

# First Steps with DAC and 3rd Party Tools



**Software Development, Quality and Documentation Tool**

Development Assistant for C V4.0 Documentation  
Release date: August 26, 2002

File name: "DAC First Steps A4.pdf"  
Version 1.3

URL: [http://www.ristancase.com/dac/v40/dac\\_download.php](http://www.ristancase.com/dac/v40/dac_download.php)

Copyright © 1990-2002 RistanCASE GmbH Switzerland. All rights reserved.

The information contained in this document is subject to change without notice.

RistanCASE GmbH makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. RistanCASE GmbH shall not be liable for errors contained herein, direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of RistanCASE GmbH.

IBM PC, XT, AT are trademarks of the International Business Machine Corporation.

Microsoft® Windows is a registered trademark of the Microsoft Corporation.

All other trademarks used in this document may be trademarks or registered trademarks of their respective owners and are hereby acknowledged.

### **Corporate Headquarters**

RistanCASE GmbH  
Zielackerstrasse 19  
CH-8304 Wallisellen-Zurich  
Switzerland

Telephone: +41 (0) 1 883 35 70  
Fax: +41 (0) 1 883 35 74

E-mail: [info@RistanCASE.com](mailto:info@RistanCASE.com)

Web: [www.RistanCASE.com](http://www.RistanCASE.com)

---

This document was designed to be distributed electronically and then printed on a laser printer on an as-needed basis. Therefore, the fonts and layout of this document have been chosen for optimal printing rather than for optimal viewing on-screen. To view this document on-screen, however, simply increase the magnification using the magnification box at the top of the window. For the best results when viewing dialog boxes on-screen, increase the magnification to 200%. Some figures have hot links on them.

# Contents

<b>I Requirements .....</b>	<b>1</b>
<b>2 Configuring DAC .....</b>	<b>2</b>
2.1 Creating a New Project.....	2
2.2 Configuring Working Directories .....	3
<i>Project Root directory</i> .....	3
<i>Referential Project Root directory</i> .....	3
<i>Header Directories</i> .....	3
<i>Database directory</i> .....	3
<i>User Help file</i> .....	4
2.3 Configuring File Types .....	4
<i>C Source file</i> .....	4
<i>Assembler Source file</i> .....	4
<i>Header file</i> .....	4
<i>Document file</i> .....	5
<i>Text file</i> .....	5
<i>Referential pairs</i> .....	5
2.4 Configuring Analysis for Symbols.....	5
2.4.1 General .....	5
<i>Environment substitutes</i> .....	6
2.4.2 C Source.....	6
<i>Maximum identifier length</i> .....	7
<i>Special table processing</i> .....	7
<i>Defines / Control file</i> .....	7
<i>Nested comments</i> .....	7
<i>C++ comments</i> .....	7
<i>Traditional</i> .....	7
<i>Ignore #line</i> .....	7
2.5 Configuring Compiler Dialect and Header Directories .....	8
<i>Source</i> .....	8
<i>Compiler header directories</i> .....	8
<i>Preinclude header file</i> .....	9

2.6 Configuring Assembler Dialect and Header Directories .....	9
<i>Source</i> .....	9
<i>Assembler header files</i> .....	10
<i>Preinclude header file</i> .....	10
<i>Dialect specific options</i> .....	10
2.7 Adding Files to the Project .....	10
2.8 Building the Database .....	11
<b>3 Integrating the Tools .....</b>	<b>13</b>
3.1 User-Defined Actions (UDA) Setup .....	13
3.2 Makefile Template Setup .....	14
3.3 Using 3rd Party Tools by Default .....	15
<b>4 Limitations of Static Code Analysis .....</b>	<b>16</b>
<b>5 Index .....</b>	<b>17</b>

---

# I Requirements

---

- **DAC - V4.0.055 or later** - (Development Assistant for C - RistanCASE). The latest version, with Demo Mode license included, can be downloaded from the following URL:  
[http://www.RistanCASE.com/dac/v40/dac\\_download.php](http://www.RistanCASE.com/dac/v40/dac_download.php)

If you are running DAC in Demo Mode, you can easily obtain a trial license and enjoy all the comforts of DAC for two weeks! For more details, choose **Technical Support** from the **Help** menu.

