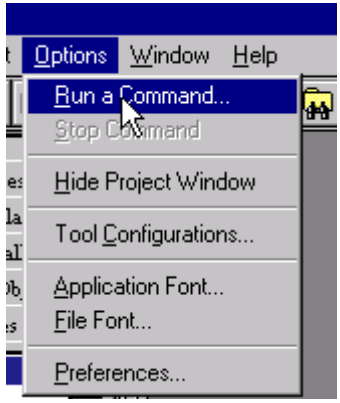
Icons for the selected tools appear on the toolbar in the order in which they are listed in the left box. Use the **Move Up** and **Move Down** buttons to order the tools as desired.

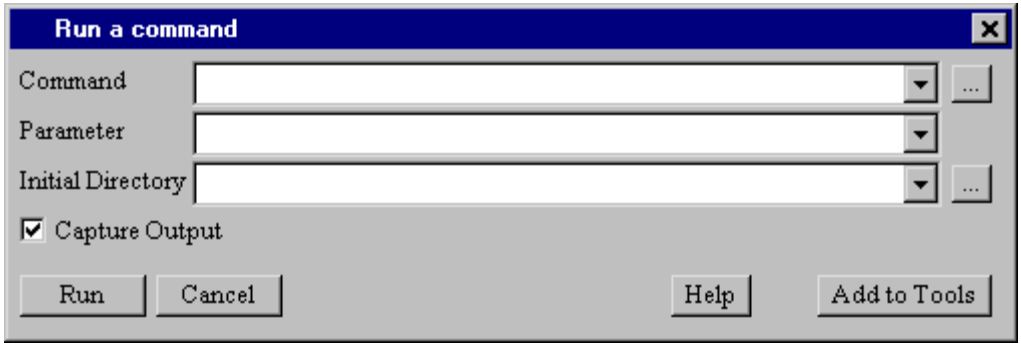
To change the icon for a particular tool, select that tool in the left box. Then click the "..." button near the bottom of the dialog and browse for an icon file (.ico) or executable file (.exe) that contains the icon you want to use.



Running External Commands

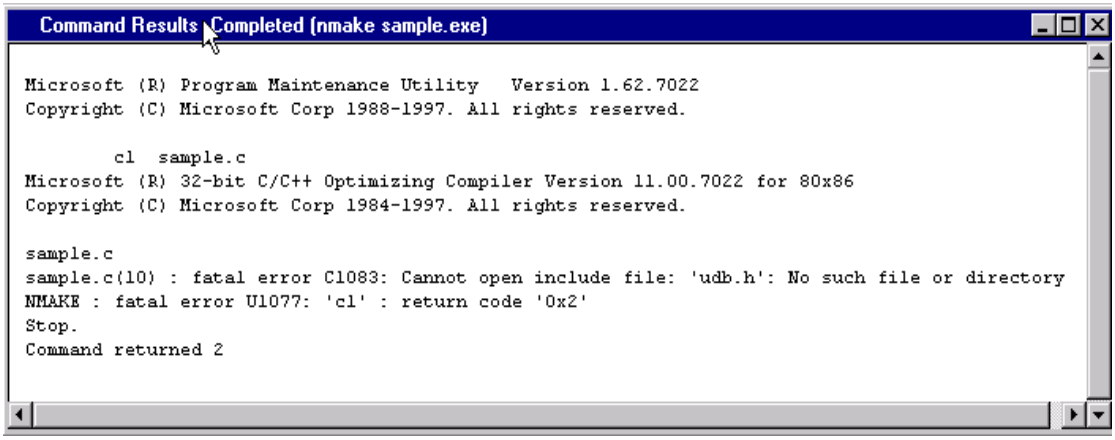
The **Options->Run a Command** menu item permits any external command to be run directly from *Understand for FORTRAN*. Common commands to invoke are compilers, configuration management tools, and PERL/C programs written using Understand's API.





While the command is executing, its output is shown in the Command Results Window if "Capture Output" is checked.

You can use the variables listed in *Configuring Tools* on page 7–4 in the Command or the Parameter field.



Note that double clicking on a line of output that has a project filename in it (optionally with a line #) opens Understand's editor to that file and line.

Tip: This works for most common output formats - if your command is generating a format we don't cover please send it to our support email address.

Chapter 8 Server Mode: Controlling from Other Programs

The chapter will show you how to control *Understand for FORTRAN* from other applications such as editors, compiler environments and debuggers. You may also write your own programs or scripts (running from the command line) that control *Understand for FORTRAN*.

The client program used in previous versions, is replaced entirely. The *Understand for FORTRAN* program itself is now used instead.

This chapter contains the following sections:

Section	Page
Understand Client	8–2
Action Commands	8–5
Communication Method	8–8
Editor Synchronization Example	8–9

Understand Client

Understand for FORTRAN listens for requests from other programs. The command line version of the program "*understand_f*" is used to send requests to the *Understand for FORTRAN* server.

A client request may be to tell the *Info Browser* and all synchronized view windows to show information about a particular entityname. Or a request may be to show a particular view of a specific subprogram or file. Or to configure or analyze a project, perform a Find in Files search, or to find an object.

The *understand_f* program accepts commands in the following forms:

- Perform an action on an entity using the current *Understand for FORTRAN* session:

```
understand_f -n[ame] entityname -file filename
  [-line line -col column] actionCommand
```

- Perform an action on an entity specifying the Understand project to use:

```
understand_f [-db database] [-new | -existing]
  -n[ame] entityname -file filename
  [-line line -col column] actionCommand
```

- Perform an action on a file entity:

```
understand_f [-db database] [-new | -existing]
  -efile entity_filename actionCommand
```

- Open any file (not project-dependent):

```
understand_f -openfile filename
```

Getting the Latest Options with -help

Since we do weekly builds of *Understand for FORTRAN*, it is quite likely that this manual may not describe all the options of the client/server interface. This is especially true if you have a printed manual.

