CODECRAFT

Welcome to Codecraft, the powerful Integrated Development Environment (IDE) for developing software natively on the Amiga. Codecraft makes it easy for you as a developer to write code, then build, execute and debug your resulting program. And everything is at your fingertips in one unified user interface.

- Shows your project tree as a hierarchical list of source files
- Integrates the AmigaOS 3.2.2 TextEdit for easy modern text editing
- Builds the project tree invoking your build system
- Runs your resulting program
- Does source level debugging with breakpoints and inspection of variables

About this User's Guide

This User's guide has the following chapters:

Getting Started

Creating a ProjectTree

Building and Running your Program

Build Configurations

Using the Debugger

Getting Started

In this chapter you will learn how to install and start Codecraft for the first time. You will also become familiar with the user interface and open an example ProjectTree.

Installing Codecraft

1. Boot your Amiga from your hard disk and wait for Workbench to load.

2. Insert the Codecraft disk.

3. Double-click on the disk icon.

4. Next double-click on the installer icon to start the installer.

5. Follow the on screen instructions until Codecraft is installed.

Besides installing Codecraft you will need a working compiler and build system of your choice.

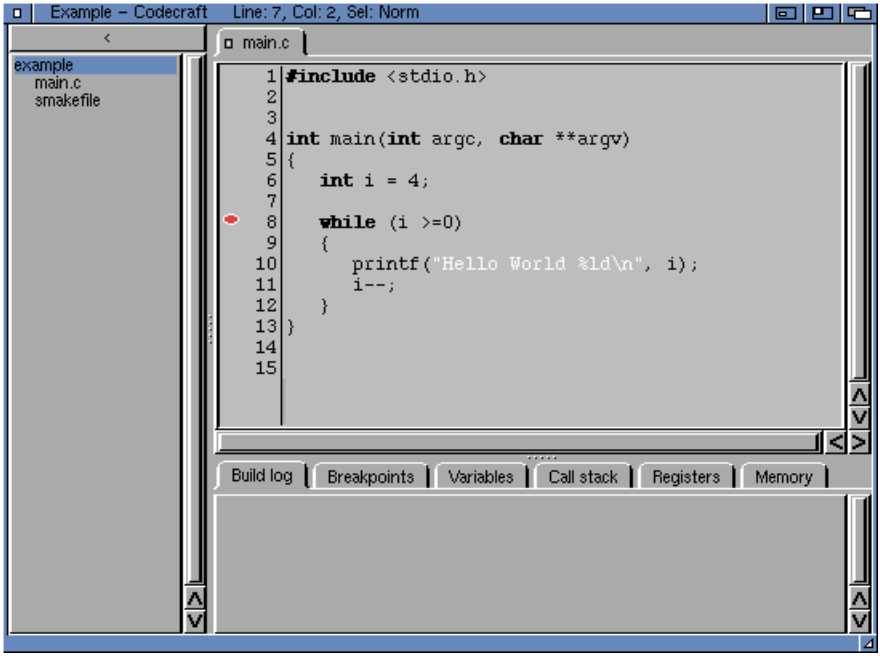
Starting Codecraft

With the installation done you will find your Codecraft at the location you have chosen. Here you will find the program icon that you need to double click to start Codecraft.

The icon on the right shows the optional Codecraft glow icon - available on the Codecraft *Install Disk*.

Getting Familiar with the Codecraft Window

Codecraft opens up in a window. Codecraft is actually the same as TextEdit from AmigaOS 3.2 except Codecraft also adds several other GUI elements to the window. Once you have loaded a ProjectTree it will look like this:



The window is divided into 4 main parts:

1. The **ProjectTree Browser** shows a list of files in your ProjectTree drawer. It also shows sub drawers in a hierarchical

tree. It is more like a file browser, and is not integrated to your build system.

2. The **Sourcefile Editor** is where you edit your sourcefiles. If you have used TextEdit (part of AmigaOS 3.2) then it should feel quite familiar.

3. The **Auxillary Browsers** provides a lot of extra info. The Build log browser is useful when you are building. The Breakpoints, Variable and Call stack browsers are useful when debugging. These browsers will be covered in more detail in their respective chapters.

4. The **Menus** should also be halfway familiar if you have used TextEdit before. The Project, Edit and Filetype menus all relate to the individual files and not the ProjectTree. The ProjectTree, Build and Debug menus are specific to Codecraft.

Opening the Example ProjectTree

Let us open an example ProjectTree. In the ProjectTree menu select the menu item named "Open". It will bring up a standard file requester where you should navigate to the place you have installed Codecraft There should be a drawer called "Example". Go into that drawer and then open the file called "example.ProjectTree".