If you don't have make utility already, and still want to use it with DAC, GNU make utility is on your disposal. **GNU make V3.75 or later** - (Free Software Foundation). The gmake.exe with RistanCASE improvements can be downloaded from the following URL:  
[http://www.RistanCASE.com/gnu/gnu\\_products.php](http://www.RistanCASE.com/gnu/gnu_products.php)

**NOTE:**

For further information on DAC, please refer to "Development Assistant for C" documentation V4.0.

It is assumed that DAC has been installed in the "*C:\Program Files\RistanCASE\Development Assistant for C*" folder, which does not have to be in the system PATH. In the text that follows this folder will be referred to as the "DAC folder."

It is also assumed that a 3rd party tool has been installed in the appropriate folder that will be referred to as the "3PT folder."

---

## 2 Configuring DAC

---

### 2.1 Creating a New Project

Start DAC and choose **New Project** from the **Project** menu. Browse through the 3PT folder and enter the project file name, for example "My project". A project file will be created.

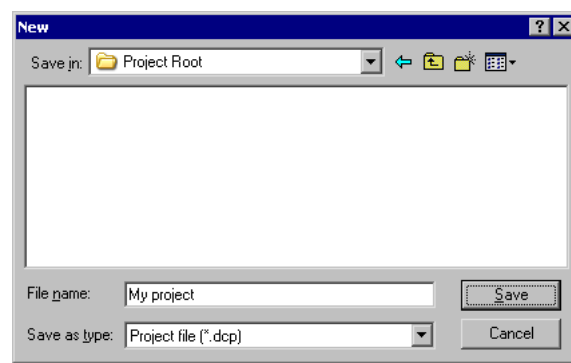


Figure 2.1 Creating a new project

Another way is to right-click in Windows Explorer to show the shortcut menu, point to **New**, and then click the **Development Assistant for C Project**.

The third way is to use the **Import Project** command from the **Project** menu. This allows you to import project from some editors or version control systems.

## 2.2 Configuring Working Directories

On the **Options** menu, click **Project** to open the **Project Options** dialog box. Use this dialog box to set project directories:

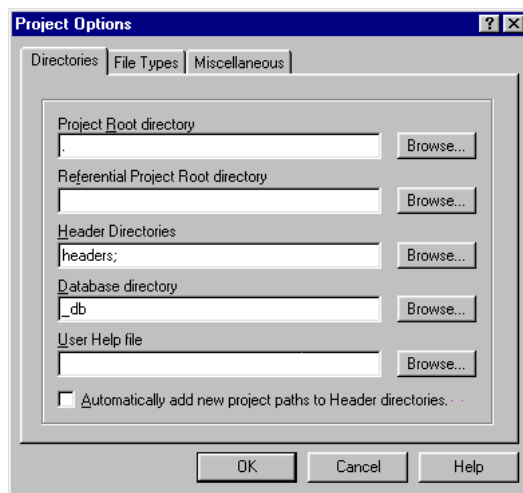


Figure 2.2 Configuring working directories

### Project Root directory

The full path is expected in the Project Root directory text box or, as a special value, a single period character, which stands for "the directory where the project file resides." The names of all files belonging to the project are considered relative to the Project Root directory if the full file path is not given.

### Referential Project Root directory

If not empty, it specifies an alternative path for searching for files that DAC fails to find in the original project path. DAC will attempt to find files with the referential extension in the Project Root directory prior to searching the referential Project Root directory. The specified path may either be full or relative to the Project Root, and it may not specify a sub-folder in the Project Root directory tree.

### Header Directories

You should specify paths to all header directories used in the project. The paths are separated by a semicolon.

### Database directory

It allows the Symbols and Software Metrics database files folder to be set. The path in question can be absolute or relative to the Project Root directory. If the Database directory is left unspecified, the Symbols and Software Metrics database files are created in the Project Root directory along

with source and other files. It is recommended that you should specify the Database directory, to keep these files from mixing.

