

*Imagix 4D*の使用方法

株式会社東陽テクニカ
ソフトウェア・ソリューション

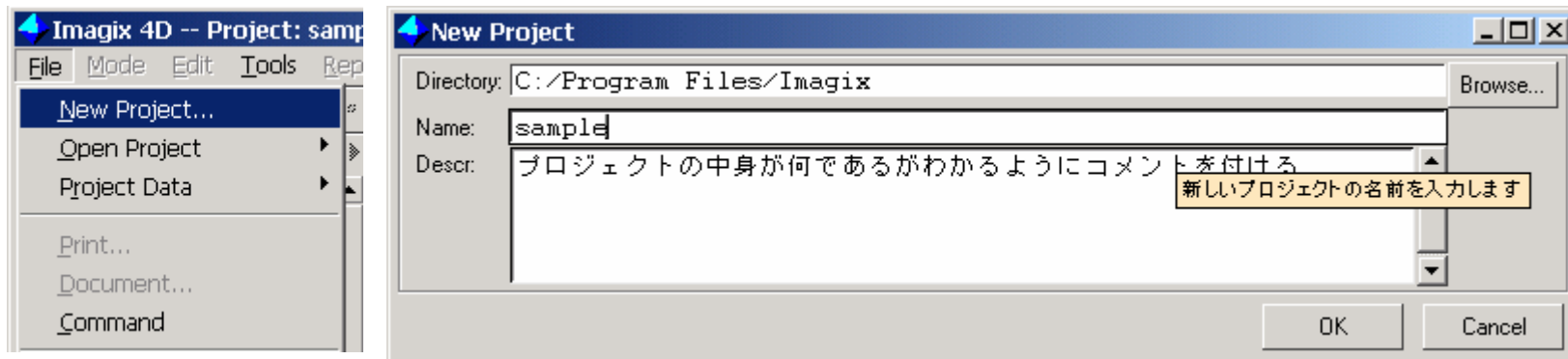
あなたのソフトウェアの構造は設計通りになっていますか？

- ソフトウェアの構造がどうなっているか知っていますか？
 - 全体像が判らず、全体の中のどの部分を担当しているかも理解できていません。
- モジュールの組み立て方を設計していますか？
 - ソースコード中心の開発で、仕様書の作成より、コーディングが先行してしまいます。
- モジュール内の構造は仕様書と一致していますか？
 - モジュール仕様書でフローチャートやPAD、HCPチャートを利用して内部の制御構造を設計していません。
- モジュールのインターフェースは始めから決まっていますか？
 - 引数や返却値の使用は少なく、グローバル変数を使った入出力が中心です。

解析からレポート作成までの手順

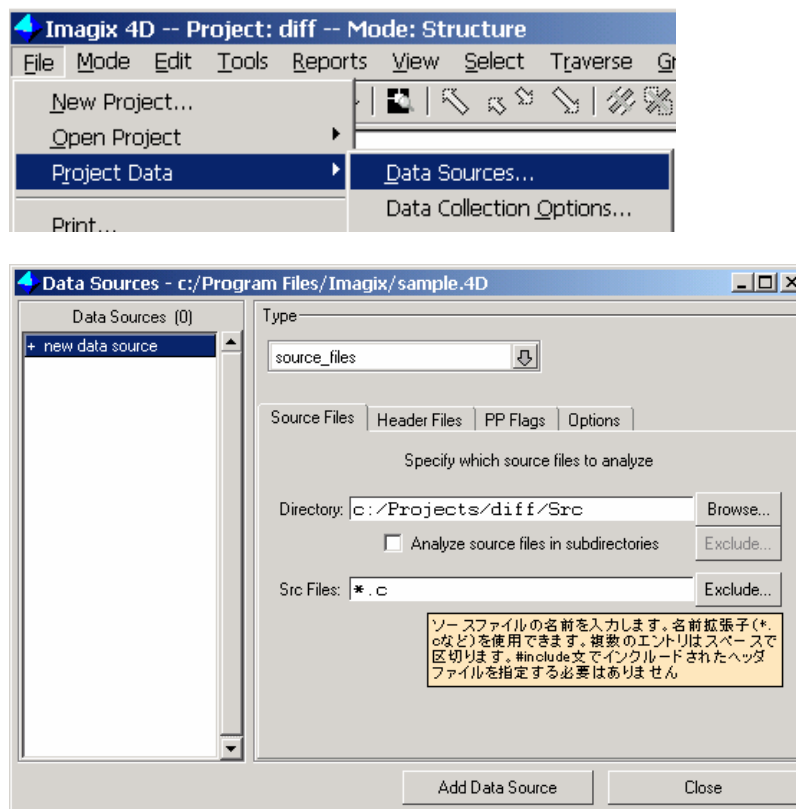
1. プロジェクトを作成します
2. データソースを作成します
3. 表示を変更します
4. モードを変更します
5. タスクを登録します
6. 割り込み関数を登録します
7. レポートを作成します
8. CSVファイルに出力します

1. プロジェクトの作成



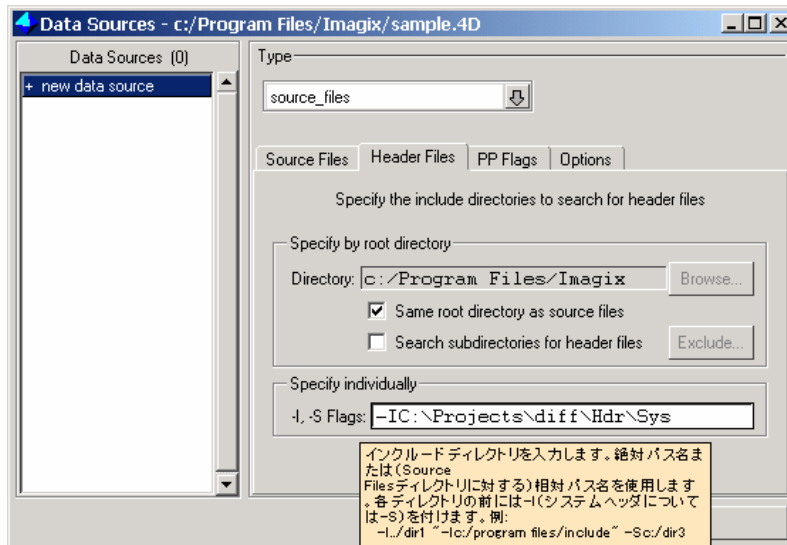
- プロジェクト・ディレクトリ
 - 解析結果が格納されるディレクトリを指定します
 - 日本語を含むパス名は使用できません
- プロジェクト名
 - プロジェクト・ディレクトリに”プロジェクト名+”4D”という名前でプロジェクトが作成されます
 - 日本語が使用できません
- コメント(任意)
 - プロジェクトに登録されているソースコードや設定、日時を記録します