As a result the ProjectTree browser will become populated and the files you had open last time will be reopened. Breakpoints you had set will be remembered. The example ProjectTree, as installed, will open as if you were in the middle of working with and will look like the above screenshot.

Creating a ProjectTree

In this chapter you will learn how to create a new ProjectTree. Because Codecraft offers a lot of flexibility you need to supply a number of properties before the ProjectTree is useful. But it is easily done in a few minutes.

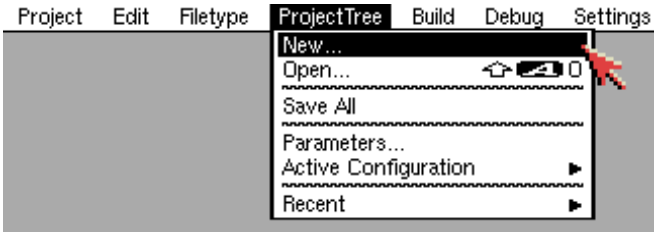
However, remember that Codecraft is not a build system nor a compiler. Please ensure you have installed those before proceeding.

What is a ProjectTree?

Codecraft uses so called ProjectTrees to store and organize the structure and the settings of a project. The creation of a ProjectTree is an essential step when setting up a new development project. Later on, when working, you just open the already existing ProjectTrees.

A ProjectTree is simply stored as a file (with .projecttree extension), and therefore it can be managed by your source control system (e.g. git, subversion, cvs), like any other file of your development project.

The ProjectTree Menu

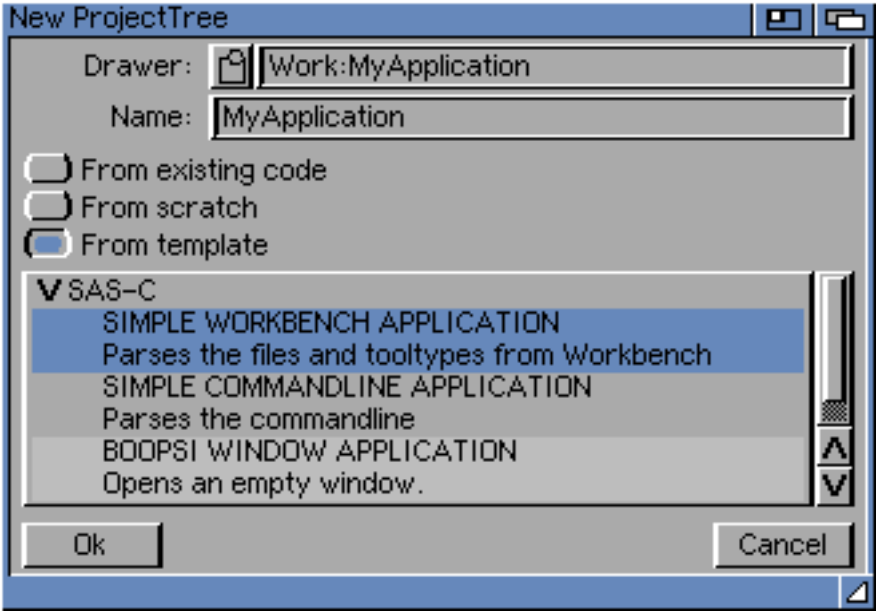


Creating a ProjectTree

When creating a ProjectTree it can be either a completely new project or one based on existing source code already in a directory somewhere.

The remainder of this chapter will setup a C language project, but you can set it up for any language you like. You can even use if for non-coding projects like documentation writing where the build system converts say plain text to AmigaGuide or something similar.

To create a new ProjectTree, choose *New* from the *ProjectTree* menu. This will open the *New ProjectTree* requester.



With this requester, you setup how to create the ProjectTree. That is:

- **Drawer** The drawer of the ProjectTree. This is either your existing project drawer or a new drawer if you are creating a new ProjectTree from scratch or from template. If the drawer doesn't exist it will be created.

- **Name** The name of the ProjectTree. This is the name of the project file in case you want more than one Projecttree inside the same drawer. But often you will simply name it the same as the drawer.

- **Radio button *From existing code*** The ProjectTree will use the existing files already in the drawer.

