

Development Assistant for C

Software Development, Quality and Documentation Tool



DAC



RistanCASE

Tools for Embedded Systems Developers

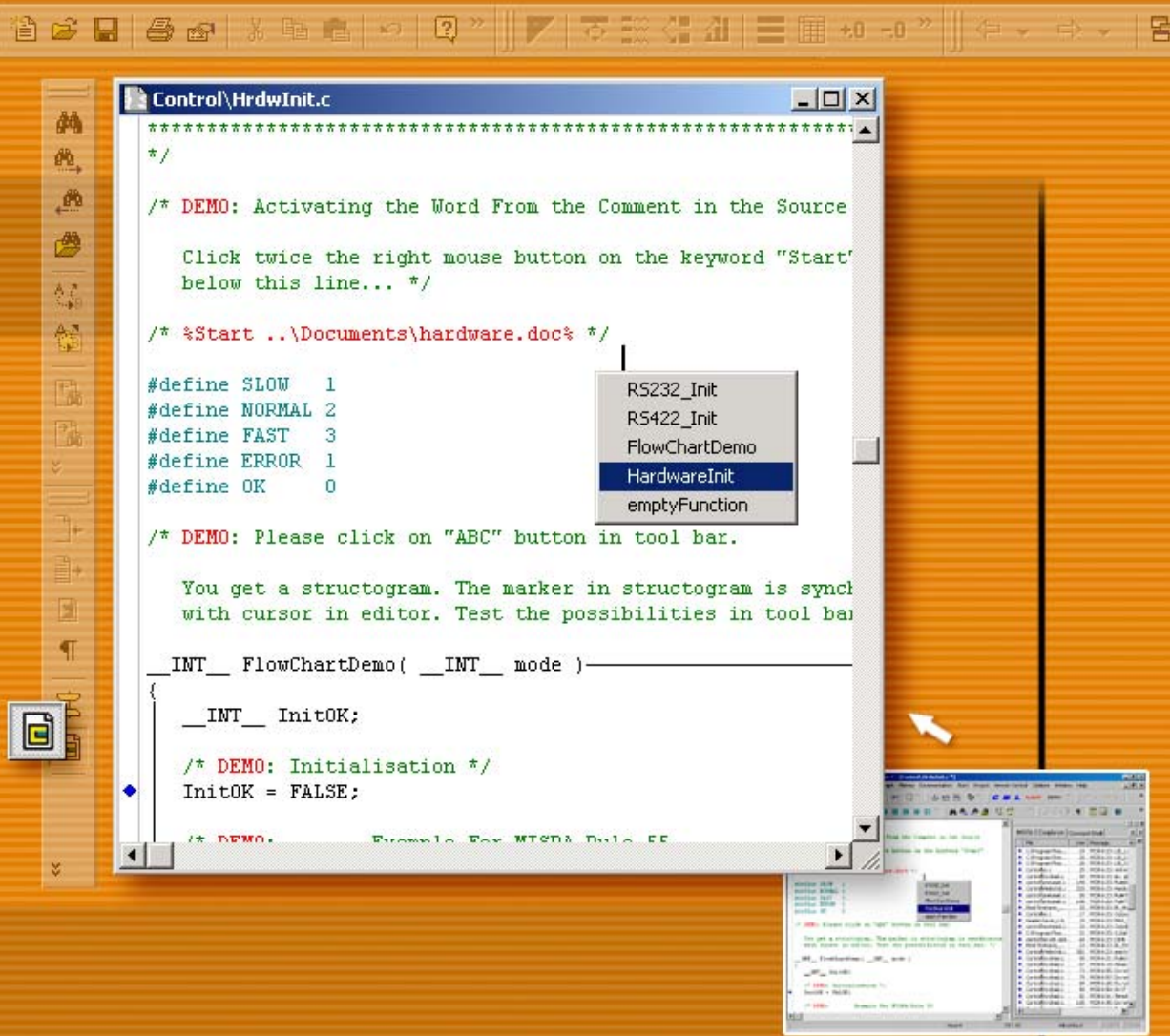
RistanCASE - Tools for Embedded System Developers



Editor

The main focus of the DAC is a fully featured programmer's editor. The DAC editor has been continually evolved to keep up with the latest trends in the world of editors and provides all the features you would expect of a modern editor, but this is just the start. The DAC contains many advanced features that allow you to produce well-documented high-quality C code.