2. データソースの作成 (ソースファイルの指定)



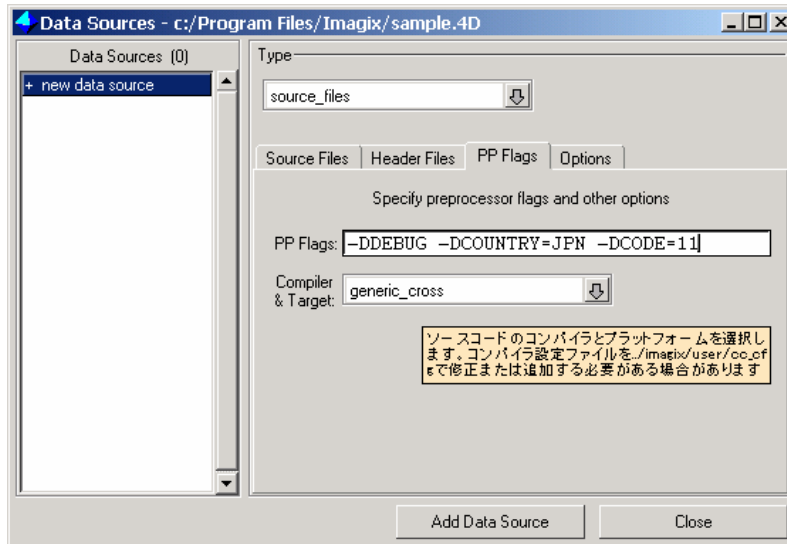
- データソースのタイプを選択します。
 - source_files
 - src_files_using_make
 - msvc_project
 - msvc_workspace
- ソースファイルのディレクトリを指定します。
 - サブディレクトリにあるソースファイルを再帰的に検索するか、を指定します。
- 検索するソースファイルの拡張子を指定します。
 - 除外したいソースファイルがあれば、指定します。(任意)

2. データソースの作成 (ヘッダファイルの指定)



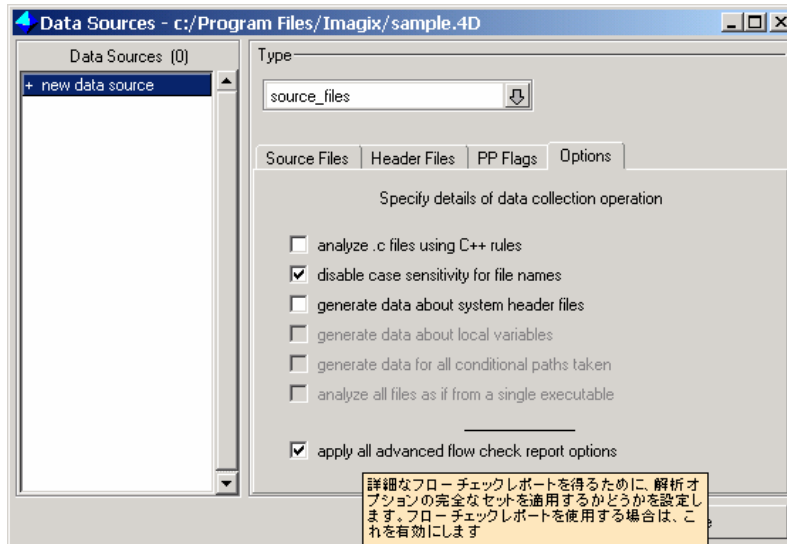
- ヘッダファイルがソースファイルと同じディレクトリにあるか、を指定します。
 - 異なるディレクトリにある場合はそのディレクトリを指定します。
 - サブディレクトリにあるヘッダファイルを再帰的に検索するか、を指定します。
- ヘッダファイルの検索パスを個別に指定します。
 - パス名にスペースが含まれる場合は、全体を二重引用符で囲みます。
 - システムヘッダには、-Sオプションを使用します。

2. データソースの作成 (プリプロセス・オプションの指定)



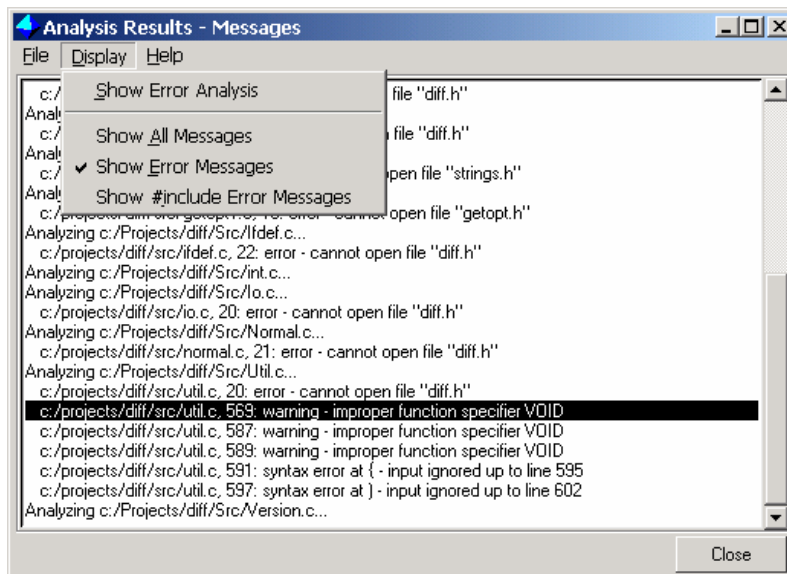
- プリプロセス・オプションを指定します。
 - コンパイル・フラグ (-D)
- コンパイラ・ターゲットを選択します。
 - コンパイラ構成ファイルは、
C:¥Program Files¥Imagix
¥user¥cc_cfg¥*.inc にコンパイラ
毎に保存されています。
 - 使用したクロスコンパイラがリスト
に無い場合はgeneric_crossを使用
してください。
 - 複数のターゲットをサポートする
コンパイラの場合、コンパイラ構
成ファイル内のマクロを調整する
必要があります。

2. データソースの作成 (解析オプションの調整)



- 構造分析レポートを使用する場合、フローチェック・レポートオプションを使用します。
 - ローカル変数を解析します。
 - 全てのファイルを一括解析します。
 - 共通ヘッダファイルの全てのプリプロセス結果を記録します。
- プロジェクトの全体の構造図をレビューする場合は、フローチェック・レポートオプションをOFFにします。