Instead, the "-help" option will also be up-to-date and we encourage you to use it from time to time as an aid to creating tools with the client interface:

```
understand_f -help
```

On UNIX systems this will print out all the options.

On Windows systems, this will start an Understand session and print out the help text in the Command Results window.

Specifying the Project Database

When launching *understand_f* commands, the existing *Understand for FORTRAN* window can be used or a new *Understand for FORTRAN* window can be launched and a new project loaded.

To use the existing *Understand for FORTRAN* window and current project, omit the **-db** specifier on the command line. You can also specify **-existing** to override any possible user-customized preferences set in the current project. For example:

```
understand_f -existing -name myEnt
 -file file.f -line 37 -ib
```

loads the *Info Browser* of the current *Understand* session with the specified entity.

To launch a new instance of *Understand*, use the **-db** specifier. You can also specify **-new** to override any possible user-customized preferences. For example:

```
understand_f -db myproj.udf -new
-name myEnt -file file.f -line 37 -ib
```

launches a new session of *Understand*, loads the database myproj.udf, and shows myEnt in the *Info Browser*.

You can also use the **-last** option to open the most recently used project if no project is currently loaded.

You can use the **-demo** option to open the sample project provided with *Understand for FORTRAN*.

Specifying the Entity

To specify the entity to be shown, use the following command options:

If the entity is a project file:

```
-efile filename
```

Or, if the entity is not a file:

```
-name entityname -file filename [-line line -col
column]
```

where:

- **-efile** - Specify the name of the file entity you wish to learn more about. This is a required argument if the entity is a file.

- **-name** - Specify the name of the entity you wish to learn more about. This is a required argument if the entity is not a file. Partial names will not be matched.
- **-file** - Specify the name of a file where the entity is referenced. This is a required argument if the entity is not a file.
- **-line** - An optional argument indicating what line number the entity you want to learn about is found at. This further clarifies where you are looking at an entity. Again, most often used when *undestand_f* is launched from an editor or other source browsing program.
- **-col** - An optional argument indicating at what column position the entity you want to learn more about is at.

Following is a sample command line to load the *Info Browser* of an existing *Understand* session with the *ral2* entity, which is referenced in the file *wppf.for*:

```
understand_f -name ral2 -file wppf.for -ib
```

You can also specify an entity using the -filter and -jumpto options:

- **-filter** "*filter_tab"* - Selects the specifed tab in the Filter Area in the *Understand for FORTRAN* interface.
- **-jumpto** "*entityname or begins with string"* - Jumps to the first entity that begins with specified string in the current or specified (with the -filter option) tab of the Filter Area in *Understand for FORTRAN*. The -jumpto option sets the current entity for use by other command line options.

Opening a File

You can also load any file into the Source Editor. Specify **-openfile** and the path to the filename:

```
understand_f -openfile wppf.for
```

will open *wppf.for* in the current directory and load it into the *Source Editor* of the existing *Understand* session.

Action Commands

Many types of actions can be driven by *understand_f*. For example, -filter, -jumpto, -ib, -ibnew, -gv, -gvnew, -edit_src, -edit, and -editnew options can be used together to control how the interface appears when it opens.

- **-addfile** "*filename*" - Add the specified file to the specified or open database.
- **-delfile** "*filename*" - Remove the specified file from the specified or open database.
- **-edit_src** - Open the declaration source for the specified entity. This is the same as double-clicking an entity in *Understand*.
- **-edit "***menu_string***"** - Open the source of the specified entity for editing. The menu_string specifies which source for the entity is opened. Valid menu_strings are listed in the following table.

Entity Menu Strings for FORTRAN
File
Definition

- **-editnew "***menu_string***"** - Open the source of the specified entity for editing in a new source window. On Windows, this is available only in Single Document Interface (SDI) mode, which is described in *Using an External Editor* on page 7–2. The menu_string (listed for -edit) specifies which source to open.
- **-entitymenu** - Pop up the entity menu for the specified entity at the current cursor position. A **Dismiss Menu** item is added to the menu as a way to deselect the menu in the other application.
- **-filter** "*filter_tab"* - Selects the specifed tab in the Filter Area in the *Understand for FORTRAN* interface.
- **-gv** "*graphical_menu_string*" - Show the specified graphical view for the specified entity. See below for list of possible values for graphical_menu_string.
- **-gvnew "***graphical_menu_string***"** - Show the specified graphical view for the specified entity in a new graphic window. See below for list of possible values for graphical_menu_string.

Graphical Menu Strings for FORTRAN
Declaration
Callby

Graphical Menu Strings for FORTRAN
Declaration File
Include
Includeby
Invocation

- **-ib** - Load the *Info Browser* with the specified entity.
- **-ibnew** - Bring up a new *Info Browser* and load with the specified entity. This is the same action as occurs in Understand when the "Reuse" button is not checked.
- **-locator** - Opens the Locate Entities dialog. If -name is specified, it is used as the entity name filter.
- **-jumpto** "*entityname or begins with string"* - Jumps to the first entity that begins with specified string in the current or specified (with -filter) tab of the Filter Area in *Understand for FORTRAN*. The -jumpto option also sets the current entity for use by the -ib, -ibnew, -gv, -gvnew, -edit_src, -edit, and -editnew options.
- **-metrics** - Generates metrics for the open or specified project.
- **-refresh** - Refreshes the currently open or specified database. Has the same effect as the **Project->Analyze Changed Files** command.
- **-rebuild** - Rebuilds the entire open or specified database. Has the same effect as the **Project->Analyze All Files** command.
- **-reports** - Generates the currently specified set of reports. Uses the most settings in the Report Configuration dialog.
- **-suppress** - Hides any errors generated as a result of the command line.