All of these features are integrated into the editing environment and they consequently become a natural part of the development process.





DAC - Software Development, Quality and Documentation Tool

```
Control\HrdwInit.c
INT__ FlowChartDemo( __INT__ mode )
{
  __INT__ Ini
  /* DEMO: In
  InitOK = FA
  /* DEMO:
  Descript
  Action:
  Loop: <<
  /* Answers */
  switch( mode )
  {
    /* Less speed, greater safety. */
    case SLOW: <<
      ; /* Next power level on. */
      break: <<
    case NORMAL: <<
      /* Normal speed, normal safety. */
      ;
      break: <<
    case FAST: <<
  }
```

Code Structure Highlighting

At the push of a button, the code can be displayed in a graphical format similar to a Nassi Schneidermann diagram. In this view the code is displayed as a set of functional blocks that instantly reveal the structure of the program. This feature is especially useful if you are maintaining the code or working with an inherited project. The structure highlighting display is fully configurable with a choice of frame styles and colors.

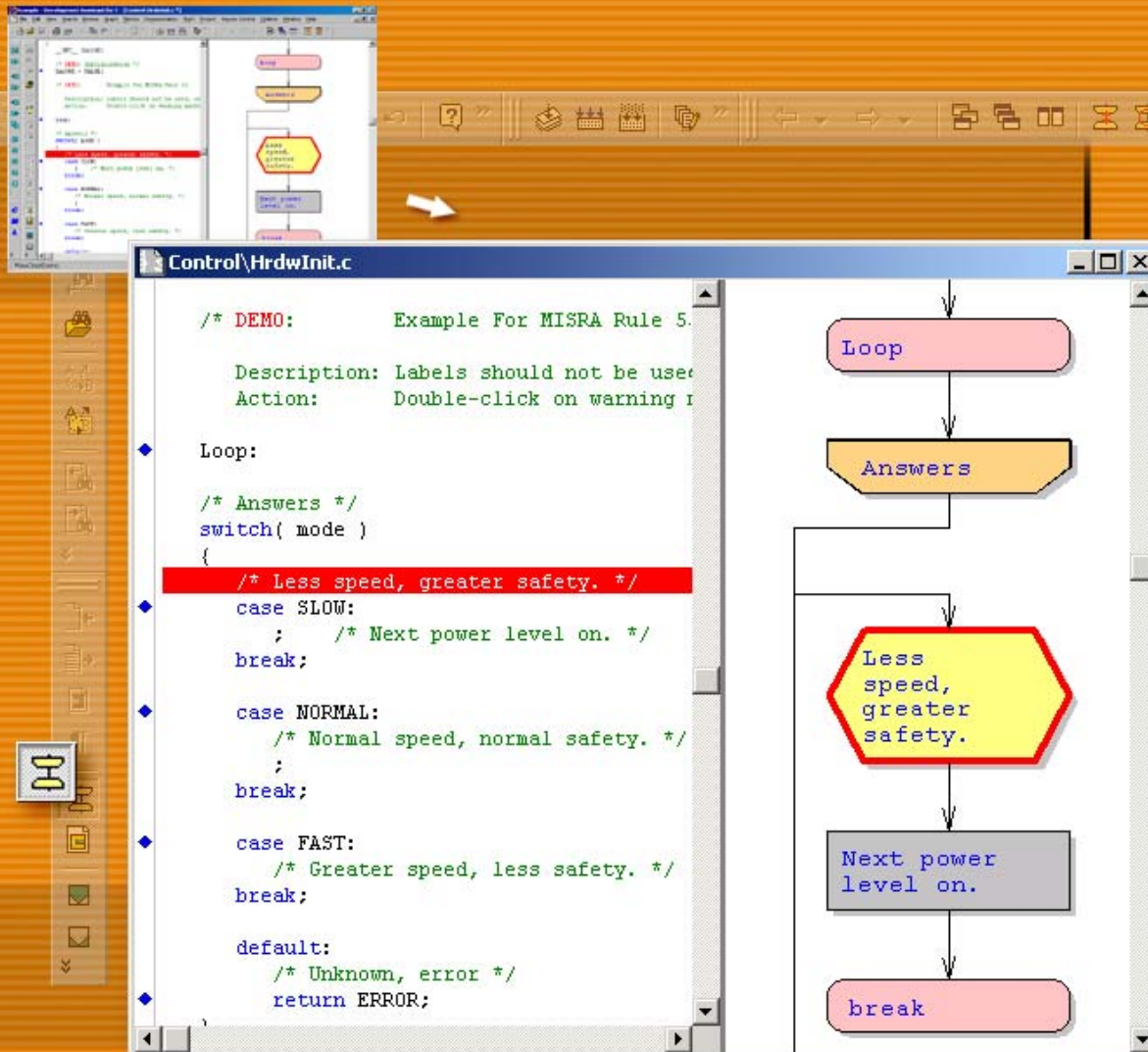


RistanCASE - Tools for Embedded System Developers



Flow Chart

DAC can also produce a flow chart of your code based on its understanding of the C language. The functional blocks of the flow chart are annotated using the existing comments in the source code. The flow chart is updated from the editor, so, any changes to the source code are immediately reflected in the flow chart. The flow chart analyzer allows for intelligent layout of the source code and comments to create a meaningful flow chart without many superfluous function blocks and meaningless statements. The