

Imagix 4D のご紹介
- C/C++ソースコード構造解析ツール -

株式会社東陽テクニカ
ソフトウェア・ソリューション

Imagix 4Dとは

- Imagix Corporation
 - 米国カルフォルニア州
- リバースエンジニアリング・ツール
 - ソフトウェアの構造情報をソースコードから逆生成し、グラフィカルに表示
 - フローチャート (if/switch/for/while/goto/break/continue,...)
 - クロスリファレンス (関数、関数と変数)
 - 構造図 (コールツリー)
 - UMLクラス図 (C++のみ)
 - ソフトウェア・コンポーネント情報の収集し、データベース化を行う
 - 変数、関数、マクロ、構造体、データ型、ファイル

*Imagix 4D*とは(続き)

- ソフトウェアの構造上の問題をレポート
 - 構造レビュー・レポート
 - 組み込みC言語用
 - メトリックス(ファイル、関数、クラス、変数)
- ソフトウェアの文書化を支援
 - HTML、RTF、テキスト
- UNIX、Linuxをサポート
 - Windows, Linux, Solaris, HP-UX

豊富なサポート・コンパイラ

- ARM Thumb C / C++
- Borland C++ Builder
- Borland C++
- Cosmic Software C Compiler
- IBM CSet++
- Diab Data C, C++
- Fujitsu C
- GNU gcc, g++
- Green Hills Software C / C++
- Hi-Tech C Compiler Family
- HP aC++
- HP cc, c89, CC
- IAR Systems C Compiler
- Keil C51/C166/C251 C Compiler
- MetaWare High C/C++
- Metrowerks CodeWarrior C, C++
- Metrowerks HIWARE C, C++
- Microchip MPLAB C
- Microtec C, C++
- MS Visual C/C++
- SCO Unix for C
- SGI MIPSpro C/C++
- Sun SPARCCompiler for C/C++
- Tasking C
- Texas Instruments C / C++
- GNU-based VX Works
- Watcom C/C++
- Z-World Dynamic C

プロセッサ固有の暗黙のマクロ定義をサポート(コンパイラ構成ファイル)

- ARM Processors
- MC68XX and ST7 Processors
- Fujitsu Processors
- Atmel AVR Microcontroller
- National Semiconductor CompactRISC CR16C
- Hitachi H8
- Motorola M68HC12, HCS12
- Texas Instruments MSP430
- 8051/80C166/MSC251 Processors
- HC08, HC12 Processors
- PIC Microcontrollers and DSPs
- Motorola 68000 Family - C166/ST10 Processors
- Texas Instruments TMS470 Microcontrollers
- Texas Instruments TMS Family of DSPs
- Zilog Z180
- 一般的な組み込みプロセッサ

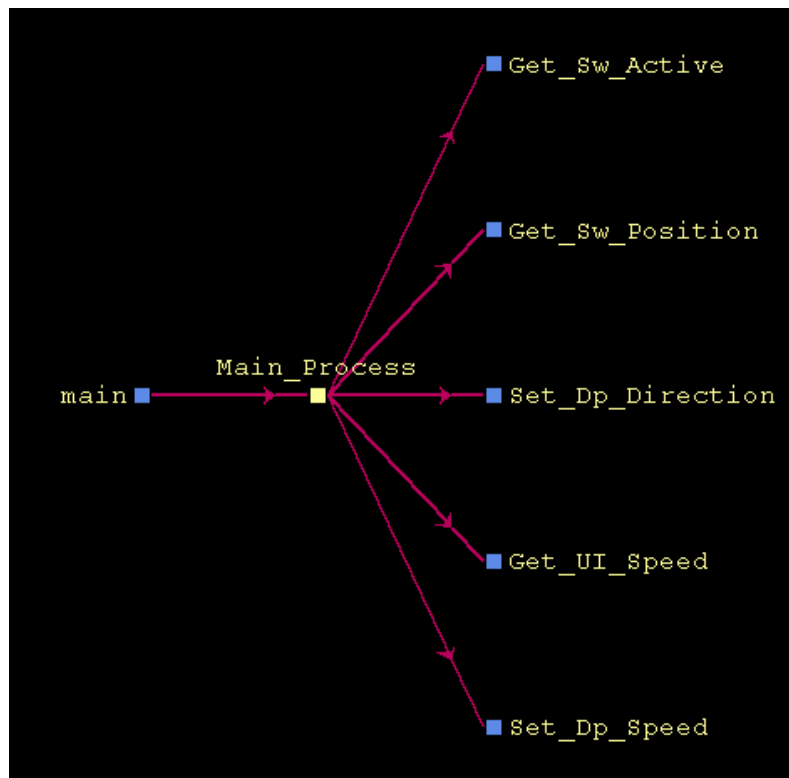
(例)

```
/* specific ARM9 family processors */
/* #define __TARGET_CPU_ARM920T */
/* #define __TARGET_CPU_ARM922T */
/* #define __TARGET_CPU_ARM940T */
/* specific ARM9E family processors */
/* #define __TARGET_CPU_ARM926EJ_S */
/* #define __TARGET_CPU_ARM946E_S */
/* #define __TARGET_CPU_ARM966E_S */
/* specific ARM10 family processors */
/* #define __TARGET_CPU_ARM1020E */
/* #define __TARGET_CPU_ARM1022E */
/* #define __TARGET_CPU_ARM1026EJ_S */
/* specific ARM11 family processors */
/* #define __TARGET_CPU_ARM1136J_S */
/* #define __TARGET_CPU_ARM1136JF_S */
```