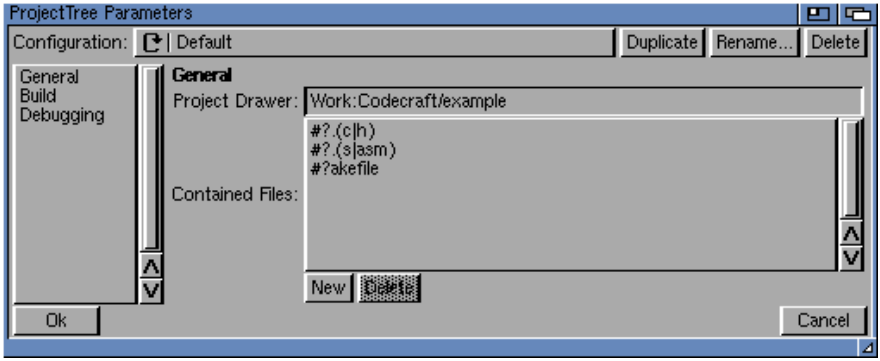
- **Radio button *From Scratch*** The ProjectTree is almost completely empty.

- **Radio button *From Template*** The ProjectTree is populated with the template that you can now select in the list below.

ProjectTree Parameters Requester

To complete the setup of a new ProjectTree, after the *New ProjectTree* requester was confirmed Codecraft presents automatically opens the *ProjectTree Parameters* requester.

The general parameters are described here, but the build and debugging parameters are described in their respective chapters.



- **Project Drawer** The drawer of the project, like given in the *New ProjectTree* requester. This is not a property, but simply information of where the ProjectTree is located.

- **Contained Files** A list of AmigaDos wildcards (e.g. #?), that Codecraft will use to show files in the ProjectTree browser. You can think of it as filtered view of your drawer structure. Drawers are only shown if there are files inside that are visible according to the wildcards.

Saving the ProjectTree

The ProjectTree is automatically saved whenever you start a build, switch to another ProjectTree or quit Codecraft.

So, in general, you don't have to save, but there is a *Save All* menu item in the *ProjectTree* menu that will explicitly save it and all the files.

Opening a ProjectTree

To open an existing ProjectTree you can either choose *Open…* in the *Project Tree* menu, or use the *Recent* menu to reopen a previously opened ProjectTree.

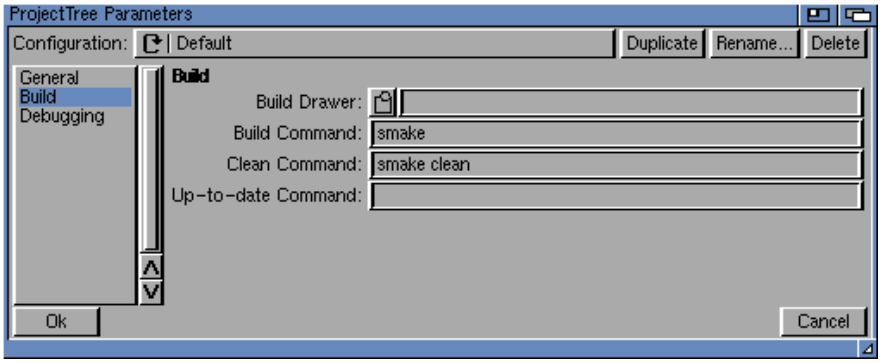
Building and Running your Program

In this chapter you will learn how to build your program. If your build system produces warnings and errors you will learn how to go through each of those. Finally you will learn how to run your program.

It is actually pretty simple.

The Related ProjectTree Parameters

In the ProjectTree parameters requester you will find several settings relevant for building and running your program.



- **Build Drawer** Before building the current directory will be changed to this location. You can specify an absolute or relative path, or simply leave it blank to stay in the project drawer

- **Build Command** The command used to build your project. It is typically *make* or *smake* or something similar.

- **Clean Command** The command used to clean your project tree from build files. It is typically *make clean* or similar.

- **Up-to-date Command** If supported by your build system, you can enter a command or script that is executed by Codecraft before running your application. The command should check whether the project is up-to-date or needs a build. A return value of 0 (zero) means that the project is current, any other return value indicates that a build is necessary. In that case, Codecraft will show a corresponding requestor.

The Build Menu

The build menu is where you actually start the build.

- **Build** This starts you build command as specified in ProjectTree parameters. First the current drawer is changed to the project drawer, and from there the current drawer is changed to your build drawer. Build drawer can be empty, relative or absolute.

- **Clean** This starts the clean command. The current drawer is set up like the *Build* menu item above.

- **Rebuild** This menu item simply executes *Build* and *Clean* menu items in succession.

- **Previous Message** Jumps to the previous message in the build log.

- **Next Message** Jumps to the next message in the build log.

Viewing the Build Log and Jumping to Errors

When you have started you build the output from the build system will start to appear in the build log. The build log is automatically shown, so you should soon see the messages scroll by.