Example Client Commands

Here are some example command lines:

- Pop up the entity menu for entity *genera* at the current mouse position:

```
understand_f -name "genera"
  -file wppf.for -entitymenu
```

- Edit the source file where entity *genera* is defined.

```
understand_f -name "genera"
  -file wppf.for -edit_src
```

- Load the *Info Browser* with the entity *genera:*

```
understand_f -name "genera"
  -file wppf.for -ib
```

- Show the graphical Invocation view for subprogram entity*genera*:

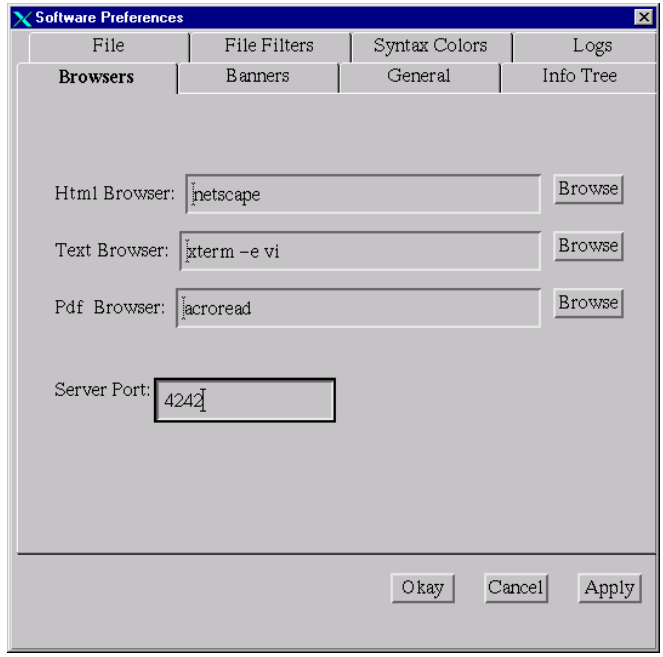
```
understand_f -name "genera"
  -file wppf.for -gv "Invocation"
```

Communication Method

The program *understand_f* uses different communication methods based on the operating system it runs on.

On Windows, Dynamic Data Exchange (DDE) is used.

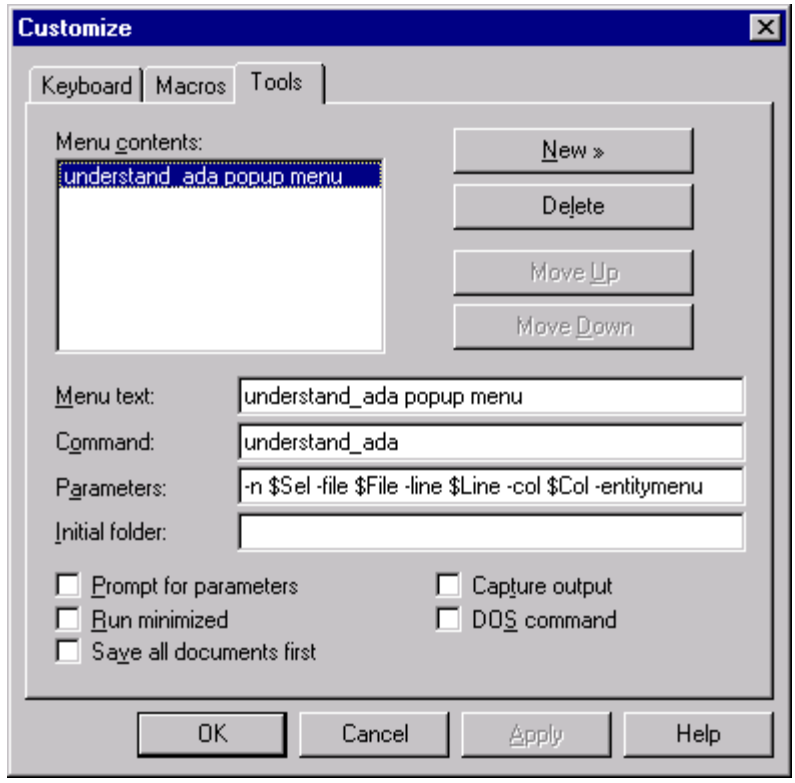
On UNIX, sockets are used to send information to the server. Since the possibility of conflicting with other programs using the same socket address is present, the server port used may be specified on the **Browser** Tab of the **Options->Preferences** menu.



If a conflict arises, change the Server Port to another number.

Editor Synchronization Example

This example launches *understand_f* from a text editor. In this case, the editor is the inexpensive (but very powerful) TextPad editor. It is available on the Internet at *http://www.textpad.com*.



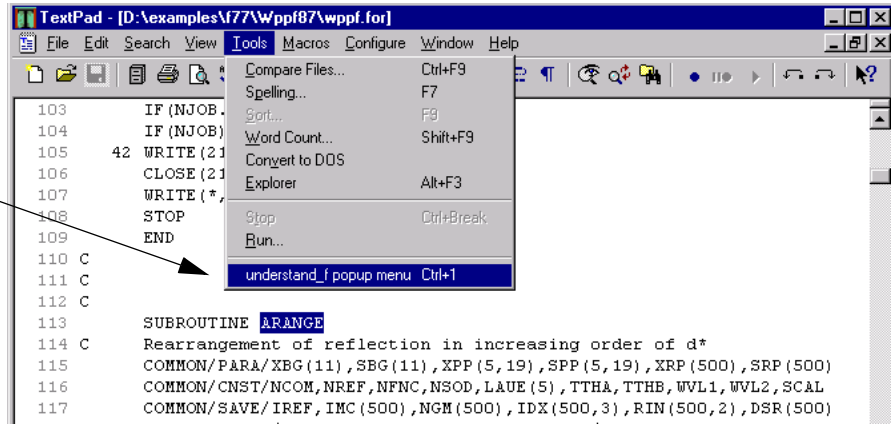
TextPad (and many other editors) offer internal variables that may be passed to external programs.

