COSMIC Software

Version 4

I.D.E.A.

Integrated Development Environment for Embedded Applications

User's Manual

PC/Windows 95/98/NT/2000/XP

Document Version 1.0 July 2004

Table of Contents

Preface

Chapter 1
IDEA Overview

Chapter 2
Running IDEA

Chapter 3
IDEA User Interface

Chapter 4
IDEA Options & Setup

Chapter 5
Managing an IDEA Project

Chapter 6
Building an IDEA Project

Preface

The *Idea User's Guide for HC12/HCS12* is a reference guide for programmers writing C programs for HC12/HCS12 microcontroller environments. It provides an overview of how IDEA works in conjunction with the compiler, and explains how to set up the compiler option, the assembler option, how to configure the linker and debug program options. It also describes the how to set up the programming support utilities included with the cross compiler. This manual assumes that you are familiar with your host operating system and with your specific target environment.

Contents of the IDEA User’s Guide

This IDEA User’s Guide consists of 6 chapters.

Chapter 1, “*IDEA Overview*” provides a brief description of IDEA and how it relates to other Cosmic programming tools.

Chapter 2, “*Running IDEA*” provides information for getting up and running quickly, including: starting IDEA, setting program options, getting help, managing a project, and exiting IDEA.

Chapter 3, “*IDEA User Interface*” provides detailed information on the IDEA GUI (Graphical User Interface), including: Title bar, Main menu, Tool bar, Project Window, Files Window, Reports Window (Errors, ...), the Status bar, and other information (for example MISRA) if applicable.

Chapter 4, “*IDEA Options & Setup*” provides information on configuring IDEA to meet your requirements, including: specifying a working directory, setting the default file type, customizing file window display, setting program options, program setup, selecting a ZAP debugger, and specifying error file display.

Chapter 5, “*Managing an IDEA Project*” provides detailed information on all aspects of managing an IDEA project, including: name, description, target file name, source files, directory, defines, include paths, tools, and documentation.

Chapter 6, “*Building an IDEA Project*” provides detailed information on the four phases of building a project, including: compiling, linking, making, and building.

CHAPTER

1

IDEA Overview

- Who is Cosmic Software ?
- What is IDEA ?
- Using IDEA
- The Idea GUI

This page intentionally left blank.

Who is Cosmic Software ?

Cosmic Software provides highly-optimized development tools for embedded microprocessors and microcontrollers.

The product line includes complete ANSI/ISO C language cross compilers, macro assemblers, linkers, utilities, ZAP (C and assembler source-level cross debuggers), and the IDEA. integrated development environment. These products are prepackaged and ready-to-run on PC/ Windows. Check out our web site (cosmic-software.com) for the latest target and host support. Cosmic Software also provides additionnal tools aimed at easing the software development process, or aimed at checking adherence of sources to a set of predefined rules/standards such as for example the MISRA C Checker

The ZAP debugger products are packaged to work off-the-shelf with popular debugging hardware configurations, such as low-cost evaluation boards or In-Circuit Emulators; the simulator versions of ZAP allow application code to be debugged entirely on a PC without access to target hardware, and can therefore simplify the development effort by providing for a “software debugging” phase before hardware/software integration.

The IDEA integrated development environment products provide a Windows-based graphical user interface (GUI) for building and managing coding projects. IDEA is fully integrated with all Cosmic tools, including compilers, assemblers, linkers, utilities, and ZAP debuggers.

What is IDEA ?

IDEA is an integrated development environment and editor for managing coding projects using Cosmic tools. IDEA is supplied in a version specific to the Cosmic tools you are using. For example, IDEA12 is designed for use with the Cosmic Freescale HC12/HCS12 cross compiler and ZAP debugger. In order to run IDEA you must have the matching cross compiler installed on your system; the ZAP debugger is optional.

With IDEA you can define and edit projects; compile, assemble and link C or assembler code; run a Make, run a Build with one or more build utilities; or run a ZAP debugger; all with a few simple mouse clicks in a user-friendly, graphical Windows interface.

Using IDEA

IDEA manipulates a certain number of objects to allow creating, defining and building an application.

At the top level one finds the "**Project**". A **Project** is composed by a **set of files**, a **set of tools**, and a number of properties that pertain to the whole project. Among these properties one finds: the project name, the project description, the directories used by the project, the paths used to locate header files and object files, the project defines, the options used for compiling, linking, ...

IDEA provides three different views of a **Project**:

1) the **File View** which displays the files used by the project and most of its properties

2) the **Data View** of the project which is based on a source analysis and displays the variables and functions used in the project's files,

3) the **Tool View** which groups all the tools and their properties.

The files that constitute the **Project** can be either specified as **individual files** or grouped in **File Groups.**

Each **individual file** posesses a number of properties among which one finds: file specific defines, file specific documentation, file specific options for compiling, ...

File Groups are composed by a set of files that share some common "property". That property is left to the user. It can be for example: a specific symbol definition, a specific compilation option, or some other property such as "belonging to the same bank", in the case of a segmented architecture. IDEA imposes no specifics as to what that property is. But all files in a **File Group** can be "handled" together by IDEA for some specific actions and they share a set of common properties (among which for example: specific symbol defintion, specific compilation flag, specific documentation, ...).

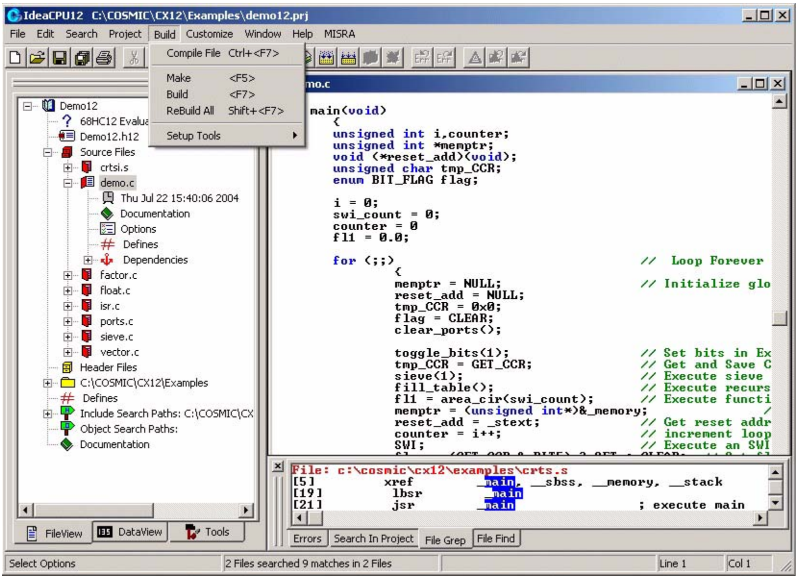
Among the standard tools manipulated by IDEA, one finds: assembler, compiler, linker, and various utilities that are used in conjunction with these. IDEA also manipulates two additionnal tools: the **MAKER** and the **BUILDER**.

The **MAKER** determines, for a specific **Project,** which files need to be recompiled/reassembled, and then does the linking of the project. It is the basic tool used to create applications.

The **BUILDER** is an additionnal tool which actually groups the execution of the **MAKER** with a certain number of pre defined utilities, or user defined utilities. It is possible to specify up to two user defined utilities to be executed before the **MAKER** process, and up to two user defined utilities to be executed after the **MAKER** process. This allows execution of some pre filters, or converters after the link is done to produce formats and files that are needed to interface with third party tools.

The Idea GUI

The IDEA GUI (graphical user interface) provides immediate access to all the tools you need to manage coding projects.



```
main(void)
    {
    unsigned int i,counter;
    unsigned int *memptr;
    void (*reset_add)(void);
    unsigned char tmp_CCR;
    enum BIT_FLAG flag;

    i = 0;
    swi_count = 0;
    counter = 0
    fl1 = 0.0;

    for (;;)                                  //  Loop Forever
            {
            memptr = NULL;                    // Initialize glo
            reset_add = NULL;
            tmp_CCR = 0x0;
            flag = CLEAR;
            clear_ports();

            toggle_bits(1);                   // Set bits in Ex
            tmp_CCR = GET_CCR;                // Get and Save C
            sieve(1);                         // Execute sieve
            fill_table();                     // Execute recurs
            fl1 = area_cir(swi_count);        // Execute functi
            memptr = (unsigned int*)&_memory;                 /
            reset_add = _stext;               // Get reset addr
            counter = i++;                    // increment loop
            SWI;                              // Execute an SWI
```

```
File: c:\cosmic\cx12\examples\crts.s
[5]        xref        _main, __sbss, __memory, __stack
[19]        lbsr        _main
[21]       jsr         _main                   ; execute main
```

Fig. 1-1: IDEA GUI

The **Project Window** at the left provides a graphical, tree-structured view of your project. Using just the Project Window, you can add or remove files from the project, set compiler options, configure build utilities, and much more. The **Project Window** includes three different panes, that give different views of your project:

1) The **File View** pane that shows the file relevant information in your project.

2) The **Data View** pane that shows information relevant to the objects your program is using, such as functions, variables, ...

3) The **Tools** pane which shows the information relevant to the tools used in the project, such as compiler, assembler, utilities, debugger, and the default options associated with these tools.

The **Files Window** at the top right allows you to open (project) files for editing and compiling. IDEA provides color-coding of Comments, Preprocessor Keywords, C Keywords, and several other coding items so that you can easily edit source code files.

The **Reports Window** at the bottom right includes a number of panes each used to report and highlight information to the user. Each pane includes a window that shows specific information. Among these windows are:

- **Errors Window**: used to show the errors of compilation. This window can be used to reach the code causing the error, and to edit it.

- **Search In Project Window**: used to report the result of a search limited to the files that are included in the project. This window can be used to reach the code where matches have occured.

- **File Grep Window**: used to report matches for a given string search in a set of files defined by the user. This window can be used to reach the code where matches have occured.

- **File Find Window**: used to report the result of a file search based on criterias provided by the user.

There may be additionnal panes in the **Reports window**, depending on what other products are installed.

All IDEA functionality is available from the **drop-down menus** under the title bar and also through a number of popup contextual menus. The most frequently used options are also available via a single click on the **Tool bar**. In addition, you can assign custom key bindings to any program option.

For a more complete description of the IDEA GUI, see **Chapter 3**, "*IDEA User Interface*".

CHAPTER
2

Running IDEA

- Starting IDEA
- Starting a New Project
- Building a Project
- Debugging a Project
- Saving and Closing a Project
- Opening the Example Project
- Specifying the Working Directory
- Setting IDEA options
- Getting Help
- Exiting IDEA

This page intentionally left blank.

Starting IDEA

From the Windows Start menu, select **Programs** > **Cosmic Tools** > **CXxx**, where *xx* stands for the Cosmic Software compiler that you are using (for example, **Programs** > **Cosmic Tools** > **CX6812** if you are using the Cosmic HC12/HCS12 compiler).

The IDEA main window appears:

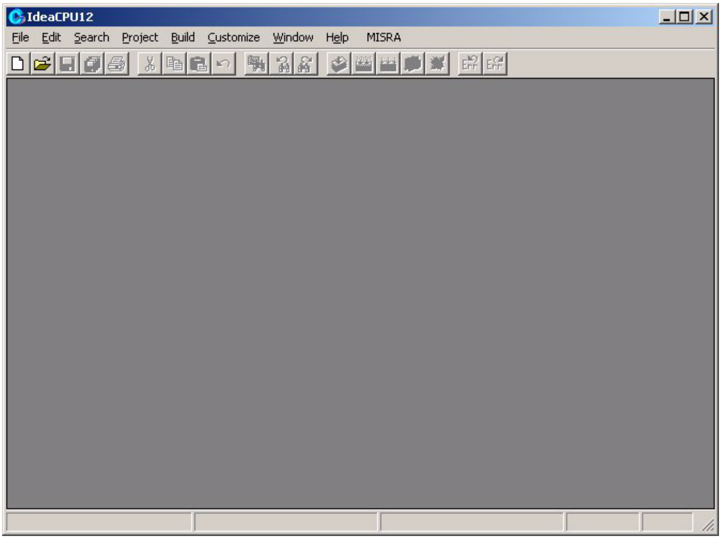
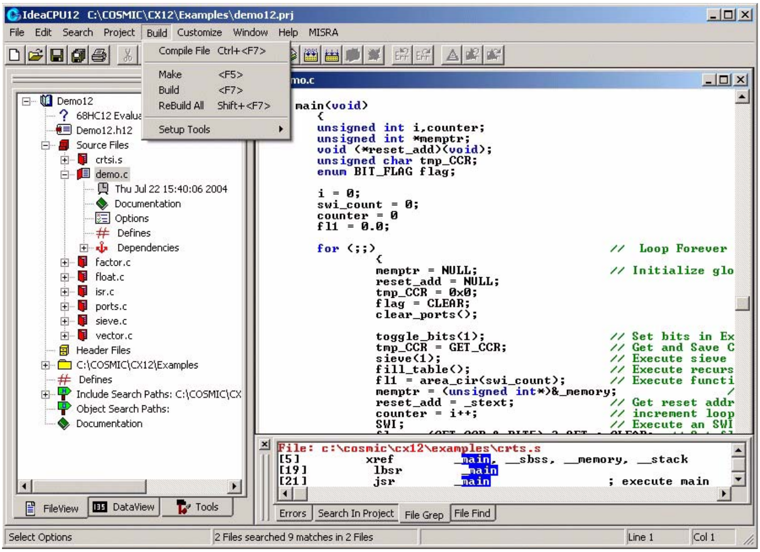


Fig. 2-1: IDEA main window

After you open a project and some files within the project, the IDEA main window appears as in the following figure.



```
main(void)
    {
    unsigned int i,counter;
    unsigned int *memptr;
    void (*reset_add)(void);
    unsigned char tmp_CCR;
    enum BIT_FLAG flag;

    i = 0;
    swi_count = 0;
    counter = 0
    fl1 = 0.0;

    for (;;)                                         //  Loop Forever
            {
            memptr = NULL;                           // Initialize glo
            reset_add = NULL;
            tmp_CCR = 0x0;
            flag = CLEAR;
            clear_ports();

            toggle_bits(1);                          // Set bits in Ex
            tmp_CCR = GET_CCR;                       // Get and Save C
            sieve(1);                                // Execute sieve
            fill_table();                            // Execute recurs
            fl1 = area_cir(swi_count);               // Execute functi
            memptr = (unsigned int*)&_memory;                        /
            reset_add = _stext;                      // Get reset addr
            counter = i++;                           // increment loop
            SWI;                                     // Execute an SWI
```

```
File: c:\cosmic\cx12\examples\crts.s
[5]         xref        _main, __sbss, __memory, __stack
[19]         lbsr        _main
[21]         jsr        _main                       ; execute main
```

Fig. 2-2: IDEA main window with Project Pane and File windows open

The IDEA main window is the principal graphical user interface (GUI) for the program.

For complete details on the components of the IDEA main window, refer to **Chapter 3**, "*IDEA User Interface*".

Starting a New Project

To start a new project, select **Project > New** from the Main menu.

The Project Window appears with a new project opened, in the **File View** pane.

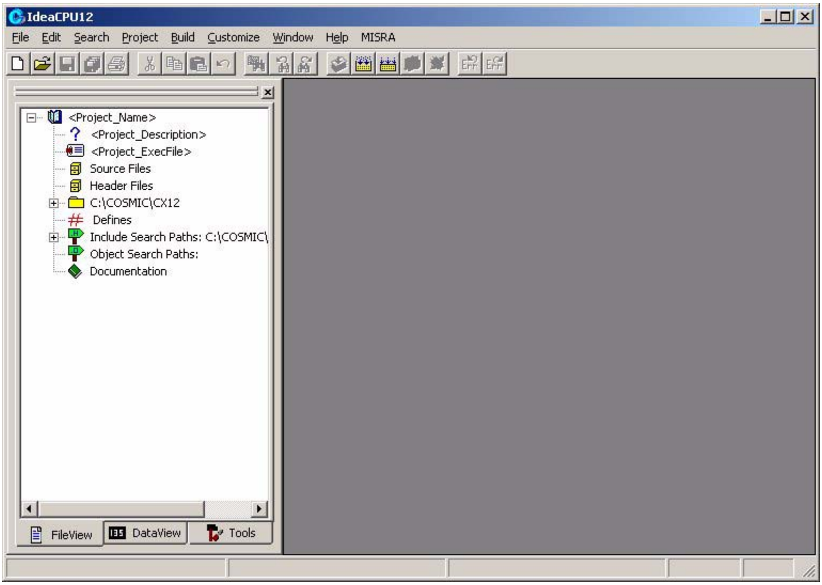


Fig. 2-3: IDEA Project Pane with new project open

The **Project** pane displays the various project components as icons in a tree-structured format, similar to Windows Explorer. Each icon in the project tree represents a project component.

Table 2-1: Project Components

📖	Project Name
?	Project Description
🗔	Project Target File Name
🗐	Project Source Files

🗐	Project Header Files
📁	Project Directory
#	Project Defines
🪧	Include Search Path
🪧	Objects SearchPath
📗	Project Documentation

A ⊞ sign next to a component icon means that sub-components are hidden below the icon. Click on the ⊞ sign or double click on the icon to display the sub-components.

A ⊟ sign next to a component icon means that the first level of sub-components below the icon is displayed. Click on the ⊟ sign or double click on the icon to hide the sub-components.

For additional details on the Project Window and its panes, refer to **Chapter 3**, “*IDEA User Interface*”.

For details on project management, refer to **Chapter 6**, “*Building an IDEA Project*”.

Building a Project

After a project is configured, you need to build the application. There are different ways to do this:

1. Right click on the **Project Name** icon in the Project window and select **Build** or **ReBuild All** from the pop-up menu.

2. Choose **Compile** (single, open file), **Build, or Rebuild All** from the **Project** pull-down menu.

3. Click on one of the following tools on the Tool bar:

	Compile tool - compiles (**.c** file) or assembles (**.s** file) an open project source file. Options are specified in the **Compiler** or **Assembler Options** dialog box.
	Build Tool - Compiles and runs the linker (and no other utilities) using the options specified for the project in the **Link Configuration** dialog box. Project source files are not checked for up-to-date status.To have the Build rebuild all files regardless of their up-to-date status, right click on the project name, select **Mark All**, and then run the Builder.
	ReBuild All Project tool - performs a Make and then runs any utilities selected in the **Builder Configuration** dialog box.

For additional details on the project building tools, refer to **Chapter 3**, "*IDEA User Interface*".

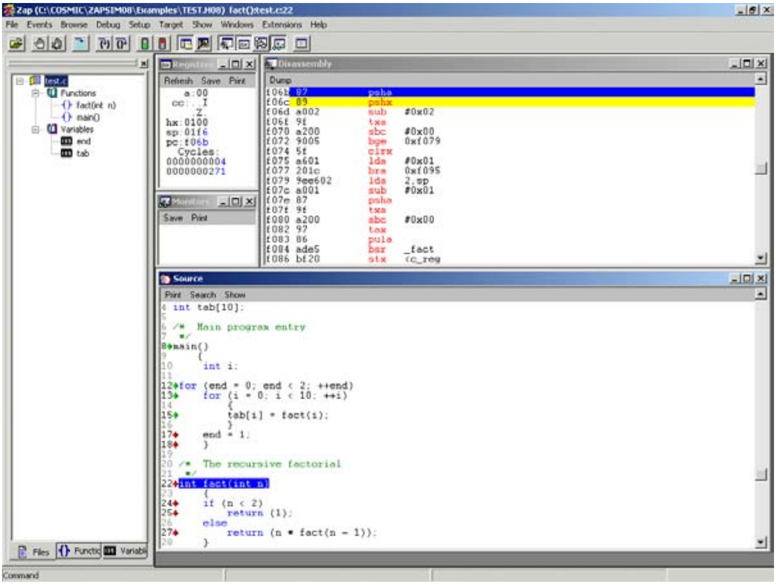
For details on building an IDEA project, refer to **Chapter 6**, "*Building an IDEA Project*".

Debugging a Project

IDEA lets you use Cosmic ZAP debuggers to debug your project.

NOTE

*Before you can use the ZAP debugger, you must first specify its location by right clicking on one of the **Debugger** tool ZAP or ✗ in the **Tools** pane on the **Project Window**; this opens a dialog box that allows you to specify the debugger location*



```
    a:00
  cc: . I
      .Z.
hx:0100
sp:01f6
pc:f06b
  Cycles:
0000000004
0000000271
```

```
f06b 87       psha
f06c 89       pshx
f06d a002     sub   #0x02
f06f 9f       txa
f070 a200     sbc   #0x00
f072 9005     bge   0xf079
f074 5f       clrx
f075 a601     lda   #0x01
f077 201c     bra   0xf095
f079 9ee602   lda   2,sp
f07c a001     sub   #0x01
f07e 87       psha
f07f 9f       txa
f080 a200     sbc   #0x00
f082 97       tax
f083 86       pula
f084 ade5     bsr   _fact
f086 bf20     stx   c_reg
```

```
int tab[10];

/*  Main program entry
 */
main()
    {
    int i;

for (end = 0; end < 2; ++end)
    for (i = 0; i < 10; ++i)
        {
        tab[i] = fact(i);
        }
    end = 1;
    }

 /*  The recursive factorial
  */
int fact(int n)
    {
    if (n < 2)
        return (1);
    else
        return (n * fact(n - 1));
    }
```

Fig. 2-4: ZAP Debugger with project target file open

When you run the ZAP Debugger from within IDEA, the ZAP Debugger automatically opens the target file for the currently loaded project.

For details on using the ZAP Debugger, refer to the ZAP User’s Guide.

Saving and Closing a Project

When you save a project, IDEA creates or updates the project file. When you close a project, IDEA checks to see if there are any unsaved changes to the project. If there are, a dialog box appears asking if you want to save the changes.

The **Project > Save** option lets you save the changes that you have made to a new or existing project. Select the **Save** option by clicking on **Save** in the **Project** menu. Alternatively, type **Alt+P+S**.

The **Project > Save As** option lets you save the changes that you have made to an existing project using a different project file name, file extension, or path. Select the **Save As** option by clicking on **Save As** in the **Project** menu. Alternatively, type **Alt+P+A**.

The **Project > Close** option lets you close the current project. Select the **Close** option by clicking on **Close** in the **Project** menu. Alternatively, type **Alt+P+O**.

Opening the Example Project

IDEA is supplied with an example project, **demo12.prj** for the Cosmic HC12/HCS12 compiler.

You can use this example project to become familiar with the principles of managing a project in IDEA.

To open the example project, select **Project->Load** in the Tool bar. In the dialog box that appears, select **demo12.prj** from the **Examples** folder. The Project window appears with the **demo12.prj** project opened.

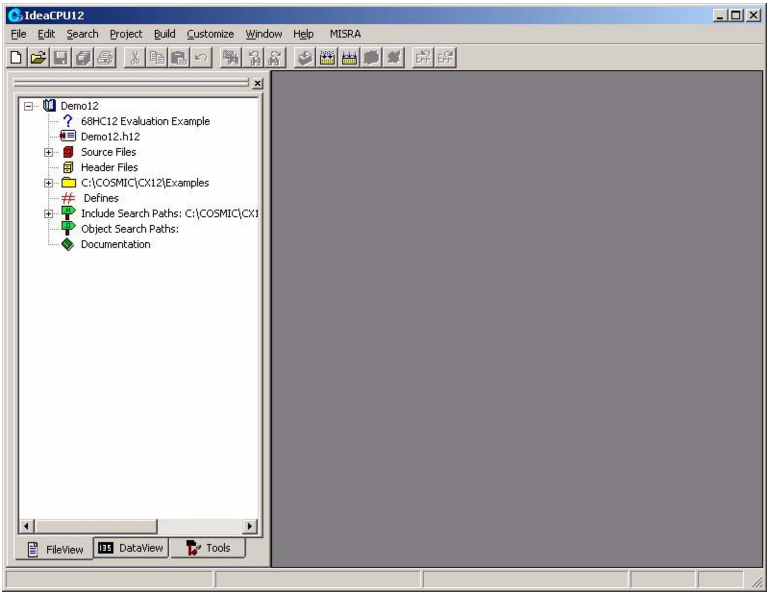


Fig. 2-5: Project pane with demo project opened

For details on managing an IDEA project, refer to **Chapter 5**, "*Managing an IDEA Project*".