Limitation: The detection of error and warning messages, is highly dependent on the compiler. For now we can only jump to messages from SAS/C, VBCC and E-VO, but if you have examples from other compilers get in touch, and we will try and add support.

If you double click on a line with warning or error, Codecraft takes you automatically to the corresponding file and line of code.

The menu items *Previous Message* and *Next Message* along with their shortcuts (Shift) F8 will also bring you to the file and line in question.

Running your Application

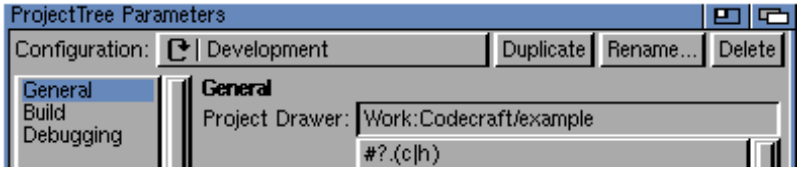
There are either two options in the *Debug* Menu to run your application:

Run In Debugger
Starts your application for debugging.

Run Without Debugger
Starts your application as if it would be started from Workbench or CLI.

Build Configurations

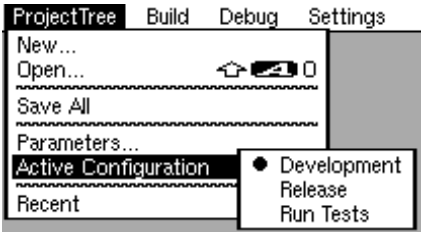
With *Build Configurations*, you can switch fast between different build parameters. To setup multiple Build Configurations of your ProjectTree, access the Build Configurations in the top of the ProjectTree Parameters Dialog.



- **Duplicate** Creates a copy of the current Build Configuration.
- **Rename** Lets you rename the current Build Configuration.
- **Delete** Deletes the current Build Configuration.

It is not possible to have no Build Configuration at all. When a new ProjectTree is created, Codecraft automatically creates a *Default* Build Configuration. Also, you can not delete the last Build Configuration. Build Configurations affect all the ProjectTree Parameters in the *Build* and *Debugging* sections. To maintain configuration-specific parameters, simply select the appropriate configuration and enter parameters for *Build* and *Debugging*.

Switching between Build Configurations



To switch between Build Configurations, use the *Active Configuration* submenu in the ProjectTree Menu.

Using the Debugger

Codecraft comes with a built-in source level debugger. A debugger is one of the most essential tools when making your program bug free.

The debugger is capable of stopping execution at breakpoints.

Stopping a Program is Impossible on the Amiga

It is important that the user manually stops a running program in the debugger. The Amiga is limited in what we can do here, so for now at least it is better to stop the program gracefully.

The reason is that the Amiga doesn't have resource tracking, so while we can stop the program from actively running we cannot safely free memory allocated by it or close its windows etc, etc.

Limited debugging with some compileres

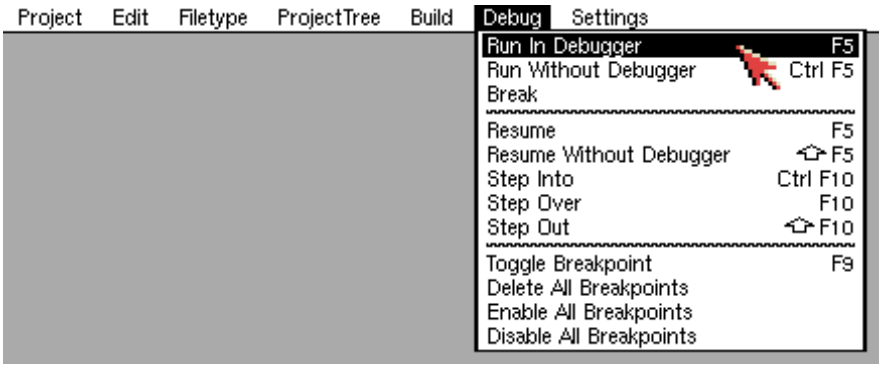
For now the debugger can only show values for programs compiled with: - **SAS/C** 6.58 and with debug=symbol. - **E-VO**

It is intended to expand this functionality to more compilers and languages in the future.

Any executable that has debug LINE hunks can be debugged using breakpoints and stepping. Many compilers produces this already. But you will not be able to see variables and their values.

The Debug Menu

The debug menu is where you run your app and control **Codecrafts** integrated debugger.



- **Run In Debugger** Starts your application for debugging by loading the debug information and prepares the breakpoints. Execution will break, as soon as a breakpoint is hit.

- **Run Without Debugger** Simply starts your application as if it would be started from Workbench or CLI. The debug information of you application is not loaded and breakpoints are ignored.

