

Using DAC with ARM Tools



Software Development, Quality and Documentation Tool



Development Assistant for C V4.0 Documentation
Release date: September 25, 2002

File name: "DAC ARM TN LT.pdf"
Version 1.3

URL: <http://www.RistanCASE.com/dac/v40/specificsupport/arm.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
Web: 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

| | |
|---|----------|
| I Requirements | 1 |
| 2 Configuring DAC | 2 |
| 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> | 4 |
| <i>Header Directories</i> | 4 |
| <i>Database directory</i> | 4 |
| <i>User Help file</i> | 4 |
| 2.3 Configuring File Types | 4 |
| <i>C Source file</i> | 5 |
| <i>Assembler Source file</i> | 5 |
| <i>Header file</i> | 5 |
| <i>Document file</i> | 5 |
| <i>Text file</i> | 5 |
| <i>Referential pairs</i> | 5 |
| 2.4 Configuring Analysis for Symbols..... | 6 |
| 2.4.1 General | 6 |
| <i>Environment substitutes</i> | 6 |
| 2.4.2 C Source | 7 |
| <i>Maximum identifier length</i> | 7 |
| <i>Special table processing</i> | 7 |
| <i>Defines / Control file</i> | 7 |
| <i>Nested comments</i> | 8 |
| <i>C++ comments</i> | 8 |
| <i>Traditional</i> | 8 |
| <i>Ignore #line</i> | 8 |
| 2.5 Configuring Compiler Dialect and Header Directories | 8 |
| <i>Source</i> | 9 |
| <i>Compiler header directories</i> | 9 |
| <i>Preinclude header file</i> | 10 |

| | |
|--|-----------|
| 2.6 Configuring Assembler Dialect and Header Directories | 10 |
| 2.7 Adding Files to the Project | 10 |
| 2.8 Building the Database | 11 |
| 3 Integrating the Tools | 13 |
| 3.1 User-Defined Actions (UDA) Setup | 13 |
| 3.2 Makefile Template Setup | 14 |
| 3.3 Using ARM Developer Suite V1.2x Tools by Default | 16 |
| 4 Index | 17 |

I Requirements

INFO:

This technical note only provides information on how to configure DAC to facilitate your work with the ARM Developer Suite V1.2x Tools, for which the document has primarily been written. Using it as a basis, DAC can easily be configured to work with other ARM tools. For further information on DAC, please refer to "Development Assistant for C" documentation V4.0.

- **DAC - V4.0.056 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

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.

- **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

In the following sections, it is assumed that your ARM tools have been installed in the "*C:\Program Files\ARMADSV1_2*" folder. In the text that follows, this folder will be referred to as the "ARM folder." Also, it is assumed that example and include files have been stored in the "*C:\ARM*" folder, which will be referred to as the "ARM Example folder." You may have to adapt the paths used in the example provided to match your current installation paths.

ARM tools and the GNU make utility folders have to be in the system PATH. The GNU make should be named "*gmake.exe*".

It is also 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."

NOTE:

Because of the current limitation of the GNU make V3.75 example and include files must be in the path that does not have spaces. Thus, you have to copy the "*Include*" subfolder of the ARM folder to the ARM Example folder.

2 Configuring DAC

To make the most of your ARM Developer Suite V1.2x tools with DAC, these are the steps to follow:

- [Creating a New Project](#)
- [Configuring Working Directories](#)
- [Configuring File Types](#)
- [Configuring Analysis for Symbols](#)
- [Configuring Compiler Dialect and Header Directories](#)
- [Configuring Assembler Dialect and Header Directories](#)
- [Adding Files to the Project](#)
- [Building the Database](#)
- [Integrating the Tools](#)

As you progress through this technical note, you will be led through the process of creating a new DAC project named "swi." The "swi" example project can be found on RistanCASE's CD-ROM in the "*DAC\Technical Notes\ARM*" folder, or downloaded from RistanCASE's web site: <http://www.RistanCASE.com/dac/v40/specificsupport/arm.php>

Supported tools by DAC - V4.0.056 or later

Supported compilers

ARM Developer Suite V1.2x

2.1 Creating a New Project

Start DAC and choose **New Project** from the **Project** menu. Browse through the "*swi*" subfolder of the ARM folder and enter the project file

name "swi". A project file will be created. The standard project file extension is ".dcp", so the created file will be named "swi.dcp".