Specifying the Working Directory

When you work on a project, you should store all the project files in a single folder. The **Project > Working Directory** option lets you specify the default directory for a project and for locating project files. For example, when you specify **File > Open** or **Project > Load**, IDEA initially looks in the working directory for files of the appropriate type.

Select the **Working Directory** option by clicking on **Working Directory** in the **Project** menu. Alternatively, type **Alt+P+W**.

Setting IDEA options

To set basic program options, select **Customize** from the Main menu. Alternatively, type **Alt+C**.

An option is set if a check mark appears before its name. To set an unchecked option, click on the desired option in the **Customize** drop-down menu. The next time you open the **Customize** drop-down menu, a check mark appears before that option's name.

An option is not set if no check mark appears before its name. To clear a checked option, click on the desired option in the **Customize** drop-down menu. The next time you open the **Customize** drop-down menu, no check mark appears before that option's name.

The **Customize** menu is covered in depth in "*Customize Menu*" in **Chapter 3**.

Getting Help

The **Help** drop-down menu provides help on the C language and on the C Library. You can also view IDEA version information.

Open the **Help** drop-down menu by clicking on **Help** in the Main menu. Alternatively, type **Alt+H**.

Exiting IDEA

To exit IDEA, click on **Exit** in the **File** menu. Alternatively, type **Alt+F+X**.

If you have selected **Auto Save before C/asm** in the **Customize** drop-down menu (**Alt+C+A**), all changed files are saved prior to exiting. If you have not selected **Auto Save before C/asm**, a dialog box appears in turn for each changed file and lets you select whether to save the file or not.

CHAPTER

3

IDEA User Interface

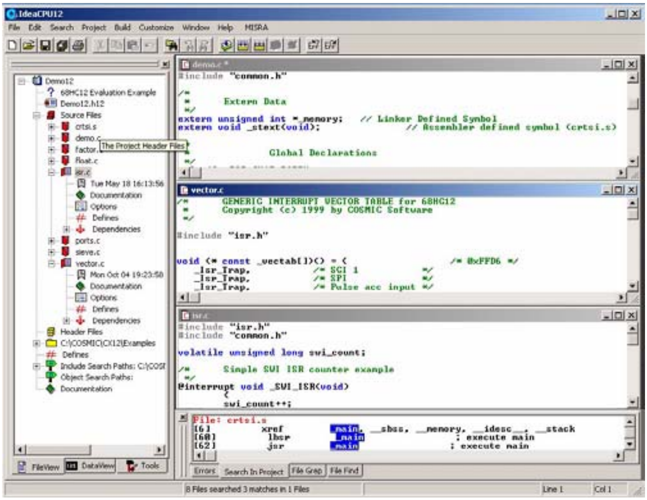
- IDEA GUI
- Title Bar
- Status Bar
- Main Menu
- Toolbar
- Project Window
- Files Window
- Errors Report

This page intentionally left blank.

IDEA GUI

Description

The following figure shows the IDEA GUI (Graphical User Interface).



```
#include "common.h"

/*
 *      Extern Data
 */
extern unsigned int *_memory;   // Linker Defined Symbol
extern void _stext(void);                  // Assembler defined symbol (crtsi.s)

 *
 *            Global Declarations
 */
```

```
/*      GENERIC INTERRUPT VECTOR TABLE for 68HC12
 *      Copyright (c) 1999 by COSMIC Software
 */

#include "isr.h"

void (* const _vectab[])() = {                    /* 0xFFD6 */
    _Isr_Trap,          /* SCI 1          */
    _Isr_Trap,          /* SPI            */
    _Isr_Trap,          /* Pulse acc input */
```

```
#include "isr.h"
#include "common.h"

volatile unsigned long swi_count;

/*      Simple SWI ISR counter example
 */
@interrupt void _SWI_ISR(void)
        {
        swi_count++;
```

```
File: crtsi.s
[6]         xref        _main, __sbss, __memory, __idesc__, __stack
[60]         lbsr        _main                         ; execute main
[62]         jsr         _main                        ; execute main
```

Fig. 3-1: IDEA GUI

The IDEA GUI consists of several components:

- Title Bar
- Status Bar
- Main Menu
- Tool Bar
- Project Window
- Files Window
- Reports Window

Each of the GUI components is described briefly in the following section and in detail later in the chapter.

GUI components

Title Bar

The Title bar gives the program name and version (for example, **IdeaCPU12**). It also provides access to the standard Windows 9x/2000/XP windows controls. For details, refer to “Title Bar” on page 32.

Status Bar

The Status bar displays helpful information after an IDEA option is executed. For details on the Status bar, refer to “Status Bar” on page 33.

Main Menu

The Main menu provides access to all IDEA functionality via its options: **File**, **Edit**, **Search**, **Project**, **Customize**, **Window**, and **Help**.

Tool Bar

The Tool bar provides one-click access to commonly used functions via groups of tools: file management, editing, searching, project building, error management, ... For details, refer to “Toolbar” on page 42.

Project Window

The Project Window displays the currently open project at the left side of the window area. The Project Window can show one of three different panes at a time:

- **File View**: Containing all of your project’s files, group of files, project paths and dependencies.

- **Data View**: Containing information about the objects used in your application such as functions, variables; this information is only available if “Project Analysis” is selected (see the **Customize** Menu).

- **Tools**: IDEA’s classic project tool tree. Set all of the compile options, linker setup, and converter mechanisms from here.

Reports Window

The **Reports** Window displays information that is the result of a request made by the user. The **Reports** Window can show one of several different panes at a time

- **Errors**: displays the errors found from the most recent compile, link, make, or build at the bottom right of the window area.
- **Search in Project**: displays search results from a search restricted to the Project's files.(See the Search Menu)
- **File Grep**: displays results from a search for a specified string in a specified set of files. (See the Search Menu)
- **File Search**: displays the results a file search command. (See the Search Menu)
- There may be additionnal panes in the **Reports** Window, depending on extensions to IDEA, and on associated installed projects.

You can navigate the IDEA GUI using any combination of mouse operations, predefined keyboard shortcuts, and custom key bindings that you create.

Mouse

You can use the mouse for most tasks in IDEA including selecting options from the Main menu and tools from the Tool bar, and working with files. All functionality can be accessed via left, right, and double clicks. Keyboard shortcuts and custom key bindings provide a convenient alternative in many repetitive situations.

For navigating the Project window and managing a project, the mouse is a much more convenient means of navigation than the keyboard.

Keyboard shortcuts

All Main menu options, Tool bar tools, and many other IDEA options can be quickly accessed from the keyboard using IDEA's predefined keyboard shortcuts. Most of the keyboard shortcuts consist of the **<Alt>** key followed by one or more letter keys.

For example, to print a file, press **<Alt>** to highlight the Main menu, **"F"** to open the File drop-down menu, and **"P"** to select the **Print** option. This key combination is shown in this manual as **Alt+F+P**.

For another example, if you have selected the **Files** icon in the Project window and right clicked to open the floating menu, you can then press **"A"** to add a file to the project files list.

In addition to the keyboard shortcuts that use the **<Alt>** key, there are a few "standard" DOS keyboard shortcuts that appear on IDEA's menus. For example, you can type **Ctrl+P** to print a file.

Custom key bindings

You can also create your own keyboard shortcuts for nearly all of the menu commands in IDEA by selecting the **Key Binding** option from the **Customize** menu.

Select the **Key Binding** option by clicking on **Key Binding** in the **Customize** menu.

The **Key Binding** dialog box appears.

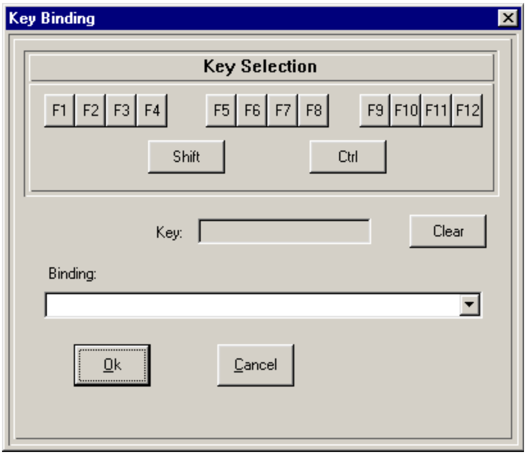


Fig. 3-2: Key Binding dialog box

You can use any combination of the **Ctrl**, **Shift**, and **Function** keys to create a new shortcut. Click on your choice(s) in the **Key Selection** field. As you click, the shortcut definition is built up in the **Key** field. To remove **Ctrl** or **Shift** from the key definition, click on either of them again. To change the **Function** key in the key definition, click on your new choice.

After you specify a key sequence, you must bind it to an action. If the key sequence you have specified is not already in use, the **Binding** field will be blank. Click on the down arrow to the right of the field and select the action that you want to associate with the key sequence from the action list.

If the key sequence you have specified is already in use, the **Binding** field will show the action with which the key sequence is associated. You can either select a new key sequence for the action or click on the **Clear** button to clear the current action from the **Binding** field.

After you create a keyboard shortcut for a menu option, the shortcut appears after the option name in the drop-down menu.

Title Bar

The Title bar gives the program name and version and provides access to the standard Windows 98/2000/XP windows controls.

Program Name and Version

The program name is followed by the version. For example, if you are using the Cosmic Freescale **HC12/HCS12** Cross Compiler, the Title bar shows “**IdeaCPU12**”.

Status Bar

The **Status** bar displays helpful information after an IDEA option is executed. It also displays error information after a compile, link, make, or build. The Status bar is divided in five sub sets:

1) The two right most subset of the status bar are used to display the current line and current column (i.e. the cursor position) in the current edit file (in the Files Window).

2) The second subset (starting from the left) is used to display service messages, such as error messages from a compilation or from an associated tool.

3) The remaining two subsets are used to provide the user with information about the status of Idea in different circumstances, for example it can be used to indicate the name of the file that is compiled.

Main Menu

The IDEA Main menu provides access to all program functionality via drop-down menus: **File**, **Edit, Search, Project**, **Build**, **Customize**, **Window**, and **Help**. There may be additionnal menu entries when associated products are installed such as **Cosmic's MISRA Checker** for example.

File Menu

The **File** drop-down menu provides options to open, read, save, compile, or print new or existing files. You can also add or remove a file from a project. You can use these options for several different types of file.

Open the **File** drop-down menu by clicking on **File** in the Main menu. Alternatively, type **Alt+F**.

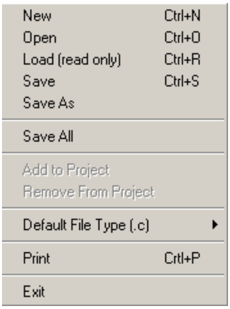


Fig. 3-3: File menu

Edit Menu

The **Edit** drop-down menu provides options to edit the contents of the currently active file. If no file is open and active, the **Edit** menu options are unavailable.

Open the **Edit** drop-down menu by clicking on **Edit** in the Main menu. Alternatively, type **Alt+E**.

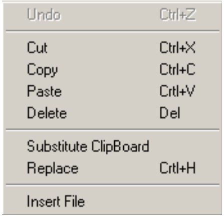


Fig. 3-4: Edit Menu

Search Menu

The **Search Menu** allows you to search among your project’s files in order to locate certain strings of code. It include file greps to allow you to search through multiple files at a time.

Open the **Search Menu** by clicking on **Search** in the **Main Menu.** Alternatively, type **Alt+S.**

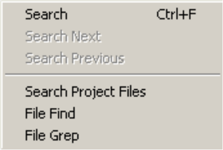
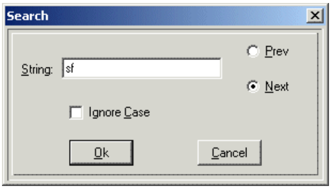


Fig. 3-5: Search Menu



String: sf

Fig. 3-6: Simple File Search

IDEA has the capabilities to do multiple types of searches within your project. These searches can consist of string searches within a project, string searches within a path, and file searches within a path. All three of these options are found in the **Search** drop-down menu.

Search Project Files

The **Search Project File** option in the **Search** menu allows you to search for a particular string in your project. Here, you can enter a string case to search and the results will be listed in your error window.

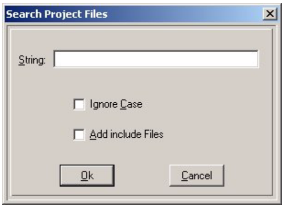
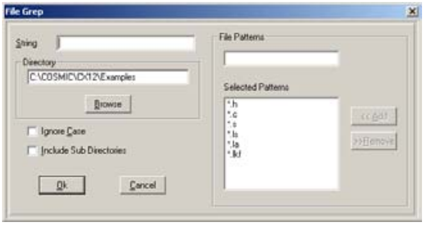


Fig. 3-7: Search Project Window

File Grep

The **File Grep** option allows you to search multiple files in a specified path for a certain string. For example, you can search a list of files in a directory for the *_main* symbol.



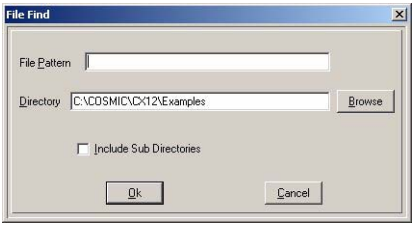
Directory: C:\COSMIC\CX12\Examples

Selected Patterns: *.h, *.c, *.s, *.ls, *.la, *.lkf

Fig. 3-8: File Grep Window

File Find

The **File Find** option allows you to search for a certain file name amidst a certain path.



Directory: C:\COSMIC\CX12\Examples

Fig. 3-9: File Find Window

Project Menu

The **Project** drop-down menu provides options to load an existing project or create a new project, and save and/or close an open project. In addition, you can add a file to a project, view its dependencies, and compile the file. You can also initiate a make command for a project and do a project build.

Open the **Project** drop-down menu by clicking on **Project** in the Main menu. Alternatively, type **Alt+P**.

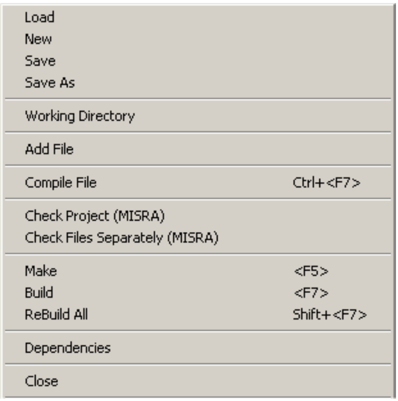


Fig. 3-10: Project menu

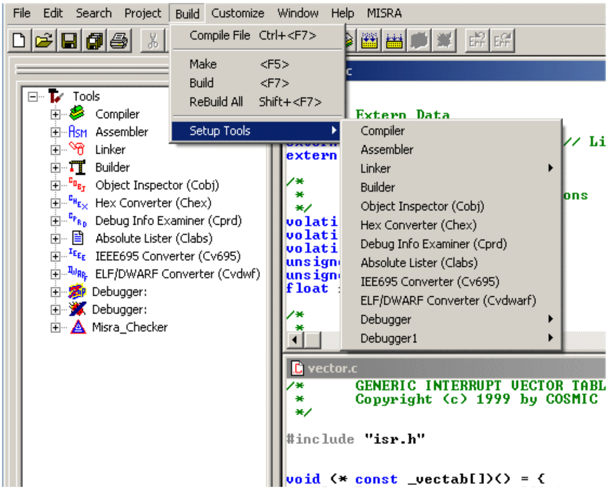
NOTE

The MISRA related options are only valid if you have installed the Cosmic's MISRA Checker in conjunction with IDEA.

Build Menu

The **Build Menu** opens a quick drop-down set of simple controls for your project. It includes the basic **Compile**, **Build**, **Re-Build** options as well as a **Tool Sub Menu,** that allows to setup the options of the various tools**.**

Open the **Build Menu** by clicking on **Build** in the Main menu. Alternatively, type **Alt+B**.



```
/*      GENERIC INTERRUPT VECTOR TABL
 *      Copyright (c) 1999 by COSMIC
 */

#include "isr.h"

void (* const _vectab[])() = {
```

Fig. 3-11: Build Menu with Tool Sub Menu

Customize Menu

The **Customize Menu** lets you set basic program options.

Open the **Customize Menu** by clicking on **Customize** in the Main menu. Alternatively, type **Alt+C**.

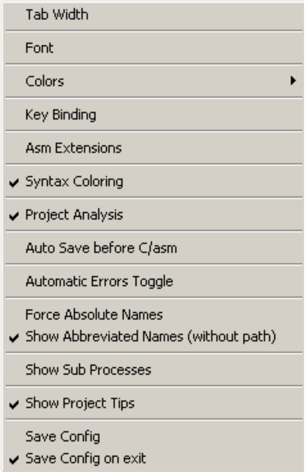
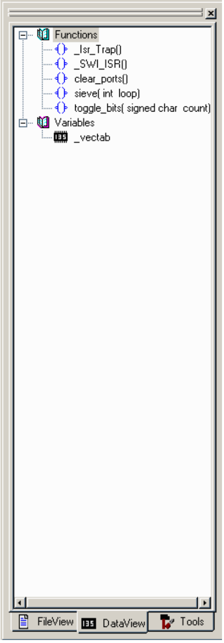


Fig. 3-12: Customize menu

Customize contains the following features you can include in your IDEA setting and your project.

- Key Bindings and Tab Widths
- ASM Extensions (don’t want to change your **.asm** to **.s** ? You don’t have to!)
- Fonts, Colors, and Syntax Coloring options
- Auto Saves, Configuration Saves
- Project Analysis (NEW!): With Project Analysis, IDEA will graphically display function and variable information in the **Dataview** pane of the **Project Window.**



Functions
_Isr_Trap()
_SWI_ISR()
clear_ports()
sieve(int loop)
toggle_bits(signed char count)
Variables
_vectab

Fig. 3-13: Project Window, Data View pane

Window menu

The **Window** drop-down menu lets you arrange all open files in IDEA. In addition, you can open the Error pane in the Reports Window.

Open the **Window** drop-down menu by clicking on **Window** in the Main menu. Alternatively, type **Alt+W**.

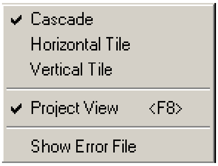


Fig. 3-14: Window menu

Help menu

The **Help** drop-down menu provides on-line help on the C language and the C Library. You can also view IDEA version information.

Open the **Help** drop-down menu by clicking on **Help** in the Main menu. Alternatively, type **Alt+H**.

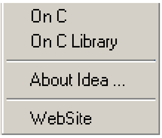


Fig. 3-15: Help menu

Toolbar

The Toolbar provides one-click access to commonly used IDEA functions via seven groups of tools: file management, project management, editing, searching, project building, error management, and system windows.

File Management

New File

New File lets you create a new file in any one of several different file types. Selecting **New File** is equivalent to selecting the **New** option in the **File** menu.

Open File

Open File lets you open and edit an existing file in any one of several different file types. Selecting **Open File** is equivalent to selecting the **Open** option in the **File** menu.

Save File

Save File lets you save the changes that you have made to a new or existing file. You can determine if a file has edits that need to be saved by looking at the window title bar. If "**(modified)**" appears after the file name, changes have been made to the file and not yet saved. Selecting **Save File** is equivalent to selecting the **Save** option in the **File** menu.

Save All

Save All lets you save your existing project in one easy step. Every file currently open and/or included in your **Project** will be saved when you click this tool. Selecting **Save All** is equivalent to selecting the **Save All** option in the **File** menu.

Print File

Print File lets you print the active file. Selecting the **Print File** tool is equivalent to selecting the **Print** option in the **File** menu.

File Editing Tools

Cut

Cut allows you to delete a selected piece of text from your current file and place its contents into the clipboard. Selecting the **Cut** tool is equivalent to selecting **Cut** from the **Edit** menu.

Copy

Copy allows you to copy a selected piece of text from your current file and place its contents into the clipboard. Selecting **Copy** is equivalent to selecting **Copy** from the **Edit** menu.

Paste

Paste allows you to place the contents of the clipboard to your current cursor location. Selecting **Paste** is equivalent to selecting **Paste** from the **Edit** menu.

Undo

Undo undoes your last action from the file you are editing. Selecting **Undo** is equivalent to selecting **Undo** from the **Edit** menu.

Search Functions

Search

Search brings up a dialog box exactly like the **Search** option located under the **Search** menu. Enter the string you wish to search for and the first instance will be located.

Search Previous

Search Previous appears after you make one successful search. This tool will scan back over the current search instances. The **Search Previous** tool does nothing if you are on the first instance of your search string. The **Search Previous** tool is indentical to the **Search Previous** selection under the **Search** menu.

Search Next

Search Next appears after you make one successful search. This tool will scan to the next search instance. The **Search Next** tool does nothing if you are on the last instance of your search string. The **Search Next** tool is indentical to the **Search Next** selection under the **Search** menu.

Project Building

Compile

Compile lets you compile (**.c** file) or assemble (**.s** file) the currently active source file. Selecting the **Compile** tool is equivalent to selecting the **Compile** option in the **Build** menu or pressing **Ctrl + F7**.

Build

Build builds all uncompiled files, re-links your project, and produces any S-records or other debug files you have specified. Clicking **Build** without modifying any source files will simply re-link your project. Selecting **Build** is equivalent to selecting the **Make** option in the **Build** menu or pressing **Ctrl + F7**.

Rebuild

Rebuild flags every source file and output file for recreation and rebuilds your entire project from scratch. All objects, listings, absolute listing, executeables, et al are rebuilt. Selecting the **Reuild** tool is equivalent to selecting the **Rebuild** option in the **Build** menu or pressing **Shift + F7**.

Debuggers and

Debuggers let you run one or two debugger(s) with the project target file loaded. Selecting a **Debugger** tool is equivalent to double clicking on the **Debugger** tool in the **Tools Pane** in the **Project Window,** or selecting the debugger from the **Setup Tools** menu in the **Build** menu.

NOTE

You must first specify the location of a debugger by right clicking a debugger icon in the ***Tools View Pane*** in the ***Project Window****. This opens a dialog that allows you to specify the debugger for the project.*

Error Management

If any errors are encountered during a compile, link, make, or build (and if the **Automatic Errors Toggle** option on the **Options** sub-menu is checked), IDEA will display the errors in the **Errors** pane of the **Reports Window**. In addition the Error buttons will be available for use.

- **Show Previous Error**
- **Show Next Error**

You can use the Error management buttons to highlight any error in the **Errors** report. When you select an error by double clicking the line that reports that error, IDEA will display the source file where the error is located in the **Files Window**.

Show Previous Error

Show Previous Error moves to the previous error in the **Errors** report.

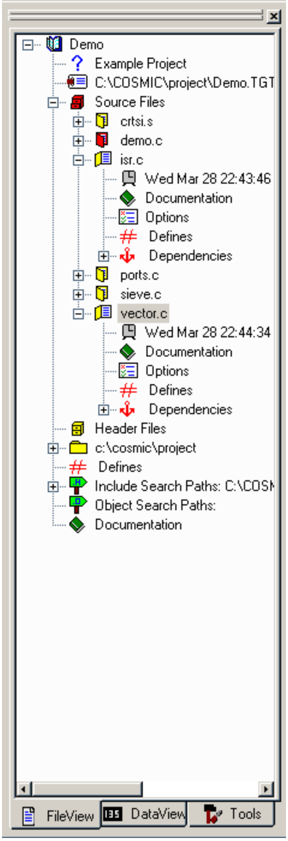
Show Next Error

Show Next Error moves to the previous error in the **Errors** window.