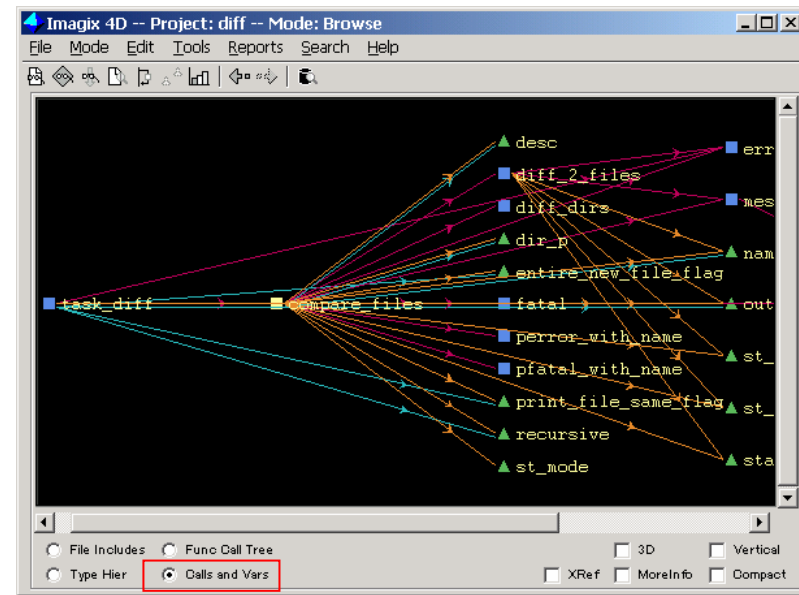
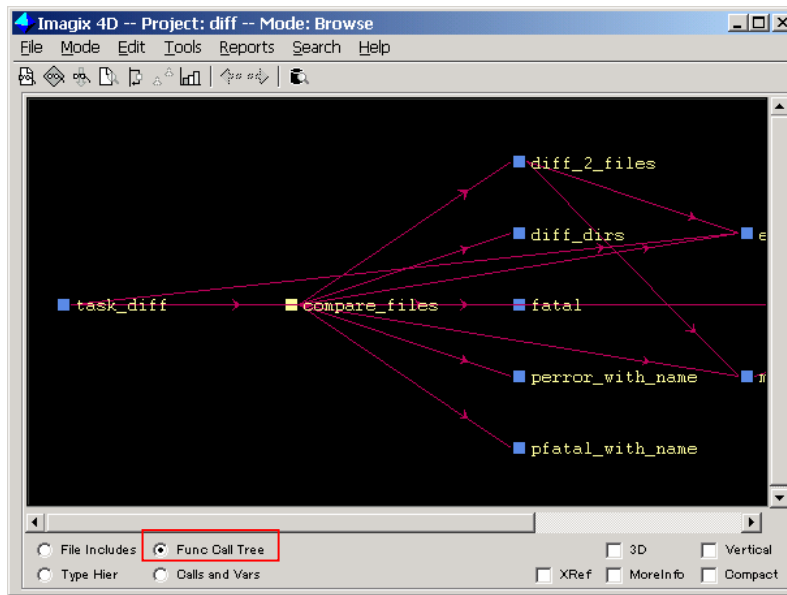
2. データソースの作成 (解析の実行)



- [Add Data Source]/[Modify Data Source]ボタンをクリックして解析を実行します。
- エラーや警告を確認して、設定の調整を行います。
 - 最初に、ヘッダファイルのインクルード・ミス进行调整します。
 - インラインアセンブラやコンパイラ固有の言語拡張进行调整します。
 - コンパイラ構成ファイルに、これらをプリプロセス時に、スキップさせる設定を追加します。

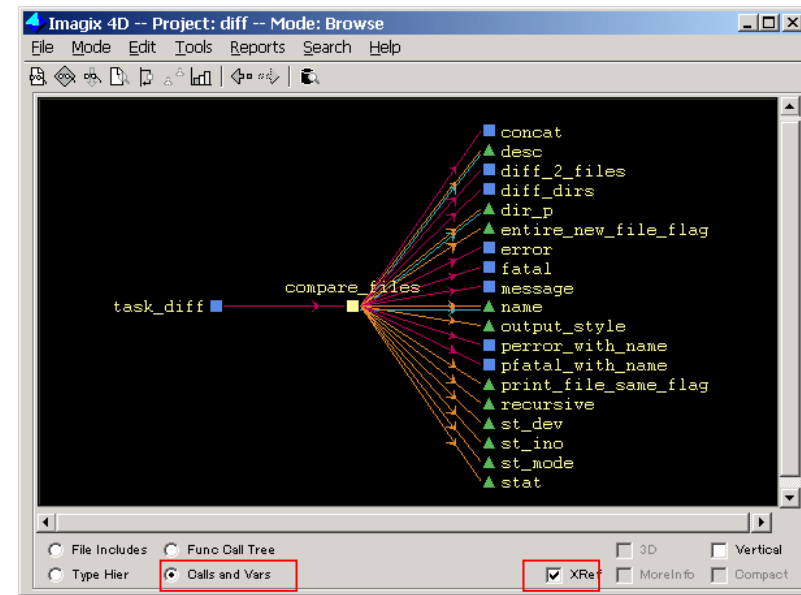
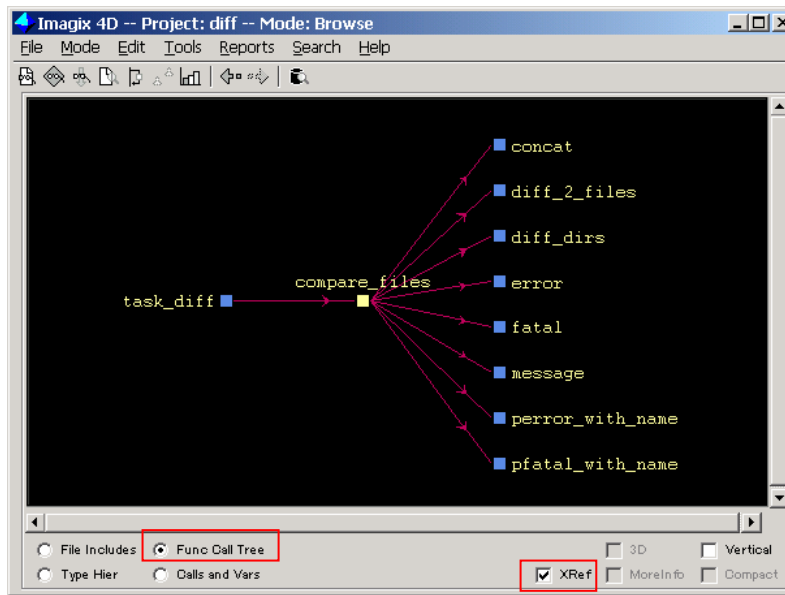
3. 表示の変更 (構造図の表示)