flow chart display and the editor window are synchronized together so that the source code is always displayed with the relevant section of the flow chart. For large modules a navigation window is available for an overview of the code and it is also possible to collapse sections of the source code such as loops into a single block in order to generate a simplified diagram.





DAC - Software Development, Quality and Documentation Tool

Static Code Analyzer / Browser

The DAC contains a static analyzer that can be used to detect many common programming errors in a complete or a partially complete project. This allows you to remove most of your bugs early in the development cycle. The analyzer understands ANSI C and extensions to the C language used by most embedded C compilers. The DAC also contains a browse feature that allows you to navigate your code like HTML. With a couple of mouse clicks you can find the definition and uses of any function or variable, all the calls to a function and all the variables used within a function.

The screenshot displays the DAC interface with several windows and menus:

- Global functions**: A list of functions including LIB_1, LIB_2, LIB_3, main, and X_AxisActivate, each with a comment indicating its type (e.g., library function or function returning).
- Uses within HardwareInit**: A window showing the uses of variables within the HardwareInit function, such as `Bl_Axis.Geschwindigkeit` and `DspE4`.
- Calls within main**: A window showing the calls to functions within the main function, including `HardwareInit`, `ReadInput`, `WriteOutput`, `X_AxisAct`, `Y_AxisAct`, `Z_AxisAct`, `Bl_AxisAct`, `InpBuf`, `events`, and `WriteC`.