### User Help file

It enables you to set the User Help file, for example, the compiler help file. The shortcut key for the User Help file can be set in the Keyboard definition file (default CTRL + SHIFT + F1).

## 2.3 Configuring File Types

On the **Options** menu, click **Project** and select the **File Types** tab. Here you can set file extensions for the basic project file types. A file type extension list consists of up to 10 file extensions separated by a white space or the "." character. Here are the default values:

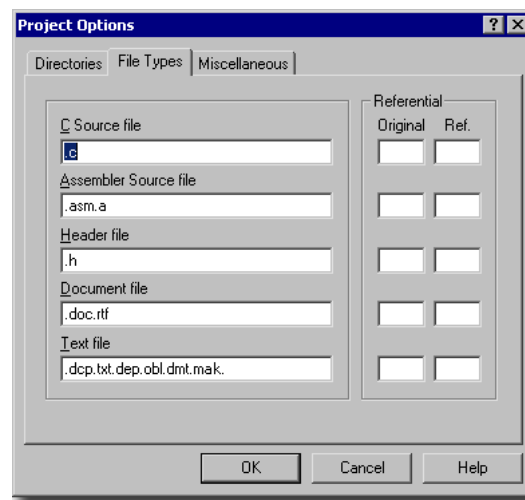


Figure 2.3 Configuring File Types

### C Source file

Files with one of the extensions listed in this box will be considered C source files. Header file extensions should not be defined within this field.

### Assembler Source file

Files with one of the extensions listed in this box will be considered included Assembler files. Module file extensions should not be defined within this field.

### Header file

Files with one of the extensions listed in this box will be considered included C header files. Module file extensions should not be defined within this field.

### Document file

Files with one of the extensions listed in this box will be considered document files.

### Text file

Files with one of the extensions listed in this box will be considered editable (text) files. If files without extensions are to be considered editable, the list ends with a period ".".

### Referential pairs

Fields for entering alternative (referential) extensions are located in this area. The original extension is entered in the Original column, while the alternative extension is entered in the Referential column. Up to five original / alternative pairs can be set. If opening the file with the original extension fails, DAC will attempt to find and open the file with the identical name and the alternative extension.

## 2.4 Configuring Analysis for Symbols

You may need to configure Analysis for Symbols Options from [General](#) and [C Source](#) sections.

### 2.4.1 General

On the **Options** menu, click **Analysis for Symbols**, and then click the **General** tab to open the following dialog box:

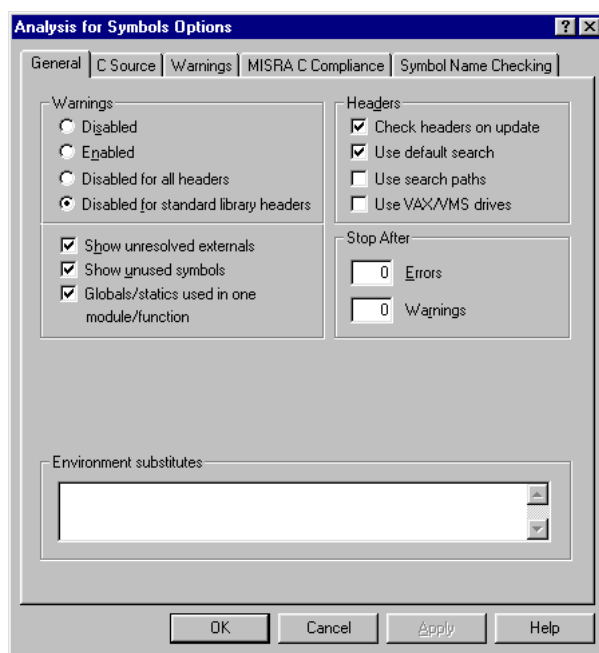


Figure 2.4 General Analysis for Symbols Options

## Environment substitutes

Environment variables defined in Windows can be made a part of various path specifications in DAC by enclosing the variable identifier in "%" characters. Apart from variables defined in the Windows environment, it is possible, in **Environment substitutes**, also to define "DAC environment variables" in the form of:

```
env_var=env_string
```

If the %env\_var% construct is found in DAC paths (project root directory, database directory, include path, %Env(env\_var), ...) %env\_var% will be replaced with an env\_string.