:関数 / :変数 / :関数呼び出し / :変数への書き込み / :変数の読み出し

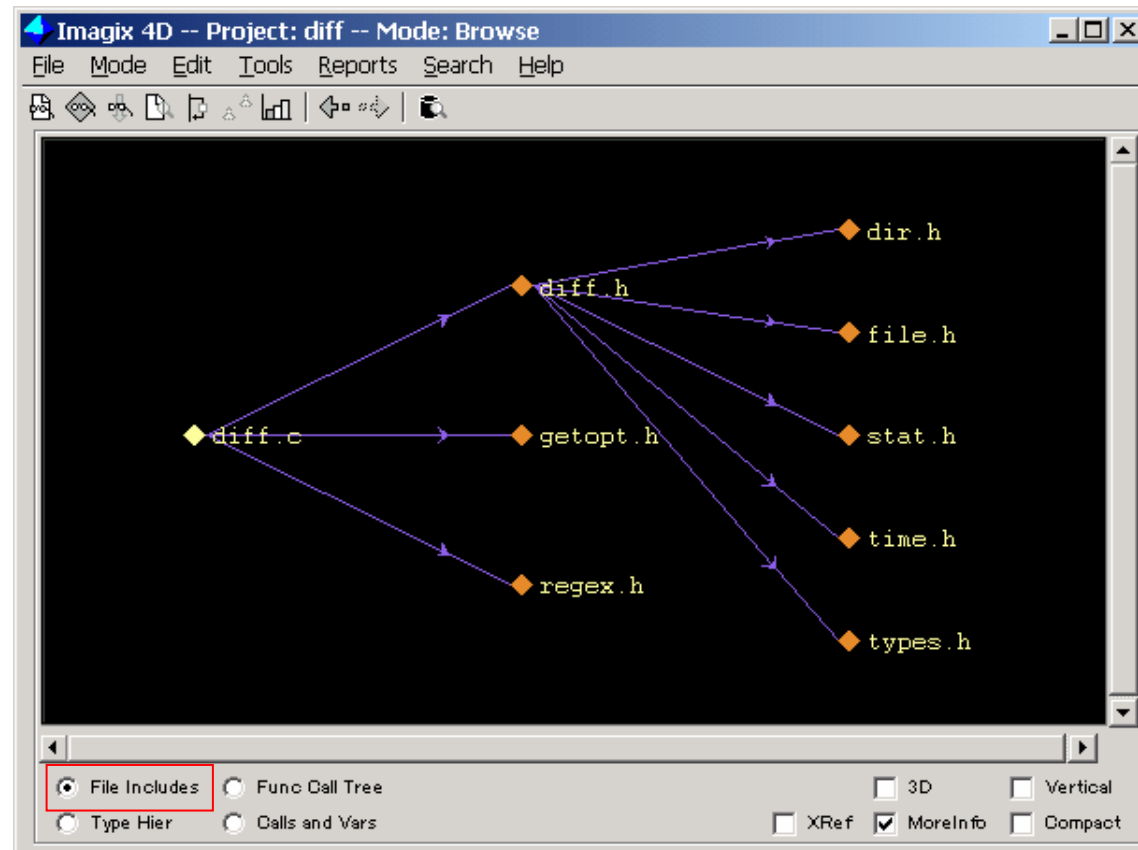


3. 表示の変更 (クロスリファレンスの表示)

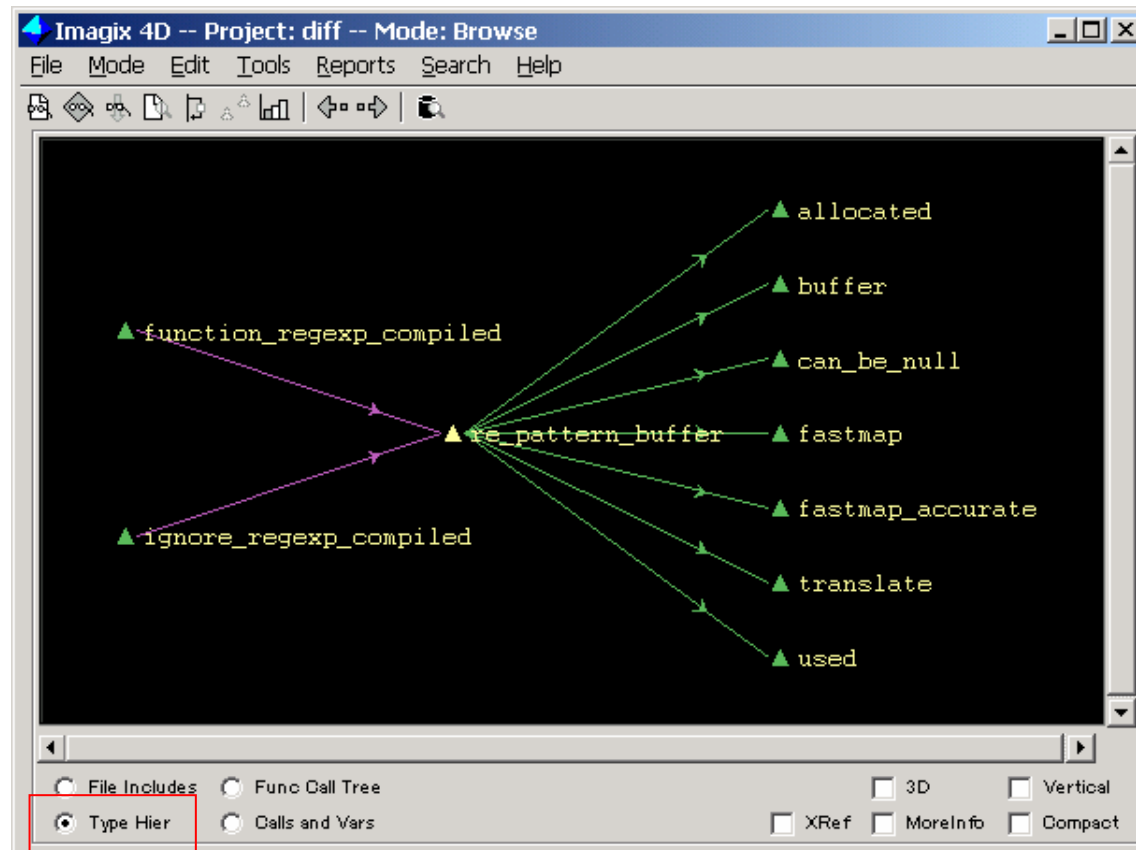
:関数 / :変数 / :関数呼び出し / :変数への書き込み / :変数の読み出し