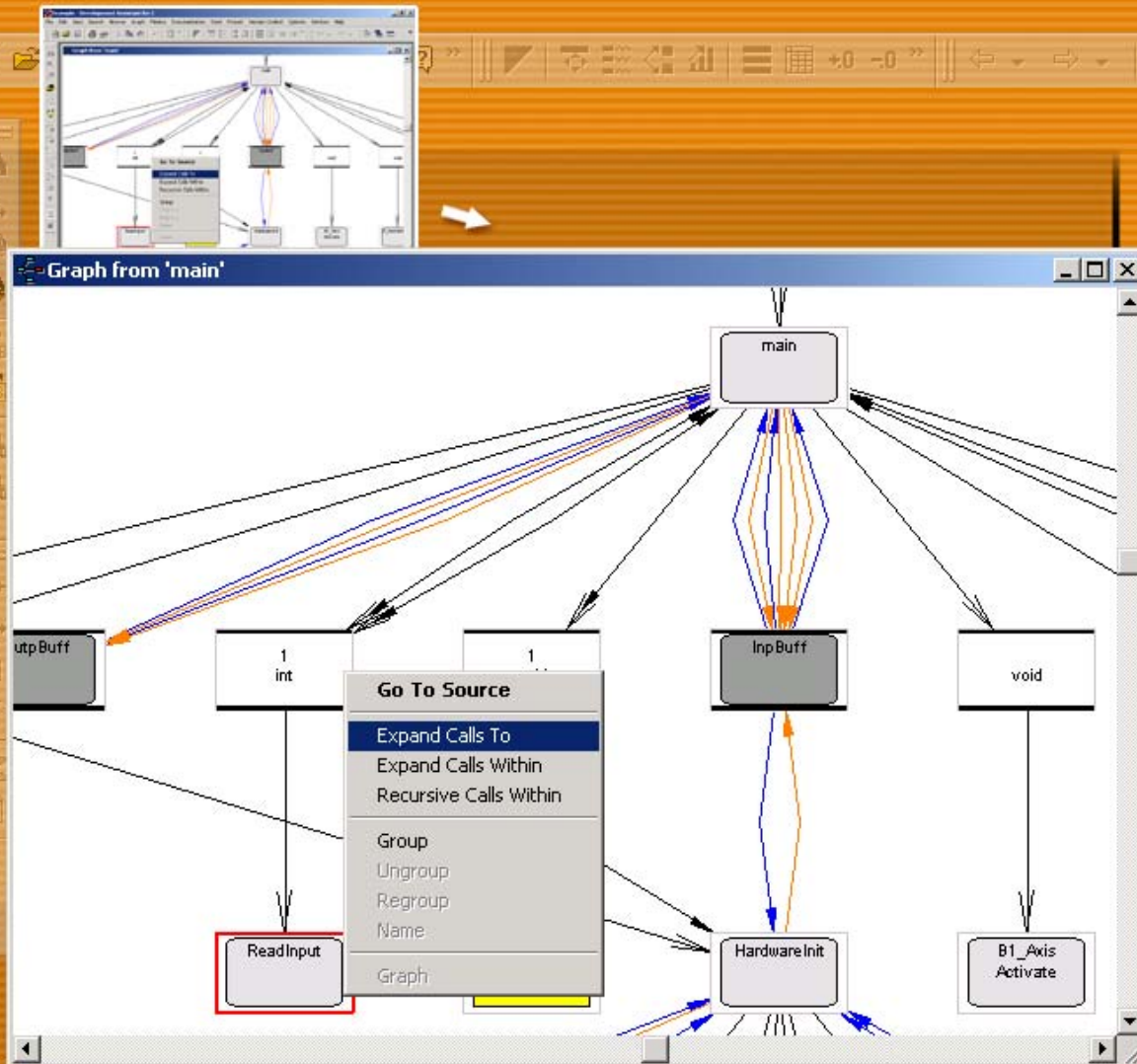
Context menus are visible over the code, offering options like Definition, Declarations, Uses, Assignments, Calls To, Calls Within, Uses Within, Graph, and Metrics. A smaller window at the bottom shows a project tree with a code editor and another analysis window open over it.