These substitutions are also necessary if the environment variables of your development environment have been set only in the DOS window in which you build project binaries. As the variables are not known at Windows level, if you intend to use them, you will have to define them in DAC, in **Environment substitutes**.

For a detailed explanation of other options in the **General** tab please consult DAC help.

### 2.4.2 C Source

On the **Options** menu, click **Analysis for Symbols**, and then click the **C Source** tab to open the following dialog box:

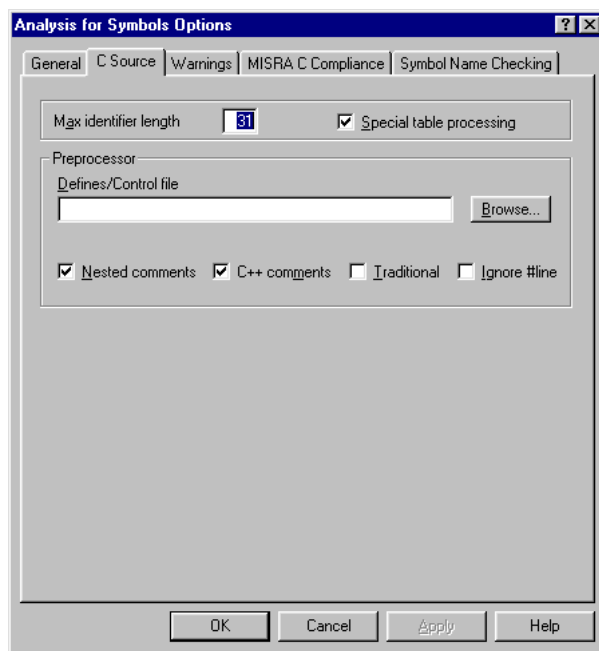


Figure 2.5 C Source Analysis for Symbols Options

**Maximum identifier length**

This is the number of significant characters in the identifiers. The remaining characters are ignored.

**Special table processing**

It determines the treatment of arrays of pointers to functions in the analysis. If this option is selected, such variables are treated as functions, otherwise, they are treated in the usual way (default).

**Defines / Control file**

It determines the list of macro definitions to be set at the beginning of the module preprocessing. This box may contain the control file name (with absolute or relative path in relation to the project root) with the prefix `@`. The prefix `@` is also used in the Project file `".dcp"` to specify included files. Therefore, you should avoid using the character `@` at the beginning of the names of files and folders that are to be used in the project. Control file lines beginning with `-i` or `-I` subsequently contain the list of directories for searching for `#include` files, separated by a semicolon. Control file lines beginning with `-d` or `-D` subsequently contain the list of macro definitions, as described in the previous paragraph. The remaining control file lines are ignored.

**Nested comments**

It allows proper recognition of nested C style comments, for example `/*  
/* ... */*/`.

**C++ comments**

It determines whether C++ comments (beginning with `"/"/`) are to be recognized while analyzing the source code.

**Traditional**

It determines if the pre-ANSI preprocessor is to be used. For the traditional preprocessor, a comment is equivalent to nothing, while for the ANSI preprocessor it is a white space. The traditional preprocessor does not delete comments within the `#define` directive, it does not recognize `"#"` and `"##"` operators, and requires an initial `"#"` in a directive at the very beginning of the line. It allows specification `foo()` if a macro `foo` takes a single argument, and permits a macro to be expanded recursively.

**Ignore #line**

It determines that the `#line` preprocessor directive be ignored.