3. 表示の変更 (ファイル・インクルードの表示)

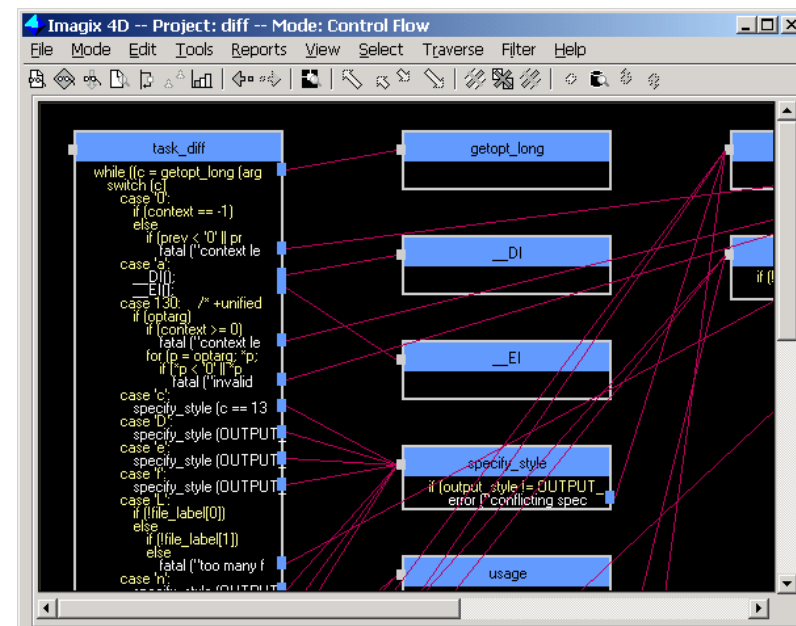
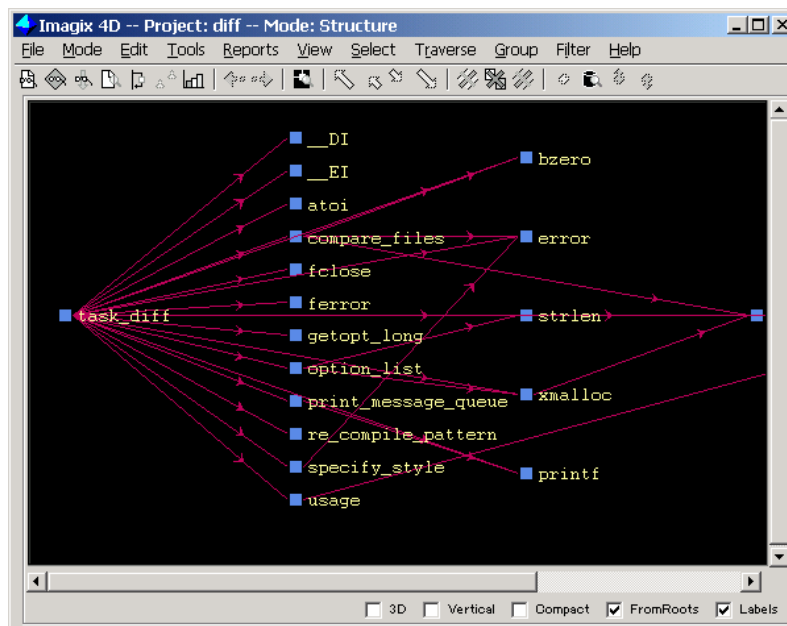


3. 表示の変更 (型情報、構造体情報の表示)

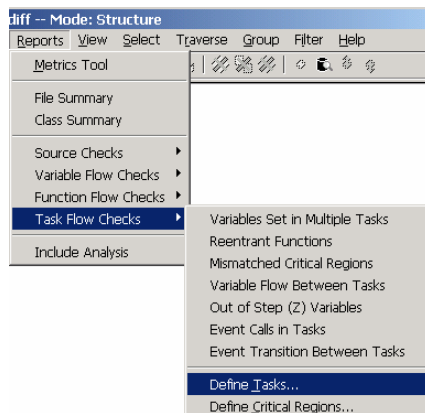
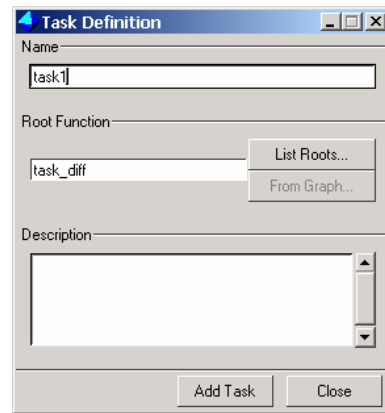


4. モードの変更 (ストラクチャー/コントロールフロー)

ModeメニューからStructureあるいはControl Flowを選択します。初期値はBrowseです。



5. タスクの登録



- 構造分析レポートの中で、Task Flow Checksに使用されます。
 - タスクの起点(先頭)の関数を登録します。
 - 起点の関数から末端の関数全てが1つのタスク(関数群)として、登録されます。
 - 未登録の場合は、親がないルート関数全てが、タスクとして登録されます。

6. 割り込み関数の登録

Critical Region Definition

Name: VxWorks

Entry / Exit Pairs:

- Interrupt Functions
- Semaphores Standard...

	Function	Param	
Entry:	semTake	1	From Graph...
Exit:	semGive	1	From Graph...

Description: Pre-defined semaphores for VxWorks operating system.

Add Region Close

Critical Region Definition

Name: INT

Entry / Exit Pairs:

- Interrupt Functions
- Semaphores Standard...

	Function	
Disable:	__DI	From Graph...
Enable:	__EI	From Graph...