RistanCASE - Tools for Embedded System Developers

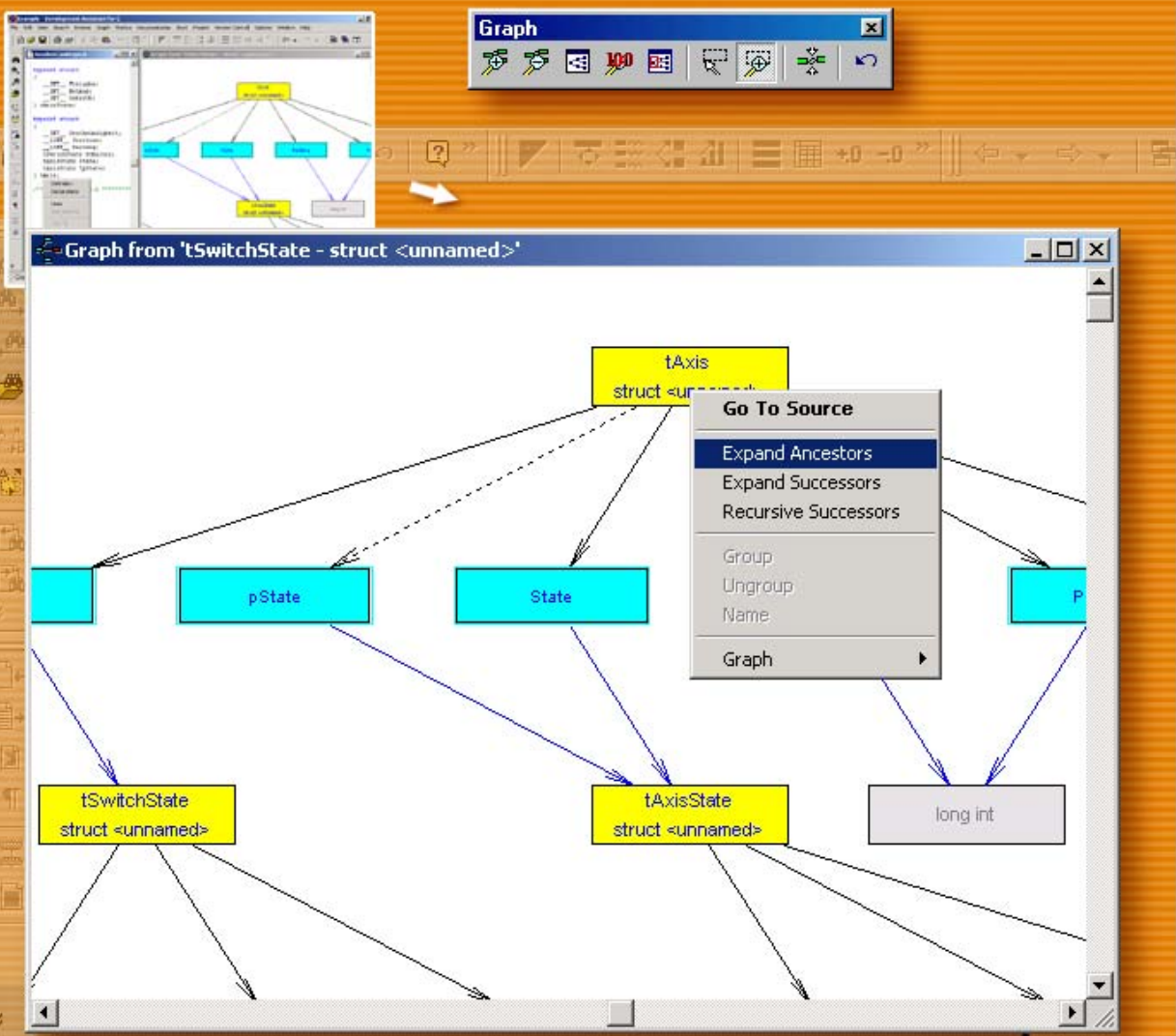


Call-Hierarchy and Data Flow Graph

DAC can also generate a call-hierarchy that graphically displays the structure of the overall program. Like the browse feature these graphs allow you to navigate your code, by clicking in a function box you can jump to the source code, by clicking a connecting arrow you can jump to the function call. The call hierarchy also shows the parameters that are passed between functions, thus exposing the data flow within the program. For large programs where the call hierarchy may be difficult to view, DAC allows to place logical groups of functions into a subsystem. This subsystem can then be expanded or viewed in a second window as required. DAC also allows you to graphically view complex data objects such as structures and unions directly from the source code.