設計図をレビューをしましょう

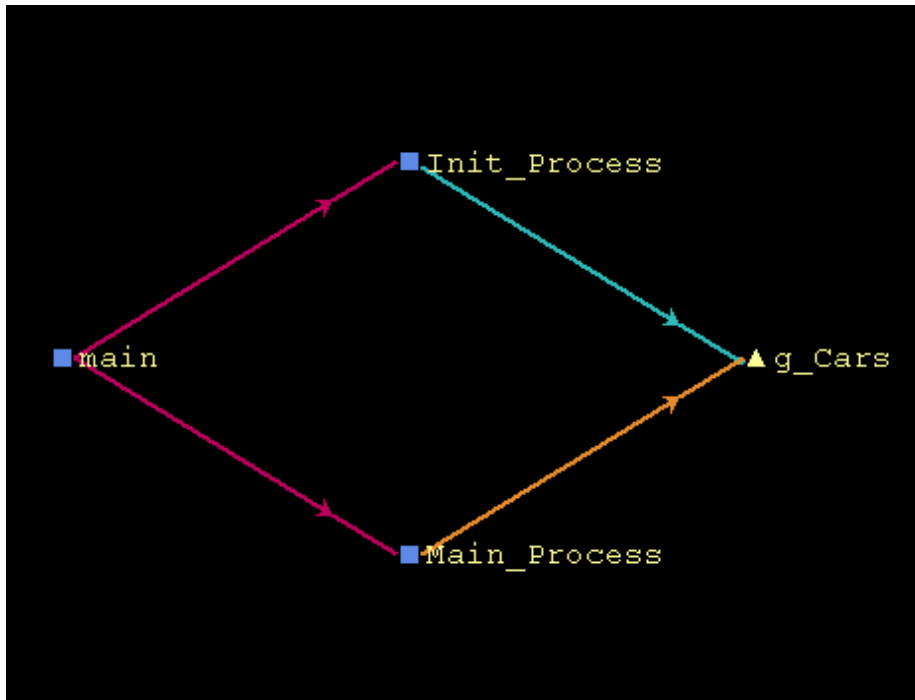
- ソースコード・レビューは意外と難しい
 - 膨大なソースコードを短時間にレビューするのは不可能
 - 他人の書いたコードを理解するには時間がかかる
- 抽象度の高い設計図でレビューをしよう
 - モジュール仕様書はあるが、構造図がない
 - ソースコードとドキュメントが乖離している
 - 規模が大きすぎる
- 最新のソースコードから作成したクロスリファレンスとフローチャートから始めましょう！