Project Window

Description

The **Project** Window is a resizeable fixture on the left side of the Main Window. The Project Window has 3 panes: **File View**, **Data View**, and **Tools**.



```
⊟ Demo
    ? Example Project
    C:\COSMIC\project\Demo.TGT
  ⊟ Source Files
      ⊞ crtsi.s
      ⊞ demo.c
      ⊟ isr.c
          Wed Mar 28 22:43:46
          Documentation
          Options
          # Defines
        ⊞ Dependencies
      ⊞ ports.c
      ⊞ sieve.c
      ⊟ vector.c
          Wed Mar 28 22:44:34
          Documentation
          Options
          # Defines
        ⊞ Dependencies
    Header Files
  ⊞ c:\cosmic\project
    # Defines
  ⊞ Include Search Paths: C:\COSN
    Object Search Paths:
    Documentation
```

Fig. 3-16: IDEA Project Window, File View Pane

Navigation

The **Project** pane window displays the various project components as icons in a tree-structured format, similar to Windows Explorer. Each icon in the project tree represents a project component.

A ⊞ sign next to an icon means that sub-components are hidden below the icon. Click on the ⊞ sign or double-click on the icon to display the sub-components.

A ⊟ sign next to an icon means that the first level of sub-components below the icon are displayed. Click on the ⊟ sign or double-click on the icon to hide the subcomponents.

Customization

When you open a project, the Project pane is always displayed to the left side of the IDEA window area. You can display or hide the Project Window. You can also adjust the width of the Project Window (up to the full width of the main window area) by clicking and dragging on its right margin.

After you close a project, the Project Window will automatically disappear.

Project Window: File View Pane

The File View pane shows the current setup for the file related components of your project.

- **Project Name**
- **Project Description**
- **Project Target File Name**
- **Project Source Files**
- **Project Header Files**
- **Project Directory**

- **Project Defines**
- **Project Include Search Paths**
- **Project Objects Search Paths**
- **Project Documentation**

Each of these components is described briefly in the following sections. For additional details on project components, refer to **Chapter 5**, "*Managing an IDEA Project*".

Project Name

The **Project Name** component lets you specify a name for the project (for example, **demo12**). It also represents the parent component for all project sub-components.

To specify a name for the project, click on the **Project Name** icon .

A text cursor appears to the right of the icon. Click on the text cursor to open a text box and enter a project name. Right click on the **Project Name** icon to view a menu of project commands.

Project Description

The **Project Description** component lets you specify a short description for the project.

To specify a description for the project, click on the **Project Description** icon ?.

A text cursor appears to the right of the icon. Click on the text cursor to open a text box and enter a short project description.

Project Target File Name

The **Project Target File Name** component lets you specify a target file name for the project (for example, **demo12.h12)**. This name is used as the root name for the linked executable.

To specify a project target file name, click on the **Project Target File Name** icon .

A text cursor appears to the right of the icon. Click on the text cursor to open a text box and enter a target file name. Be sure to include the target file name extension; this extension depends on the target processor and is the same one as used by the compiler/linker. Right click on the **Project Target File Name** icon to view a menu containing target file commands.

Project Source Files

The **Project Source Files** component lets you specify the C and Assembly language source files to be included in the project.

Files can either be included individually or they can be included in **File Groups**. **File Groups** are a set of file that same some common property, which can for example be a special set of flags, or it could represent for example the various functions that need be linked in a specific way, or that must grouped in a specific memory segment, ...

It is left to the user to decide whether his project can take advantage of **File Groups** and how his own files are gouped.

Right click on the **Project Source Files** icon to view a menu containing source file management commands.

Project Header Files

The **Project Headers Files** component lets you specify the header source files that are used in your project. This is only for reference and to avoid mixing source files which are compiled and header files which are not compiled

Project Directory

The **Project Directory** component is used to set the working directory for the project. This is typically where the source code files for the project are located.

Right-click on the **Project Directory** icon to open the **Path Editor** dialog box and set the path to the source files.

Project Defines

The **Project Defines** component lets you specify #define options for the project.

Right-click on the **Project Defines** icon # to open the **#defines** dialog box and specify up to twenty user-defined preprocessor symbols.

Project Include Search Paths

The **Project Include Search Paths** component lets you specify include paths for the compiler (**-i** > option).

Right-click on the **Project Include Search Paths** icon to open the **Include Path Editor** and specify up to twenty include paths for the project.

Project Object Search Paths

The **Project Object Search Paths** component lets you specify object paths for the project.

These paths will be used by the **MAKE** utlity included in **IDEA** to check whether a file needs to be recompiled or not.

This allows a user to set his object files in a different directory than his source files.

Right-click on the **Project Object Search Paths** icon to open the **Object Path Editor** and specify up to twenty Object paths for the project.

Project Documentation

The **Project Documentation** component shows all documents that are associated with the project.

Right click on the **Project Documentation** icon and select **Add Doc** to associate a documentation file with the project.

NOTE

*If when loading a project **IDEA** finds a documentation file specification whose name is "**startup.tip**", this file will automatically be displayed in the **Files Window**.*

Project Window: Data View Pane

The **Data View** pane of the Project Window displays the project analysis results that IDEA generates for your project.

In order for this information to be displayed, make sure the **Project Analysis** option in the **Customize** drop-down menu is checked.

The **Data View** displays the functions and variables associated with your project.

Double-clicking a member in either the function or variable section will open up the proper source file containing that object.

Right clicking a member will display a popup menu with actions related to the type of object selected by the click.

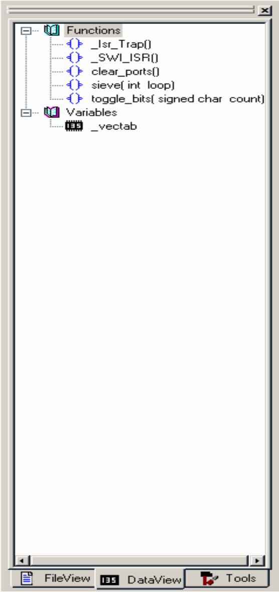


Fig. 3-17: Project Window: Data View

Project Window: Tool View

The **Tool View** section of the **Project** pane is a representation of IDEA’s classic Tools tree. **Tool View** lets you set project default options for:

- Compiler Options
- Assembler Options
- Linker Options
- Builder
- Debugger

The **Builder** component lets you configure build utilities for the project, including:

- Object Inspector (cobj)
- Hex Converter utility (chex)
- Debug Info Examiner utility (cprd)
- Absolute Lister utility (clabs)
- IEEE695 Converter utility (cv695)
- ELF/Dwarf Converter utility (cvdwarf)

Double click on the **Project Tools** icon to display the project tools. By default, the tool tree is expanded.

Compiler tool

The **Project Compiler** tool lets you set the default compiler options that are used to compile all files for the project.

Right click on the **Project Compiler** icon to open the **Compiler Options** dialog box.

Assembler tool

The **Project Assembler** tool lets you set the default assembler options that are used to compile all assembly language files for the project.

Right click on the **Project Assembler** icon ASM to open the **Assembler Options** dialog box.

Linker tool

The **Project Linker** tool lets you set options for the project linker. You can also specify a linker command file and edit the file.

Right click on the **Project Linker** icon to view a menu containing linker commands.

Builder tool

The **Project Builder** tool lets you specify utilities for building the project.

Right click on the **Project Builder** icon to open the **Builder Configuration** dialog box.

Object Inspector (*cobj*) utility

The *cobj* utility lets you inspect relocatable object files or executable output by the assembler or linker. The *cobj* utility can be used to check the size and configuration of relocatable object files or to output information from their symbol tables.

Right click on the **Object Inspector** icon COBJ and select **Options** to open the **Options** dialog box.

Hex Converter (*chex*) utility

The *chex* utility translates executable images produced by *clnk* to one of several hexadecimal interchange formats.

Right click on the **Hex Converter** icon CHEX and select **Options** to open the **CHEX Configuration** dialog box.

Debug Info Examiner (*cprd*) utility

The *cprd* utility extracts and prints information about functions and data objects from an object module or executable image that has been compiled with the **+debug** option.

Right click on the **Debug Info Examiner** icon and select **Options** to open the **CPRD Configuration** dialog box.

Absolute Lister (*clabs*) utility

The *clabs* utility processes relocatable C and Assembly listing files with the associated executable file to produce absolute listings with updated code and address values.

Right click on the **Absolute Lister** icon and select **Options** to open the **CLABS Configuration** dialog box.

IEEE695 Converter utility

The *cv695* utility converts a file produced by the linker into IEEE695 format.

Right click on the **IEEE695 Converter** icon and select **Options** to open the **CV695 Configuration** dialog box.

ELF/DWARF Converter utility

The *cvdwarf* utility converts a file produced by the linker into ELF/ Dwarf format.

Right click on the **DWARF Converter** icon and select **Options** to open the **DWARF Configuration** dialog box.

Debugger tool

The **Debugger** tools let you specify one or two debuggers for the project.

Right-click on a **Project Debugger** icon or to open a dialog box that allows you to specify the location of a debugger.

After you select a debugger, the path and filename appear after the **Project Debugger** icon.

Managing a project

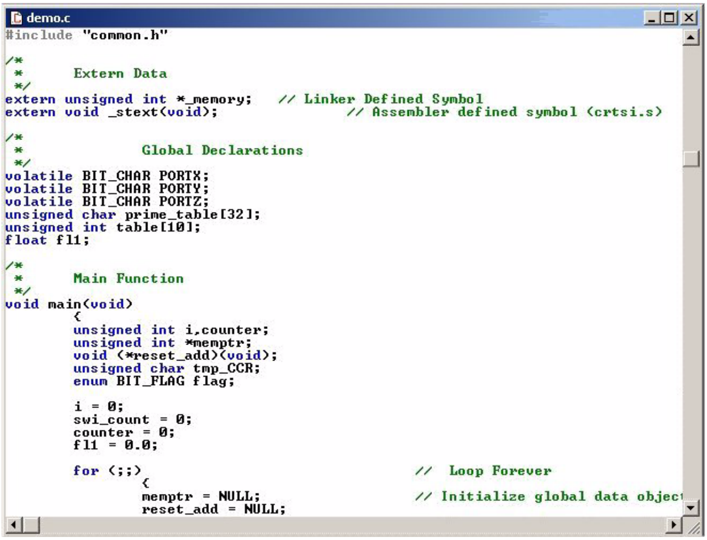
You manage an IDEA project by selecting the various components and tools in the Project Pane and entering information into the associated dialog boxes and fields.

For complete details on managing a project, refer to **Chapter 5**, "*Managing an IDEA Project*".

Files Window

Description

The **Files** window display(s) one or more open file.



```
#include "common.h"

/*
 *      Extern Data
 */
extern unsigned int *_memory;   // Linker Defined Symbol
extern void _stext(void);               // Assembler defined symbol (crtsi.s)

/*
 *              Global Declarations
 */
volatile BIT_CHAR PORTX;
volatile BIT_CHAR PORTY;
volatile BIT_CHAR PORTZ;
unsigned char prime_table[32];
unsigned int table[10];
float fl1;

/*
 *      Main Function
 */
void main(void)
        {
        unsigned int i,counter;
        unsigned int *memptr;
        void (*reset_add)(void);
        unsigned char tmp_CCR;
        enum BIT_FLAG flag;

        i = 0;
        swi_count = 0;
        counter = 0;
        fl1 = 0.0;

        for (;;)                                //  Loop Forever
                {
                memptr = NULL;                  // Initialize global data objec
                reset_add = NULL;
```

Fig. 3-18: IDEA Files Window

When you first run IDEA, the Files Window area is blank. To open a file window, select **New**, **Open**, or **Load (read only)** from the **File** menu or click on the **New File**, **Open File**, buttons in the Tool bar.

If a project is open, double click on the **Files** component in the Project Window to display a list of files included in the project. Right click on a file name to display a pop-up menu from which you can **Open** or **Load (read only)** the file.

Navigation

Each open file is displayed in its own file window. You can make an open file active by clicking anywhere in the file window if it is visible.

You can also make an open file active by selecting it from the list at the bottom of the Window drop-down menu.

The Title bar in a file window functions in the same manner as the IDEA Title bar. Refer to "Title Bar" on page 32 for a description of this functionality.

The icon at the left side of the Title bar differs according to the type of file that is open in that window. The file type icons are listed below.

.C	C code source file (**.c**)
.S	Assembler source file (**.s**)
.H	Header file (**.h**)
LKF	Linker command file (**.lkf**)
?	"Other" file type (**.ls**, **.la**, **.lkf**, etc.)

Customization

When you open a file, it is displayed in the window area to the right of the project window (if a project is open). If no project is open, the file window occupies the entire window area.

As you open more files, they are displayed in one of three ways, depending on the option selected in the Window drop-down menu. The file window display options are:

- **Cascade**
- **Horizontal Tile**
- **Vertical Tile**

Project file types

A project may be composed of several different file types:

- **C source file (.c)**
- **Assembler file (.s)**
- **Header file (.h)**
- **Link file (.lkf)**
- **Listing file (.ls)**
- **Absolute listing file (.la)**

In addition to the above files, several other file types may be created by IDEA:

- **Project file (.prj)** - A project file contains options for a project and is similar in structure to a Windows **.ini** file.
- **Project target file** - the extension depends on the target processor.
- **Project map file (.map)**
- **Object file (.o)**
- **Errors file (.err)** - The file **idea.err** contains the last errors that resulted from a compile, link, make, or build.

Managing Project Files

Each IDEA project is composed of a series of source files. icon ▤ in the **File View** section of the **Project** pane.

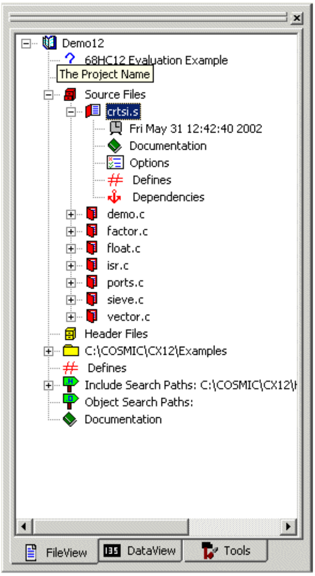


Fig. 3-19: IDEA Project Pane

Right click on the **Source Files** icon to view a menu containing source file management commands.

For complete details on managing source files, refer to **Chapter 5**, "*Managing an IDEA Project*".

Adding source files to the project

You can add source files to the project using the **Add File** command.

- To add a source file using the **Add File** command, right click on the **Project Source Files** icon, select **Add File**, and select the file to add from the **Add File** dialog box.

You can select more than one source file at a time using standard Windows conventions for selecting and files.

Adding File Groups to the project

You can add **File Groups** to the project using the **Add Group** command.

- To add a **File Group**, right click on the **Project Source Files** icon, select **Add Group**, you will then have to enter the File Group name which should be unique. Once a **Group** is created you can right click the Group icon to obtain the **File Group** specific popup menu.

Working with source files

Each source file included in the project, whether it is an individual file or it is part of a **File Group** is listed next to a **Source File icon**. The **Source File** icon lets you view the source file and its attributes.

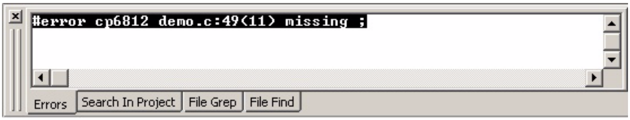
Right click on the **Source File** icon to view a menu containing source file commands.

For complete details on working with individual source files, refer to **Chapter 5**, "*Managing an IDEA Project*".

Errors Report

Description

If the **Automatic Errors Toggle** option on the **Options** sub-menu is checked, the **Errors Report** is opened automatically if any errors are found during a compile, link, make, or build.



```
#error cp6812 demo.c:49(11) missing ;
```

Fig. 3-20: Error Report

The **Errors Report** display can be toggled on or off using the **Show Error File** option on the **Errors** sub-menu.

Navigation

If any errors are encountered during a compilation, IDEA opens an **Errors** window and lists the errors. In addition, two **Error** buttons appear in the Tool Bar.

Err	Go to previous error in list
Err	Go to next error in list

You can use the error list navigation buttons to highlight any error in the **Errors** report. When you highlight an error in the **Errors** report, the cursor in the Files Window moves to the point where the error occurred.

CHAPTER
4

IDEA Options & Setup

This page intentionally left blank.

Overview

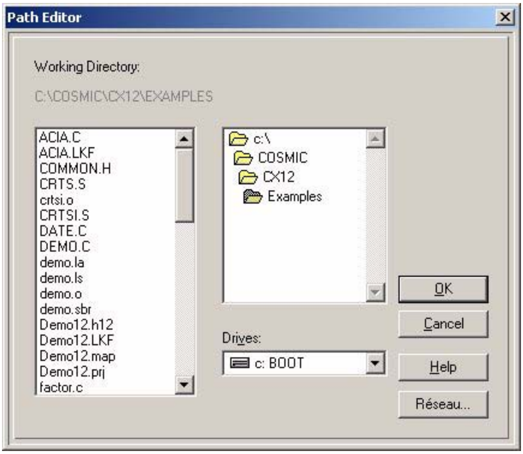
This chapter explains how to set options and defaults for IDEA. You may want to change some of these prior to working on a project.

Working Directory

You can specify a working directory for your projects and for locating files. IDEA uses the working directory when you select options such as **File > Open**, **Project > Load**, etc... .

Specify the working directory by clicking on **Working Directory** in the **Customize** menu. Alternatively, type **Alt+S+W**.

The **Path Editor** dialog box appears.



Path Editor

Working Directory:

C:\COSMIC\CX12\EXAMPLES

ACIA.C
ACIA.LKF
COMMON.H
CRTS.S
crtsi.o
CRTSI.S
DATE.C
DEMO.C
demo.la
demo.ls
demo.o
demo.sbr
Demo12.h12
Demo12.LKF
Demo12.map
Demo12.prj
factor.c

c:\
COSMIC
CX12
Examples

Drives: c: BOOT

Fig. 4-1: Path Editor dialog box

You can specify a working directory by selecting the appropriate drive and double clicking on the appropriate folders. The contents of the currently selected folder appear in the list window at the left.

Default file type

You can specify a default file type for many file operations. IDEA uses the default file type when you select options such as **File > New**, **File > Open**, etc.

The current default file type is shown after the **Default File Type** option on the **File** drop-down menu (for example, **Default File Type (.c)**).

To change the default file type, select **Default File Type** in the **File** menu. Alternatively, type **Alt+F+D**.

A drop-down list appears with the available file types and associated file extensions:

- **C source (.c)**
- **Assembler (.s)**
- **Header file (.h)**
- **Link file (.lkf)**
- **Listing file (.ls)**
- **Absolute Listing file (.la)**
- **Undefined (.*, any file extension)**

Highlight the desired default file type and click on it.

Files Window display

You can have multiple file windows open while managing a project. IDEA lets you arrange the file windows in three ways.

Cascade

The **Cascade** option arranges all open file windows in a cascade.

Select the **Cascade** option by clicking on **Cascade** in the **Window** menu. Alternatively, type **Alt+W+C**.

After you select this option, a check mark appears before the option name.

Horizontal Tile

The **Horizontal Tile** option arranges all open file windows in a horizontal tiling.

Select the **Horizontal Tile** option by clicking on **Horizontal Tile** in the **Window** menu. Alternatively, type **Alt+W+H**.

After you select this option, a check mark appears before the option name.

Vertical Tile

The **Vertical Tile** option arranges all open file windows in a vertical tiling.

Select the **Vertical Tile** option by clicking on **Vertical Tile** in the **Window** menu. Alternatively, type **Alt+W+V**.

After you select this option, a check mark appears before the option name.

Program Options

You can set basic program options using the **Cutomize** drop-down menu.

Open the **Cutomize** menu by clicking on **Cutomize** in the Main menu. Alternatively, type **Alt+C**.

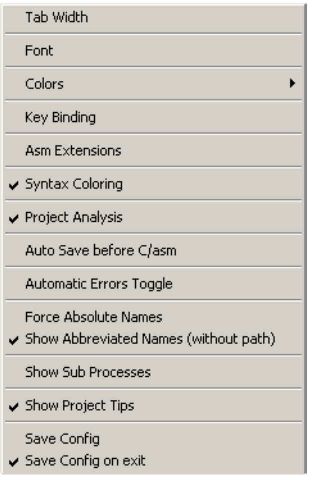


Fig. 4-2: Cutomize menu

An option is set if a check mark appears before its name. To set an unchecked option, click on it in the **Cutomize** drop-down menu. The next time you open the **Cutomize** drop-down menu, a check mark appears before the option name.

An option is not set if no check mark appears before its name. To clear a checked option, click on it in the **Cutomize** drop-down menu. The next time you open the **Cutomize** drop-down menu, no check mark appears before the option name.

Syntax Coloring

The **Syntax Coloring** option, if set, provides color coding in your project source files to assist you in programming. The following table lists the type of text that is color coded and the default color.

Comments	Green
Preprocessor Keyword	Light Red
C Keyword	Blue
C Library Function	Red
Assembler Mnemonics	Red
Assembler Directives	Blue
Link Directives	Red

If this option is not set, all source file text is in black.

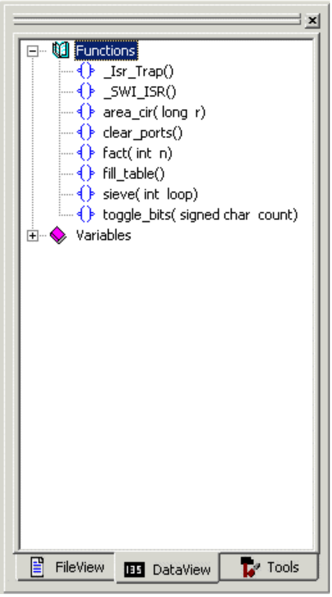
You can change the default color for each item using the **Colors** option in the **Customize** menu. Refer to “Colors” on page 75 for details.

To toggle the **Syntax Coloring** option, select **Syntax Coloring** for the **Customize** drop-down menu. Alternatively, you can type **Alt+C+Y**.

Project Analysis

The **Project Analysis** option in the **Customize** drop-down menu adds **Function** and **Variable** lists to the Data View Pane in the **Project Window**.

The figure below shows a typical Data pane with functions and variables listed:



```
⊟ Functions
   {} _Isr_Trap()
   {} _SWI_ISR()
   {} area_cir( long r)
   {} clear_ports()
   {} fact( int n)
   {} fill_table()
   {} sieve( int loop)
   {} toggle_bits( signed char count)
⊞ Variables
```

Fig. 4-3: Data View: Function and Variable Lists

If this option is set, IDEA parses each source file in the project to display all function and variable definitions. This makes it easier to organize a project and monitor function and variable usage.

If this option is not set, functions and variables are not shown.

To toggle the **Project Analysis** option, select **Project Analysis** from the **Customize** drop-down menu. Alternatively, you can type **Alt+C+J**.

Auto Save before C/asm

The **Auto Save before C/asm** option, if set, automatically saves a C or Assembly source code file before it is compiled or assembled via a **Project/Build > Compile File**, **Project/Build > Make**, or **Project/ Build > ReBuild All** command. In addition, it automatically saves before you exit IDEA.

If this option is not set, you are asked whether you wish to save the file. If you select **Yes**, the file is saved and the compilation or assembly proceeds. If you select **No**, the file is not saved and the compilation or assembly proceeds using the last saved version of the file, not the current version of the file.

To toggle the **Auto Save before C/asm** option, select **Auto Save before C/asm** from the **Customize** drop-down menu. Alternatively, you can type **Alt+C+A**.

Automatic Errors Toggle

The **Automatic Errors Toggle** option, if set, automatically opens the **Errors** window when errors are detected after a compile, link, make, or build operation.