## 2.5 Configuring Compiler Dialect and Header Directories

An additional path configuration must be performed to specify the location of the compiler library header file (needed for DAC symbol analysis), if you changed your compiler in recent time

On the **Options** menu, click **Compiler** to open the **Compiler Options** dialog box. This dialog box contains options which enable you to set C source code analysis parameters:

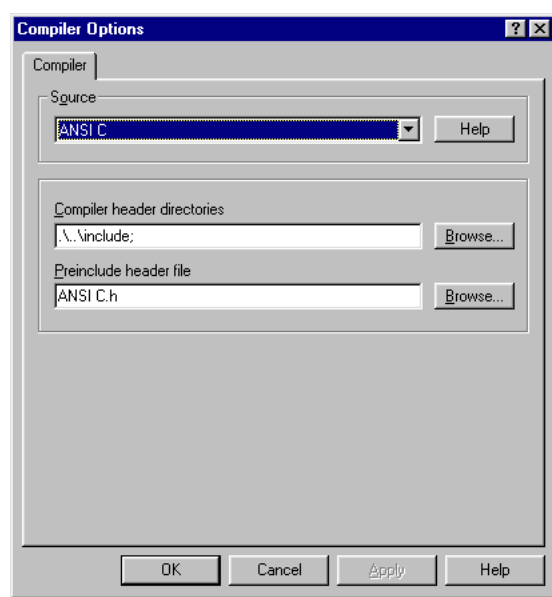


Figure 2.6 Compiler Options

### Source

The supported C dialects of the C language used in the active project can be selected in this box.

### Compiler header directories

It enables you to set the list of directories which are to be searched for the files named using the "#include" directive. Different directories within a list are separated by a semicolon character. Only listed directories are searched for files whose names are written enclosed in brackets ("<" and ">"). The search for files whose names are written enclosed in quotation marks ("") starts in the folder in which the source file containing the "#include" directive resides.

The list of header directories can be assigned to a file. In that case, the **Compiler header directories** box contains the file name (with absolute or relative path in relation to the Project Root) with the @ prefix. The directories listed in the file are separated by a semicolon or a new line (no semicolon necessary). The prefix @ is also used in the Project file ".dcp"

to specify included files. Therefore, you should avoid using the character @ at the beginning of the names of files and folders that are to be used in the project.

### Preinclude header file

It enables you to set the name of the file which will be included automatically at the beginning of every C source module during the analysis, as if the #include directive containing the name of the file in question enclosed in quotation marks (") was present in the first line of the analyzed source code. The preinclude file is used to specify predefined macros, variable and function declarations for a particular compiler which are not set by default in DAC analysis. To adapt this file to your needs, on the **File** menu, point to **Configuration Files** and click **Compiler Preinclude File**.

## 2.6 Configuring Assembler Dialect and Header Directories

On the **Options** menu, click **Assembler** to open the **Assembler Options** dialog box. This dialog box contains options which enable you to set assembler source code analysis parameters:

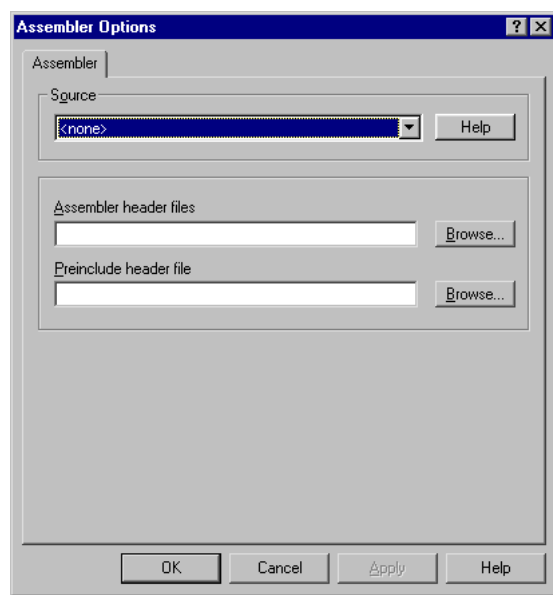


Figure 2.7 Assembler Options

### Source

The supported assembler language used in the current project. It is possible to select a particular dialect, or the value **<none>** which means that parts of the C code containing inline assembler code will be ignored

(information on symbols will not be retrieved, but no syntax errors will occur either).

### **Assembler header files**

It determines a list of directories which are to be searched for the files named within the `INCLUDE` directive. Directories are separated by a semi-colon.

### **Preinclude header file**

If a file is specified in this edit box (typed in or selected using the **Browse** button), it will be included in each assembler file prior to commencing assembler analysis. This is identical to entering the `INCLUDE` directive in each assembler file.