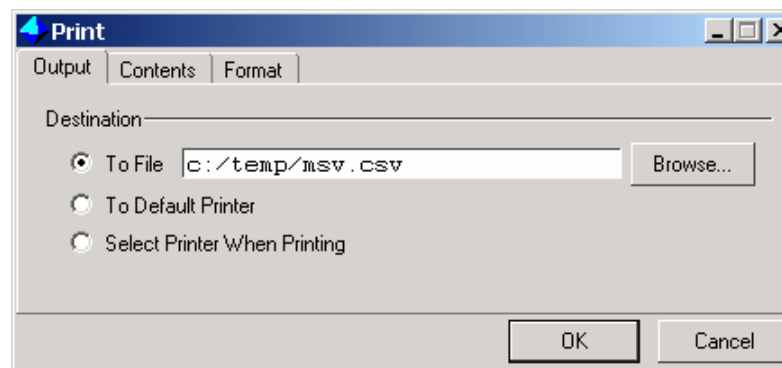
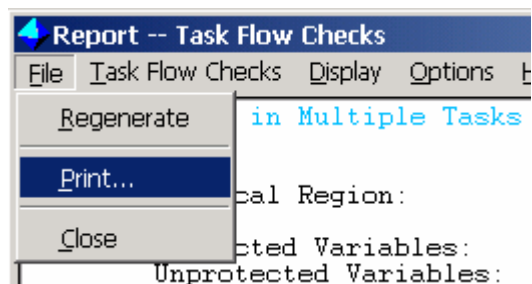
Description: 保護領域のExit関数の名前を入力します
割り込み禁止関数と割り込み許可関数を登録します。

Modify Region Close

7. レポートの作成

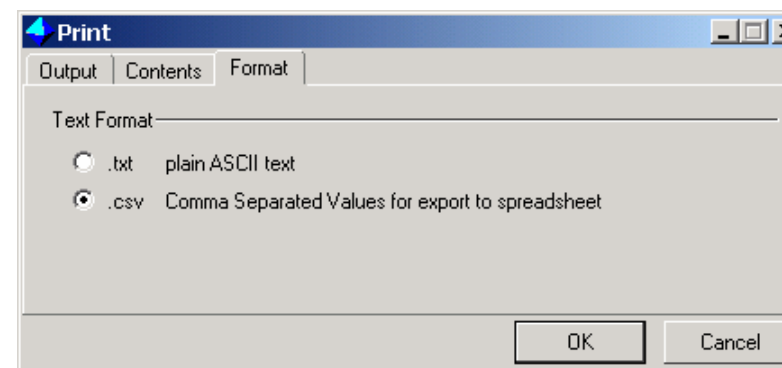
- チェックリスト
 - タスク登録が完了していますか？
 - 割り込み禁止・許可の関数やセマフォの登録が完了していますか？
 - フローチェック・レポートオプションを有効にして解析が実行されていますか？

8 .CSVファイルの出力



Microsoft Excel - msv.csv

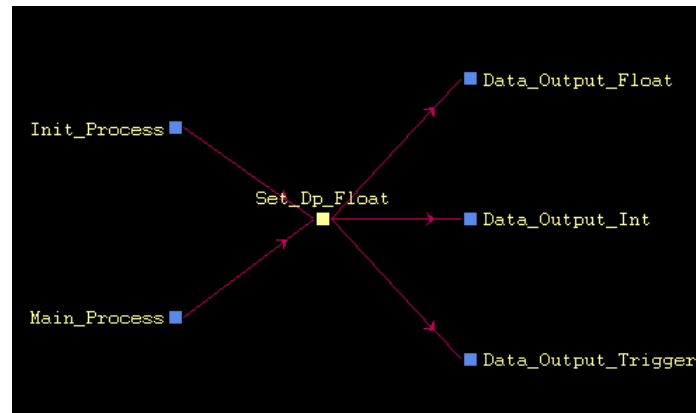
Name	Members	Root	task_diff	Usage
task1	104	Root	task_diff	
task2	56		task_diff3	
Variable	File (Line)			
Task		Line Number of Usage		Usage
				User of Variable
current	ioc (30)			
task1				
		587	U	read_files ioc (580)
		590	U	read_files ioc (580)
		599	U	read_files ioc (580)
		620	U	read_files ioc (580)
task2				
		523	U	make_3way_diff3.c (457)
		582	U	make_3way_diff3.c (457)



一般的な使い方

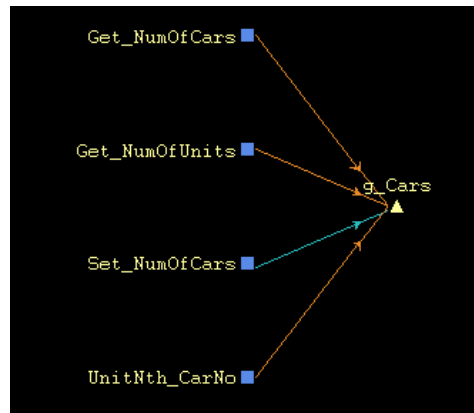
- 関数の影響範囲を確認する
- 変数の影響範囲を確認する
- ソフトウェアの全体構造を見易くする
- ソフトウェアの全体構造を比較する
- 変数のアクセス順を追跡する
- 未使用関数を確認する
- 複雑な関数内の制御構造を確認する
- 類似関数を確認する

関数の影響範囲を確認する (クロスリファレンス)



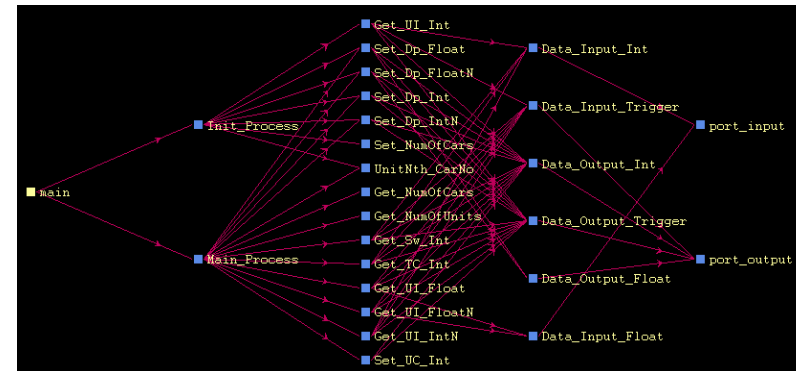
- 親関数を確認する(対象関数の左側)
 - 青い四角 (関数、選択時は黄色)、赤い矢印 (関数呼び出し)
 - 対象関数を修正した時、影響が波及する
 - 親関数の数はFan INメトリックスとしてカウントされる
 - 間接の親関数を含む親関数の総数はTransitive Fan IN メトリックスとしてカウントされる

変数の影響範囲を確認する (変数と関数の関係図)