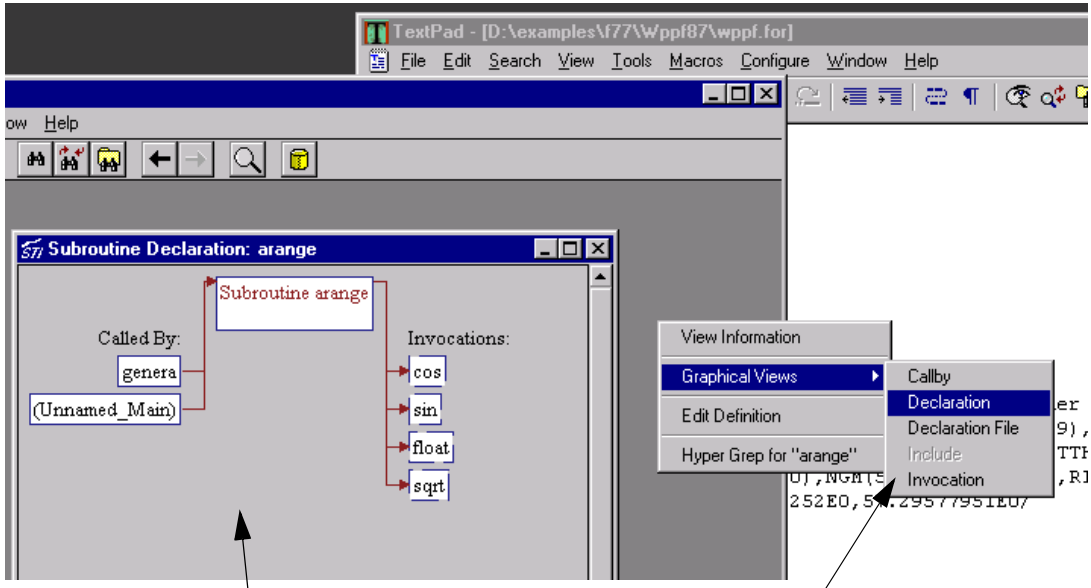
In this case we pass $Sel (text currently highlighted) as the entity name argument, $File as the filename, $Line as the line number, and $Col as the column where the entity can be found. In this example, the right-click menu of the entity is displayed at the current mouse position and whatever selection is chosen from the right-click menu will be loaded into the *Understand* window.

Now, after highlighting any entity in TextPad, we can choose the **Tools->Understand_f Popup Menu** option to activate

Understand's right-click menu for that entity. The selection from the right-click menu is then displayed in the *Understand* window.

Editing in another program and you want information about this entity. Simply highlight the entity press **CTRL+1** or select item from **Tools** menu.





Understand's right click menu for the editor-selected entity appears. Choose any item from the right click menu and the Understand window will automatically be loaded with the selected view.

Chapter 9 Command Line Processing

This chapter shows how to create an *Understand for FORTRAN* database by analyzing your source code, and how to generate reports from the command line. Command-line processing can be used in a batch file for automatic re-building and report generation of projects.

Two command-line programs are described in this chapter:

- *undftn*, which analyzes sources and creates *Understand for FORTRAN* databases
- *repftn,* which generates reports

This chapter contains the following sections:

Section	Page
Using undftn	9–2
Generating Reports	9–8

Using undftn

A command line tool for creating and building *Understand for FORTRAN* databases is *undftn*. This command line program parses your source code to populate the database with information that can be browsed or reported on.

A project database file (.udf) is created when a project is configured and analyzed using either the graphical interface of *Understand for FORTRAN* or the command line tool "undftn".

The "undftn" command line takes this form:

```
undftn -db database [-create] [files] [Options]
```

where *database* is the project database file and is a required argument.

Create a new database using the -create_77 or -create_90 or -create_95 option and then load the files into the database with the -add option. To parse all files, specify -rebuild. Once the database is initially populated, subsequent parsing can be incremental, meaning that a file only needs to be re-parsed when it has changed, or something it depends on changes. For this incremental parsing, specify -refresh.

Refer to the sections that follow for details on all the available command line options and how to create and analyze a project.

Permanent Vs. One Time Options

"undftn" lets you control every aspect of the parsing in the same way the Project Configure dialog does in *Understand for FORTRAN*'s GUI. One important difference is that with "undftn" options can be permanent or just for the invocation of "undftn".

Here is the rule: Any option specified when creating a database will be permanently stored as an option to be used with future uses of "undftn" (unless overridden or reset using the *Understand for FORTRAN* GUI).

So...

```
undftn -db foo.udf -create -include /usr/myincludes
```

permanently adds "/usr/myincludes" to the include path, while

```
undftn -db foo.udf -include /usr/myincludes/test
```

uses that include path only for this invocation of "undftn".

Getting Help on Command Line Options

Since we do weekly builds of *Understand for FORTRAN*, it is quite likely that this manual may not describe all the options of the "undftn" command line. This is especially true if you have a printed manual.

Any command in the *Understand* suite lists its command line options and provides a brief explanation when given the "-h" option:

```
undftn -h
```

Command Line Options

The only required argument for all commands using "undftn" is the project database file. The following table lists the command line options for "undftn".