- **Break** Breaks the current execution immediately.

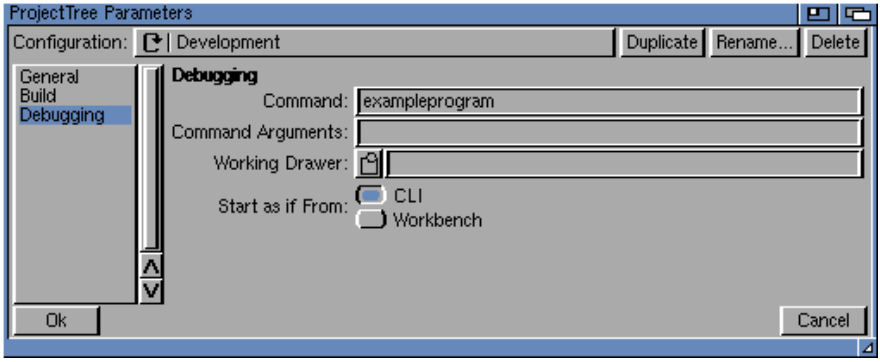
- **Resume** Continues the execution of your application when a breakpoint was hit or the execution was stopped using *Break*.

- **Resume Without Debugger** Continues the execution of your application by temporarily disabling all breakpoints. This allows to gracefully run the application to end, without further interruption by breakpoints. The breakpoints are reenabled on the next debugger run. Because of the shared memory architecture of the Amiga, running debugging sessions cannot be terminated immediately without endangering the stability of the system. Therefore, it is necessary to stop running debugging sessions by gracefully exiting the debugged

application. And for doing so, *Resume Without Debugger* is very convenient.

- **Step Into** Executes the next program line. Steps into functions, if the program line to execute is a function call.
- **Step Over** Executes the next program line, but steps over function calls by executing the whole function.
- **Step Return** Continues execution until the current function is exited, then breaks the execution.
- **Toggle Breakpoint** Sets or removes a breakpoint on the current position of the cursor, or on the nearest code line possible.
- **Delete All Breakpoints** All breakpoints are removed immediately.
- **Disable All Breakpoints** All breakpoint are disabled - i.e. they stay in place but the debugger does not stop when a breakpoint is hit.

The Related ProjectTree Parameters



- **Command** The name of your executable. You can specify either a path relative to you *build* drawer, or an absolute path.

- **Command Arguments** The arguments to supply to your program.

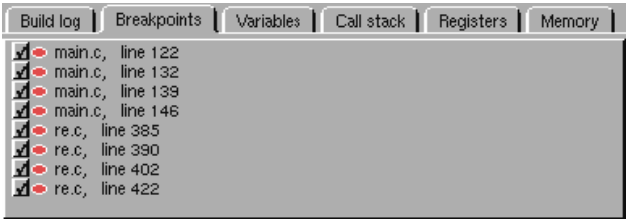
- **Working Drawer** The drawer where the program is running. You maybe have some test files here etc.

- **Start as if From** Radio buttons to choose whether you application is started as CLI or Workbench application. When from *Workbench* is selected, your application gets send the usual WBStartup message.

In general, Codecraft runs you application with a Stack size of 100000.

The Breakpoints Browser

The Break Points Browser is located in the tab **Breakpoints**, below the code editor in the Codecraft window.

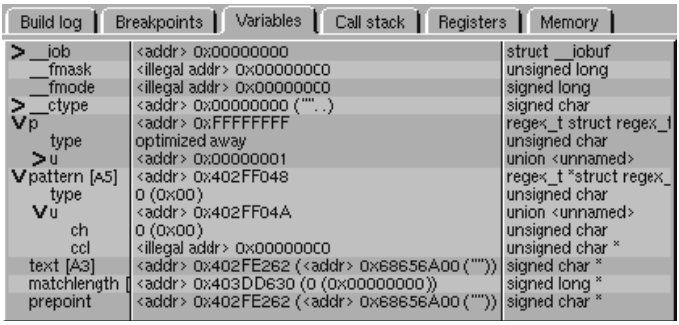


The Break Point Browser lists all breakpoints in the current ProjectTree by displaying the file and line the breakpoint is set. By double clicking an entry, Codecraft takes you to the file and line of the breakpoint. Further, by checking/unchecking the checkbox, you can enable/disable the corresponding breakpoint.

The Debugger Variables Browser

When you run your application in debug mode (see Using the Debugger) and a breakpoint is hit, the application execution breaks. In this state you can watch different information of your app.