If this option is not set, errors are reported by default in the IDEA Status bar. The **Errors** window can be opened manually using the **Show Error File** option on the **Errors** sub-menu.

To toggle the **Automatic Errors Toggle** option, select **Automatic Errors Toggle** from the **Customize** drop-down menu. Alternatively, you can type **Alt+C+E**.

Force Absolute Names

The **Force Absolute Names** option, if set, adds the full path for all source files in the project folder to the linked executable. In addition, the full path is shown in the **Project Source Files** list.

If this option is not set, the path for all source files in the project folder is not added to the linked executable and is not shown in the **Project Source Files** list.

NOTE

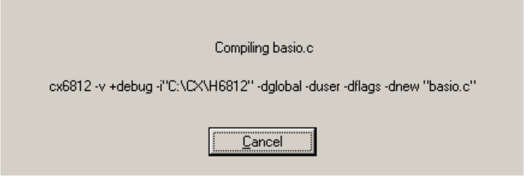
Project files which are not located in the project folder always use the full path.

To toggle the **Force Absolute Names** option, select **Force Absolute Names** from the **Customize** drop-down menu. Alternatively, you can type **Alt+C+N**.

Show Sub Processes

The **Show Sub Processes** option, if set, instructs IDEA to show all sub-processes during a compilation, link, make, or build.

If this option is not set, a simple dialog appears during a compilation.



Compiling basio.c

```
cx6812 -v +debug -i"C:\CX\H6812" -dglobal -duser -dflags -dnew "basio.c"
```

Fig. 4-4: Compilation status dialog box

If this option is set, in addition to the dialog above, a DOS window appears with details on the compilation subprocesses.

To toggle the **Show Sub Processes** option, select **Show Sub Processes** from the **Customize** drop-down menu. Alternatively, you can type **Alt+C+B**.

Show Project Tips

The **Show Project Tips** option, if set, shows names for project components in the Project Window.

If this option is not set, names are not shown.

To toggle the **Show Tips** option, select **Show Tips** from the **Customize** drop-down menu. Alternatively, you can type **Alt+C+P**.

Save Config

The **Save Config** option immediately saves the current IDEA configuration. Unlike the other options on the **Customize** submenu, the **Save Config** option is not a toggle.

To save the current IDEA configuration, select **Save Config** from the **Customize** drop-down menu. Alternatively, you can type **Alt+C+S**.

Save Config on exit

The **Save Config on exit** option, when set, saves the current IDEA configuration when you exit the program.

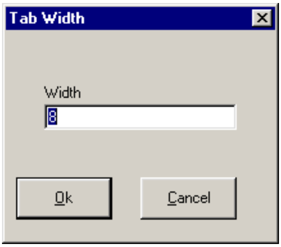
If this option is not set, IDEA will use the last saved configuration when it starts up again.

To toggle the **Save Config on exit** option, select **Save Config on exit** from the **Customize** drop-down menu. Alternatively, you can type **Alt+C+X**.

Tab Width

The **Tab Width** option lets you specify the number of spaces that the tab key moves the cursor in a source file. The default is 8.

Select the **Tab Width** option by clicking on **Tab Width** in the **Customize** menu. Alternatively, type **Alt+C+T**. The **Tab Width** dialog box appears.



Width: 8

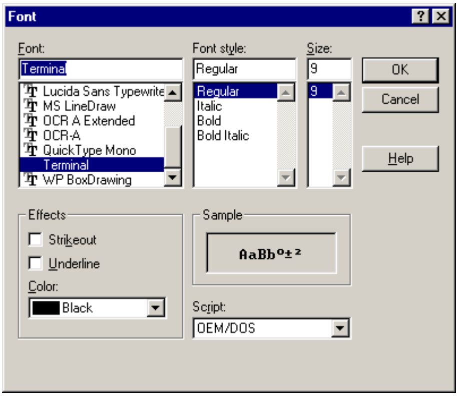
Fig. 4-5: Tab Width dialog box

Specify a tab width in the **Width** field and click on **OK**.

Font

The **Font** option lets you specify the default font for viewing and editing source files.

Select the **Font** option by clicking on **Font** in the **Customize** menu. Alternatively, type **Alt+C+F**. The **Font** dialog box appears.



Font: Terminal

Font style: Regular

Size: 9

Color: Black

Script: OEM/DOS

Sample: AaBb°±²

Fig. 4-6: Font dialog box

The default font is **Terminal**, **Regular** style, **9 point** size, **black**. You can select a new font from the **Font** list and view it in the **Sample** field. You can also specify font style, size, and color.

Colors

The **Colors** option lets you specify the default font color for certain types of text in your source files. If you have set the **Syntax Coloring** option in the **Customize** menu, IDEA recognizes several different items in a project file and automatically displays them in a color that you can easily distinguish.

IDEA can display the following items in different colors:

- **Comments**
- **Preprocessor keywords**
- **C Keywords**
- **C Library functions**
- **Assembler mnemonics**
- **Assembler directives**
- **Link directives**

For details on the **Syntax Coloring** option and the default colors for these items, refer to “Syntax Coloring” on page 70.

Select the **Colors** option by moving the cursor over **Colors** in the **Customize** menu. Another menu appears with a list of items for which you can change the color. Click on the appropriate item.

Alternatively, type **Alt+C+C**, move the cursor over the appropriate item, and click again.

The **Colors** menu appears below.

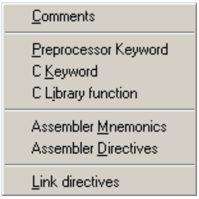


Fig. 4-7: Colors menu

After you click on one of the above items, the **Color** dialog box appears.

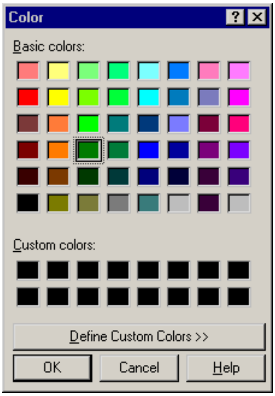


Fig. 4-8: Color dialog box

You can select a basic color from the **Basic colors** array or select a custom color from the **Custom colors** array. The current color for the item is surrounded by a black border.

The **Custom colors** array is initially all black. You can create your own colors by clicking on the **Define Custom Colors >>** button. This expands the **Color** dialog box so that you can specify the custom color exactly.

After you define a custom color, it appears in the **Custom colors** array for all items.

Key Binding

The **Key Binding** option lets you specify keyboard shortcuts for nearly all of the menu commands in IDEA.

Select the **Key Binding** option by clicking on **Key Binding** in the **Customize** menu. Alternatively, type **Alt+C+K**.

The **Key Binding** dialog box appears.

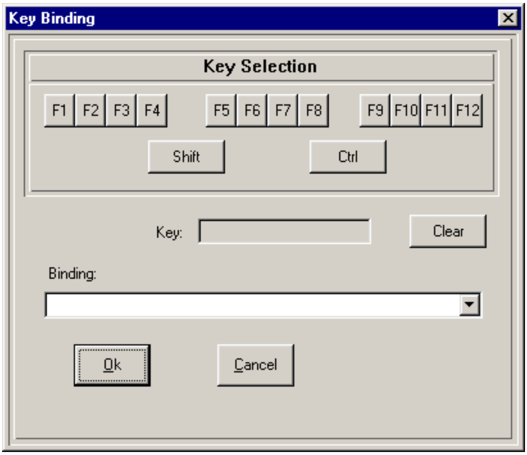


Fig. 4-9: Key Binding dialog box

You can use any combination of the **Ctrl**, **Shift**, and **Function** keys to create a new shortcut. Click on your choice(s) in the **Key Selection** field. As you click, the shortcut definition is built up in the **Key** field. To remove **Ctrl** or **Shift** from the key definition, click on them again. To change the **Function** key in the key definition, click on your new choice.

After you specify a key sequence, you must bind it to an action. If the key sequence you have specified is not already in use, the **Binding** field will be blank. Click on the down arrow to the right of the field and select the action that you want to associate with the key sequence from the action list.

If the key sequence you have specified is already in use, the **Binding** field will show the action with which the key sequence is associated. You can either select a new key sequence for the action or click on the **Clear** button to clear the current action from the **Binding** field.

After you create a keyboard shortcut for a menu option, the shortcut will appear after the option name in the drop-down menu.

Working Directory

You can specify a working directory for your projects and for locating files. IDEA uses the working directory when you select options such as **File > Open**, **Project > Load**, etc.

Specify the working directory by clicking on **Working Directory** in the **Project** menu. Alternatively, type **Alt+P+W**.

Refer to “Working Directory” on page 66 for additional information.

Asm Extensions

The **Asm Extensions** option lets you specify additional file extensions that the assembler will recognize. The default file extension for assembler files is **.s**.

Select the **Asm Extensions** option by clicking on **Asm Extensions** in the **Customize** menu. Alternatively, type **Alt+C+M**.

The **Asm Extensions** dialog box appears.

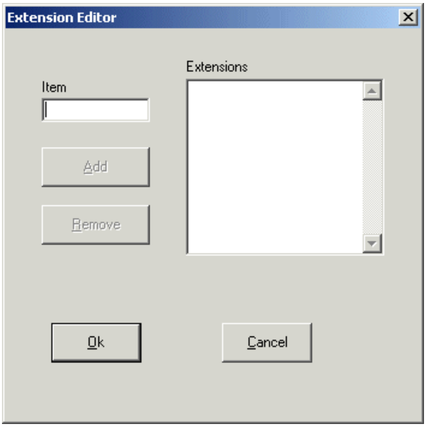


Fig. 4-10: Asm Extensions dialog box

The **Asm Extensions** dialog box lets you build a list of file name extensions that the assembler will recognize during assembly. The default extension for assembler files is **.s**.

Enter an extension (for example, **.asm**) in the **Item** field, and then click on the **Add** button to add the extension to the **Extensions** list.

To remove an extension from the list, select the extension and then click on the **Remove** button.

Debuggers

You can use appropriate debuggers to debug your project in IDEA. Before you can use a debugger, you must specify its location. You can do this in one of two ways:

1) From the **Project Window**, select the **Tools** Pane. Right click on one of the **Debuggers** icon or to open a dialog box that allows you to specify the location of a debugger.

2) The Main menu path of **Build > Setup Tools > Debugger > Application** allows you to specify the location of the debugger(s).

After you select a debugger, the path and filename appears after the **Debugger** icon in the **Tool Pane** and the Toolbar icon will turn on.

Errors Report

IDEA lets you display and scroll through any errors encountered during a compile, link, make, or build. The **Automatic Errors Toggle** option in the **Customize** drop-down menu and the **Show Error File** option in the **Errors** drop-down menu control the display of the **Errors** report.

Customize > Automatic Errors Toggle option

The **Automatic Errors Toggle** option, if set, automatically opens the **Errors** report when errors are detected after a compile, link, make, or build operation.

If this option is not set, errors are reported by default in the IDEA Status bar. The **Errors** report can be opened manually using the **Show Error File** option on the **Window** sub-menu.

To toggle the **Automatic Errors Toggle** option, select **Automatic Errors Toggle** from the **Customize** drop-down menu. Alternatively, you can type **Alt+C+E**.

Window > Show Error File option

The **Show Error File** option, when checked, shows the most recent error file generated for a compile, link, make, or build. If the **Show Error File** option is unchecked, the error file is hidden.

Toggle the **Show Error File** option by clicking on **Show Error File** in the **Window** menu. Alternatively, type **Alt+W+E**.

CHAPTER
5

Managing an IDEA Project

- Overview
- Opening a project
- Project Name
- Project Description
- Project Target File Name
- Project Source Files
- Project Header Files
- Project Directory
- Project Defines
- Project Include Search Paths
- Project Object Search Paths

Overview

This chapter provides complete details on managing an IDEA project. You can use it to create and build a new project.

If you have not created an IDEA project yet, IDEA is supplied with an example project (for example, “**demo12.prj**” for the Cosmic HC12/HCS12 compiler) You can use this example project to become familiar with the principles of managing an IDEA project.

Opening a project

To open a new project, select **Project** > **New** from the Main Menu or type **Alt+P+N**.

The **File View** pane of the **Project Window** appears at the left of the window area with a new, untitled project opened.

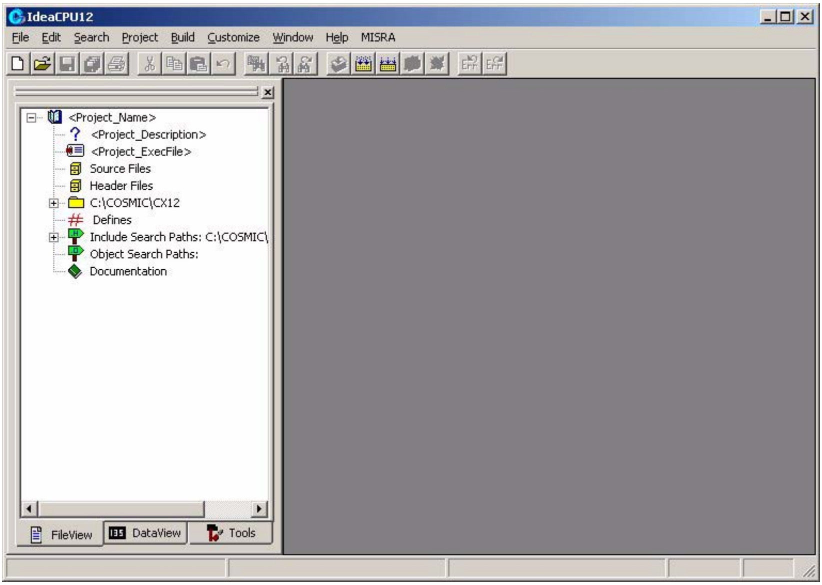


Fig. 5-1: Project Window with new project opened

To open an existing project, select **Project** > **Load** from the **Main Menu** or type **Alt+P+L**.

In the dialog box that appears, select the project from the folder where it is located. The Project Window appears with the selected project opened.

A project is composed of ten different project components:

- Project Name

- Project Description
- Project Target File Name
- Project Source Files
- Project Header Files
- Project Directory
- Project Defines
- Project Include Search Paths
- Project Object Search Paths
- Project Documentation

Each of these components is described in detail in the following sections.

Project Name

The **Project Name** component lets you specify a name for the project. It also represents the parent component for all project sub-components.

To specify a name for the project, click on the **Project Name** icon. A text cursor appears to the right of the icon. Click on the text cursor to open a text box and enter a project name.

Right click on the **Project Name** icon to view a menu of project commands. These commands are shown in the following table:

Add File/File Group	Adds a source file to the project.
Add Group	Adds a File Group to the project, this group can then contain source files.
Save	Saves the project.
Save As	Saves the project with a new name.
Make	Checks source file up-to-date status and dependencies. Then selectively compiles or assembles any out-of-date files and runs the Linker. The icons in the **Project Source Files** folder are colored yellow.
Build	Performs a **Make** as described above and then runs any utilities selected in the **Builder Configuration** dialog box.
Rebuild All	Force all files to be compiled/assembled (whether they are up to date or not), then performs a **Make** as described above and then runs any utilities selected in the **Builder Configuration** dialog box.
Mark All	Marks all project source files for recompile/assemble without changing the file time/date stamp. The icons in the **Project Source Files** folder are colored orange.

Touch All	Marks all project source files for recompile/ assemble and updates all project source files with the current system date and time stamp. The icons in the **Project Source Files** folder are colored red.
Documentation	Adds a document file to the project.

Project Description

The **Project Description** component lets you specify a short description for the project.

To specify a description for the project, click on the **Project Description** icon ?. A text cursor appears to the right of the icon. Click on the text cursor to open a text box and enter a short project description.

Project Target File Name

The **Project Target File Name** component lets you specify a target file name for the project. This name is used as the root name for the linked executable.

To specify a project target file name, click on the **Project Target File Name** icon ▤.

A text cursor appears to the right of the icon. Click on the text cursor to open a text box and enter a target file name. Be sure to include the target file name extension; for example, "**.h12**" for the Cosmic HC12/HCS12 compiler.

Right click on the **Project Target File Name** icon to view a menu containing target file commands. These commands are shown in the following table:

Inspect Object	Runs the **Object Inspector** utility (***cobj***) on the target file.
Show Debug	Runs the **Debug Info Examiner** utility (***cprd***) and opens the project debug file in read-only mode.
Produce Hex Records	Runs the **Hex Converter** utility (***chex***), which translates executable images produced by the *clnk* linker to one of several hexadecimal interchange formats.
Produce Absolute Listings	Runs the **Absolute Lister** utility (***clabs***) to generate absolute listings.
Produce IEEE Output	Runs the **IEEE695 Converter** utility (***cv695***) to generate IEEE695 debug format.
Produce ELF/DWARF Output	Runs the **ELF/DWARF Converter** utility (***cvdwarf***) to generate ELF/Dwarf debug format.
Debug File	Runs the first debugger and loads the linked executable.
Delete	Deletes the project target file. A pop-up dialog box asks you to confirm the deletion

Project Source Files

The **Project Source Files** component lets you specify the C and Assembly language source files to be included in the project. (Please note you only specify the source files, NOT the header files).

Right click on the **Project Source Files** icon to view a menu containing source file management commands. These commands are shown in the following table:

Add File	Adds a source file to the project.
Add Group	Adds a group to the project. This group can then be used to group source files
Touch All	Updates all project source files with the current system date and time stamp and marks them for recompile/assemble when a **Make** or **Build** is executed. The icons in the **Project Source Files** folder are colored red.
Mark All	Marks all project source files for recompile/ assemble when **Make** or **Build** is executed. This option does not change the time-date stamp of the files. The icons in the **Project Source Files** folder are colored orange.

Adding File Groups to the project

File Groups are a convenient way to group a number of source files that share some kind of common property. It can be any property, and for example:

- a set of common compilation flags
- a set of common common defines
- a specific link request for this set of files
- .....

File Groups have a number of attributes:

- Options for the C source files that are part of the group
- Options for the assembler files that are part of the group
- “Defines” that are common to all the files
- Documentation files

You can still specify such properties for every file that is apart of the group. Such properties override the Group properties, or complement them.

You can add **File Groups** to the project using the **Add Group** command.

- To add a **File Group** right click on the **Project Source Files** icon, select **Add Group**. An unnamed Group is then created and you then must enter the Group name.

Adding source files to the project

You can add source files to the project using the **Add File** command.

- To add an individual source file using the **Add File** command, right click on the **Project Source Files** icon, select **Add File**, and select the file to add from the **Add File** dialog box.
- To add a source file in a **File Group** using the **Add File** command, right click on the icon of the target **File Group**, select **Add File**, and select the file to add from the **Add File** dialog box.

Working with File Groups

Each File Group included in the project is listed next to a **File Group** icon 📁. The **File** Group icon lets you view the source file and its attributes.

Right click on the **File Group** icon to view a menu containing **File Groups** commands. These commands are shown in the following table:

Add File	Adds an individual file to the **Group**.
Make	Insures that all files in the **Group** are compiled/assembled if necessary.
Mark All	Marks all source files in the **Group** for recompile/assemble without changing the file time/date stamp. The File icons in the **Group** are colored orange.
Touch All	Updates source files in the **Group** with the current system date and time stamp and marks them for recompile/assemble when a **Make** or **Build** is executed. The file icons in the **Group** are colored red.
C Files Options	Sets common C compilation options for all C source files in the **Group**.
S Files Options	Sets common C asembly options for all assembler source files in the **Group**.
Remove	Removes a Group and all of its contents.
Documentation	Add a documentation file to the list of documents of the **Group**.

Working with Source Files

Each source file included in the project whether it is an individual file or it is part of a **File group** is listed next to a **Source File** icon. The **Source File** icon lets you view the source file and its attributes.

Right click on the **Source File** icon to view a menu containing source file commands. These commands are shown in the following table:

Edit	Opens the source file for editing.
Load (read only)	Opens the source file in read-only mode.
Compile	Compiles or assembles the source file. The source file icon is colored yellow if the **Compile** is successful.
Mark	Marks the source file for rebuilding. The **Source File** icon is colored orange.

Touch	Updates the source file with the current system date and time stamp and marks it for rebuilding. The **Source File** icon is colored red and the **Source File Time Stamp** icon is updated with the new date and time.
Replace with Backup	Uses the last backup available file to restore the source file.
Remove	Removes the source file from the project.
Properties	Shows file properties, which include full file name, last modification, and dependencies list.
Options	Opens the **Compiler** (or **Assembler**) **Options for Source File** dialog box, where you can specify options for the source file. Refer to "*Source File Options*" on page 96.
Defines	Opens the **#defines** dialog box, where you can specify compiler define options for the source file. Refer to "*Source File Defines*" on page 98.
Documentation	Adds a document file for the source file. Refer to "*Source File Documentation*" on page 96.

Source Files Components

The source file components, whether it is an individual file or it is part of a File Group are described in the following table:

🖫	Source File Time Stamp
📗	Source File Documentation
☰	Source File Options
#	Source File Defines
⚓	Source File Dependencies

Source File Time Stamp

The **Source File Time Stamp** component and icon shows the day, date, and time that the file was last saved or “touched”.

Source File Documentation

The **Source File Documentation** component shows all documents that are associated with the source file.

Right click on the **Documentation** icon and select **Add Doc** to associate a documentation file with the source file.

The **Document** component lets you view, edit, or remove a document associated with a source file. The appearance of the **Document** icon varies, depending on the type of document.

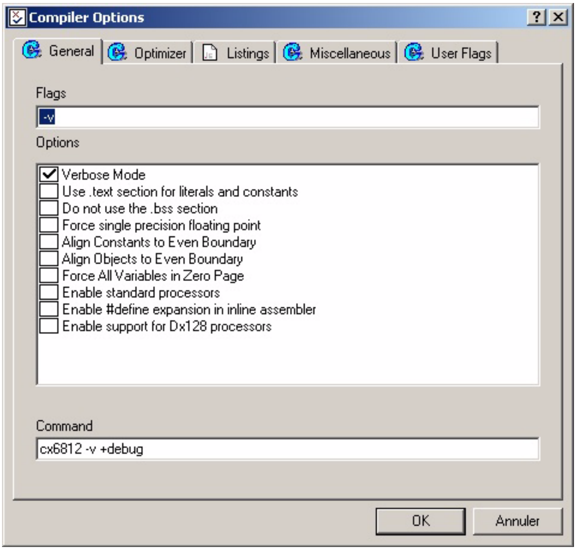
Right-click on the icon of a file included in the **Document** component to view a menu containing documentation file commands. The documentation file commands are described in the following table:

Load (read only)	Opens the document in read-only mode.
Open	Opens the document using the appropriate Windows-registered application.
Remove	Removes the document from the project.

Source File Options

The **Source File Options** component lets you specify compiler or assembler options for a source file, whether it is an individual source file or it is part of a **File Group**. These options override the default project compiler or assembler options.

Right-click on the **Source File Options** icon to open the **Compiler** (or **Assembler**) **Options for Source File** dialog box.



Flags

-v

Options

- ☑ Verbose Mode
- □ Use .text section for literals and constants
- □ Do not use the .bss section
- □ Force single precision floating point
- □ Align Constants to Even Boundary
- □ Align Objects to Even Boundary
- □ Force All Variables in Zero Page
- □ Enable standard processors
- □ Enable #define expansion in inline assembler
- □ Enable support for Dx128 processors

Command

```
cx6812 -v +debug
```

Fig. 5-2: Compiler Options for Source File dialog box

The **Compiler** (**Assembler**) **Options** dialog box has five tabs:

- General options
- Optimizer options
- Listings options
- Miscellaneous options
- User Flags

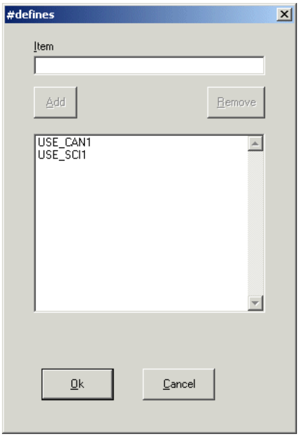
Choose a tab and select the desired options. Selected options are displayed in bold and unselected options are grayed out. To select an option simply click on the option description. To deselect an option click on the option again.