Option	Description
-a[dd] file [files] -a[dd] @filelist.txt	Specify one or more files to add to the database. Files can be specified in two ways: • **Individually:** Separate filenames with spaces. For example -add *.for adds all *.for files in the current directory to the project. Absolute paths are added to the project. For relative paths, use the -reladd option. • **In a text file:** Use -add @filelist.txt to add all files that are listed in that file to the project. The file must contain one file per line. Full or relative paths may be used. A # sign in the first column of a line in the listfile indicates a comment. Wildcards are expanded by the command shell. This option may be used multiple times on the same command line.
-create_77 -create_90 -create_95	Create a new project that will contain FORTRAN 77, 90, or 95 code. This cannot be changed once the database is created. Creating a new database overwrites an existing database of the same name.
-db database	Specify the name of the database to create or open. An extension of .udf is provided if no extension is given.

Option	Description
-d[elete] file [files] -d[elete] @filelist.txt	Specify one or more files to delete from the database. Wildcards are expanded by the command shell. Files can be specified in two ways: individually, separated by spaces, or in a text file, with one file per line. For example -delete *.for removes all *.for files in the current directory from the project. Or use -delete @filelist.txt to delete all files that are listed in that file to the project. The file must contain one file per line. Full or relative paths may be used. A # sign in the first column of a line in the listfile indicates a comment. This option may be used multiple times on the same command line.
-error file	Specify a file to which error messages should be logged. No reporting is the default.
-format [fixed | free]	Specify if source to be processed uses free format (statements cross lines) or fixed format (statements end at a line or column)
-h[elp]	Show the command line options.
-include includesPath or -I includesPath	An optional argument to add a path to the analysis include path. This include path will be saved in your project database file. This argument may be specified once, with multiple directories separated by spaces. If directories have spaces in their name, surround entire set of directories with " ".
-libunit_defs [before|after|both|none]	Specifies to collect comments for a given declaration from before/after, or never, the declaration of the entity.
-list [status]	Lists files in the project. Specify "status" to report the current status of each file: "Okay" if current or "Changed" if file has been modified since last parse.
-quiet	Use for quiet processing, i.e. status messages will not be written to the screen. This argument is optional. By default, some status messages are written to standard output.
-rebuild	Performs a full reparse of all project source files whether they have been modified or not.

Option	Description
-refresh	Performs an incremental reparse of the project source files. Only modified and dependent files are re-parsed. This is the default action when no other option is specified. If project parameters have changed since the last build, be sure to use -rebuild instead of -refresh.
-reladd	Add a relative source file reference to a project.
-rellist	List source files with relative references.
-truncate_column column-number	Specify column to truncate analysis of a line at. Common are 72 or 132. Used only when -fixed is chosen.
-typeobject_defs [before|after|both|none]	Specifies to collect comments for a given declaration from before, after, or never, the declaration of the typeobject entity.
-v[erbose]	Use for verbose processing, i.e. status messages will be written to the screen. This argument is optional.

The following sections show example uses of undftn.

Creating a New Project

There are several ways to create a new project file using "undftn". Of course you can use the graphical interface of *Understand for FORTRAN* to first create your project, but here we will examine a few ways to do it from the command line.

To create a new, empty, project called sample.udf:

```
undftn -create_77 -db sample.udf
```

Adding Files to a Project

If you have a small number of source files then it may be easiest to just supply their names to the analyzer using the wildcarding abilities of your operating system shell. In this example we will process all source files in the folder:

```
undf -db myproject -add *.for
```

In some cases there may be too many file locations to use the *-add* technique. A common command line limitation is 255 characters. A directory with hundreds or thousands of files may easily exceed this limit. In this case, or when you want more fine-grained/repeatable control over what files are processed, you should create a "listfile". This file must have a format of one filename per line:

```
c:\myfiles\myproject\myproject.f
c:\myfiles\myproject\support.f
c:\myfiles\myproject\io.f
h:\shared\allprojects\file2.f
h:\options\file3.f
h:\options\file4.f
h:\options\file5.f
```

is processed via the *-add* command line option:

```
undftn -db myproject -add @myfiles.lis
```

Note that there is no limit on the number of files listed in the list file.

Creating a Database and Adding Sources in One Step

You can create and add files to the database all in one command. For example:

```
undftn -db myproject.udf -create_77 -add *.f
```

Creating a List of Files

On UNIX here are a couple ways to create such a file:

- Use the 'ls' command, as in:

```
ls *.f  > my_project.lst
```

- Use the 'find' command to recurse sub-directories, as in:

```
find . -name "*.f " -print > my_project.lst
```

In a Windows command shell:

- Use the dir command with the /b option:

```
dir /b *.f  > my_project.lst
```

- Use the /s option to recurse sub-directories, as in:

```
dir /b /s *.f  > my_project.lst
```

Analyzing a Project

The "undftn" command line program allows you to analyze (or re-analyze) a previously created project database. Refer to the preceding section if you haven't yet created and configured your project.

When analyzing a project, you have two options to choose from. You may re-analyze all files with the -rebuild option, or only those files that have changed with the -refresh option.

If you are doing your first analysis after creating a new project, it doesn't matter which option you choose as it will parse all files regardless. However, if you are performing this function on a regular basis, you may prefer to do an incremental analysis where only the

modified files and any other files dependent on those files are re-analyzed.

For example, to parse all files in the project with the following command:

```
undftn-db myproject.udf -rebuild
```

Or, to perform an incremental analysis, re-parsing only those files that have changed or other dependent files, use the command:

```
undftn -db myproject.udf -refresh
```

Keeping a Database Updated

Once you have successfully parsed your code for the first time, keeping the database up to date is simple:

```
undftn -db myproject.udf -refresh
```

This command checks each file previously parsed into the repository to see if it has changed. If it has changed then it will re-parse it, and any files that depend on it (and so on) until the database is fully refreshed.

Use this feature to keep your database up to date while keeping your parsing to a minimum.