関数のコールツリー図



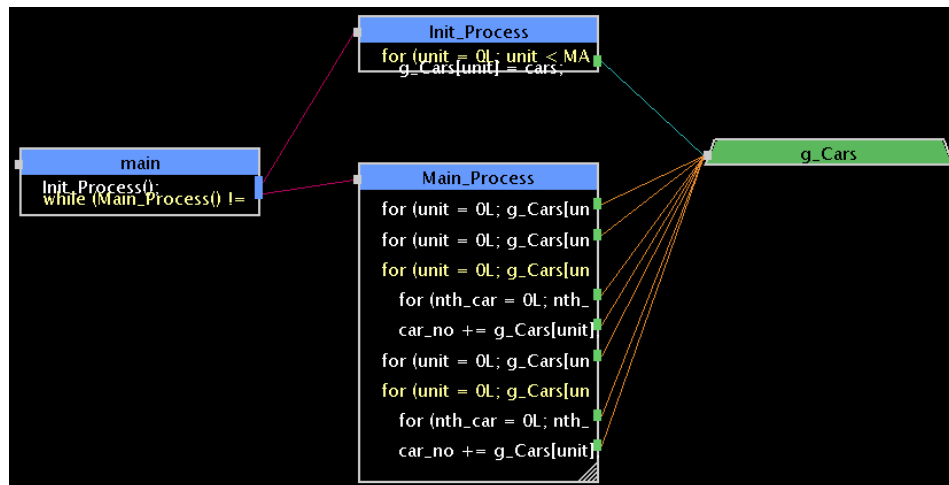
- 関数の呼び出し関係を表示
 - 関数ポインタによる呼び出しも表示可能
 - コール先の関数名が静的に記述されている場合のみ
- クロスリファレンスの表示
 - 親関数 (左側)
 - 対象が変更された場合の影響範囲を特定
 - サブ関数 (右側)
 - 単体テスト時の依存度を特定

関数と変数の関係図



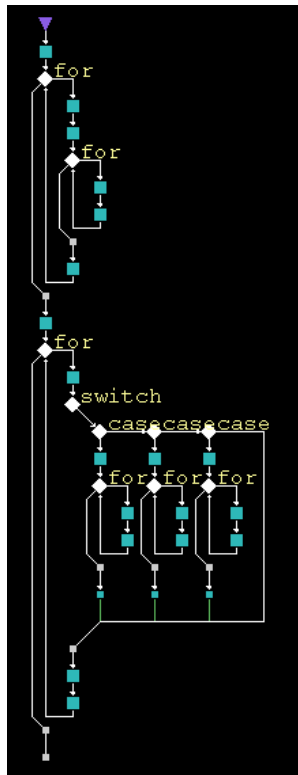
- 関数と変数の関係を表示
 - ブルーの線: セット(代入)
 - オレンジの線: リード(参照)
 - リード・セットの場合は両方
 - 変数の使用状況を直感的に把握
 - 対象の変数に関連する全ての関数を表示
 - ローカル変数も表示可能
 - 解析時のオプションにより関数のパラメータなどのローカル変数も表示可能

コントロールフロー図



- 関数コール、変数アクセスの状態をより詳細にグラフ表示
 - 回数、条件、順序の状態を把握
 - 関係線が回数分表示
 - 関連する条件式は黄色で表示
 - 関連するコード行のみを関数ボックス内に行番号順に表示
 - 未初期化・未設定の可能性のある変数をレビュー
 - ルート関数からの関数の実行順序を考慮し、変数への静的なアクセス順序を確認

関数のフローチャート



```
{
long cab, car_no, unit, cars, car_no;
/* 画面 初期化 */
for (cab = CAB_WEST; cab <= CAB_EAST; ++cab) {
  for (cab = CAB_WEST; cab <= CAB_EAST; ++cab) {
    Set_Dp_Direction(cab, DIR_OFF);
    Set_Dp_Speed(cab, FLOAT_OFF);
  }
  for (car_no = 0L; car_no < MAX_CARNO; ++car_no)
    for (car_no = 0L; car_no < MAX_CARNO; ++car_no)
      Set_Dp_Car(cab, car_no, DIR_OFF);
      Set_Dp_Weight(cab, car_no, FLOAT_OFF);
      Set_Dp_Rate(cab, car_no, FLOAT_OFF);
      Set_Dp_Cooler(cab, car_no, CTL_OFF);
      Set_Dp_Heater(cab, car_no, CTL_OFF);
      Set_Dp_Temperature(cab, car_no, FLOAT_OFF);
    }
  for (car_no = 0L; car_no < MAX_CARNO; ++car_no)
    for (cab = CAB_WEST; cab <= CAB_EAST; ++cab) {
      car_no = 0L;
      for (unit = 0L; unit < MAX_UNITS; ++unit) {
```