The source file compiler or assembler options will override the default project compiler or assembler options.

Source File Defines

The **Source File Defines** component lets you specify compiler #define options for a source file.

Right-click on the **Source File Defines** icon # to open the **#defines** dialog box and specify up to twenty user-defined preprocessor symbols.



USE_CAN1
USE_SCI1

Fig. 5-3: Source File #defines dialog box

To add a symbol to the list, enter the symbol in the **Item** field and click on **Add**. To remove a symbol from the list, select the symbol and click on **Remove**.

You can also add project #defines to the source file #define list by clicking on **Add Project Defines**.

Refer to "*Project Defines*" on page 102 for details.

After you add #defines, they appear as individual sub-components in the **Defines** list, each one after a **Define** icon DEFINE.

The define symbol is shown to the right of the icon. In addition, the day, date, and time that the #defines were last updated is shown next to the **Source File Defines** icon.

Source File Dependencies

The **Source File Dependencies** component and icon let you view the files that are named in the source file #includes.

The **File** icon shows a file that is named in the source file #includes. If the file name is enclosed in brackets, it is a system include file and is not typically modified in each project.

If the file name is enclosed in quotes, it is a user include file and can be modified. Right-click on the **File** icon to display a menu include file commands. The include file commands are described in the following table:

Open	Opens the file with the application in the Windows extension (File Types) Registry
Edit	Opens the file in IDEA for editing
Load (read only)	Opens the file in read-only mode.
Open	Opens the file for editing.
Touch	Updates the file with the current system date and time and marks it for rebuilding.
Delete	Removes the file from your system

Source File Functions

The **Function** icon in the Data View pane shows a function in the source file and lists all of the variables local to that function. Right-click on the **Function** icon to open the source file at the function.

Source File Variables

The **Source File Variables** component and icon in the Data View Pane let you view the variables that are local to the source file.

The **Variable** icon shows a variable declared with the source file.

NOTE

Functions *and* ***Variables*** *appear in the Data View pane only if* ***Project Analysis*** *is selected from the* ***Customize*** *menu.*

Project Header Files

The header files component lets you specify the header files of your project. This is just provided for conveniency.

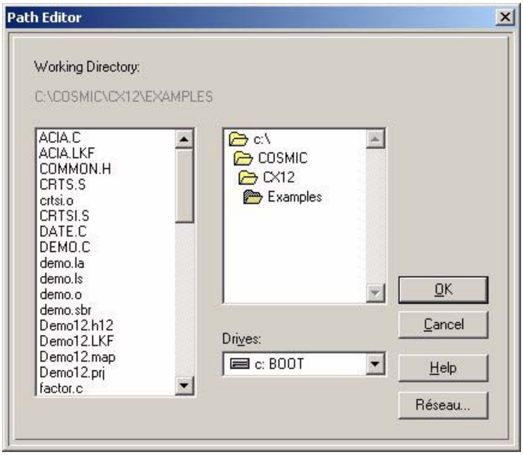
NOTE

*Specifying header files is **optional** and should only be used as a reference to the user. IDEA does not use thsi component in any way.*

Project Directory

The **Project Directory** component is used to set the working directory for the project. This is typically where the source code files for the project are located.

Right-click on the **Project Directory** icon 📁 to open the **Path Editor** dialog box and set the path to the source files.



Working Directory:
C:\COSMIC\CX12\EXAMPLES

Drives: c: BOOT

Fig. 5-4: Path Editor dialog box

The **Folder** icon 📁 shows folders in the project directory.

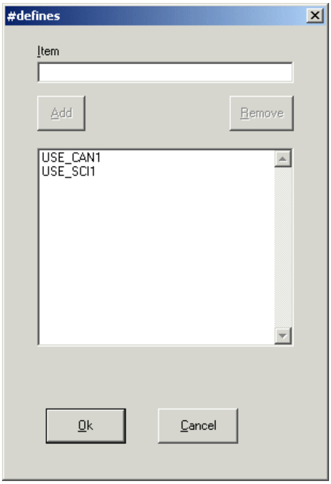
The **File** icon 📄 shows files in the project directory.

Right-click on the **File** icon to display a menu with file commands. The file commands are described in the following table:

Load (read only)	Opens the file in read-only mode.
Open	Opens the file for editing.
Touch	Updates the file with the current system date and time and marks it for rebuilding.

Project Defines

The **Project Defines** component lets you specify #define options for the project. Right-click on the **Project Defines** icon # to open the #defines dialog box and specify up to twenty preprocessor symbols.



USE_CAN1
USE_SCI1

Fig. 5-5: Project #defines dialog box

To add a symbol to the list, enter the symbol in the **Item** field and click on **Add**. To remove a symbol, select it and click on **Remove**.

Symbols defined as project #defines can be imported into source file #defines. Refer to "*Source File Defines*" on page 98 for details.

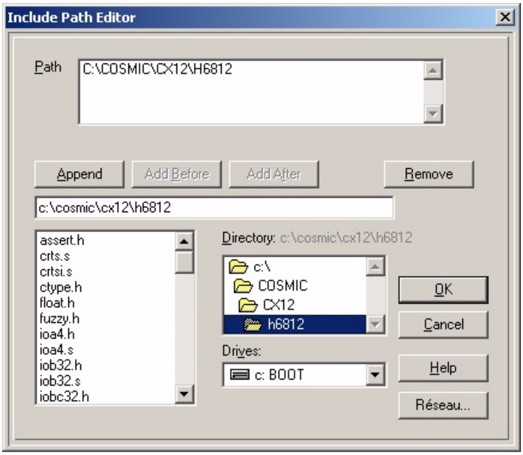
After you add project #defines, they appear as individual sub-components in the **Defines** list, each one after a **Define** icon DEFINE.

The define symbol is shown to the right of the icon. In addition, the day, date, and time that the project #defines were last updated is shown next to the **Project Defines** icon.

Project Include Search Paths

The **Include Search Paths** component lets you specify include paths for the compiler (**-i** > option).

Right-click on the **Include Search Paths** icon to open the **Include Path Editor** and specify up to twenty include paths for the project.



Include Path Editor

Path: C:\COSMIC\CX12\H6812

c:\cosmic\cx12\h6812

assert.h
crts.s
crtsi.s
ctype.h
float.h
fuzzy.h
ioa4.h
ioa4.s
iob32.h
iob32.s
iobc32.h

Directory: c:\cosmic\cx12\h6812

c:\
COSMIC
CX12
h6812

Drives: c: BOOT

Fig. 5-6: Include Path Editor

You can specify paths in any desired order. The paths are searched from top to bottom by the compiler.

You can use the **Drives** and **Directory** fields to specify the include path. Click on **Append** to add the path to the bottom of the list in the **Path** field.

To position the new path before or after the selected path, select an include path in the **Path** field and click on **Add Before** or **Add After**.

After you add include paths, they appear in order next to the **Include Search Paths** icon and as components in the **Include Search Paths** list, each one after a **Folder** icon.

Include Path: Folders and Files

The **Include Path Folder** icon 🗀 shows folders for include file paths in the project directory.

The **Folder** icon 🗀 shows folders in an include file path. The **File** icon 🗎 shows files in an include file path.

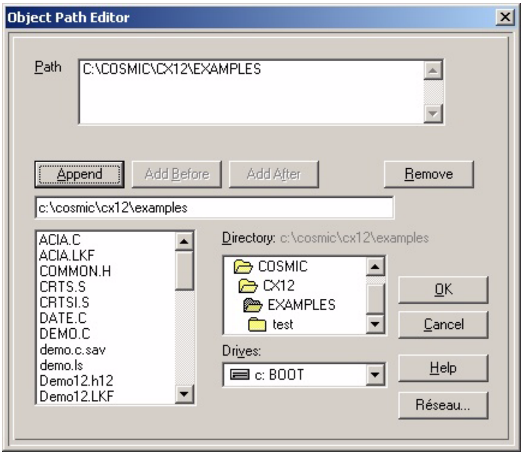
Right-click on the **File** icon to display a menu with the following commands:

Load (read only)	Opens the file in read-only mode.
Open	Opens the file for editing.
Touch	Updates the file with the current system date and time and marks it for rebuilding.

Project Object Search Paths

The **Object Search Paths** component lets you specify object paths for the **Maker**. These are the paths that will be used to check whether a source file has an associated object file, and whether this object file is up to date.

Right-click on the **Object Search Paths** icon to open the **Object Path Editor** and specify up to twenty include paths for the project.



Object Path Editor

Path: C:\COSMIC\CX12\EXAMPLES

c:\cosmic\cx12\examples

Directory: c:\cosmic\cx12\examples

```
ACIA.C
ACIA.LKF
COMMON.H
CRTS.S
CRTSI.S
DATE.C
DEMO.C
demo.c.sav
demo.ls
Demo12.h12
Demo12.LKF
```

```
COSMIC
  CX12
    EXAMPLES
      test
```

Drives: c: BOOT

Fig. 5-7: Object Path Editor

You can specify paths in any desired order. The paths are searched from top to bottom.

You can use the **Drives** and **Directory** fields to specify the include path. Click on **Append** to add the path to the bottom of the list in the **Path** field.

To position the new path before or after the selected path, select an include path in the **Path** field and click on **Add Before** or **Add After**.

After you add include paths, they appear in order next to the **Objects Search Paths** icon and as components in the **Objects Search Paths** list, each one after a **Folder** icon.

Objects Path: Folders and Files

The **Folder** icon 📁 shows folders in an object search path.

The **File** icon 📄 shows files in an object search path.

Right-click on the **File** icon to display a menu with the following commands:

Load (read only)	Opens the file in read-only mode.
Open	Opens the file for editing.
Touch	Updates the file with the current system date and time and marks it for rebuilding.

Project Documentation

The **Project Documentation** component shows all documents that are associated with the project.

IDEA also supports a special type of documentation called statup tip. If one of the documenation attached to the project is called “startup.tip” it will be automatically displayed in a read only window when the project is loaded. This allow to display some user specific “tips” when the project is loaded.

Right click on the **Documentation** icon and select **Add Doc** to associate a documentation file with the project.

The **Document** icon lets you view, edit, or remove a document associated with the project. The appearance of the icon varies, depending on the type of document.

Right-click on the **Document** icon to view a menu containing documentation file commands. These commands are described in the following Table:

Load (read only)	Opens the document in read-only mode.
Open	Opens the document for editing.
Remove	Removes the document from the project.

Project Tools

The **Project Tools** icon in the Tool pane lets you set project default options for:

- **Compiler**
- **Assembler**
- **Linker**
- **Builder**
- **Debugger**

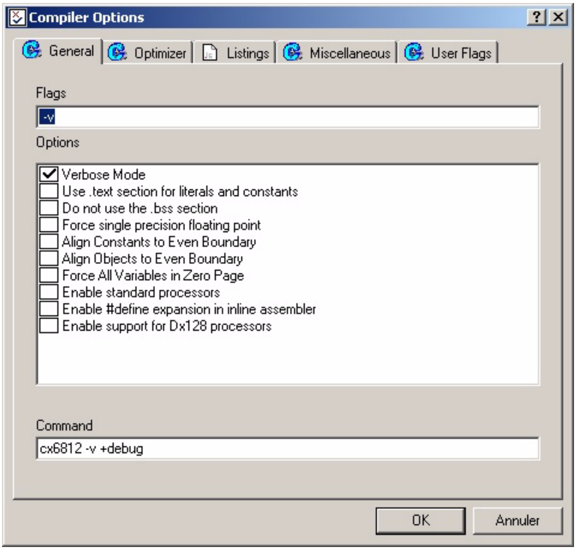
The **Builder** component lets you configure build utilities for the project, including:

- **Object Inspector (cobj)**
- **Hex Converter (chex)**
- **Debug Info Examiner (cprd)**
- **Absolute Lister (clabs)**
- **IEEE-695 Converter (cv695)**
- **ELF/Dwarf Converter (cvdwarf)**

Default Compiler Options

The **Compiler** component lets you set the default compiler options that are used to compile all C code (**.c**) files in a project.

Right click on the **Compiler** icon to open the **Compiler Options** dialog box. You can also double-click on the **Compiler** icon to display the **Compiler Options** icon and then right-click on the **Compiler Options** icon to open the **Compiler Options** dialog box.



Flags

-v

Options

☑ Verbose Mode
☐ Use .text section for literals and constants
☐ Do not use the .bss section
☐ Force single precision floating point
☐ Align Constants to Even Boundary
☐ Align Objects to Even Boundary
☐ Force All Variables in Zero Page
☐ Enable standard processors
☐ Enable #define expansion in inline assembler
☐ Enable support for Dx128 processors

Command

```
cx6812 -v +debug
```

Fig. 5-8: Compiler Options dialog box

The **Compiler Options** dialog box has five tabs:

- **General options**
- **Optimizer options**
- **Listings options**
- **Miscellaneous options**
- **User Flags**

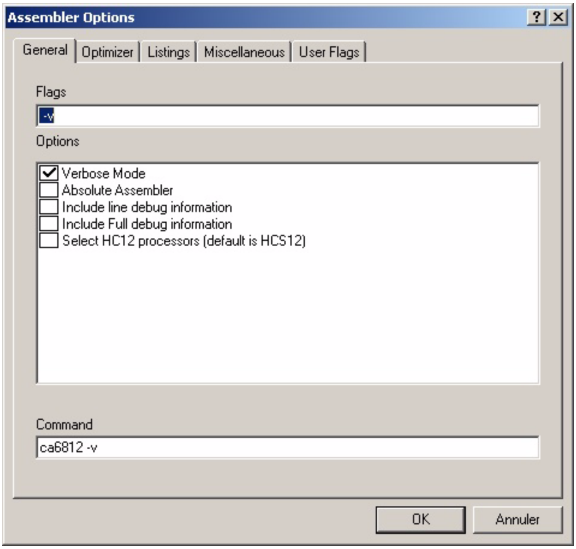
Choose a tab and select the desired options. Selected options are displayed in bold and unselected options are greyed out. To select an option, simply click on the option description. To deselect an option, click on the option again.

The default compiler options can be overridden by setting compiler options for the individual source files.

Default Assembler Options

The **Assembler** component lets you set the default assembler options that are used to assemble all assembly language (**.s**) files in a project.

Right click on the **Assembler** icon Asm to open the **Assembler Options** dialog box. You can also double-click on the **Assembler** icon to display the **Assembler Options** icon and then right-click on the **Assembler Options** icon to open the **Assembler Options** dialog box.



Flags
-v

Options
☑ Verbose Mode
□ Absolute Assembler
□ Include line debug information
□ Include Full debug information
□ Select HC12 processors (default is HCS12)

Command
ca6812 -v

Fig. 5-9: Assembler Options dialog box

The **Assembler Options** dialog box five tabs:

- **General options**
- **Optimizer options**
- **Listings options**
- **Miscellaneous options**
- **User Flags**

Choose a tab and select the desired options. Selected options are displayed in **bold** and unselected options are greyed out. To select an option, simply click on the option description. To deselect an option, click on the option again.

The default assembler options can be overridden by setting assembler options for the individual source files.

Default Linker Options

The **Linker** component lets you set the default ***clnk*** **utility** options that are used to link all files in a project. You can also specify a linker command file and edit the file.

Right click on the **Linker** icon to view a menu containing linker commands. The linker commands are described in the following table:

Options	Opens the **Link Configuration** dialog box.
Edit Command File	Opens the project link command file for editing.
Change Command File	Opens the **Select Linker Command File** dialog box.

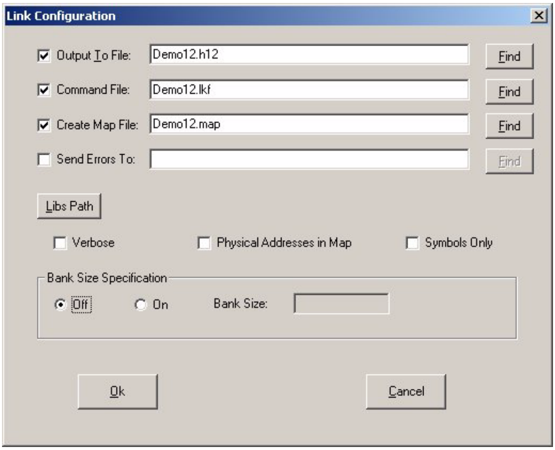
You can also double-click on the **Linker** icon to display the **Linker Options** icon and the **Linker Command File** icon .

Linker Configuration

Select **Options** from the **Project Linker** menu (or right-click on the **Linker Options** icon) to open the **Link Configuration** dialog box.

The **Link Configuration** dialog box lets you specify:

- **Linker options**
- **Libraries path option**
- **Reporting mode option**
- **Memory banking option**



☑ Output To File: Demo12.h12

☑ Command File: Demo12.lkf

☑ Create Map File: Demo12.map

☐ Send Errors To:

☐ Verbose ☐ Physical Addresses in Map ☐ Symbols Only

Bank Size Specification

● Off ○ On Bank Size:

Fig. 5-10: Link Configuration dialog box

Specifying linker options

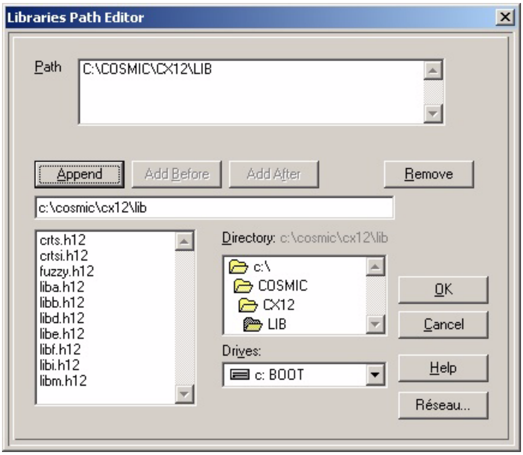
The Link Configuration dialog box lets you specify ***clnk* utility** options. These options are described in the following table:

Output file	**(-o option)**: writes output to the specified file. This option is required and has no default value.
Command file (.lkf)	The linker command file. This option is required and has no default value.
Map file	**(-m option)**: produces map information for the program being built to the specified file.
Error file	**(-e option)**: logs errors in the text file specified instead of displaying the messages on the screen.

After you select any one of these files, you can click on the **Find** button to specify the file name and path.

Specifying the libraries path

Click on the **Libs Path** button to open the **Libraries Path Editor** and set a path to the compiler library (-l > option).



Path: C:\COSMIC\CX12\LIB

c:\cosmic\cx12\lib

crts.h12
crtsi.h12
fuzzy.h12
liba.h12
libb.h12
libd.h12
libe.h12
libf.h12
libi.h12
libm.h12

Directory: c:\cosmic\cx12\lib

Drives: c: BOOT

Fig. 5-11: Libraries Path Editor

You can specify up to twenty library paths in any order. The paths are searched from top to bottom. After you add paths, they appear in order next to the **Libs Path** button.

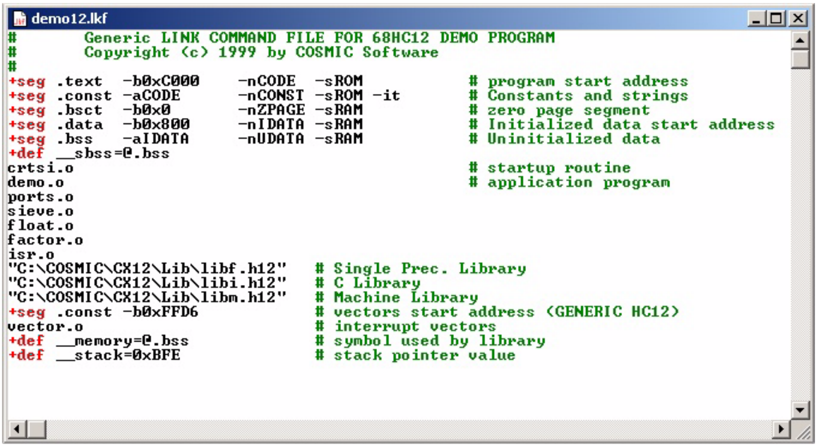
Other linker options you can set are described in the following table:

Verbose	**(-v option)**: be verbose.
Symbols Only	**(-s option)**: create an output file containing only an absolute symbol table, but still with an object file format.
Memory Banking	**(-bs option)**: enter the **size** of the page to be used. The size is translated to the correct **-bs** option for the linker. For example the default page size for HC12/HCS12 paging is 0x4000 which translates to a **-bs14**. The default value for most processors is **0** (bank switching **disabled**).

Editing the linker command file

Before you can edit a linker command file, you must first check the **Command File** check box in the **Link Configuration** dialog box and then specify a linker command file name and path.

Select **Edit Command File** from the **Project Linker** menu (or right-click on the **Linker Command File** icon) to open the linker command file for editing.



```
#       Generic LINK COMMAND FILE FOR 68HC12 DEMO PROGRAM
#       Copyright (c) 1999 by COSMIC Software
#
+seg .text  -b0xC000    -nCODE  -sROM            # program start address
+seg .const -aCODE      -nCONST -sROM -it        # Constants and strings
+seg .bsct  -b0x0       -nZPAGE -sRAM            # zero page segment
+seg .data  -b0x800     -nIDATA -sRAM            # Initialized data start address
+seg .bss   -aIDATA     -nUDATA -sRAM            # Uninitialized data
+def __sbss=@.bss
crtsi.o                                          # startup routine
demo.o                                           # application program
ports.o
sieve.o
float.o
factor.o
isr.o
"C:\COSMIC\CX12\Lib\libf.h12"   # Single Prec. Library
"C:\COSMIC\CX12\Lib\libi.h12"   # C Library
"C:\COSMIC\CX12\Lib\libm.h12"   # Machine Library
+seg .const -b0xFFD6            # vectors start address (GENERIC HC12)
vector.o                        # interrupt vectors
+def __memory=@.bss             # symbol used by library
+def __stack=0xBFE              # stack pointer value
```

Fig. 5-12: Linker command file for demoxx.prj

To edit the linker command file, you can:

1) make changes directly in the file using the options in the **Edit** menu. Type **Alt+E** to view the editing options. You can also right click to view a menu of editing options.

2) use the linker specific menu by right clicking in the linker file. The options provided by this menu are:

Insert File Allows you to insert an object file in the link command. The file will be inserted at the current cursor position.

Insert File from List Allows to insert a set of files from the files that belong to the project. You will be presented with a list of files and you can then select the files you want to include. They will included at the current cursor position.

Insert File List Will insert all the files belonging to the project.

Insert Group Allows to insert a group of files in the link file. Group of files are inserted via a **+inc** directive which will be inserted at the current cursor position.

Insert Libs Allows you to select a set of library to be included in the link file. These libraries will be included at the current cursor position.

Insert Default Libs Inserts the list of default libraries at the current cursor position.

Insert Segment Inserts a segment definition at the current cursor position. You will be presented with a dialog to define the properties of the segment.

Insert Symbol Definition Allows to insert a symbol definition at the current cursor position. You will be presented with a dialog to define the properties of the symbol.

The link file uses syntax coloring to highlight some of the linker control directives, and the comments that are included

By double clicking on the **+seg** directive you will be presented with a dialog that will allow you to modifiy this directive if needed.