Generating Reports

Text and/or HTML reports can be generated from a previously created *Understand for FORTRAN* project and database through the command line program *"repftn"*. The database (.udf file) may be created by using either *Understand for FORTRAN* or the *"undftn"* command line program.

The *"repftn"* command line syntax for *repftn* is:

```
repftn -db udcfile [report-options] [format_options]
```

In general, the most recent setting used in the *Understand for FORTRAN* environment is the default setting used for an option on the command line. Such settings are stored in the project database file, but can be overridden on the command line.

Details for the options are provided in the following sections.

General repftn Options

The general options specify what type of reports to generate (text or HTML) and where they should be created. Details of each general option follows:

Option	Description
-db database	Specify the database file to be used to create reports. This is the .udf file created by undftn or *Understand for FORTRAN*.
-help	Provides on-line guidance for command line syntax.
-html [directory]	Use this option to generate HTML versions of the reports. If the directory specified by html_directory exists, it is used, if not it is created. By default, the directory .\html is used. The "home" file of the directory is index.html. Also specify one of the -htmlall, -htmlalpha, or -htmlsplit options to specify if and how the HTML files are to be split up. By default, reports are split alphabetically. The -html option may not be used if -text or -prefix is used.
-htmlall	Use this option to specify that each HTML report is to be kept in one HTML file. Using this option on large projects may cause html browser problems when loading large files. This argument is optional and cannot be used with -htmlalpha or -htmlsplit.
-htmlalpha	Generate multiple files for each HTML report, splitting up the files alphabetically by the first character of each entity name. The "home" file of the directory is index.html. This is the default. This argument is optional and cannot be used with -htmlall or -htmlsplit.

Option	Description
-htmlsplit n	Generate multiple files for each HTML report, splitting up the files into "n" entities per file. By default, the report is split into 500 entities per file. The "home" file of the directory is index.html. This argument is optional and cannot be used with -htmlall or -htmlalpha.
-metrics filename	Export metrics information to a comma-delimited file of name "filename".
-quiet	Run quietly (don't output message about what report is being generated and so on). This option is useful when using *repftn* in cron jobs or other batch creation situations.
-separate prefix	Used to break each text report into a separate ASCII file. The prefix is prepended to each generated file. If desired, a separate directory location may be specified as part of the prefix. See *Report File Naming Conventions* on page 6–3. This option may not be used if -html or -text options are used.
-text output_file	Specify the output filename. Use this option to create an ASCII output file containing all reports. Cannot be used in conjunction with the -separate or -html options.

Report Options

repftn offers a variety of reports. These report options are used to turn the generation of specific reports on or off. These options are not applicable when using the single -text report.

If no report options are specified, all reports are turned on and will be generated. Specifying one or more reports to be "on" turns off all reports except those explicitly specified to be on. Specifying one or more reports to be "off" turn on all reports except those explicitly turned off. Each report option is listed below:

Report Option	Description
-rep_complexity [on|off]	Turn the Program Unit Complexity report on or off.
-rep_dec_not_used [on|off]	Turn the Unused Objects, Unused Program Units, and Unused Types reports on or off.
-rep_dectree [on|off]	Turn the Declaration Trees report on or off.
-rep_dict [on|off]	Turn the Data Dictionary report on or off.
-rep_extension [on|off]	Turn the FORTRAN Extension Usage report on or off.
-rep_implicit [on|off]	Turn the Implicitly Declared Objects report on or off.
-rep_include [on|off]	Turn the Include report on or off.
-rep_invtree [on|off]	Turn the Invocation report on or off.
-rep_metrics_file [on|off]	Turn the File Metrics report on or off.

Report Option	Description
-rep_metrics_project [on|off]	Turn the Project Metrics report on or off.
-rep_metrics_pu [on|off]	Turn the Program Unit Metrics report on or off.
-rep_object [on|off]	Turn the Object Cross Reference report on or off.
-rep_program_unit [on|off]	Turn the Program Unit Cross Reference report on or off.
-rep_simple_invtree [on|off]	Turn the Simple Invocation Tree report on or off.
-rep_type [on|off]	Turn the Type Cross Reference report on or off.

Refer to *Text and HTML Reports* on page 6–1 for more information on the different reports generated.

Report Format Options

The following report formatting options are available.

Format Option	Description
-fmt_fullname [on|off]	Use full entity names in the invocation tree and metrics reports. The default is to use shortnames.
-intrinsics [on|off]	Include intrinsics in reports.
-repeat_inv_subtrees [on|off]	Repeat invocation subtrees in the Invocation Tree report. This option can also be set for graphic views. See *Duplicate Subtrees Menu* on page 3–26.

Extensions Used by the -separate Option

The *-separate* option (described above) is used to break ASCII output reports into multiple reports. On larger projects this can make for more manageable output file sizes.

For instance, if this command is used,

```
repftn -db myproject -separate test
```

a variety of files beginning with test and ending in different extensions are created. Refer to *Report File Naming Conventions* on page 6–3 for a complete list of all file extensions used.

Creating All Reports in an HTML Directory

The following command generates all reports, create a directory called *html* and fills it with the HTML versions of each report:

```
repftn -db my_project -html html
```

The resulting file ***index.html*** is the home page of the report.

Turning Off Some Reports

This command generates all reports except the Object Cross Reference and Program Unit Complexity reports:

```
repftn -db my_project -html html -rep_object off -rep_complexity off
```

This command generates only the textual Data Dictionary and Invocation Tree reports:

```
repftn -db my_project -separate rep_ -rep_dict on -rep_invtree on
```

Generating All Text Reports

Generate one text file of the specified name with the -text option.

```
repftn -db sample.udf -text textreports.txt
```

Generate multiple text files with the -separate option and specify a directory path and a common filename prefix.

```
repftn -db sample.udf -separate rep020324
```

The file extensions of each text file will denote the separate reports. Refer *Report File Naming Conventions* on page 6–3 for details on the file extensions used for each report. All available text reports are always generated when this option is specified.