One of this information is the current values of the variables of your application. The variables are displayed by switching to the ***Variables*** tab in bottom tab bar in the Codecraft user interface:



The variables and their values are displayed in three columns:

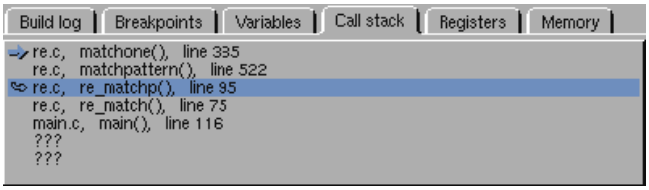
First Column The first column shows the name of the variable and some additional information. If the variable is a complex element like an *Array* or *Struct* for example, the variable can be expanded and nested elements are displayed. If a variable is stored in a register, the name of the register is displayed.

Second Column The second column shows the value of the variable. The display-format of the value is chosen automatically by Codecraft, depending on the variable type.

Third Column The third column shows the datatype of the variable. Datatype names are shown as used in the code. This means that typedefs are displayed as is and are not substituted by their basic type names.

The Debugger Call Stack

When your application is stopped by hitting a breakpoint, you can inspect the call hierarchy of your application using the ***Call stack*** tab, also located in the bottom tab bar.

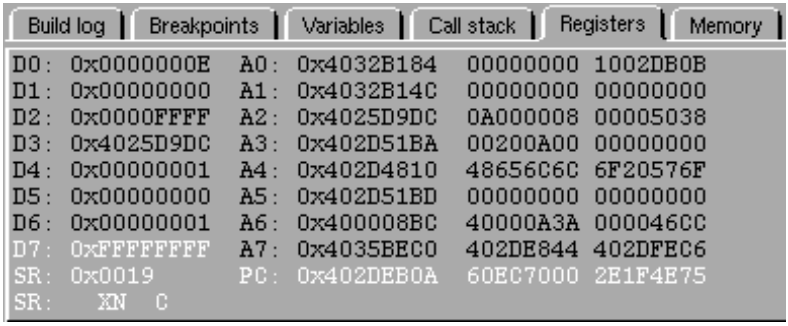


The call stack shows the source file, name of the function and line number your application has called to reach the current breakpoint. The blue arrow on the first entry of the call stack indicates the position of the current program execution.

When you double click on the other entries in the call stack Codecraft navigates you to the corresponding line of code. At the same time the entry you double clicked will get a blue curved arrow, and the variable browser is updated to reflect the variables at that place in the code.

Registers View

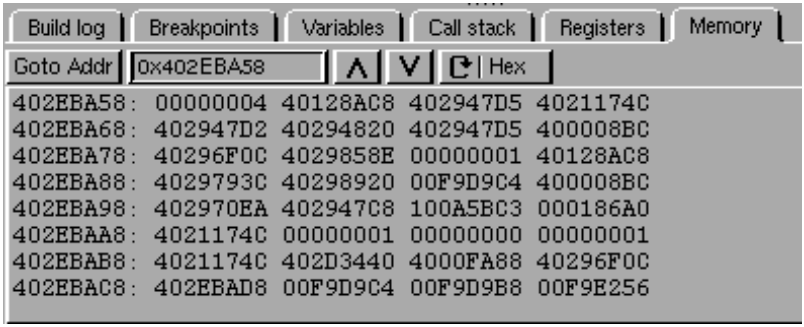
The Register View displays the current values of the CPU registers during debugging.



It shows the names and values of the data registers D0-D7 and A0-A7 for addresses, along with the program counter (PC) and status register (SR). By examining these registers, you can monitor changes, diagnose issues, and verify instruction execution.

Memory Browser

The Memory Browser View allows you to inspect memory contents during debugging.

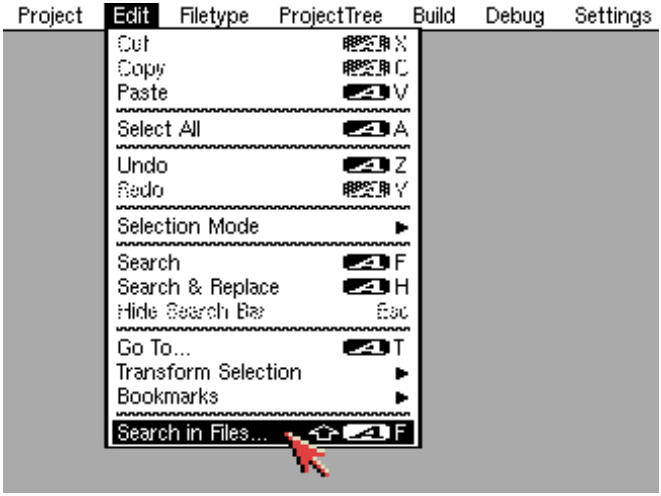


This view helps diagnose issues by providing a detailed look at how data is stored and manipulated. To display a memory area, you can either type the memory address to display into the *Goto Addr* field, or you can double click a variable value in the *Debugger Variable Browser* to display directly to the memory of the variable.