Changing the linker command file

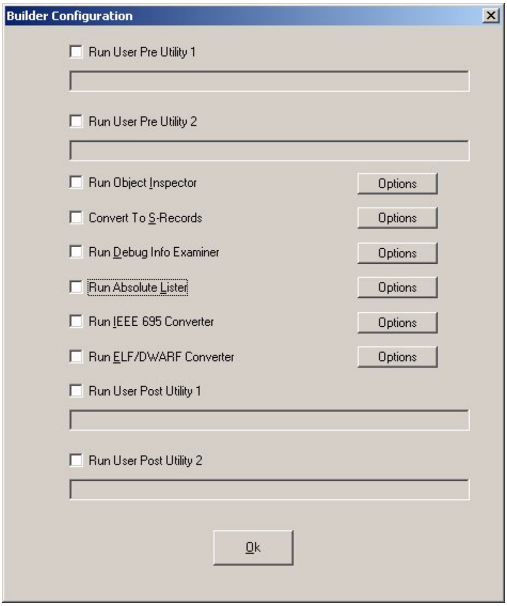
Select **Change Command File** from the **Project Linker** menu to change the linker command file. The **Select Linker Command File** dialog box lets you specify a file name and path for the new linker command file.

After you select a new command file, the **Command File** check box is checked in the **Link Configuration** dialog box, and the linker command file name and path are displayed.

Project builder utilities

The **Project Builder** component lets you specify utilities for building the project.

Right click on the **Project Builder** icon to open the **Builder Configuration** dialog box. You can also double-click on the **Project Builder** icon to display the **Builder Options** icon, and then right-click on the **Builder Options** icon to display the **Builder Configuration** dialog box.



□ Run User Pre Utility 1

□ Run User Pre Utility 2

□ Run Object Inspector

□ Convert To S-Records

□ Run Debug Info Examiner

□ Run Absolute Lister

□ Run IEEE 695 Converter

□ Run ELF/DWARF Converter

□ Run User Post Utility 1

□ Run User Post Utility 2

Fig. 5-13: Builder Configuration dialog box

The **Builder Configuration** dialog box contains check boxes that let you specify which builder utilities to run. The builder utilities are described in the following table.

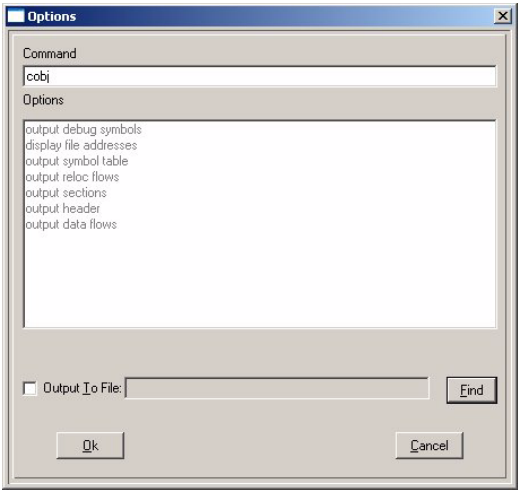
Run User Pre Utility 1	Runs the specified user utility before linking. You can specify a path and filename for the utility.
Run User Pre Utility 2	Runs the specified user utility before linking. You can specify a path and filename for the utility.
Run Object Inspector	Runs the ***cobj*** utility to examine object modules. If you select **Run Object Inspector** and then click on the **Options** button, the **Options** dialog box appears. Refer to "*Object Inspector utility*" on page 120 for details.
Convert to S-Records	Runs the ***chex*** utility to translate object module format to hexadecimal format. If you select **Convert to S-Records** and then click on the **Options** button, the **CHEX Configuration** dialog box appears. Refer to "*Hex Converter Utility*" on page 121 for details.
Run Debug Info Examiner	Runs the ***cprd*** utility to print debugging information about functions and data objects. If you select **Run Debug Info Examiner** and then click on the **Options** button, the **CPRD Configuration** dialog box appears.
Run Absolute Lister	Runs the ***clabs*** utility to generate absolute listings. If you select **Run Absolute Lister** and then click on the **Options** button, the **CLABS Configuration** dialog box appears. Refer to "*Absolute Lister Utility*" on page 124 for details
Run IEEE 695 Converter	Runs the ***cv695*** utility to generate IEEE695 format. If you select **Run IEEE 695 Converter** and then click on the **Options** button, the **CLABS Configuration** dialog box appears. Refer to "*IEEE695 Converter Utility*" on page 126 for details.

Run ELF/Dwarf Converter	Runs the ***cvdwarf*** utility to generate ELF/Dwarf format. If you select **Run ELF/Dwarf Converter** and then click on the **Options** button, the **CVDWARF Configuration** dialog box appears. Refer to “*Configuring the CVDWARF Converter utility*” on page 128 for details.
Run User Post Utility 1	Runs the specified user utility after linking. You can specify a path and filename for the utility.
Run User Post Utility 2	Runs the specified user utility after linking. You can specify a path and filename for the utility.

Object Inspector utility

The ***cobj*** utility lets you inspect relocatable object files or executable output by the assembler or linker. The ***cobj*** utility can be used to check the size and configuration of relocatable object files or to output information from their symbol tables.

Right click on the **Object Inspector** icon and select **Options** to open the **Options** dialog box. You can also double-click on the **Object Inspector** icon to display the **Options** icon and then right-click on it to display the **Options** dialog box.



Command
cobj

Options
output debug symbols
display file addresses
output symbol table
output reloc flows
output sections
output header
output data flows

□ Output To File:

Fig. 5-14: COBJ Utility Options dialog box

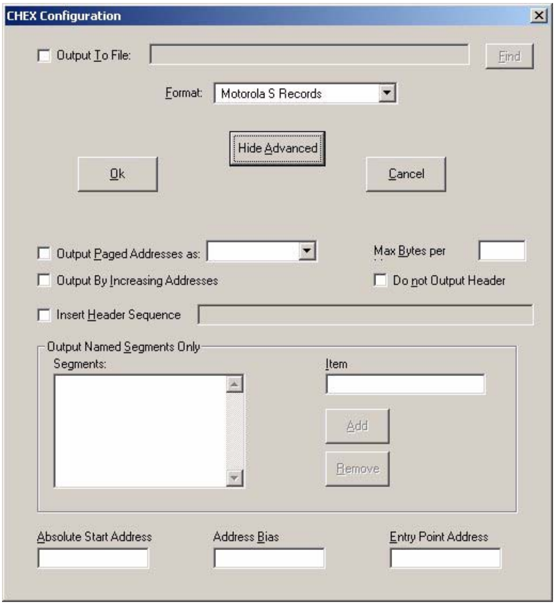
Selected options are displayed in bold and unselected options are greyed out. To select an option simply click on the option description and it is added to the command line. To deselect an option click on the option again.

You can also specify a path and file name to receive the **Object Inspector** output. This file may be in relocatable format or executable format.

Hex Converter Utility

The *chex* utility translates executable images produced by the ***clnk*** utility to one of several hexadecimal interchange formats.

Right click on the **Hex Converter** icon and select **Options** to open the **CHEX Configuration** dialog box. You can also double-click on the **Hex Converter** icon to display the **Options** icon and then right-click on it to display the **CHEX Configuration** dialog box.



CHEX Configuration

☐ Output To File:

Format: Motorola S Records

☐ Output Paged Addresses as:

Max Bytes per

☐ Output By Increasing Addresses

☐ Do not Output Header

☐ Insert Header Sequence

Output Named Segments Only

Segments:

Item

Absolute Start Address

Address Bias

Entry Point Address

Fig. 5-15: CHEX Configuration dialog box

Debug Info Examiner Utility

The *cprd* utility extracts and prints information about functions and data objects from an object module or executable image that has been compiled with the **+debug** option.

Right click on the **Debug Info Examiner** icon and select **Options** to open the **CPRD Configuration** dialog box. You can also double-click on the **Debug Info Examiner** icon to display the **Options** icon and then right-click on it to display the **CPRD Configuration** dialog box.

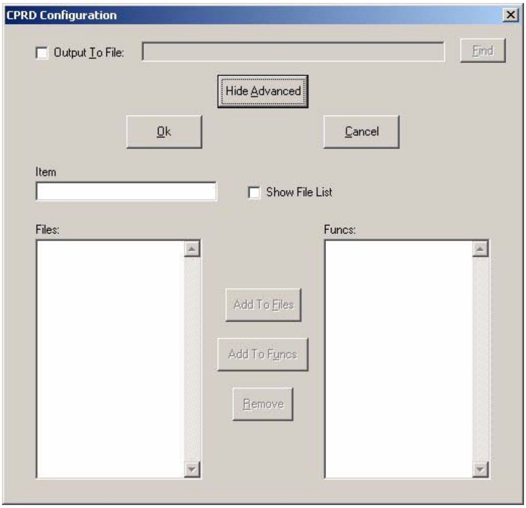


Fig. 5-16: CPRD Configuration dialog box

The **CPRD Configuration** dialog box lets you build a list of files and functions for debugging purposes. Enter a file or function name in the **Item** field, and then click on **Add to Files** to add the item to the **Files** list or **Add to Funcs** to add the item to the **Functions** list.

If you check the **Show File List** check box, the **Item** field changes to a **File List** field, with a drop-down list of the files in the project directory. Select a file from the list and then click on the **Add to Files** button to add it to the **Files** list.

To remove an item from either list, select the item and then click on the **Remove** button.

Each file in the **Files** list is processed with the **-fl** option, which prints debugging information about the file. By default, the ***cprd*** utility prints debugging information on all C source files.

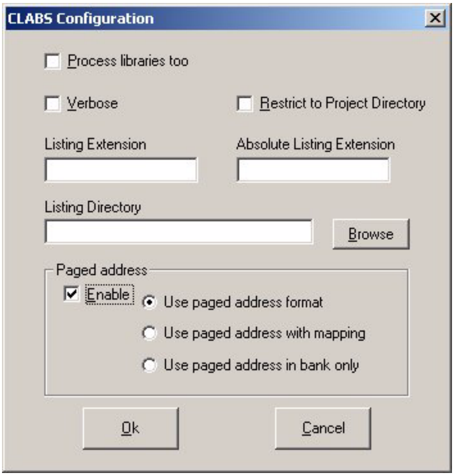
Each function in the **Functions** list is processed with the **-fc** option, which prints information only about the function. By default, the ***cprd*** utility prints debugging information on all functions in a file.

You can also specify a path and file name to receive the debugger output. This is equivalent to the ***cprd*** utility **-o** option. By default, the ***cprd*** utility writes debugging information to the terminal screen.

Absolute Lister Utility

The ***clabs*** utility processes relocatable C and Assembly listing files with the associated executable file to produce absolute listings with updated code and address values.

Right click on the **Absolute Lister** icon 🗎 and select **Options** to open the **CLABS Configuration** dialog box. You can also double-click on the **Absolute Lister** icon to display the **Options** icon ☰ and then right-click on it to display the **CLABS Configuration** dialog box.



CLABS Configuration

☐ Process libraries too

☐ Verbose ☐ Restrict to Project Directory

Listing Extension:

Absolute Listing Extension:

Listing Directory:

Paged address: ☑ Enable

● Use paged address format

○ Use paged address with mapping

○ Use paged address in bank only

Fig. 5-17: CLABS Configuration dialog box

The *clabs* utility options are described in the following table:

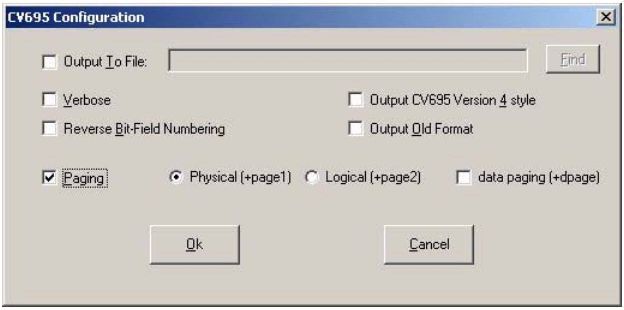
Process Libraries too	**(-a** option**)** **-** process also files located in libraries.
Verbose	**(-v** option**)** **-** the name of each module of the application is output to STDOUT.
Restrict to Project Directory	**(-l** option**)** **-** process files in the project directory only. The default is to process all files of the application.

Listing Extension	(**-r** option) **-** specify the input file extension. The default is “**.ls**”.
Absolute Listing Extension	(**-s** option) **-** specify the output file extension. The default id “**.la**”
Paged Address	(**-p** option) - output addresses of banked segment.

IEEE695 Converter Utility

The *cv695* utility converts a file produced by the linker into IEEE695 format.

Right click on the **IEEE695 Converter** icon IEEE and select **Options** to open the **CV695 Configuration** dialog box. You can also double-click on the **IEEE695 Converter** icon to display the **Options** icon ☰ and then right-click on it to display the **CV695 Configuration** dialog box.



□ Output To File:

□ Verbose

□ Output CV695 Version 4 style

□ Reverse Bit-Field Numbering

□ Output Old Format

☑ Paging ● Physical (+page1) ○ Logical (+page2) □ data paging (+dpage)

Fig. 5-18: CV695 Configuration dialog box

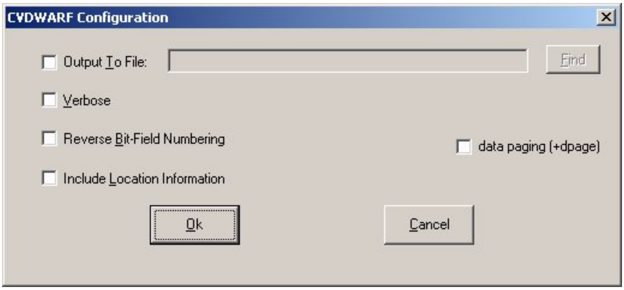
The *cv695* utility options are described in the following table:

Paging	(**+page# option**) - this option is currently meaningful for the **HC12/HCS12** only. This option specifies the address format for bank-switched code. If you check the **Paging** check box, three options appear to the right: **Physical (+page1)** - the application is banked and the *cv695* utility outputs physical addresses. This is the default if **Paging** is checked. **Logical (+page2)** - the application is banked and the *cv695* utility outputs addresses in paged mode: <page><offset_in_page>. This is equivalent to the old **+paged** flag. **data paging (+dpage)** - the application uses data paging.
Output to File	(**-o option**) - you can specify a path and file name to receive the *cv695* utility output. By default, the *cv695* utility outputs to the file whose name is obtained from the input file by replacing the filename extension with "**.695**".
Verbose	**(-v option) -** the ***cv695*** utility displays information about its activity.
Reverse BitField Numbering	**(-rb option) -** reverses bitfield from left to right.

Configuring the CVDWARF Converter utility

The *cvdwarf* utility converts a file produced by the linker into ELF/Dwarf format.

Right click on the **CVDWARF Converter** icon and select **Options** to open the **CVDWARF Configuration** dialog box. You can also double-click on the **CVDWARF Converter** icon to display the **Options** icon and then right-click on it to display the **CVDWARF Configuration** dialog box.



CVDWARF Configuration

☐ Output To File:

☐ Verbose

☐ Reverse Bit-Field Numbering

☐ data paging (+dpage)

☐ Include Location Information

Fig. 5-19: CVDWARF Configuration dialog box

The *cvdwarf* utility options are described in the following table:

Verbose	**(-v** option**) -** the ***cvdwarf*** utility displays information about its activity.
Paging	(**+page#** option) - this option is currently meaningful for the **HC12/HCS12** only. This option specifies the address format for bank-switched code. If you check the **Paging** check box, three options appear to the right: **Physical (+page1)** - the application is banked and the *cv695* utility outputs physical addresses. This is the default if **Paging** is checked. **Logical (+page2)** - the application is banked and the *cv695* utility outputs addresses in paged mode: `<page><offset_in_page>`. This is equivalent to the old **+paged** flag. **data paging (+dpage)** - the application uses data paging.
Reverse BitField Numbering	**(-rb** option**) -** reverses bitfield from left to right.
Include location information	(**-loc** option) location lists are used in place of location expressions whenever the object whose location is being described can change location during its lifetime. THIS POSSIBILITY IS NOT SUPPORTED BY ALL DEBUGGERS.
Output to File	(**-o** option) - you can specify a path and file name to receive the *cvdwarf* utility output. By default, the *cvdwarf* utility outputs to the file whose name is obtained from the input file by replacing the filename extension with "**.elf**".

Project Debuggers

IDEA provides in its Tools Configuration simultaneous support for two debuggers. Right-click on one of the **Project Debugger** icons ZAP or 🐞 to open a dialog box that allows you to specify the selected debugger's properties.

After you select a program, the path and filename appears after the **Project Debugger** icon.

Once you have specified a debugger, you can double click on the **Debugger** icon to run the ZAP debugger with the project target file opened. You can also run the debugger by clicking on one of the **Debugger** tools ZAP or 🐞 in the Tool bar.

CHAPTER

6

Building an IDEA Project

- Overview
- Compiling a Project
- Linking a Project
- Making a Project
- Building a Project

This page intentionally left blank.

Overview

After a project is set up and configured, you need to build it. There are four processes involved in building a project:

- **Compiling**
- **Linking**
- **Making**
- **Building**

Each of these processes is described briefly below and in detail later in the chapter.

Compiling

The **Compile** process compiles (**.c** files) or assembles (**.s** files) an open project source file. Compiler options are specified in the **Compiler Options** dialog box. Assembler options are specified in the **Assembler Options** dialog box.

Linking

The **Link** process runs the linker (and no other utilities) using the options specified for the project in the **Link Configuration** dialog box. Project source files are not checked for up-to-date status.

Making

The **Make** process first checks source file up-to-date status and dependencies. It then selectively compiles or assembles any out-of-date files and runs the **Linker**.

Building

The **Build** process performs a **Make** and then runs any utilities selected in the **Builder Configuration** dialog box. To have the **Builder** rebuild all files regardless of their up-to-date status, right click on the project name, select **Rebuild All** from the **Builder**.

Compiling a Project

The Compile process compiles (**.c** files) or assembles (**.s** files) an open project source file.

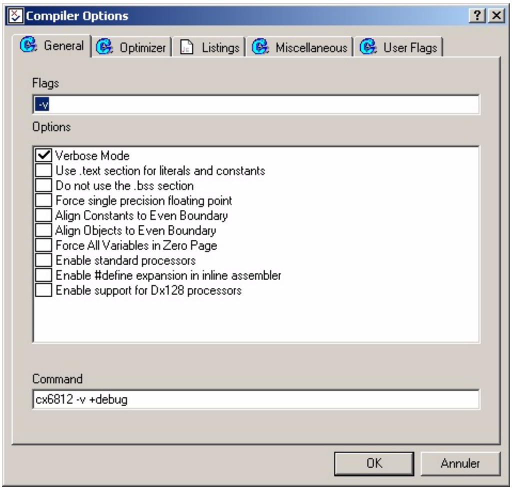
Specifying Compiler Options

Compiler options are specified in the **Compiler Options** dialog box. Assembler options are specified in the **Assembler Options** dialog box. You can specify compiler and assembler options for a project and for individual source files.

Project compiler options

Project compiler options apply to all (**.c**) source files in a project. You can override the project compiler options for one or more source files by specifying source file compiler options.

Double click on the **Tools** icon in the Tools Pane of the Project window to show the project tools. Then right click on the **Compiler** icon and select **Options** (or double click on the **Compiler** icon and right click on the **Options** icon). The **Project Compiler Options** dialog box appears.



Flags

-v

Options

☑ Verbose Mode
☐ Use .text section for literals and constants
☐ Do not use the .bss section
☐ Force single precision floating point
☐ Align Constants to Even Boundary
☐ Align Objects to Even Boundary
☐ Force All Variables in Zero Page
☐ Enable standard processors
☐ Enable #define expansion in inline assembler
☐ Enable support for Dx128 processors

Command

cx6812 -v +debug

Fig. 6-1: Project Compiler Options dialog box

The Title bar says “**Compiler Options**” and does not show a file name, indicating that these compiler options are for the entire project.

The **Project Compiler Options** dialog box is identical to the **Source File Compiler Options** dialog box. The available options for both are detailed in “*Compiler options*” on page 136.

Source file compiler options

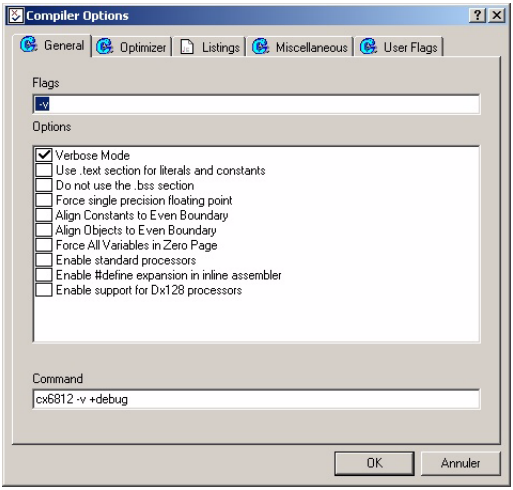
Source file compiler options apply only to a single (**.c**) source file. They override the project compiler options.

You can specify source file compiler options in one of two ways:

1) In the File View Pane of the Project window, double click on the **Files** icon to show the project files. Then right click on the **File** icon for the appropriate file and select **Options** from the pop-up menu.

2) In the Project window, double click on the **Files** icon to show the project files. Then double click on the **File** icon for the appropriate file to show the file attributes. Finally, right click on the **Options** icon .

The **Source File Compiler Options** dialog box appears.



Flags

-v

Options

☑ Verbose Mode
☐ Use .text section for literals and constants
☐ Do not use the .bss section
☐ Force single precision floating point
☐ Align Constants to Even Boundary
☐ Align Objects to Even Boundary
☐ Force All Variables in Zero Page
☐ Enable standard processors
☐ Enable #define expansion in inline assembler
☐ Enable support for Dx128 processors

Command

cx6812 -v +debug

Fig. 6-2: Source File Compiler Options dialog box

The Title bar says "**Compiler Options for: filename.c**", indicating that these compiler options are for a source file only.

The **Source File Compiler Options** dialog box is identical to the **Project Compiler Options** dialog box.

The available options for both are detailed in the next section.

Compiler options

The **Compiler Options** dialog box has five tabs:

- **General options**
- **Optimizer options**
- **Listings options**
- **Miscellaneous options**
- **User Flags**

Choose a tab and select the desired options. Selected options are displayed in bold and unselected options are greyed out. To select an option simply click on the option description. To deselect an option click on it again.

The default compiler options can be overridden by setting compiler options for the individual source files.

Listed below are the available compiler options. For additional compiler options and target-specific options, consult you compiler documentation.

Compiler General options

Verbose Mode (-v)

Be “verbose”. Before executing a command, print the command, along with its arguments, to STDOUT. The default is to output only the names of each file processed. Each name is followed by a colon and new line.

Use .text section for literals and constants (+nocst)

Output literals and constants in the code section **.text** instead of the specific section **.const**.

Do not use the .bss section (+nobss)

Do not use the **.bss** section for variables allocated in external memory. By default, such uninitialized variables are defined into the **.bss** section. This option is useful to force all variables to be grouped into a single section.

Align Object to Even Boundary (+even)

Align any object larger than one byte on an even boundary.

Compiler Optimizer options

Do not widen char and float arguments (+nowiden)

Do not widen char and float arguments. By default, char arguments are promoted to int before being passed as an argument.

Compiler Listings options

Generate Listings (-l)

Merge the C source listing with assembly language code; the listing output defaults to <*file*>*.ls*.

Compiler Miscellaneous options

Force prototyping (-pp)

Tells the parser to enforce prototype declaration for functions. An error message is issued if a function is used and no prototype declaration is found for it. By default, the compiler accepts both syntaxes without any error.

Generate Debug Information (+debug)

Produce debug information to be used by the debug utilities provided with the compiler and by any external debugger.

Number bits from MSB to LSB in bitfields (+rev)

Reverse the bitfield filling order. By default, bitfields are filled from the less significant bit (*lsb*) towards the most significant bit (*msb*) of a memory cell. If the **+rev** option is specified, bitfields are filled from the *msb* to the *lsb*.