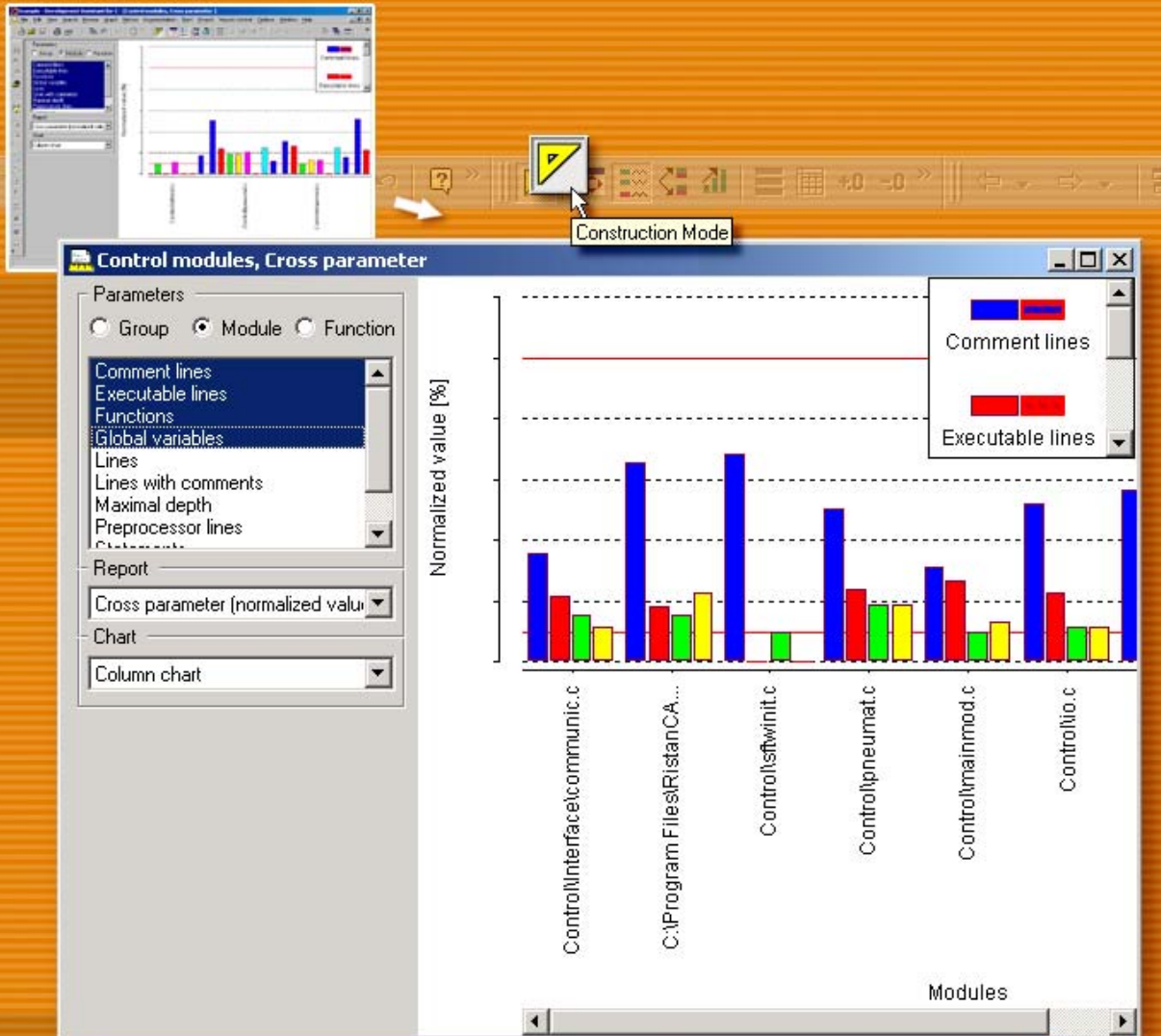
DAC - Software Development, Quality and Documentation Tool



Type-Hierarchy Graph

The type hierarchy graph allows you to view complex C objects such as structures and unions in a graphical format. The Type Graph allows you to instantly visualize the data structures within your program and, like the call hierarchy graph, the graphical display is directly linked to the original source code.