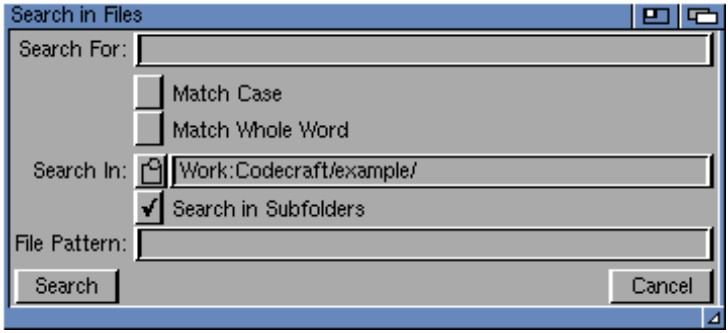
Further, you can switch the display of the memory data from *Hex* to *ASCII* to match the kind of data you want to inspect.

Search in Files

Codecraft extends the editor functions of TextEdit. One of the extensions is *Search in Files*. Searching in files can be accessed via the **Edit** menu.

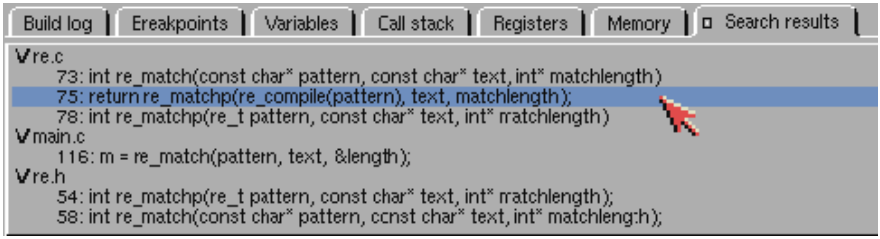


In the "Search in Files" dialog, enter the search criteria you want to search for and press **Search** to start the search.



- **Search For** The text you want to search for.
- **Match Case** When checked, the search is case sensitive.
- **Match Whole Word** When checked, the search finds only exact matches of the whole word.
- **Search In** The folder within which to search.
- **Search in Subfolders** When checked, the text will be searched in all subdirectories of the directory specified by _"Search In_".
- **File Pattern** By entering a *File Pattern*, you specify the types of files to include in the search. The regular *Amiga* style wildcards can be used - e.g. '#?.h' would search in header files only.

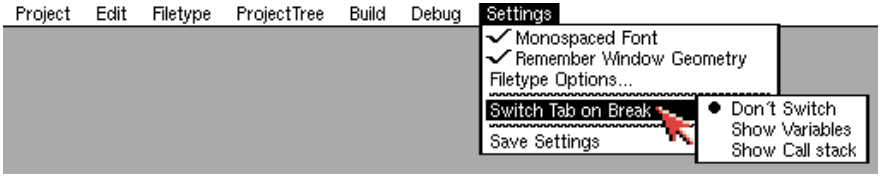
Codecraft displays the *Search results* in an additional tab in the **Auxillary Browsers** section.



The search results are grouped per file and the corresponding lines containing the searched text are displayed. Double-clicking on one of the lines navigates to the corresponding location in the source code in the editor.
Close the *Search Results* tab to end the search and discard the search results.

Codecraft Settings

In addition to the settings that TextEdit provides, Codecaft provides some more specific settings.



- **Switch Tab on Break** provides the option to make Codecraft automatically switch to a specific tab of the bottom tab bar during debugging, when a breakpoint is hit. You can choose either the *Variables* tab, the *Call stack* tab or not to switch to another tab.

About Project Templates

Project Templates are a convenient and easy way to create a base for a new development project in Codecraft. Project Templates are presented in the *New ProjectTree* requester.

A Project Template usually consists of a collection of files and resources that are prefilled with code and data, to create a specific development project from it. Like a CLI application, an Amiga library, etc.

When creating a new Project Tree from a template, Codecraft copies the files and data of the project template to the newly created Project Tree directory, and adjusts the content of the copied template files according to your new Project Tree specifics.

Templates shipped with Codecraft

The list of ProjectTree Templates shipped with Codecraft is constantly expanding. Here are descriptions of some of them:

Simple Workbench Application
The *Simple Workbench Application* template creates a project with necessary startup code to run as a Workbench application. That is responding to the Workbench startup message and parsing the application Tooltypes.