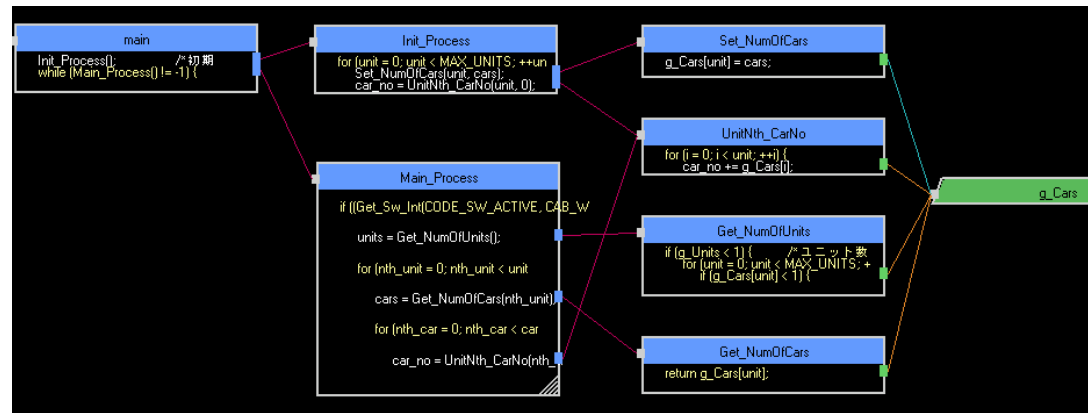
- 変数にアクセスする関数と線の色を確認する
 - 緑の三角 (変数、選択時は黄色)
 - ブルーの矢印 (代入アクセス)、オレンジの矢印 (参照アクセス)
 - 書き込んでいる関数の数はFunctions Setting メトリックスとしてカウントされる
 - 読み込んでいる関数の数はFunctions Reading メトリックスとしてカウントされる

ソフトウェアの全体構造を比較する (構造図)



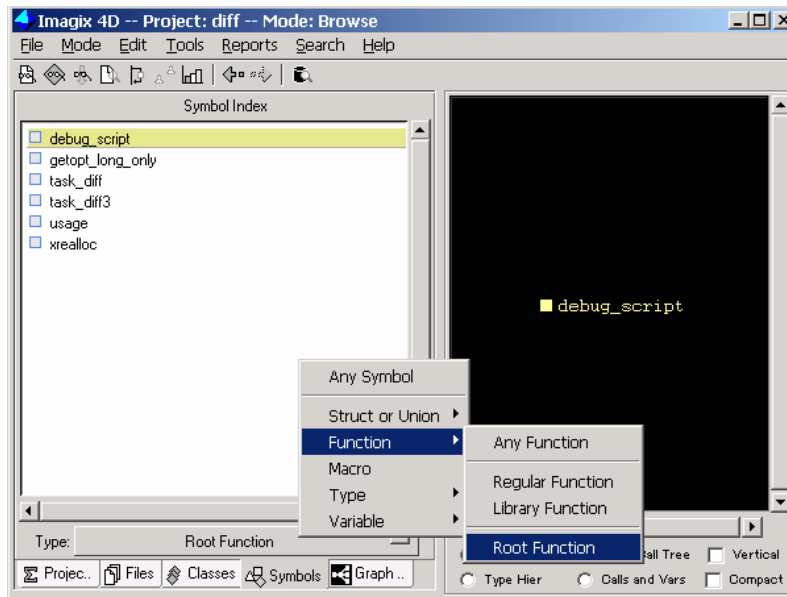
- リファクタリング前(左)、リファクタリング後(右)
 - 階層を跨ぐ関数呼び出しの有無、量を確認する
 - 関数ポインタの使用の有無を確認する
 - 階層ごとのモジュール数の変化を確認する

変数のアクセス順を追跡する (コントロールフロー図)



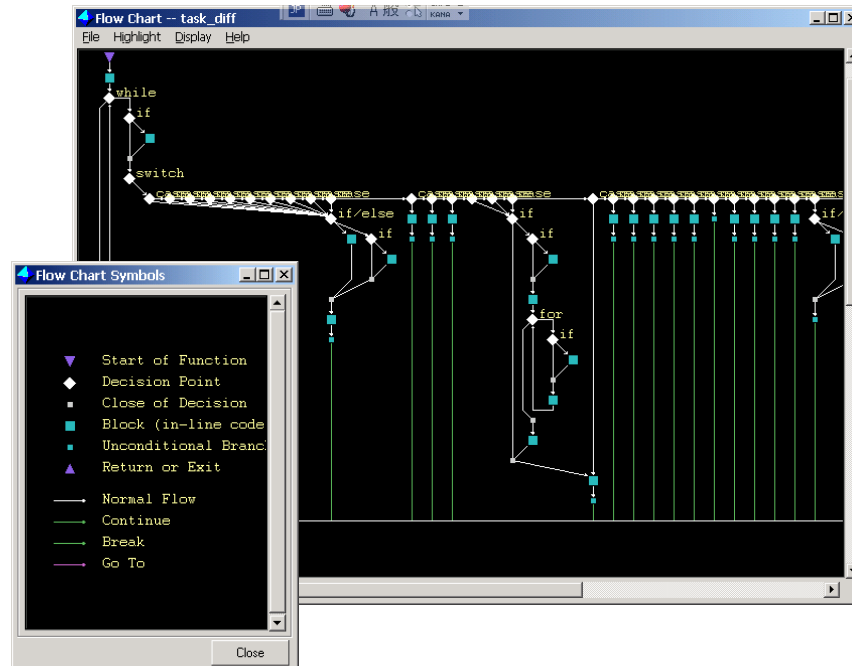
- 関数の呼び出し順を考慮して変数へのアクセスを確認する
 - 関数呼び出し、変数アクセスを行っているソース行が表示される(白色文字)
 - 該当行に到達するまでの条件式の行が表示され、インデントされる。(黄色文字)
 - 関数呼び出しがソースコードの順番に表示される。(赤い矢印)
 - 変数への書き込み(ブルーの矢印)、読み出し(オレンジの矢印)

未使用関数を確認する (シンボルリスト・ルート関数)



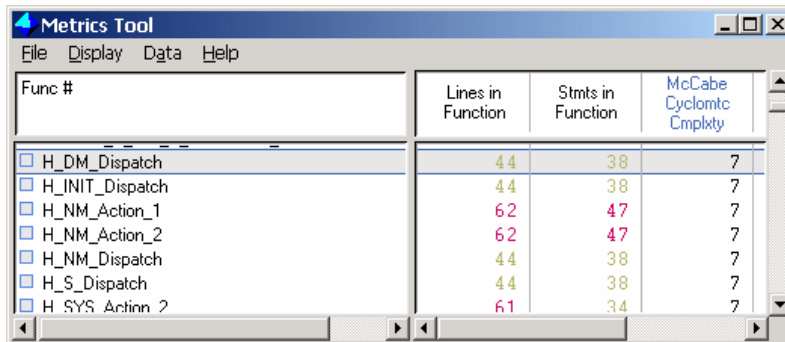
- シンボルウィンドウのタイプセレクタから[Root Functions] を選択します。
 - main関数、割り込み関数、タスクの先頭の関数以外がリストアップされた場合、未使用(誰からも呼び出されていない)の可能性があります。
 - コンパイルには使用されていないソースファイルを誤って一緒に解析していないか、を確認してください。