Generating All HTML Reports

HTML reports may be split into multiple files alphabetically or by number of entities. Specify -htmlalpha to split each report into multiple HTML files (one file per alpha character). Use -htmlsplit and specify the number of entities desired per HTML file. Or, use -htmlall to create only one HTML file per report generated. Refer to *Report File Naming Conventions* on page 6–3 for details on the files generated. For example:

- For one HTML file per report:

  ```
  repftn -db sample.udf -html -htmlall
  ```

- For multiple HTML files per report, split alphabetically:

  ```
  repftn -db sample.udf -html -htmlalpha
  ```

- For multiple HTML files per report, up to 250 entities per file:

  ```
  repftn -db sample.udf -html -htmlsplit 250
  ```

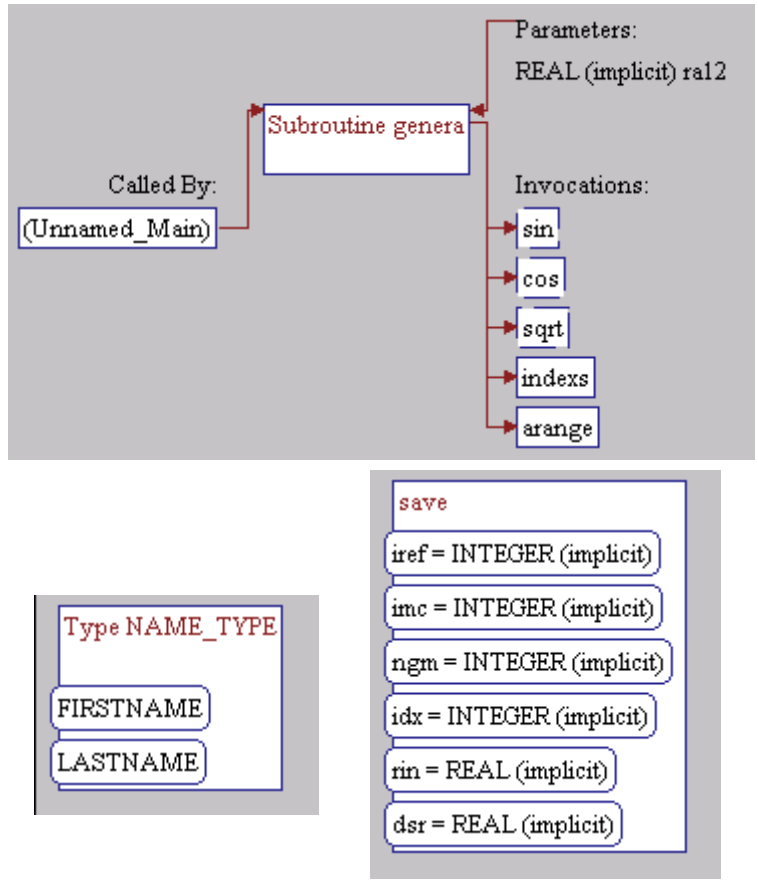
Generating Metrics Reports

Use the -metric option to create a comma-delimited project metrics file which can be used in spreadsheet programs. Refer to *Exporting Project Metrics Info* on page 6–18 for a sample file. For example:

```
repftn -db sample.udf -metrics sample-metrics.txt
```

Appendix A Graphical Notation

The following figure shows symbols used in graphic views.



The following symbols are used by *Understand for FORTRAN* to represent various language constructs.

- Parallelogram denotes an interface
- Double parallelogram denotes a module
- Rectangle denotes block data, entry, function, main program, or subroutine
- Slashed Rectangle denotes an intrinsic function or intrinsic subroutine

- Octagon denotes a common block, datapool, namelist, or pointer block
- Oval denotes a variable or dummy argument
- Double Oval denotes a file
- Hexagon denotes a derived type
- All dashed lines denote an unresolved or unknown entity.

Index

A

analyze sources
  after editing a file 2–17
  using command line tool 9–2
  using the graphical interface 2–16

B

build project
  Build Dialog annotated explanation 2–16
  like a compiler 2–2
  starting the analysis 2–16
  using command line tools 9–2

C

C API 6–3
call by view
  explained 1–15
  explanation of view 1–15
client program syntax 8–2
code file declaration view
  example picture 1–18
  explanation 1–18
color
  use in editor 4–4
command line control 8–2
complexity
  summary by program unit 6–12
configure project 2–4
cross reference
  HTML and Text reports 6–7
cross references
  visiting where an entity is used 3–14

D

database
  parsing 2–2
declaration view
  brief explanation 3–7
  code file 1–18
  examples of 3–19
  subroutine 1–17
declaration views
  examples of 3–19

E

editor
  annotated screen dump 4–2
  bracket matching 4–10
  color coding 4–4
  configuring and using as external tool 7–1
  detailed explanation 4–2
  key mappings 4–7, 4–8
  replacing text 4–5
  string searching 4–5
  using to synchronize views 8–9
entity
  browsing all 5–3
  filtering list of 5–7
  name truncation and wrapping 3–22
extensions
  usage report 6–13

F

filter
  automatic filters in locator window 5–5
  by selection in locator window 5–6
  dialog in locator window 5–7
  manual filter in locator window 5–7
  with regular expressions 5–8
filter area
  in GUI 3–4
  tabs in 3–4
Find in Files
  detailed explanation 5–10
  introduction 3–6
  overview 1–10
fortran extensions
  usage report 6–13

FORTRAN versions supported 1–2
fullnames 3–23
functions
  showing or hiding unresolved functions 3–24