Simple Commandline Application
The *Simple Commandline Application* template creates a command line application, running in the Shell. The template code also reads some

command line parameters, as a starting point for your own implementation.

BOOPSI Window Application
The *BOOBSI Window Application* template creates a Workbench application that just opens an empty window. It is a good starting point for any gui application.

Creating custom templates

Codecraft comes already with a set of predefined project templates. However, it is easily possible to add further, individual templates for your own development projects. For this, a separate folder for user templates is created when Codecraft was installed. That folder is named *UserTemplates* and it is located in the Codecraft program folder.

To create your own project templates, you have to create an folder within the *UserTemplates* folder for each of your project templates. Create all files of your project template in the respective project template directory, like:

- source files
- smake file
- ProjectTree file for your template
- all sorts of other resources for your template (images, sources, whatsoever)

Important conventions for project template files

In order for Codecraft to be able to customize the files of a project

template when creating a plain ProjectTree from it, the following conventions must be used:

Renaming of files

Any file which name is or contains "TEMPLATE" will be renamed during project creation by replacing "TEMPLATE" with the name of the Project Tree.

Example:

Let's assume you create a template that should contain a #?.c file with some code and a #?.info icon file. When creating a project from the template, you want the c file and the icon get renamed with the project name.

Names of the template files: TEMPLATE.info
TEMPLATE.c

Names after project creation (e.g. MyAmigaApp.projecttree):
MyAmigaApp.info
MyAmigaApp.c

Replacing of text tokens

It is possible to insert the name of your project (= ProjectTree name) in the files when a project is created from a template. E.g. for the name of the binary to be created by the makefile, of just the application name shown in the window title. Codecraft searches for the token *%TEMPLATE%* in all files of your template (except binary files) and will replace *%TEMPLATE%* by your project name.

Example:

Let's assume the name of your project is MyAmigaApp and you want the name to appear in the window title of your workbench application.

Code in the template that creates a window:
windowObject = NewObject(WINDOW_GetClass(), NULL,
WINDOW_Position, WPOS_CENTERSCREEN,
WA_Activate, TRUE,
WA_Title, **"%TEMPLATE%"**,
WA_DragBar, TRUE,
WA_CloseGadget, TRUE,
etc…

Code after project creation:
windowObject = NewObject(WINDOW_GetClass(), NULL,
WINDOW_Position, WPOS_CENTERSCREEN,
WA_Activate, TRUE,
WA_Title, **"MyAmigaApp"**,
WA_DragBar, TRUE,
WA_CloseGadget, TRUE,
etc…

This feature is also very handy to adjust smakefile files included in templates. For further reference and more examples, you can inspect the project templates provided with Codecraft, they are located in the *Templates* folder inside the Codecraft application folder.

One further note: You can organize your templates by putting them into subfolders within the *UserTemplates* folder. This will make Codecraft to show the templates in the *New ProjectTree* Dialog sorted into groups, named by the folder name.

Appendix

About the Codecraft Project

For news and to download the latest version of Codecraft, visit the official Codecraft homepage at: http://boemann.dk/codecraft/.

Codecraft development is hosted on GitLab, see: https://gitlab.com/boemann/codecraft.

Currently we do decided on how to report report bugs or submit suggestions. But, for now, you can use the gitlab feature or write in the forum at developer.amigaos3.net.

Copyright

Requirements

Hardware
Technically, Codecraft runs an Amiga with a plain 68000 CPU, some Fast RAM (e.g. 4MB) and a hard drive to hold the system and the development environment. However, such a system would be really slow and compiling stuff would take a long time.

Therefore, we recommend the following system equipment to run development using Codecraft:

- Amiga with 68030 CPU running at 25MHz
- 8 MB Fast RAM
- Flicker Fixer to run PAL Highres Interlaced
- Hard drive or SD Card

These are the minimum requirements to work meaningfully with Codecraft. A faster system makes it even better!

Software
At least AmigaOS 3.2.2 is currently required to run Codecraft 1.0 and greater. Since Codecraft relies heavily on TextEdit, which is part of the AmigaOS distribution, AmigaOS 3.2.2 is a mandatory requirement.

SRC6 Hunk Documentation

As part of the Codecraft project, the Amiga SRC6 hunk format, as generated by SAS/C, was reverse engineered. This was necessary to develop the Codecraft Debugger. Because of the complexity of the SRC6 Hunk format, this is still a work in progress.

You can find the current documentation of the SRC6 hunk format on the Codecaft GitLab page:

https://gitlab.com/boemann/codecraft/-/blob/master/SRC6%20description.txt