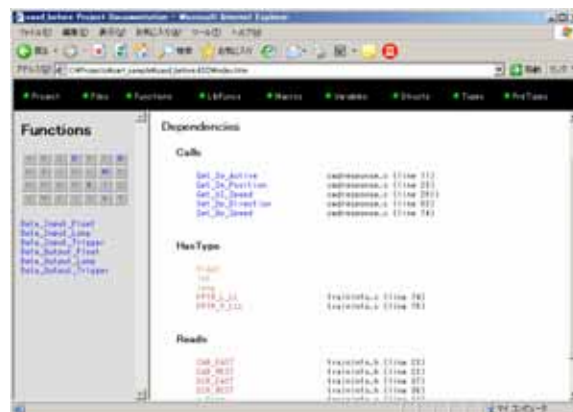
● 制御構造の様子を的確に把握

- ノーマルモード
 - 実際のソースコードの同時表示が可能
- コンパクトモード
 - switchブロックのbreak抜けのチェック
 - gotoの使用のチェック
 - ifブロックでのcontinueとbreakの使用のチェック
- ハイライト機能
 - 関数コール箇所
 - 変数アクセス箇所
 - 検索ヒット箇所

ソフトウェアの構造上の問題をチェック

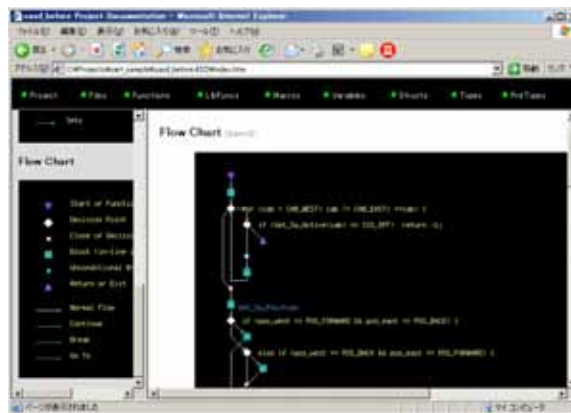
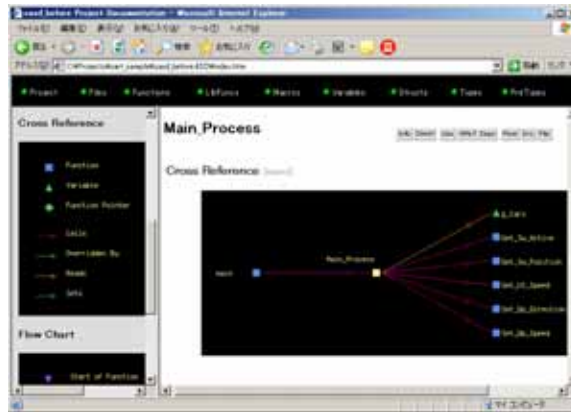
- 9種類の構造レビュー・レポートを作成
 - 読み込まれるだけの変数
 - 書き込まれるだけの変数
 - 一度も使用されることのない変数
 - 意味のない代入をしている変数
 - 再帰関数
 - 多重書き込み変数
 - リエントラント関数
 - 割り込みの禁止・許可の対応
 - 複数タスクから共有されている変数
 - 1周期前の値を使用している可能性のある変数(Z変数)

HTMLドキュメント



- **定義情報**
 - ファイル名、行番号、種別
 - スコープ
- **コメント**
 - 定義の前のコメント
- **使用状況**
 - 参照・代入の状況
 - 関数呼び出しの状況
- **依存状況**
 - サブ関数、参照変数、代入変数
- **ソースファイル**
 - オブジェクト
 - ソースファイル全体

HTMLドキュメント(続き)



- クロスリファレンス
 - 対象関数の親関数とサブ関数を表示
 - 関数ポインタも同時表示
 - 代入・参照している変数を関係線の色で表示
- フローチャート
 - 分岐の種別とそのソースコード表示
 - 表示する情報をカスタマイズ可能

ライセンス形態と価格

- ネットワーク・フローティング・ライセンス
 - ライセンス・サーバーを1台設定
 - 異機種プラットフォーム間でライセンスは共有可能
 - ライセンスの一時持ち出しが可能
- 価格
 - ss_sales@toyo.co.jpまでご連絡ください
- 評価版
 - Imagix社Webサイトからインストーラを入手
<http://www.imagix.com/trial/trial.html>
 - 評価ライセンスをss_support@toyo.co.jpまでお申し込み