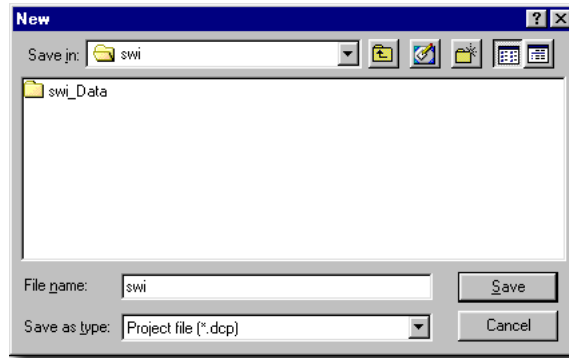


Figure 2.1 Creating a new project

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:

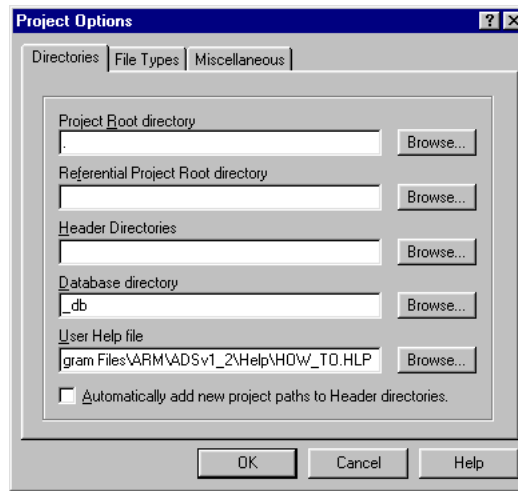


Figure 2.2 Configuring working directories

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. In our example, keep the single period character for the Project Root directory.

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 subfolder in the Project Root directory tree. In this example it should be left empty.

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. Enter "_db" in the box.

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). Browse through the "Help" subfolder of the ARM folder and select the "HOW_TO.HLP" help file. This will give you an entry point for ARM Help.

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. To ensure the maximally efficient use of ARM Developer Suite V1.2x tools, fill in the boxes as shown in the following picture:

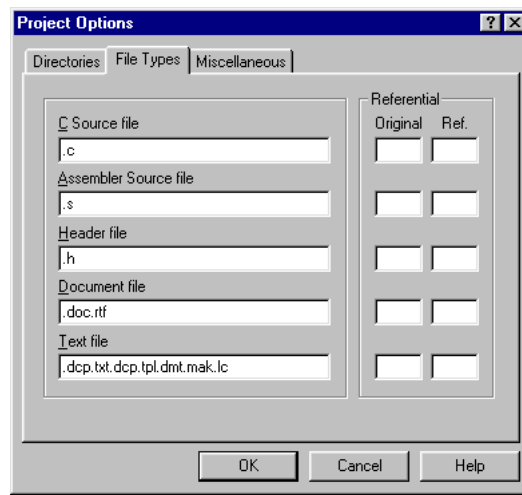


Figure 2.3 Configuring File Types

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. Enter ".c" in the box.

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. Enter ".s" in the box.

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. Enter ".h" in the box.

Document file

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

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 ".". Enter ".txt.dcp.dmt.mak.tpl.lc" in the box.