RistanCASE - Tools for Embedded System Developers



Software Metrics

The metrics module within DAC contains over 40 of the most important algorithms used to measure aspects of your source code such as complexity, testability and code quality. These metrics can be used to build up an objective view of the existing code. Metrics also allow certain limits, i.e. maximum code complexity or language use, to be imposed during the development process. Pre-defined templates can be made so that the same analysis can be applied on different projects or by several programmers.

DAC - Software Development, Quality and Documentation Tool



Documentation Generator

DAC also takes the hard work out of project documentation. With the Automatic documentation generator you can incorporate free text and all the rich graphical and textual information within DAC into a formal document.

DAC contains a unique documentation markup language DTML that allows you to design a documentation template that can be applied to every module in your project. This allows you to design the template once and then generate the vast bulk of your project documentation automatically. Since this information is generated automatically, project documentation can be kept continuously up to date. DAC contains templates for common documents and the wizard helps you define new templates so you don't have to learn the DTML script language.

RistanCASE - Tools for Embedded System Developers

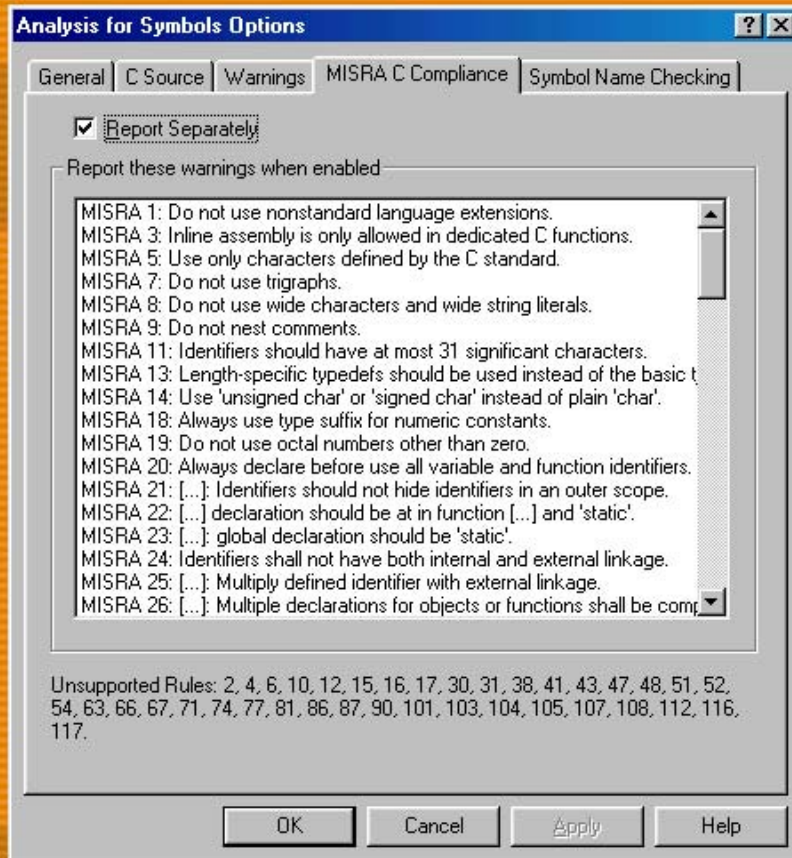


MISRA C Compliance Check

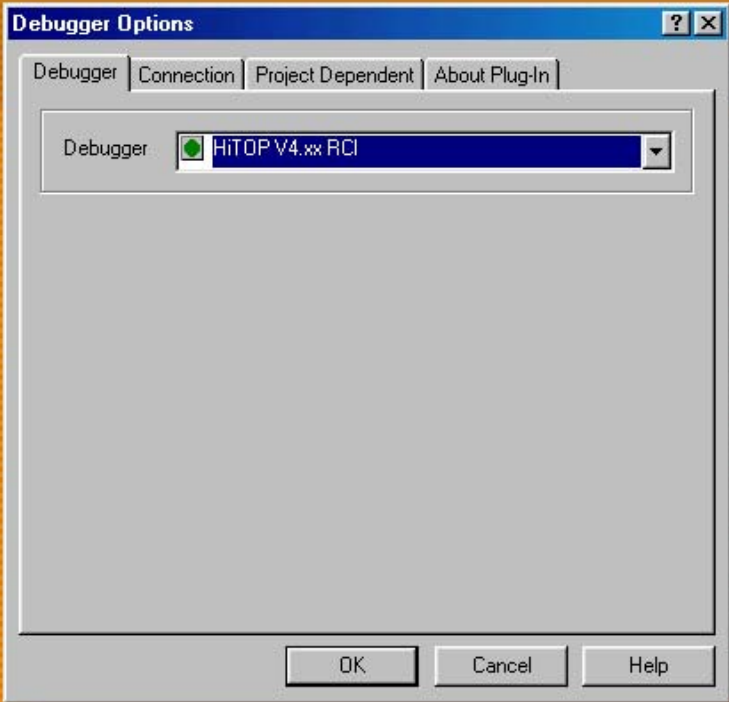
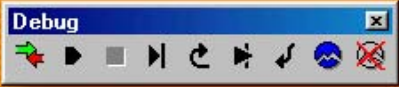
The MISRA (Motor Industry Software Research Association) "Guidelines for the use of the C language in vehicle based software", known as MISRA C is a set of 127 rules that implement a subset of the C language. The guidelines are designed to make C more suitable for use in safety related systems. MISRA C was originally written for the automotive industry but it is now in widespread use across the whole C programming world.