複雑な関数内の制御構造を確認する

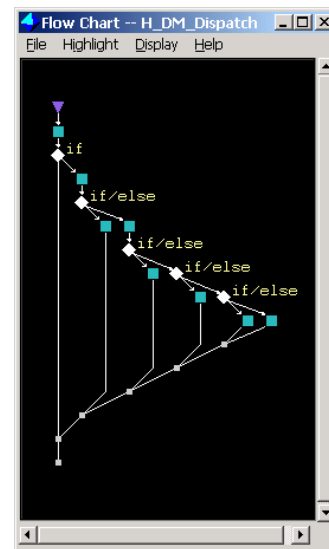
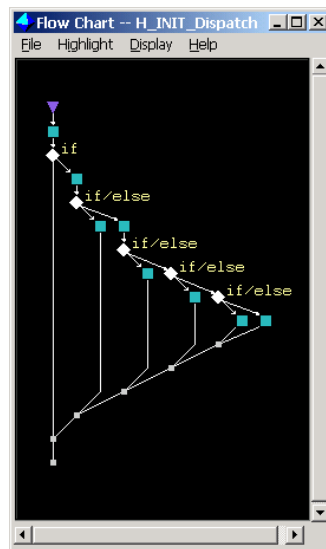


- 理解し難い構造を確認します。
 - 数多くの出口が存在していないかどうか？
 - goto文が使用されていないか？
 - switch/case以外でbreak文が使用されていないか？
 - switch/caseでbreak文を書き忘れていないか？
- 複雑な関数の検出には McCabe Cyclomatic Complexity メトリックスを使用します。

類似関数を確認する

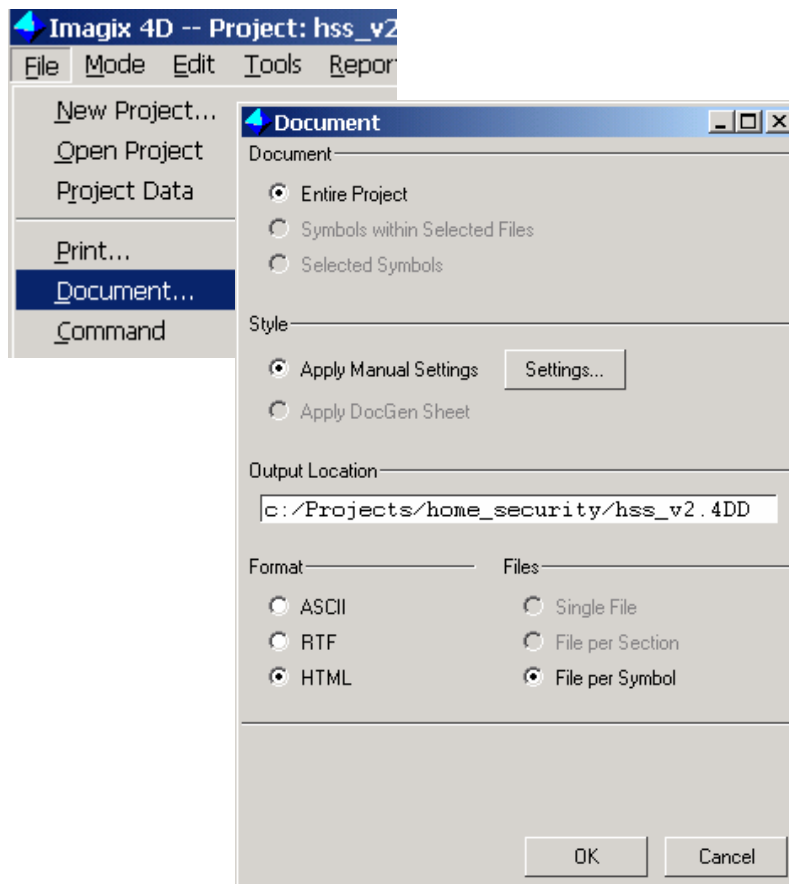


Func #	Lines in Function	Stmts in Function	McCabe Cyclomatic Cmplxy
<input type="checkbox"/> H_DM_Dispatch	44	38	7
<input type="checkbox"/> H_INIT_Dispatch	44	38	7
<input type="checkbox"/> H_NM_Action_1	62	47	7
<input type="checkbox"/> H_NM_Action_2	62	47	7
<input type="checkbox"/> H_NM_Dispatch	44	38	7
<input type="checkbox"/> H_S_Dispatch	44	38	7
<input type="checkbox"/> H_SYS_Actinn ?	61	34	7



- メトリックツールから数値の類似が見られる関数をピックアップします。特に、以下のメトリクスに着目します。
 - ソース行数
 - McCabe Cyclomatic Complexity
- 候補の関数のフローチャート同士を比較します。
 - ほぼ、同じ構造の場合は、コピー・ペーストによる類似関数の可能性が高いと考えられます。

コードレビュー用のドキュメントを作成する



- HTML形式のドキュメントを作成します。
 - プロジェクト全体を指定します。
 - 出力先のディレクトリを指定します。
 - フォーマットでHTMLを指定します。

コードレビュー用のドキュメントを作成する (サンプル画面・クロスリファレンス)

The screenshot shows a Microsoft Internet Explorer window titled "diff Project Documentation - Microsoft Internet Explorer". The address bar displays "C:\Projects\diff\diff.4DD\index.htm". The browser's menu bar includes "ファイル(E)", "編集(E)", "表示(V)", "お気に入り(A)", "ツール(T)", and "ヘルプ(H)". The navigation bar contains "戻る", "検索", "お気に入り", and "移動 リンク".

The main content area is titled "diff Project Documentation" and features a navigation menu with the following items: Project, Files, Functions, LibFuncs, Macros, Variables, Structs, Types, and PreTypes. The "Functions" section is active, displaying a list of functions: add_change, all loca, and analyze_hunk. A grid of letters (A-Z, @, ?) is also visible.

The main content area displays the function "make_3way_diff" with tabs for "Info", "Cmnt", "Use", "XRef", "Deps", "Flow", "Src", and "File". Below the tabs, the "Cross Reference [legend]" section shows a diagram illustrating the relationships between functions. The diagram features a central node "make_3way_diff" (represented by a yellow square) and several other nodes: "task_diff3" (blue square), "fatal" (blue square), "next" (green triangle), "ranges" (green triangle), and "using_to_d" (blue square). Arrows indicate dependencies or relationships between these functions.