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

For this example 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:

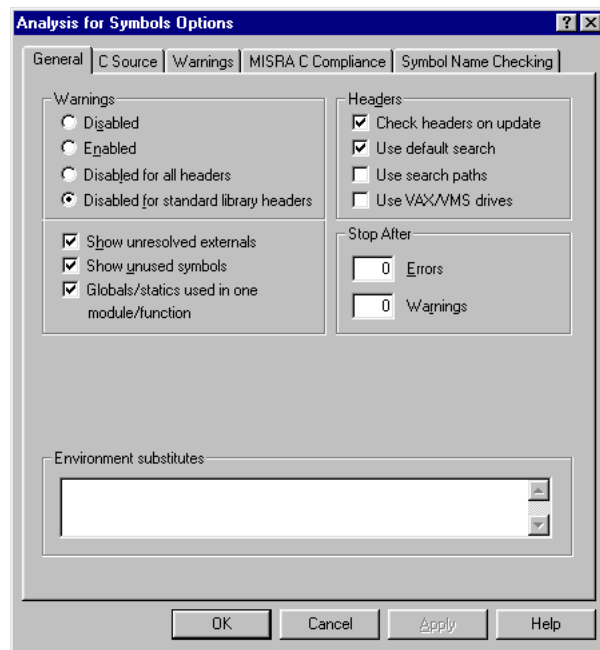


Figure 2.4 General Analysis for Symbols Options

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 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:

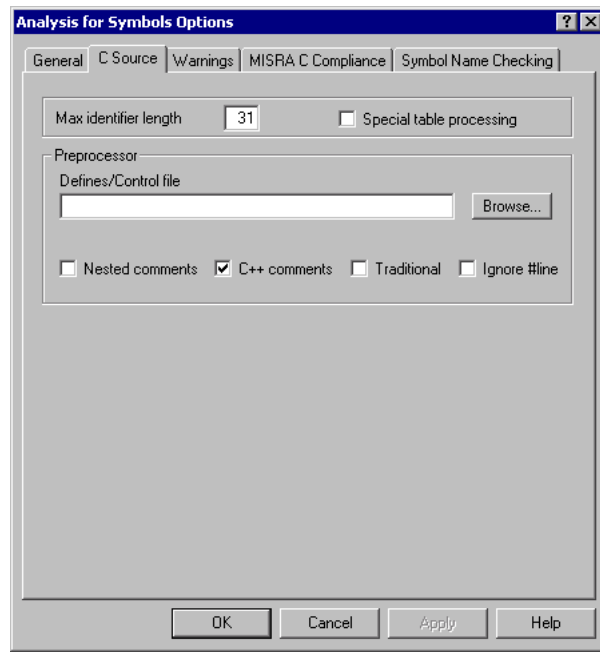


Figure 2.5 C Source Analysis for Symbols Options

Maximum identifier length

This is the number of significant characters in the identifiers. The remaining characters are ignored. The default value is 31.

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 if 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 ARM Developer Suite V1.2x C compiler library header file (needed for DAC symbol analysis). 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:

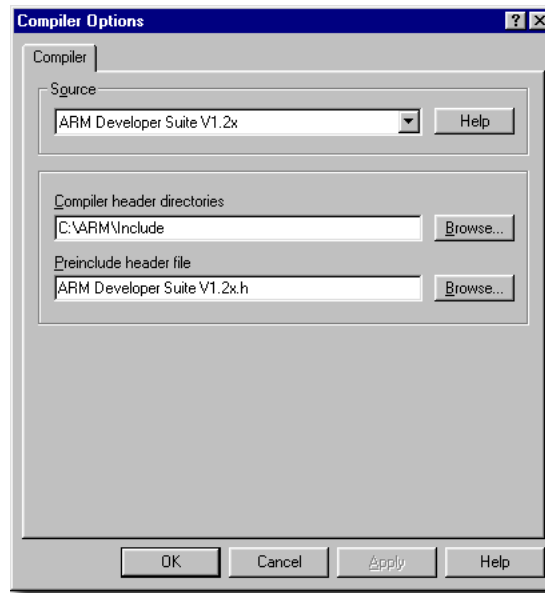


Figure 2.6 Compiler Options

Source

The supported C dialects of the C language used in the active project can be selected in this box. In the "swi" example project the ARM Developer Suite V1.2x C dialect is used.