Compiler User Flags (-d*^)

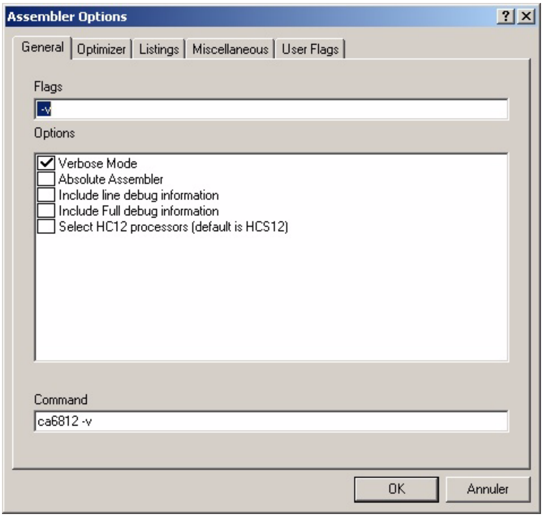
The **User Flags** tab allows you to specify up to twenty preprocessor symbols (**#define**). The form of the definition is **-dsymbol[=value]**; the symbol is set to **1** if **value** is omitted.

Project assembler options

Project assembler options apply to all assembly (**.s**) source files in a project. You can override the project assembler options for one or more source files by specifying source file assembler options.

Double click on the **Tools** icon in the Tools Pane of the Project Window to show the project tools. Then right click on the **Assembler** icon Asm and select **Options** (or double click on the **Assembler** icon and right click on the **Options** icon).

The **Project Assembler Options** dialog box appears.



Flags

-v

Options

☑ Verbose Mode
□ Absolute Assembler
□ Include line debug information
□ Include Full debug information
□ Select HC12 processors (default is HCS12)

Command

ca6812 -v

Fig. 6-3: Project Assembler Options dialog box

The Title bar says “**Assembler Options**” and does not show a file name, indicating that these assembler options are for the entire project.

The **Project Assembler Options** dialog box is identical to the **Source File Assembler Options** dialog box.

The available options for both are detailed in “*Assembler options*” on page 141.

Source file assembler options

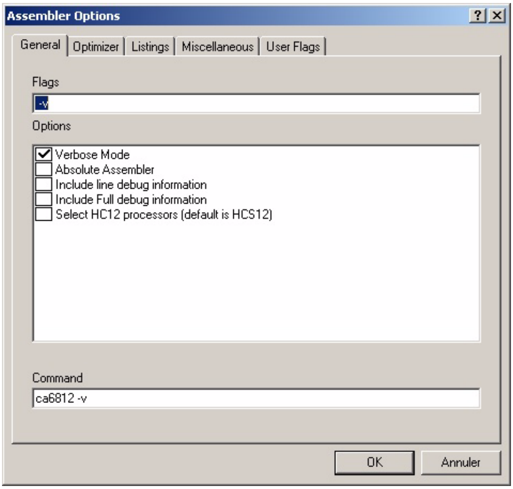
Source file assembler options apply only to a single (**.s**) source file. They override the project assembler options.

You can specify source file assembler options in one of two ways:

1) In the Project window, double click on the **Files** icon to show the project files. Then right click on the **File** icon for the appropriate file and select **Options** from the pop-up menu.

2) In the Project window, double click on the **Files** icon to show the project files. Then double click on the **File** icon for the appropriate file to show the file attributes. Finally, right click on the **Options** icon .

The **Source File Assembler Options** dialog box appears.



Flags: -v

Options

☑ Verbose Mode
☐ Absolute Assembler
☐ Include line debug information
☐ Include Full debug information
☐ Select HC12 processors (default is HCS12)

Command: ca6812 -v

Fig. 6-4: Source File Assembler Options dialog box

The Title bar says “**Assembler Options for: filename.s**”, indicating that these assembler options are for a source file only.

The **Source File Assembler Options** dialog box is identical to the **Project Assembler Options** dialog box. The available options for both are detailed in the following section.

Assembler options

The Assembler Options dialog box has five tabs:

- **General options**
- **Optimizer options**
- **Listings options**
- **Miscellaneous options**
- **User Flags**

Choose a tab and select the desired options. Selected options are displayed in bold and unselected options are greyed out. To select an option click on the option description. To deselect an option click on it again.

The default assembler options can be overridden by setting assembler options for the individual source files.

Listed below are the available assembler options. For additional assembler options, consult you compiler documentation.

Assembler General options

Include Full debug information (-xx)

Add debug information in the object file for any label defining code or data.

Include line debug information (-x)

Add line debug information to the object file.

Absolute Assembler (-a)

Map all sections to absolute, including the predefined ones.

Verbose Mode (-v)

Display the name of each file that is processed.

Assembler Optimizer options

Do not optimize branches (-b)

Do not optimize branch instructions. By default, the assembler replaces long branches by short branches wherever a shorter instruction can be used, and short branches by long branches wherever the displacement is too large. This optimization also applies to jump and jump to subroutines instructions.

Assembler Listings options

Output a listing (-l)

Create a listing file. The name of the listing file is derived from the input file name by replacing the suffix with the extension *.ls*.

Use form feed in listing (-ff)

Use formfeed character to skip pages in listing instead of using blank lines.

Force Title in Listings (-ft)

Output a title in the listing (date, file name, page). By default, no title is output.

Assembler Miscellaneous options

Keep All local symbols (-pl)

Put locals in the symbol table. They are not published as externals and will be displayed only in the linker map file.

Accept old MOTOROLA syntax (-m)

Accepts the old Motorola syntax. That is, the assembler will accept labels starting in the first column without a terminating colon. Also, the assembler will accept “*” as a comment delimiter instead of “;”.

Make all symbols Public (-p)

Mark all defined symbols as **public**. This option has the same effect as adding an **xdef** directive for each label.

Make all equates Public (-pe)

Mark all symbols defined by an **equ** directive as **public**. This option has the same effect than adding a **xdef** directive for each of those symbols.

Output Cross References (-c)

Produce cross-reference information. The cross-reference information will be added at the end of the listing file; this option forces the **-l** option.

Assembler User Flags (-d*>)

The **User Flags** tab allows you to specify user-defined symbols, up to a maximum of twenty. This option is equivalent to using an **equ** directive in each of the source files. The form of the definition is **-dname=value**, where **name** is defined to have the value specified by **value**.

Compiling (assembling) a File or a Project

Compiling (assembling) a file

You can compile (assemble) a file in one of four ways:

1) If the file is open in a **File** window, click anywhere on the File window to make the file the currently active file, and then select the **Compile** tool on the Tool bar.

2) If the file is open in a **File** window, click anywhere on the File window to make the file the currently active file, and then select **File > Compile** from the IDEA menu or type **Alt+F+C**.

3) If the file is open in a File window, click anywhere on the File window to make the file the currently active file, and then select **Project > Compile File** from the IDEA menu or type **Alt+P+C**.

4) In the Project window, double click on the **Files** icon to show the project files. Then right click on the **File** icon for the appropriate file and select **Compile** from the pop-up menu. This method lets you compile (assemble) a file that is not open.

After the file starts compiling (assembling), a Status box appears showing the progress during compilation. In addition, if you have selected **Show Sub Processes** from the **Customize** drop-down menu, a MS-DOS window opens and shows all processes during compilation.

Compiling (assembling) a project

You can compile (assemble) a project by doing a **Project Make** or **Project Build**. Refer to "*Making a Project*" on page 158 or "*Building a Project*" on page 159 for details.

Linking a Project

The **Link** process runs the linker (and no other utilities) using the options specified for the project in the **Link Configuration** dialog box. Project source files are not checked for up-to-date status.

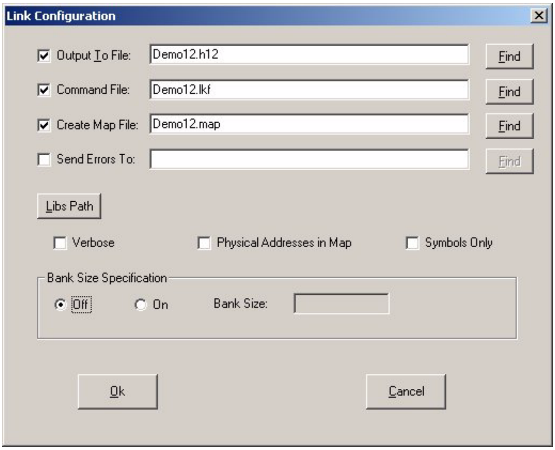
You can run the project linker by selecting the **Link** tool on the Tool bar.

A Status box appears showing the progress during linking. In addition, if you have selected **Show Sub Processes** from the **Customize** drop-down menu, a MS-DOS window opens and shows all processes during linking.

Specifying linker options

Double click on the **Tools** icon in the Project window to show the project tools. Then right click on the **Linker** icon and select **Options** (or double click on the **Linker** icon and right click on the **Options** icon).

The **Link Configuration** dialog box appears.



Link Configuration

☑ Output To File: Demo12.h12

☑ Command File: Demo12.lkf

☑ Create Map File: Demo12.map

☐ Send Errors To:

☐ Verbose ☐ Physical Addresses in Map ☐ Symbols Only

Bank Size Specification

● Off ○ On Bank Size:

Fig. 6-5: Link Configuration dialog box

The **Link Configuration** dialog box lets you specify:

- Linker options
- Libraries path option
- Reporting mode option
- Memory banking option

Linker options

After you select any one of these file options, you can click on the **Find** button to specify the file name and path.

Output file option (-o)

Writes output to the specified file. This option is required and has no default value.

Command file (.lkf)

The linker command file. This option is required and has no default value.

Map file option (-m)

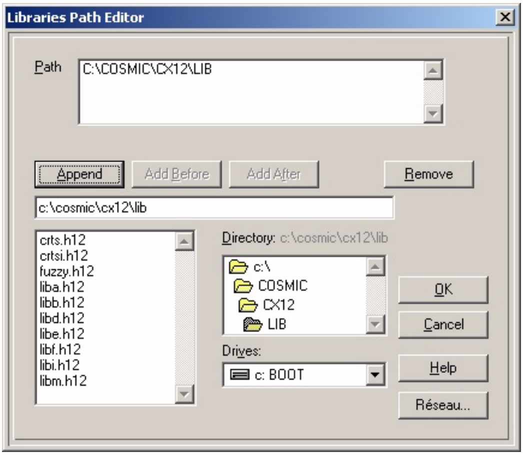
Produces map information for the program being built to the specified file.

Error file option (-e)

Logs errors in the text file specified instead of displaying the messages on the screen.

Libraries path option (-l)

Click on the **Libs Path** button to open the **Libraries Path Editor** and set a path to the compiler library.



Libraries Path Editor

Path: C:\COSMIC\CX12\LIB

c:\cosmic\cx12\lib

crts.h12
crtsi.h12
fuzzy.h12
liba.h12
libb.h12
libd.h12
libe.h12
libf.h12
libi.h12
libm.h12

Directory: c:\cosmic\cx12\lib

c:\
COSMIC
CX12
LIB

Drives: c: BOOT

Fig. 6-6: Libraries Path Editor

You can specify up to twenty library paths in any order. The paths are searched from top to bottom. After you add paths, they appear in order next to the **Libs Path** button in the **Link Configuration** dialog box.

Reporting Mode options

Verbose option (-v)

Be verbose.

Symbols Only option (-s)

Create an output file containing only an absolute symbol table, but still with an object file format.

Memory Banking option (-bs)

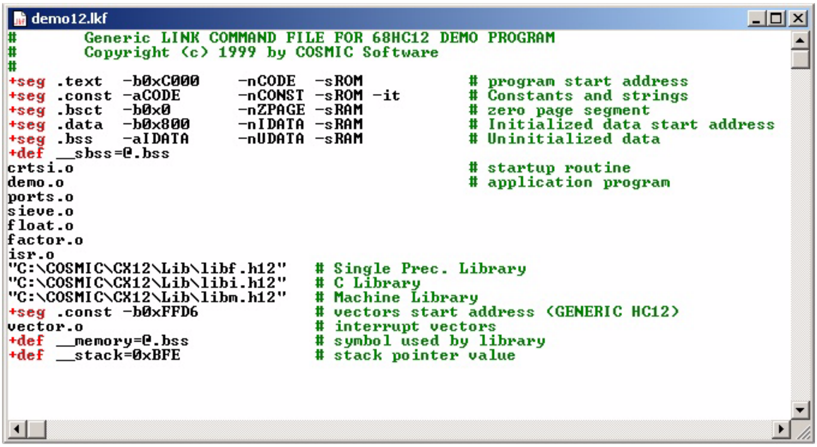
Enter the size of the page to be used. The size is translated to the correct **-bs** option for the linker. For example the default page size for 68HC12 paging is 0x4000 which translates to a **-bs14**. The default value for most processors is **0** (bank switching disabled).

Editing the Linker Command File

Before you can edit a linker command file, you must first check the **Command File** check box in the **Link Configuration** dialog box and then specify a linker command file name and path.

Right click on the **Linker** icon in the Tools Pane in the Project Window and select **Edit Command File** (or double click on the **Linker** icon and right click on the **Command File** icon). If you haven't specified a Linker Command File, IDEA will generate a generic one.

The **Linker Command File** is opened in a File window.



```
#       Generic LINK COMMAND FILE FOR 68HC12 DEMO PROGRAM
#       Copyright (c) 1999 by COSMIC Software
#
+seg .text  -b0xC000    -nCODE  -sROM         # program start address
+seg .const -aCODE      -nCONST -sROM -it     # Constants and strings
+seg .bsct  -b0x0       -nZPAGE -sRAM         # zero page segment
+seg .data  -b0x800     -nIDATA -sRAM         # Initialized data start address
+seg .bss   -aIDATA     -nUDATA -sRAM         # Uninitialized data
+def __sbss=@.bss
crtsi.o                                       # startup routine
demo.o                                        # application program
ports.o
sieve.o
float.o
factor.o
isr.o
"C:\COSMIC\CX12\Lib\libf.h12"       # Single Prec. Library
"C:\COSMIC\CX12\Lib\libi.h12"       # C Library
"C:\COSMIC\CX12\Lib\libm.h12"       # Machine Library
+seg .const -b0xFFD6                # vectors start address (GENERIC HC12)
vector.o                            # interrupt vectors
+def __memory=@.bss                 # symbol used by library
+def __stack=0xBFE                  # stack pointer value
```

Fig. 6-7: Linker command file

To edit the linker command file, you can make changes directly in the file using the options in the **Edit** drop-down menu. Type **Alt+E** to view the editing options.

You can also right click to view a pop-up menu of editing options.

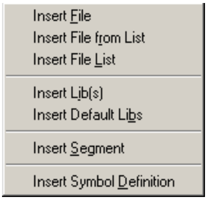


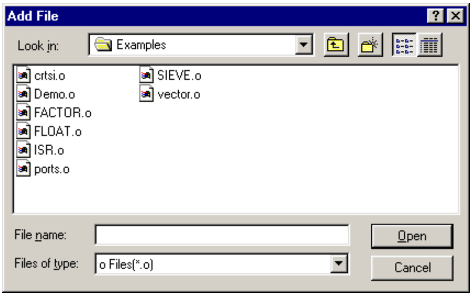
Fig. 6-8: Linker command file editing options

Insert File option

The **Insert File** option lets you insert the name of a single object file (**.o**) in the linker command file.

To insert an object file name at the current cursor position, right click and select **Insert File** from the pop-up menu.

When you select **Insert File**, IDEA looks for all object files with an **.o** extension in the default working directory (for example, IDEA12). The **Add File** dialog box that appears lists all files found.



Look in: Examples

crtsi.o
Demo.o
FACTOR.o
FLOAT.o
ISR.o
ports.o
SIEVE.o
vector.o

File name:

Files of type: o Files(*.o)

Fig. 6-9: Add File dialog box

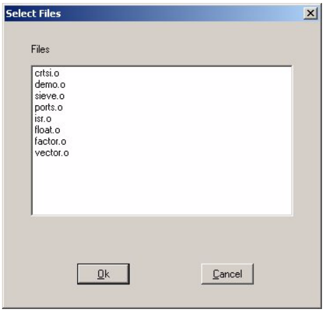
Select a file name and then select open to insert the file name at the current cursor position.

Insert File from List option

The **Insert File from List** option lets you insert the name of one or more object files (**.o**) in the linker command file.

To insert the object file name(s) at the current cursor position, right click and select **Insert File from List** from the pop-up menu.

When you select **Insert File from List**, IDEA looks for all object files with an **.o** extension in the default working directory. The **Select Files** dialog box that appears lists all files found.



crtsi.o
demo.o
sieve.o
ports.o
isr.o
float.o
factor.o
vector.o

Fig. 6-10: Select Files dialog box

Select a single file name by clicking on it.

Select a contiguous list of file names by clicking on the first name, holding down the **Shift** key, and then clicking on the last name.

Select multiple non-contiguous file names by clicking on the first name, holding down the **Ctrl** key, and then clicking on the names of other files in succession.

Insert File List option

The **Insert File List** option lets you insert the names of all object files with an **.o** extension in the default working directory.

To insert the object file names at the current cursor position, right click and select **Insert File List** from the pop-up menu.

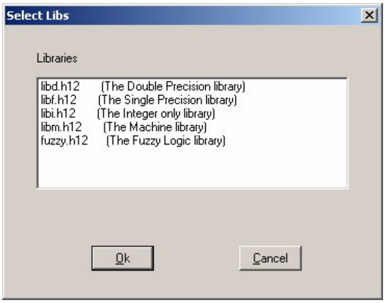
Insert Lib(s) option

The **Insert Lib(s)** option lets you insert the name of one or more library files (**.h***) in the linker command file.

To insert the library file name(s) at the current cursor position, right click and select **Insert Lib(s)** from the pop-up menu.

When you select **Insert Lib(s)**, IDEA looks for all library files with an **.h*** extension in the default libraries path(s). Libraries paths are specified in the **Link Configuration** dialog box. Refer to “*Libraries path option (-l)*” on page 146 for details.

The **Select Libs** dialog box lists all libraries found.



```
libd.h12     (The Double Precision library)
libf.h12     (The Single Precision library)
libi.h12     (The Integer only library)
libm.h12     (The Machine library)
fuzzy.h12    (The Fuzzy Logic library)
```

Fig. 6-11: Select Libs dialog box

Select a single library name by clicking on it. Select a contiguous list of library names by clicking on the first name, holding down the **Shift** key, and then clicking on the last name.

Select multiple non-contiguous library names by clicking on the first name, holding down the **Ctrl** key, and then clicking on other library names in succession.

Insert Default Libs option

The **Insert Default Libs** option lets you insert the names of the default libraries for your compiler.

The default libraries are:

libd.h* (Double-precision library)

libi.h* (Integer-only library)

libm.h* (Machine library)

To insert the default library names at the current cursor position, right click and select Insert **Default Libs** from the pop-up menu.

NOTE

Please refer to the Linker chapter in your compiler manual for a complete description of the default libraries.

Insert Segment option (+seg <segment> <options>)

The **Insert Segment** option lets you insert a segment in the linker command file. A segment is a logically unified block of memory in the executable image.

To insert a segment at the current cursor position, right click and select **Insert Segment** from the pop-up menu. The **Segment Definition** dialog box appears.

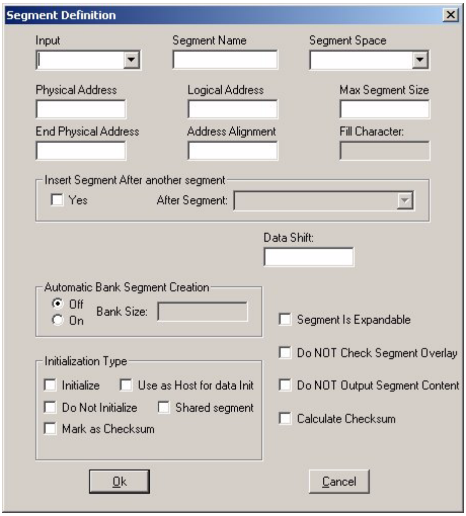


Fig. 6-12: Segment Definition dialog box

Input option

Specifies the segment type (for example, **.text**, **.const**, **.data**, etc.)

Segment Name option (-n)

Sets the output name of the segment to the value specified. Segment output names have at most fifteen characters; longer names are truncated.

Segment Space option (-s)

Defines a space name for the segment. This segment will be verified for overlapping only against segments defined with the same space name.

Physical Address option (-b)

Sets the physical start address of the segment to the value specified.

Logical Address option (-o)

Sets the logical start address of the segment to the value specified. The default is to set the logical address equal to the physical address.

Max Segment Size option (-m)

Sets the maximum size of the segment to the number of bytes specified. The default size is 65536 bytes.

Insert Segment after another segment option (-a)

Makes the current segment follow the segment specified. Options **-b** (**Physical Address**) and **-o** (**Logical Address**) cannot be specified if this option is specified.

Output Symbols Only option (-c)

Does not output any code/data for the segment.

Do NOT Check Segment Overlay option (-v)

Does not verify overlapping for the segment.

Automatic Bank Segment Creation option (-w*)

Activates automatic bank segment creation and sets the window size for banked applications.

Bank Number option (-p)

Defines the bank number of the segment. This information will be used in case of bank switching instead of the computation based on the **-b** (**Physical Address**) and **-bs** (**Window Shift**) values.

Shared segment option (-is)

Marks the segment as shared data.

Use as Host for data Init option (-it)

Uses the segment to host the descriptor and image copies of initialized data used for automatic data initialization.

Initialize option (-id)

Initializes the segment.

Do Not Initialize option (-ib)

Does not initialize the segment.

Insert Symbol Definition option (+def*)

The **Insert Symbol Definition** option lets you insert a symbol definition in the linker command file.

To insert a symbol definition at the current cursor position, right click and select **Insert Symbol Definition** from the pop-up menu. The **Symbol Definition** dialog box appears.

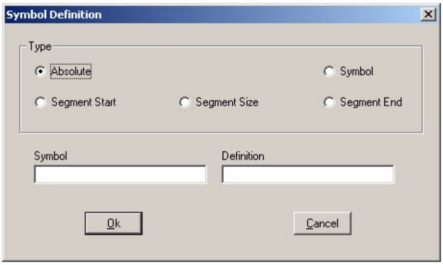


Fig. 6-13: Symbol Definition dialog box

Symbol Type option

Specifies the symbol type. The choices available are:

- **Absolute**
- **Segment End**
- **Symbol**

Symbol option

Specifies the linker-defined symbol.

Definition option

Specifies the symbol definition.

If the symbol type is **Segment End**, the choices are any defined segments. For example, **.text**.

If the symbol type is **Symbol**, the choices are any defined symbols. For example, **__memory**, or **__stack**.

Changing the Linker Command File

You can change the linker command file in one of two ways:

1) Double click on the **Tools** icon in the Project window to show the project tools. Then right click on the **Linker** icon and select **Change Command File**.

2) Select **Tools** from the Main menu. In the **Tool Browser**, right click on the **Linker** tool and select **Change Command File**.

The **Select Linker Command File** dialog box lets you specify a file name and path for the new linker command file.

After you select a new command file, the **Command File** check box is automatically checked in the **Link Configuration** dialog box, and the linker command file name and path are displayed.

Marking Files

To mark a single file for recompile/assemble without changing the time/date stamp of the file, double click on the **Files** icon in the Project window to show the project files. Then right click on the **File** icon for the appropriate file and select **Mark** from the pop-up menu. The color of the **File** icon changes from yellow to orange .

To mark all files for recompile/assemble without changing the time/date stamp of the files, right click on the **Project Name** icon in the Project window and select **Mark All** from the pop-up menu. The color of all the **File** icons changes from yellow to orange.

Touching Files

To mark a single file for recompile/assemble and update the time/date stamp of the file to current, double click on the **Files** icon in the Project window to show the project files. Then right click on the **File** icon for the appropriate file and select **Touch** from the pop-up menu. The color of the **File** icon changes from yellow to red .