G

graphical interface
  introduction 1–4

H

hierarchy views
  brief explanation 3–7
  call by 1–15
  examples of 3–18
  explained 1–15
  introduction 1–11
  short explanations 3–7
HTML
  creating HTML reports 9–10
  generating HTML reports 9–8
  level used 6–3
Hyper Grep
  now called Find in Files 3–6
  now Find in Files 1–10

I

include by view
  example picture 1–16
  explained 1–16
include files
  specifying in project configuration 2–9
include view
  example picture 1–16
  explanation 1–16
information browser
  detailed explanation 3–9
  introduction 3–4
  overview 1–12
information views
  introduction 1–11
intrinsics
  controlling if they show up on graphics 3–22
invocation view
  example picture 1–15
  explained 1–15
  explanation 1–15

K

keyboard mappings 4–7

L

layout
  crossing 3–23
  non-crossing 3–23
locator window
  annotated screen dump 5–3
  detailed usage 5–3
  filter by selection 5–5, 5–6
  purpose 3–5
  regular expressions 5–8
  sorting columns in 5–4

M

macros
  defining project specific 2–10
  setting include paths to ensure proper setting of macros 2–9
makefile
  using for project information 2–2
McCabe complexity 6–12
menus
  levels 3–24
  scale settings 3–21
  showing/hiding calledbys 3–26
  showing/hiding calls 3–26
  showing/hiding intrinsics 3–22
  showing/hiding parameters 3–25
  showing/hiding unresolveds 3–24
  text 3–22
  text menu in graphical browsers 3–22

metrics
exporting for spreadsheet 6–18
exporting to comma delimited file 6–18
file 6–18
program units 6–17
project 6–17, 6–18

N

names
controlling truncation and wrapping in graphics 3–22
short or full names in graphics 3–23

O

options
configuring and using external tools 7–1
in project configuration 2–8
overview of information provided 1–11

P

parameters
showing or hiding in drawings 3–25
PERL 6–3
posters
configuring 3–28
printing/creating 3–27
pre-processor
#define 2–10
#ifdef 2–10
printing
choosing levels in hierarchical views 3–24
choosing levels in hierarchy views 3–24
layout of hierarchical views 3–23
overview 3–27
poster printing 3–27
shrink to fit 3–27
project
adding include files 2–9
adding source files 2–4
by drag and drop 2–5
building from the command line 9–2
creating new project 2–3
needs same info as compiler 2–2
project specific macros 2–10

Q

quality
metric reports 6–12

R

read-only file access 2–5
regular expressions
in locator window 5–8
use in locator window filtering 5–8
repftn command line program 9–8
reports
cross reference 6–2
explanation 6–6
introduction 6–2
data dictionary 6–6
declaration tree 6–9
entity index 6–5
examples of how to create 9–10
exporting metrics data for spreadsheet 6–18
exporting of metrics to spreadsheet 6–18
file metrics 6–18
file naming conventions 6–3
fortran extension usage 6–13
function complexity 6–12
function metrics report 6–17
generate using command line method 9–8
generating ASCII reports 9–9
generating HTML reports 9–8
generating multi-file ASCII reports 9–9
HTML level used 6–3
implicitly declared objects 6–14
include file cross reference 6–11
introduction 1–20, 6–2
invocation tree 6–10
metrics

detailed explanation 6–12
introduction 6–2
metrics reports
details 6–16
object cross reference 6–7
output formats 6–3
program unit complexity 6–12
program unit cross reference 6–7
program units metrics 6–17
project metrics 6–17, 6–18
quality reports
details 6–12
introduction 6–2
selecting which to generate 2–20
simple invocation tree 6–11
specify format 2–18
structure 6–2, 6–9
structure reports
introduction 6–2
turning on and off 2–18
type cross reference 6–8
unused objects 6–14
unused types 6–15
reports - ASCII format 6–3
repository 2–2
introduction 2–2
right-click
available everywhere 1–6
in locator window 5–4
Sources tab 2–6
right-click menu
filter by selection 5–5, 5–6
in locator window 5–3

S

scale
menu in graphical browsers 3–21
setting 3–21
searching
for strings in editor 4–5
for strings in project 3–6
for strings in source files of project 5–10
Go To Line... in editor 4–6
server mode
client program syntax 8–2
configuring UNIX client/server port 8–8
hooking an editor up 8–9
introduction 8–2
short names 3–23
source
browsing in editor 4–2
detailed explanation 4–2
introduction to source code editor 1–14
overview of source code editor 1–14
printing 4–10
source code
introduction to browsing 1–14
quickly visiting 3–13
source files
adding to project 2–4
moved or renamed files 2–6
source views
introduction 1–11
overview 1–11
spreadsheet file of metrics data 6–18
structure reports
declaration tree 6–9
invocation tree 6–10
Invocation Tree Report 6–10
simple invocation tree 6–11
structure views
brief explanation 3–7
introduction 1–11, 1–17
output report 6–9
overview 1–11
subroutine declaration view
example picture 1–17
explained 1–17

T

text menu
in graphical browser menus 3–22
tools
configuring external tools 7–1

U

Understand
Parts of the GUI 3–3
parts of the GUI 3–3
What is it? 1–2
undftn
command line options 9–3
UNIX
printing on UNIX machines 3–28
use and configuration of sockets for server mode 8–8
unused objects
report 6–14
unused types
report 6–15

V

views
code file declaration 1–18
controlling layout 3–21
hierarchy of entity relationships 1–15
hierarchy type - explained 1–15
include 1–16
include by 1–16
source code editor 1–14
source code views - introduction 1–14
structure views 1–17
subroutine declaration 1–17

W

Windows
ASCII/Text Output format 6–3
ASCII/Text output format 6–3
use of DDE in client/server mode 8–8
write access
preventing 2–5