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 the 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.

The path of the Compiler libraries for the used CPU should be defined as the "Include" subfolder of the ARM Example folder.

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. The one corresponding to the ARM Developer Suite V1.2x C Compiler is: "ARM Developer Suite V1.2x.h". This file contains declarations of ARM Developer Suite V1.2x C Compiler predefined macros. 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

The assembler support for ARM Developer Suite V1.2x has not been released yet. On the **Options** menu, click **Assembler**, and then in the **Source** box select **<none>**.

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 the "swi" example will now be added to the project.

In the **Explorer View**, browse through the ARM Example folder, right-click on the "swi" subfolder and, from the shortcut menu, choose **Add to project**. All file types configured in the previous section [Configuring File Types](#), are now added to the project. Remove the files not belonging to the project.

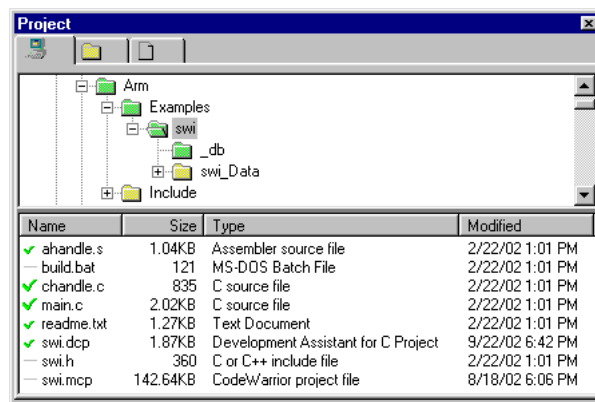


Figure 2.7 Project Window - Explorer View (Undocked mode)

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 in the "swi" example, 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 as illustrated below (for all files included in the project):

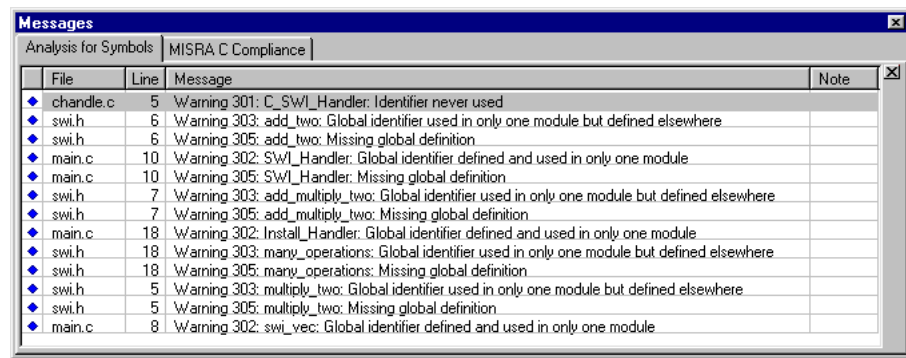


Figure 2.8 Messages Window - Analysis for Symbols tab (Undocked mode)

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.

In the DAC **Project Window**, select the **Logical View** tab and unfold fields to get an overview of your project:

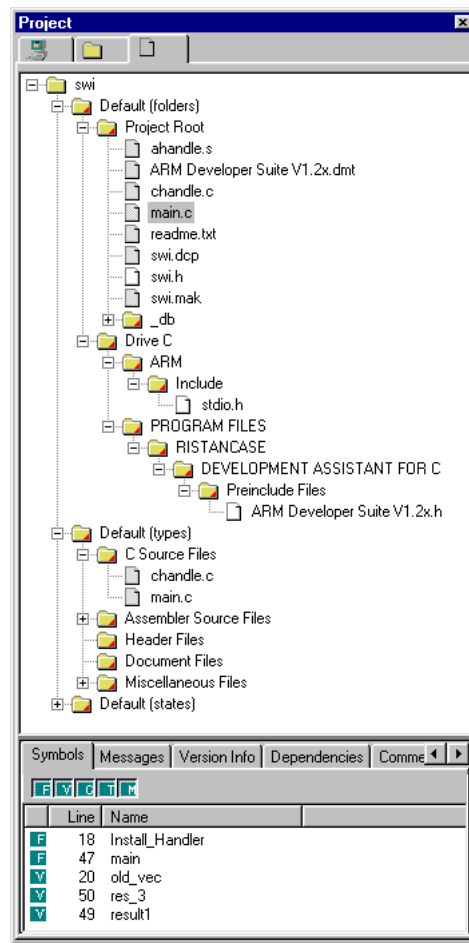


Figure 2.9 Project Window - Logical View (Undocked mode)

3 Integrating the Tools

3.1 User-Defined Actions (UDA) Setup

To integrate the ARM Developer Suite V1.2x tools and GNU make 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 ARM tools. The action is contained in the "*ARM Developer Suite V1.2x.tpl*" file located in the "*UDA Templates*" subfolder of the DAC folder. You can edit it and adjust it to your needs. RistanCASE would appreciate your sharing additional features and ideas with us. Now click **OK**.

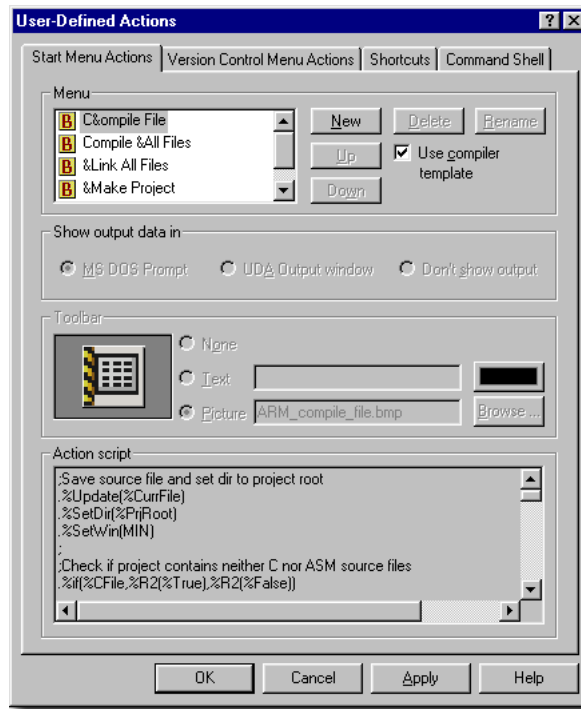







Figure 3.1 User-Defined Actions - Start Menu Actions

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.

INFO:

You inevitably have to make the necessary customizations of the appropriate file for your compiler. Follow the directions given in the template file and consult your compiler tools literature if and when necessary.

Since manual editing is a very tedious and error-prone process, an example makefile template *"ARM Developer Suite V1.2x.dmt"* has been provided in the *"Makefile Generator Templates"* subfolder of the DAC folder.

Another file that should be customized to reflect the properties of the target system on which a program is to be executed is the linker command file. If the *"ARM Developer Suite V1.2x.dmt"* makefile template is expanded, the linker command file will automatically be generated along with the makefile. The template file section responsible for the linker command line file generation is at the end of the template and is commented.

To enable automatic makefile generation, do the following:

- Copy the file *"ARM Developer Suite V1.2x.dmt"* from the *"Makefile Generator Templates"* subfolder of the DAC folder to the *"swi"* subfolder of the ARM folder.
- Customize *"ARM Developer Suite V1.2x.dmt"* to your needs.
- On the **Options** menu, click **Makefile Generator**, and then select **Generate makefile or its components**.
- In the **Makefile template** box enter *"ARM Developer Suite V1.2x.dmt"* or use the **Browse** button to find it.
- Click **OK**.
- On the **Start** menu, click **Generate Makefile**. The makefile for your project will be generated automatically.