### **Dialect specific options**

Various assemblers also have a set of options which differs depending on the assembler selected in the **Source** option. See the Technical Note describing assembler dialects for more details.

## **2.7 Adding Files to the Project**

In the **Project Window** the **Explorer View** replaces the Windows Explorer and supplies you with plenty of information on directories containing project files. It also gives you the possibility of adding files to the project. All files needed to run project will now be added to the project.

In the **Explorer View**, browse and select the root folder of your project. Right-click and choose **Add to project**. All file types configured in the previous section [Configuring File Types](#), are now added to the project:

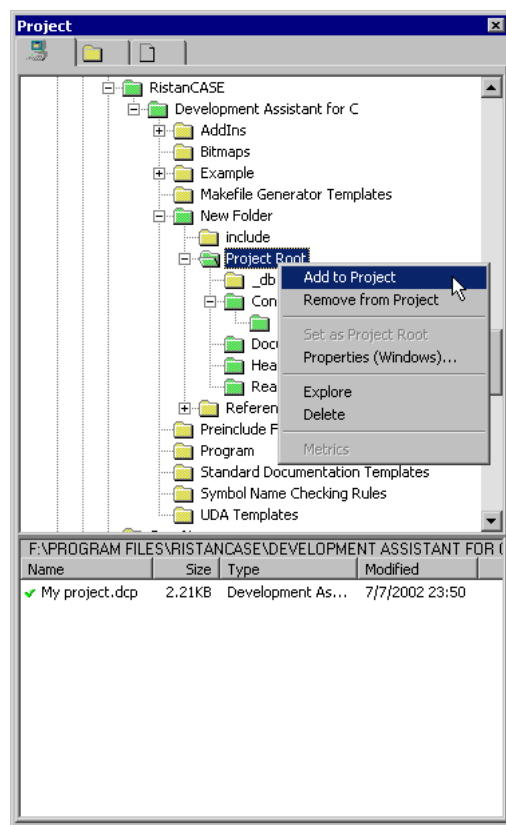


Figure 2.8 Project Window - Explorer View (Undocked mode)

Another alternative is to add each file separately by selecting each one and adding it to the project.

## 2.8 Building the Database

Development Assistant for C provides the static code analysis of C and ASM source files, and generates various data based on the results.

The analysis of the project source files and the generation of the database are divided into two phases: the analysis of individual program modules and the generation of data on global symbols usage. The results of the analysis are saved in database files on the disk, allowing them to be used in DAC later on. You can choose between the unconditional analysis of all project files and the analysis of changed source files only, using the **Build Database** and the **Update Database** commands from the **Start** menu, respectively. The **Update Database** command will optionally check if the include files used in program modules have been changed as well.

To build the database, on the **Start** menu click the **Build Database** command. This command performs the unconditional analysis of all project files and creates a database containing information on the analyzed source code. The errors and warnings detected during this operation are displayed in the **Messages** window.

DAC after using the **Build Database** command on the **Start** menu, with displayed **Messages** window, the **Project** window with selected **Logical View** tab (for overview of the project), and the **Editor** window with Structure Highlighting on and a non structured Flow Chart of the code, is shown on the following figure:

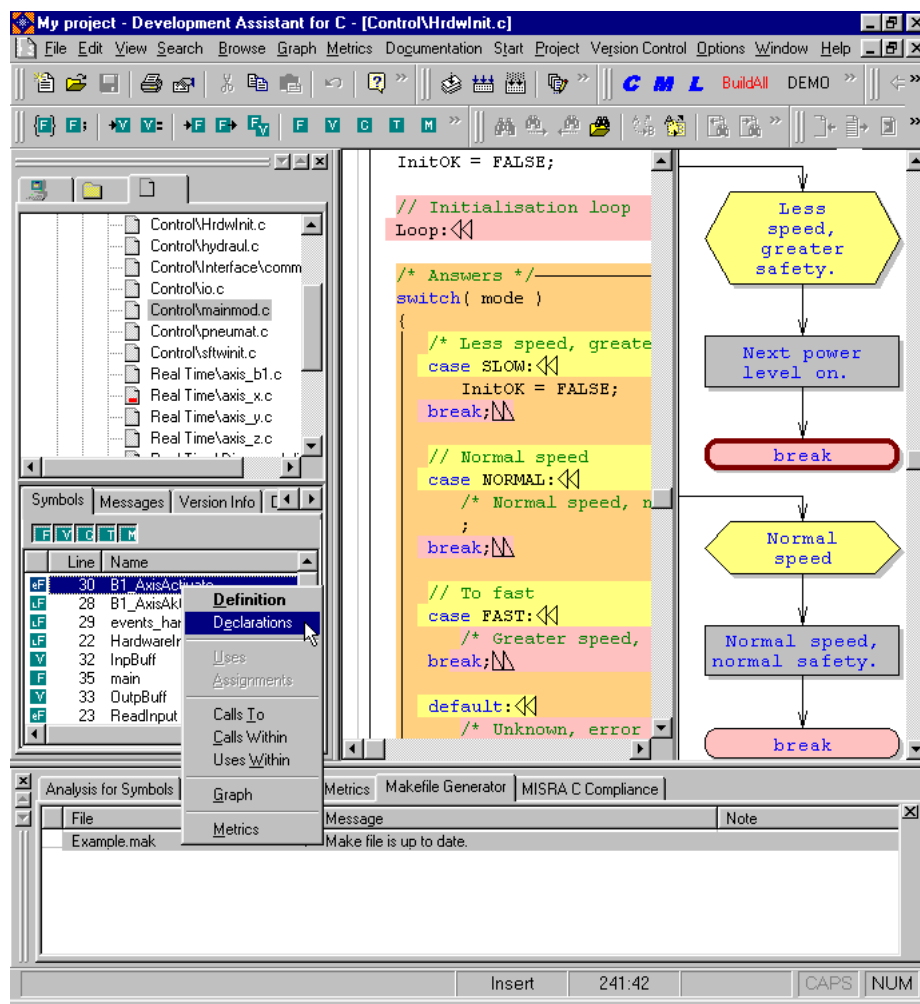


Figure 2.9 DAC after using the **Build Database** command on the **Start** menu

Once the analysis of all project files has been completed, the new database file containing the information on global symbols is created. Please refer to the DAC manual for further information on how symbols information can be used.

---

## 3 Integrating the Tools

---

### 3.1 User-Defined Actions (UDA) Setup

To integrate 3rd party tools into DAC, on the **Option** menu, click **User-Defined Actions**, and then click the **Start Menu Actions** tab.

Now check the **Use compiler template** check box. By selecting this option, you have decided on what is probably the most common manner of using 3rd party tools. The action is contained in the compiler template file located in the "*UDA Templates*" subfolder of the DAC folder. The compiler template file corresponds to the chosen compiler from the **Compiler Options** dialog box. You can edit it and adjust it to your needs. RistanCASE would appreciate your sharing additional features and ideas with us. Now click **OK**.