MISRA C is available in book form from www.misra.org.uk and www.hitex.co.uk. MISRA C is written in a clear and easily accessible style.

The majority of the MISRA C rules are statically checkable. The DAC analyzer has specific MISRA C reporting.



DAC - Software Development, Quality and Documentation Tool



Debugger Support

The DAC can be linked directly to software simulators and debuggers. This allows you to examine program flow directly within your editor and on the flow chart display. The DAC allows you to control the external debugger by setting breakpoints starting/stopping the execution and single stepping.

RistanCASE - Tools for Embedded System Developers

Other Features:

Project Window, Formatter, Command Shell, Analysis for Symbols, Message Window, Makefile Generator, Version Control System Support, Project Importer, DDE Server



Supports tools for:

PIC, M16C, MSP430, H8S, CRI6, C166, ST6, TMS320, ST7, ST9, HC05, HC08, HC11, HC12, 8051 as well as for many other processors



RistanCASE

RistanCASE GmbH Zielackerstrasse 19 CH-8304 Wallisellen Switzerland
Tel. ++ 41 (0)1 883 35 70 Fax ++ 41 (0)1 883 35 74
E-mail: sales@RistanCASE.com Web: www.RistanCASE.com



Symbol Name Check

A standard naming convention for symbols in your code can make a project far easier to read and maintain. At a glance you can tell if a variable is global, local, static, etc. The DAC allows you to define symbol naming conventions and the analyzer will then check that these have been followed.

User-Defined Actions

A fully-featured macro language is an integral part of the DAC. This macro language allows you to automate common tasks within the DAC and also integrate external tools, such as cross compiler toolsets, word processors, etc. The DAC comes with a library of pre-defined macros that can be used to set up a fully custom environment.