To mark all files for recompile/assemble and update the time/date stamp of the files to current, right click on the **Project Name** icon in the Project window and select **Touch All** from the pop-up menu. The color of all the **File** icons changes from yellow to red.

Making a Project

The **Make** process first checks source file up-to-date status and dependencies. It then selectively compiles or assembles any out-of-date files and runs the **Linker**. You can selectively mark files for recompile/ assemble when a **Make** is executed using the **Mark**, **Mark All**, **Touch**, and **Touch All** options.

To summarize, you can run a project build in one of three ways.

1) Select the **Build Project** tool or the Rebuild Project tool on the Tool bar.

2) Select the **Build/Rebuild All** option from the Project menu or type **Alt+P+B**.

3) Right click on the **Project Name** icon in the Project window and select **Build** from the pop-up menu.

A **Build Project** box appears, showing the progress during the build. In addition, if you have selected **Show Sub Processes** from the **Options** drop-down menu, a MS-DOS window opens and shows all processes during the build.

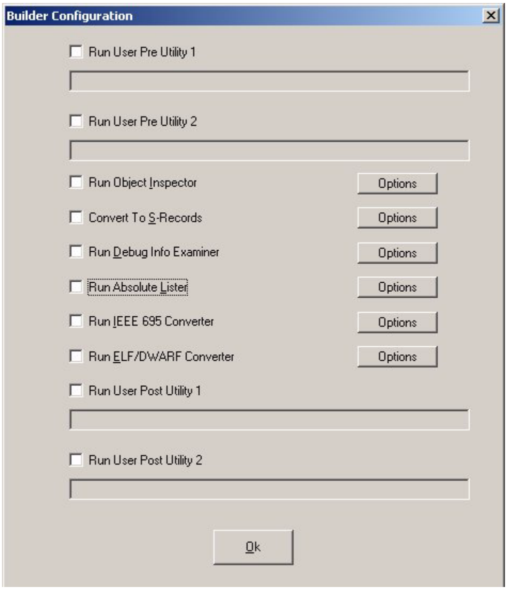
Building a Project

The **Build** process performs a **Make** as described in the preceding section and then runs any utilities selected in the **Builder Configuration** dialog box. You can selectively mark files for rebuilding using the **Mark**, **Mark All**, **Touch**, and **Touch All** options.

Specifying builder options.

Right click on the **Builder** icon in the Tools Pane of the Project Window and select **Options** (or double click on the **Builder** icon and right click on the **Options** icon).

The **Builder Configuration** dialog box appears.



Builder Configuration

☐ Run User Pre Utility 1

☐ Run User Pre Utility 2

☐ Run Object Inspector

☐ Convert To S-Records

☐ Run Debug Info Examiner

☐ Run Absolute Lister

☐ Run IEEE 695 Converter

☐ Run ELF/DWARF Converter

☐ Run User Post Utility 1

☐ Run User Post Utility 2

Fig. 6-14: Builder Configuration dialog box

The **Builder Configuration** dialog box contains check boxes that let you specify which builder utilities to run.

The following table describes the builder utilities.

Run User Pre Utility 1	Runs the specified user utility before the make. You can specify a path and filename for the utility you wish to run.
Run User Pre Utility 2	Runs the specified user utility before the but after the Utility above if specified. You can specify a path and filename for the utility you wish to run.
Run Object Inspector	Runs the ***cobj*** utility to examine object modules. If you select **Run Object Inspector** and then click on the **Options** button, the **Options** dialog box appears. Refer to "*Object Inspector utility*" on page 162 for details.
Convert to S-Records	Runs the ***chex*** utility to translate object module format to hexadecimal format. If you select **Convert to S-Records** and then click on the **Options** button, the **CHEX Configuration** dialog box appears. Refer to "*Hex Converter utility*" on page 164 for details.
Run Debug Info Examiner	Runs the ***cprd*** utility to print debugging information about functions and data objects. If you select **Run Debug Info Examiner** and then click on the **Options** button, the **CPRD Configuration** dialog box appears. Refer to "*Debug Info Examiner utility*" on page 167 for details.
Run Absolute Lister	Runs the ***clabs*** utility to generate absolute listings. If you select **Run Absolute Lister** and then click on the **Options** button, the **CLABS Configuration** dialog box appears. Refer to "*Absolute Lister utility*" on page 169 for details.
Run IEEE 695 Converter	Runs the ***cv695*** utility to generate IEEE695 format. If you select **Run IEEE 695 Converter** and then click on the **Options** button, the **CV695 Configuration** dialog box appears. Refer to "*IEEE 695 Converter utility*" on page 171 for details.

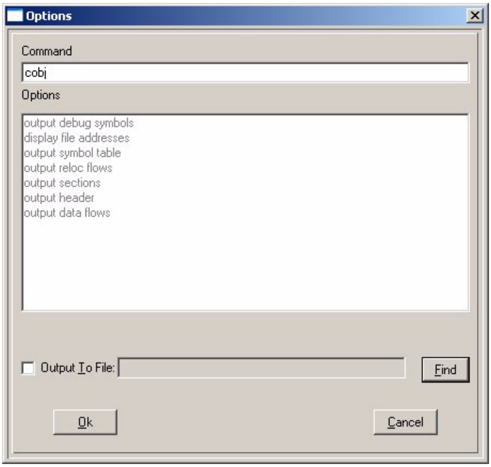
Run ELF/Dwarf Converter	Runs the ***cvdwarf*** **utility** to generate ELF/Dwarf format. If you select **Run ELF/Dwarf Converter** and then click on the **Options** button, the **CVDWARF Configuration** dialog box appears. Refer to "*CVDWARF Converter utility*" on page 173 for details.
Run User Post Utility 1	Runs the specified user utility after the make. You can specify a path and filename for the utility you wish to run.
Run User Post Utility 2	Runs the specified user utility after the make and after the above utility if specified. You can specify a path and filename for the utility you wish to run.

Object Inspector utility

If you select **Run Object Inspector** and then click on the **Options** button, the **Options** dialog box appears.

Alternatively, you can right click on the **Object Inspector** icon and select **Options** to open the **Options** dialog box. You can also double-click on the **Object Inspector** icon to display the **Options** icon and then right-click on it to display the **Options** dialog box.

The **Options** dialog box lets you specify options for the ***cobj*** utility, which lets you inspect relocatable object files or executable output by the assembler or linker. The ***cobj*** utility can be used to check the size and configuration of relocatable object files or to output information from their symbol tables.



Command: cobj

Options:

- output debug symbols
- display file addresses
- output symbol table
- output reloc flows
- output sections
- output header
- output data flows

☐ Output To File:

Fig. 6-15: Object Inspector Options dialog box

Command options

Selected options are displayed in bold and unselected options are greyed out. To select an option simply click on the option description and it will be added to the command line. To deselect an option click on it again.

The following Table lists the available *cobj* utility options. For additional *cobj* utility options, consult you compiler documentation.

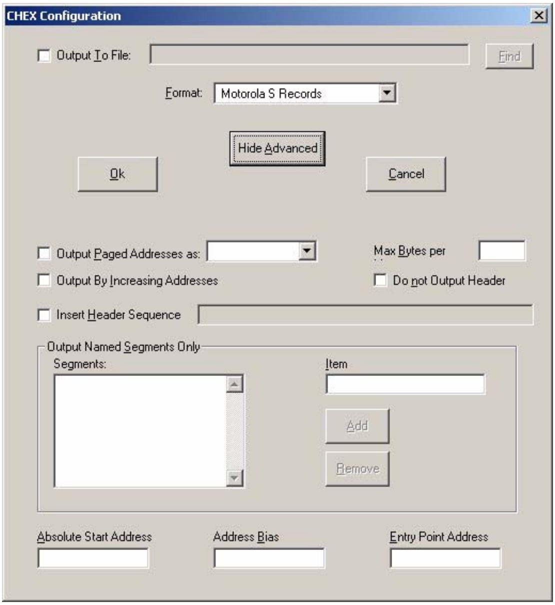
Output debug symbols	(**-x** option) - displays the debug symbol table.
Display file addresses	(**-v** option) - displays seek addresses inside the object file.
Output symbol table	(**-s** option) - displays the symbol table.
Output relocation flows	(**-r** option) - outputs the relocation part of each section in symbolic form.
Output sections	(**-n** option) - displays the name, size and attribute of each section.
Output header	(**-h** option) - displays all the fields of the object file header
Output data flows	(**-d** option) - outputs the data part of each section in hexadecimal format.
Output file	You can specify a path and file name to receive the **Object Inspector** output. This file may be in relocatable format or executable format.

Hex Converter utility

If you select **Convert to S-Records** and then click on the **Options** button, the **CHEX Configuration** dialog box appears.

Alternatively, you can right click on the **Hex Converter** icon CHEX and select **Options** to open the **CHEX Configuration** dialog box. You can also double-click on the **Hex Converter** icon to display the **Options** icon ☰ and then right-click on it to display the **CHEX Configuration** dialog box.

The **CHEX Configuration** dialog box lets you specify options for the *chex* utility, which translates executable images produced by the *clnk* linker to one of several hexadecimal interchange formats.



Format: Motorola S Records

Fig. 6-16: CHEX Configuration dialog box

Motorola S Records	(**-fm option**) - produces S1 and S2 records as needed
Motorola S2 Records	(**-f2 option**) - produces S2 records only.
Motorola S3 Records	(**-f3 option**) - produces S3 records only.
Intel Hex	(**-fi option**) - produces Intel hex format
Absolute Start Address	(**-a## option**): the output address of the first byte. The argument file is considered as a pure binary file
Address Bias	(**-b## option**): subtract from any address before output.
Entry point	(**-e## option**): define ## as the entry point address encoded in the dedicated record of the output format, if available.
Max Bytes per line	(**-m option**): maximum data bytes per line. The default is 32 bytes per line
Insert Header Sequence	(**+h* option**) - if such a sequence exists for the selected format
Do not Output Header	(**-h option**) - if such a sequence exists for the selected format
Output Paged Addresses	(**-p option**) - using a paged format <page_number><logical_address>, instead of the default format <physical>
Output Paged Addresses	(**-pn option**) - behaves as **-p** but only when logical address is inside the banked area (use for Noral debugger
Output Paged Addresses	(**-pp option**) - behaves as **-p** but uses paged addresses for all banked segment, mapped or unmapped (used for Promic tools)

Output named segments only	(**-n** option). Up to twenty different named segments can be specified. To add a named segment to the **Segments** field, enter the named segment in the **Item** field and click on the **Add** button. To remove a named segment from the **Segments** field, select the segment and click on the **Remove** button.
Output by Increasing Addresses	(**-s option**)? Sort the output addresses in increasing order
Output to File	(**-o option**): the default is STDOUT
No Output segment	(**-x* option**): do not output segment whose name is equal to string *.

Debug Info Examiner utility

If you select **Run Debug Info Examiner** and click on **Options**, the **CPRD Configuration** dialog box appears.

Alternatively, you can right click on the **Debug Info Examiner** icon CPRD and select **Options** to open the **CPRD Configuration** dialog box. You can also double-click on the **Debug Info Examiner** icon to display the **Options** icon ☰ and then right-click on it to display the **CPRD Configuration** dialog box.

The **CPRD Configuration** dialog box lets you specify options for the *cprd* utility, which extracts and prints information about functions and data objects from an object module or executable image that has been compiled with the **+debug** option.

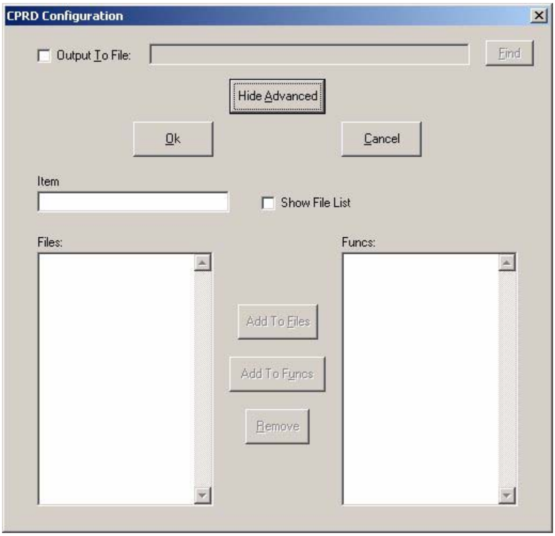


Fig. 6-17: CPRD Configuration dialog box

The **CPRD Configuration** dialog box lets you build a list of files and functions for debugging purposes. Enter a file or function name in the **Item** field, and then click on **Add to Files** to add the item to the **Files** list or **Add to Funcs** to add the item to the **Functions** list.

If you check the **Show File List** check box, the **Item** field changes to a **File List** field, with a list of the files in the project directory. Select a file from the list and click on the **Add to Files** button to add it to the **Files** list.

To remove an item from either list, select the item and then click on the **Remove** button.

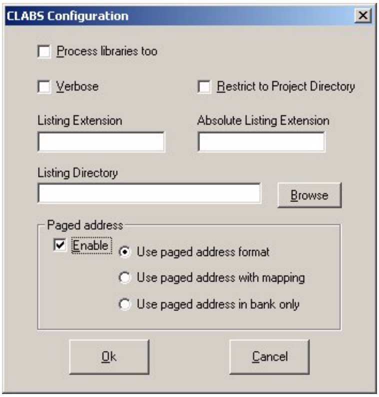
Print file debugging information	(**-fl** option) - each file in the **Files** list is processed with the **-fl** option, which prints debugging information about the file. By default, *cprd* prints debugging information on all C source files.
Print function debugging information	(**-fc** option) - each function in the **Functions** list is processed with the **-fc** option, which prints information about the function only. By default, *cprd* prints debugging information on all functions in a file.
Print debugging information to file	(**-o** option) - you can also specify a path and file name to receive the debugger output. By default, *cprd* writes debugging information to the terminal screen.

Absolute Lister utility

If you select **Run Absolute Lister** and then click on the **Options** button, the **CLABS Configuration** dialog box appears.

Alternatively, you can right click on the **Absolute Lister** icon and select **Options** to open the **CLABS Configuration** dialog box. You can also double-click on the **Absolute Lister** icon to display the **Options** icon and then right-click on it to display the **CLABS Configuration** dialog box.

The **CLABS Configuration** dialog box lets you specify options for the *clabs* utility, which processes relocatable C and Assembly listing files with the associated executable file to produce absolute listings with updated code and address values.



CLABS Configuration

☐ Process libraries too

☐ Verbose ☐ Restrict to Project Directory

Listing Extension

Absolute Listing Extension

Listing Directory

Paged address

☑ Enable

● Use paged address format

○ Use paged address with mapping

○ Use paged address in bank only

Fig. 6-18: CLABS Configuration dialog box

Process Libraries too	(**-a** option) **-** process also files located in libraries.
Verbose	(**-v** option) **-** the name of each module of the application is output to STDOUT.

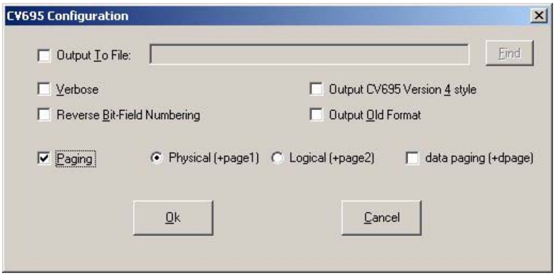
Restrict to Project Directory	**(-l** option**) -** process files in the project directory only. The default is to process all files of the application.
Listing Extension	**(-r** option**) -** specify the input file extension. The default is "**.ls**".
Absolute Listing Extension	**(-s** option**) -** specify the output file extension. The default id "**.la**"
Paged Address	(**-p** option) - output addresses of banked segment.

IEEE 695 Converter utility

If you select **Run IEEE 695 Converter** and then click on the **Options** button, the **CV695 Configuration** dialog box appears.

Alternatively, you can right click on the **IEEE695 Converter** icon IEEE and select **Options** to open the **CV695 Configuration** dialog box. You can also double-click on the **IEEE695 Converter** icon to display the **Options** icon and then right-click on it to display the **CV695 Configuration** dialog box.

The **CV695 Configuration** dialog box lets you specify options for the *cv695* utility, which converts a file produced by the linker into IEEE695 format.



CV695 Configuration

□ Output To File:

□ Verbose

□ Output CV695 Version 4 style

□ Reverse Bit-Field Numbering

□ Output Old Format

☑ Paging

● Physical (+page1)

○ Logical (+page2)

□ data paging (+dpage)

Fig. 6-19: CV695 Configuration dialog box

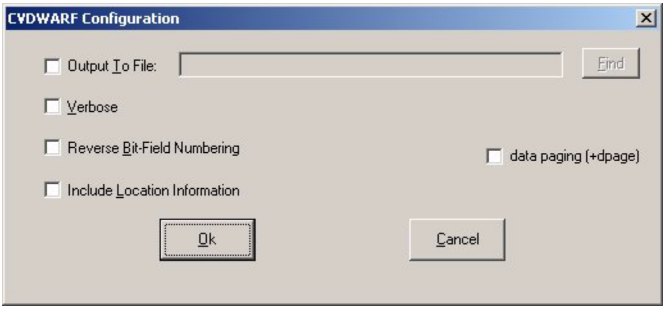
Verbose	**(-v option) -** the ***cv695*** utility displays information about its activity.
Reverse Bit-Field Numbering	**(-rb option) -** reverses bitfield from left to right.
Output to File	(**-o option**) - you can specify a path and file name to receive the *cv695* utility output. By default, the *cv695* utility outputs to the file whose name is obtained from the input file by replacing the filename extension with “**.695**”.

Paging	(**+page# option**) - this option is currently meaningful for the MC68HC12 only. This option specifies the address format for bank-switched code. If you check the **Paging** check box, three options appear to the right: **Physical (+page1)** - the application is banked and the *cv695* utility outputs physical addresses. This is the default if **Paging** is checked. **Logical (+page2)** - the application is banked and the *cv695* utility outputs addresses in paged mode: <page><offset_in_page>. This is equivalent to the old **+paged** flag. **data paging (+dpage)** - the application uses data paging.

CVDWARF Converter utility

The ***cvdwarf* utility** converts a file produced by the linker into ELF/ Dwarf format.

Right click on the **CVDWARF Converter** icon and select **Options** to open the **CVDWARF Configuration** dialog box. You can also double-click on the **CVDWARF Converter** icon to display the **Options** icon and then right-click on it to display the **CVDWARF Configuration** dialog box.



CVDWARF Configuration

□ Output To File:

□ Verbose

□ Reverse Bit-Field Numbering

□ data paging (+dpage)

□ Include Location Information

Fig. 6-20: CVDWARF Configuration dialog box

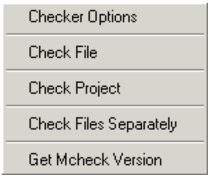
The *cvdwarf* utility options are described in the following table:

Verbose	(**-v** option) **-** the ***cvdwarf*** utility displays information about its activity.
Paging	(**+page#** option) - this option is currently meaningful for the **HC12/HCS12** only. This option specifies the address format for bank-switched code. If you check the **Paging** check box, three options appear to the right: **Physical (+page1)** - the application is banked and the *cv695* utility outputs physical addresses. This is the default if **Paging** is checked. **Logical (+page2)** - the application is banked and the *cv695* utility outputs addresses in paged mode: `<page><offset_in_page>`. This is equivalent to the old **+paged** flag. **data paging (+dpage)** - the application uses data paging.
Reverse BitField Numbering	(**-rb** option) **-** reverses bitfield from left to right.
Include location information	(**-loc** option) location lists are used in place of location expressions whenever the object whose location is being described can change location during its lifetime. THIS POSSIBILITY IS NOT SUPPORTED BY ALL DEBUGGERS.
Output to File	(**-o** option) - you can specify a path and file name to receive the *cvdwarf* utility output. By default, the *cvdwarf* utility outputs to the file whose name is obtained from the input file by replacing the filename extension with “**.elf**”.

MISRA Compliance Plug-in

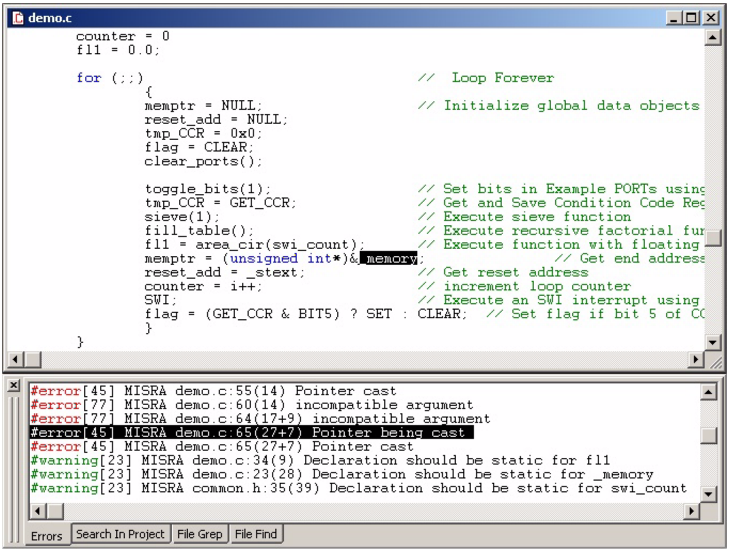
The Cosmic MISRA Checker Plug-in for IDEA aids in the production or well structured C language code using the guidelines prescribed by the Motor Industry Software Reliability Association (MISRA) document entitled “Guidelines for the Use Of the C Language In Vehicle Based Software”. As some rules cannot be checked statically, the Cosmic MISRA Checker is designed to be used in conjunction with the MISRA guidelines document.

Cosmic IDEA with MISRA plug-in statically scans a project for MISRA rule compatability based on the options selected. Select **Checker Options** from the **MISRA** menu to configure which rules are to be checked. The checker options should list the different rules you can choose from. If this menu is grayed out then the plug-in is not installed properly.



Once you select the appropriate options, you can instruct IDEA to run the MISRA checker by selecting **Check Project** in the MISRA drop down menu to check the entire project including rules with file interdependency. Select **Check Files Separately** to check the entire project one file at a time ignoring rules with file interdependencies or Select **Check File** to check the file in the active edit window.

Checking files individually or on the whole project is also accessible through the **Project** menu.



```
        counter = 0
        fl1 = 0.0;

        for (;;)                                // Loop Forever
                {
                memptr = NULL;                  // Initialize global data objects
                reset_add = NULL;
                tmp_CCR = 0x0;
                flag = CLEAR;
                clear_ports();

                toggle_bits(1);                 // Set bits in Example PORTs using
                tmp_CCR = GET_CCR;              // Get and Save Condition Code Reg
                sieve(1);                       // Execute sieve function
                fill_table();                   // Execute recursive factorial fur
                fl1 = area_cir(swi_count);      // Execute function with floating
                memptr = (unsigned int*)&_memory;               // Get end address
                reset_add = _stext;             // Get reset address
                counter = i++;                  // increment loop counter
                SWI;                            // Execute an SWI interrupt using
                flag = (GET_CCR & BIT5) ? SET : CLEAR;  // Set flag if bit 5 of CC
                }
        }
```

```
#error[45] MISRA demo.c:55(14) Pointer cast
#error[77] MISRA demo.c:60(14) incompatible argument
#error[77] MISRA demo.c:64(17+9) incompatible argument
#error[45] MISRA demo.c:65(27+7) Pointer being cast
#error[45] MISRA demo.c:65(27+7) Pointer cast
#warning[23] MISRA demo.c:34(9) Declaration should be static for fl1
#warning[23] MISRA demo.c:23(28) Declaration should be static for _memory
#warning[23] MISRA common.h:35(39) Declaration should be static for swi_count
```

Index