NOTE:

On every start of the **Build** or **Update the Database** command from the **Start** menu, DAC will automatically check if the makefile needs to be regenerated.

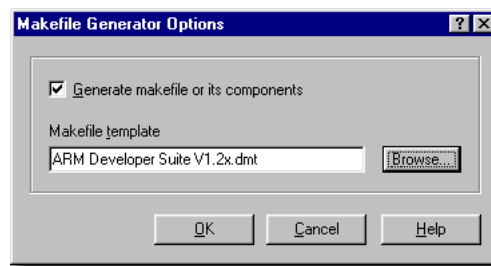


Figure 3.3 Makefile Generator Options

The report of the automatic project makefile generation process is displayed in the **Messages** window under the **Makefile Generator** tab. After the project makefile has successfully been created, you can use the **Link All Files**, **Make Project**, or **Build Project** UDA to generate the "*Ad.out*" file.

The generated "*Ad.mak*" file can be opened for inspection. Please, bear in mind that any changes made in it would be overwritten as soon as any of the following actions from the **Start** menu is performed: **Build Database**, **Update Database** or **Generate Makefile**. This means that if any changes to makefiles are to be made, the make template file should be edited instead.

DAC UDA handles compiler error reports and displays them in the **Messages** window under the **Compiler Messages** or **Linker Messages** tab, depending on the UDA performed.

3.3 Using ARM Developer Suite V1.2x Tools by Default

If you want your newly created project files to automatically inherit the proper setup for ARM Developer Suite V1.2x tools, you should also customize the default DAC project file "*dac.dcp*", which is located in the "*Program*" subfolder of the DAC folder.

4 Index

Symbols

#include 8–9
 .dcp 3
 @ 7
 @ prefix 9
 _db 4

A

Ad.mak 16
 Ad.out 16
 ARM
 Developer Suite V1.2x 1–2, 4, 8–10, 13,
 16
 ARM Developer Suite V1.2x.dmt 15
 ARM Developer Suite V1.2x.h 10
 ARM Developer Suite V1.2x.tpl 13

B

Build Project 14, 16
 Building the Database 11

C

C Source 7
 C:\Program Files\ARM\ADSv1_2 1
 Compile
 All Files 14
 File 14
 Compiler
 header directories 9
 Help 4
 libraries 9
 Configuring
 Analysis for Symbols 6

*Assembler Dialect and Header
 Directories* 10
*Compiler Dialect and Header
 Directories* 8
 DAC 2
 File Types 4
 Working Directories 3

D

-D 8
 -d 8
 DAC
 Demo Mode 1
 toolbar 14
 DAC - V4.0.055 1
 dac.dcp 16
 DAC\Technical Notes\ARM 2
 Database 4
 Build 11, 16
 directory 4
 Update 11, 16
 Defines / Control file 7

E

example
 swi 2, 9–11
 Explorer View 10

F

File Types tab 4
 Free Software Foundation 1

G

General 6

Generate Makefile 16
Generate makefile or its components 15
gmake 1
GNU 1

I
-I 8
-i 8
Integrating the Tools 13

L
license
 Demo Mode 1
 trial 1
Link All Files 14, 16
Logical View 12

M
Make Project 14, 16
Makefile
 Generator 15
 Generator Templates 15
 Template Setup 14
Maximum identifier length 7
Messages window 11

N
New Project 2

O
Options 15

P
prefix @ 7
Preinclude file 10
Project
 Adding Files to the Project 10
 Creating a New Project 2
 Root Directory 3
 Window 10, 12

R
Referential 4
Requirements 1

S
Software Metrics 4
Source 9
Special table processing 7
Start 11, 14

Start Menu Actions 13
swi.dcp 3
Symbols 4

U
UDA buttons 14
Use compiler template 13
User
 Defined Actions (UDA) Setup 13
 Help File 4
Using ARM Developer Suite V1.2x
 Tools by Default 16