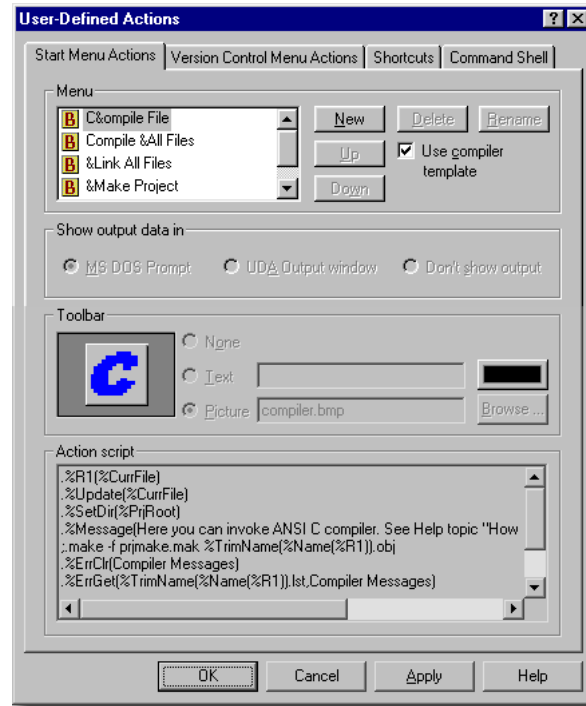







Figure 3.1 User-Defined Actions - Start Menu Actions

On the **View** menu, point to **Toolbars**, and then click **User-Defined Actions**. Five UDA buttons should now appear on the DAC toolbar, and the items **Compile File**, **Compile All Files**, **Link All Files**, **Make Project**, and **Build Project** on the **Start** menu. On the toolbar, appearing from left to right, the UDA buttons are:



Figure 3.2 UDA toolbar (undocked mode)

- Compile File 
- Compile All Files 
- Link All Files 
- Make Project 
- Build Project 

### 3.2 Makefile Template Setup

In order to completely define the **Compile File**, **Compile All Files**, **Link All Files**, **Make Project**, and **Build Project** UDA with the gmake utility, a project makefile must be provided. This file should have the same name

as the current project and the extension *".mak"*. It can be edited manually or generated automatically.

The template files are generally used to define all the necessary macros, commands, rules, and so on, which facilitate automatic project makefile generation by means of the Makefile Template Macro Language instructions expansion.

How to set **Makefile Generator Options** for particular tool, please refer to specific Technical Note for that tool.

### **3.3 Using 3rd Party Tools by Default**

If you want your newly created project files to automatically inherit the proper setup for 3rd party tools, you should also customize the default DAC project file *"dac.dcp"*, which is located in the *"Program"* subfolder of the DAC folder.

---

## 4 Limitations of Static Code Analysis

---

Since DAC Static Code Analysis supports a great number of dialects, it stands to reason that each and every one of these dialects cannot be supported in minute detail.

However, if you find that any of these limitations prevents you from using DAC in the optimal way, please, send the description of that limitation to: [support@RistanCASE.com](mailto:support@RistanCASE.com).

You can find more about these limitations in User Manual, chapters: Pre-processor, and Dealing with Syntax Extensions.

---

## 5 Index

---



---

### Symbols

#include 7-8  
 @ 7  
 @ prefix 8

---

### A

Assembler Options 9

---

### B

Build Project 14  
 Building the Database 11

---

### C

C Source 6  
 Compile  
   *All Files* 14  
   *File* 14  
 Compiler  
   *header directories* 8  
 compiler help 4  
 Configuring  
   *Analysis for Symbols* 5  
   *Assembler Dialect and Header Directories* 9  
   *Compiler Dialect and Header Directories* 8  
   *DAC* 2  
   *File Types* 4  
   *Working Directories* 3

---

### D

-D 7  
 -d 7

### DAC

*Demo Mode* 1  
   *toolbar* 14  
 DAC - V4.0.055 1  
 dac.dcp 15  
 Database 3  
   *Build* 11  
   *directory* 3  
   *Update* 11  
 Defines / Control file 7

---

### E

Explorer View 10

---

### F

File Types tab 4  
 Free Software Foundation 1

---

### G

General 5  
 gmake 1  
 GNU 1

---

### I

-I 7  
 -i 7

---

### L

license  
   *Demo Mode* 1  
   *trial* 1  
 Limitations of Static Code Analysis 16  
 Link All Files 14

---

**M**

Make Project 14  
Makefile Template Setup 14  
Maximum identifier length 7  
Messages window 11

---

**N**

New Project 2

---

**P**

prefix @ 7  
Preinclude file 9  
Project  
    *Adding Files to the Project* 10  
    *Creating a New Project* 2  
    *Root Directory* 3  
    *Window* 10

---

**R**

Referential 3  
Requirements 1

---

**S**

Software Metrics 3  
Source 8  
Special table processing 7  
Start 11, 14  
Start Menu Actions 13  
Symbols 3

---

**U**

UDA  
    *buttons* 14  
    *Toolbar* 14  
Use compiler template 13  
User  
    *-Defined Actions (UDA) Setup* 13  
    *Help File* 4  
Using 3rd Party Tools by Default 15