©2014 by Viosoft Corporation, Inc.
All rights reserved. This document and the associated non-open source Arriba software are the sole property or licensed by Viosoft Corporation. Each contains proprietary information. Reproduction or duplication by any means of any portion of this document without the prior written consent of Viosoft Corporation. is expressly forbidden. Viosoft Corporation. reserves the right to make changes to the specifications described herein at any time and without notice in order to improve design or reliability of Arriba. The information in this document has been carefully checked for accuracy; however, Viosoft Corporation. makes no warranty pertaining to the correctness of this document.
Eclipse is licensed under the Eclipse Public License (EPL). GDB and GCC are licensed under the GNU General Public License (GPL).

Trademarks
Arriba and Arriba Connect are trademarks of Viosoft Corporation. All other product and company names are trademarks or registered trademarks of their respective holders.

Warranty Limitations
Viosoft Corporation. makes no warranty of any kind that the Arriba products will meet the USER's requirements, or will operate in the manner specified by the USER, or that the operation of the Arriba products will operate uninterrupted or error free, or that any defects that may exist in the Arriba products will be corrected after the warranty period. Viosoft Corporation. makes no warranties of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose, with respect to the Arriba products. No oral or written information or advice given by Viosoft Corporation., its dealers, distributors, agents, or employees shall create any other warranty or in any way increase the scope of this warranty, and licensee may not rely on any such information or advice.
About This Guide
This guide provides comprehensive information about Arriba, the Eclipse-based development tool from Viosoft Corporation. It is intended for the embedded real-time software developer using the C or C++ programming language. The developer should be familiar with standard embedded development practices and the C programming language.

Organization
Chapter 1 Provides a basic overview of Arriba and how it fits in the development process.
Chapter 2 Provides the basic steps to install and use Arriba.
Chapter 3 Illustrates the opening, building and running of a small example system.
Chapter 4 Describes the custom Arriba perspective.
Chapter 5 Discusses the concept of workspaces in Arriba
Chapter 6 Describes creating and building projects in Arriba.
Chapter 7 Describes creating and editing files in Arriba.
Chapter 8 Describes running and debugging executable images in Arriba.

Guide Conventions
Italics typeface denotes book titles, emphasizes important words, and indicates variables.
Boldface typeface denotes file names, key words, and further emphasizes important words and variables.

Customer Support Center
For the latest product information, visit the Viosoft Corporation web site and select the "Support" menu option to find the latest online support information, including information about the latest Arriba product releases.

In addition to this User Guide, there are general Eclipse information resources available from within Arriba. These can be found by selecting the “Help -> Help Contents” menu item. Doing so will result in the following menu:
The Workbench User Guide provides a more general overview of the Eclipse while the C/C++ Development User Guide provides an overview of the C/C++ flavor of Eclipse. Since Arriba is based on Eclipse / CDT, both documents are pertinent. However, Viosoft Corporation has made a variety of enhancements that are documented only in this guide.

What We Need From You
When contacting Viosoft Corporation for support, please supply us with the following information in an e-mail message so we can more efficiently resolve your support request:

1. A detailed description of the problem, including frequency of occurrence and how it can be reliably reproduced.
2. Attach the project and/or files that cause the problem.
3. The version of Arriba that you are using is readily available on the top line of the IDE, as follows:
4. More detailed version information can be obtained by selecting the “Help -> About Arriba”, as follows:

This will yield the following screen with the necessary version information:

**Where to Send Comments About This Guide**
The staff at Viosoft Corporation is always striving to provide you with better products. To help us achieve this goal, e-mail any comments and suggestions to the Customer Support Center at:

support@viosoft.com
Enter “Arriba User Guide” in the subject line.
Libraries Options ........................................................................................................................... 57
Diagnostic Options .......................................................................................................................... 58
MPLAB XC32 Assembler Settings ................................................................................................... 59
  General Settings ............................................................................................................................... 60
  Other Options .................................................................................................................................. 61
Deleting Projects ............................................................................................................................... 62
Chapter 7. Arriba File Creating and Editing ........................................................................ 63
  Creating a New File ............................................................................................................................ 63
  Deleting a File ................................................................................................................................... 66
  Editing an Existing File ...................................................................................................................... 66
  Saving Changes ................................................................................................................................. 66
Chapter 8. Running and Debugging Executable Images .................................................. 67
  Debug Configurations ..................................................................................................................... 67
  Launching The New Connection ...................................................................................................... 68
  Terminating a Connection ................................................................................................................ 68
  Program Download .......................................................................................................................... 69
  Program Control ............................................................................................................................... 69
  Resume ............................................................................................................................................... 69
  Breakpoint .......................................................................................................................................... 71
  Single-Step Mode .............................................................................................................................. 72
  Single Step Into ............................................................................................................................... 72
  Single Step Over ............................................................................................................................... 73
  Step Return ....................................................................................................................................... 73
  Program Viewing .............................................................................................................................. 73
  Program Code Viewing .................................................................................................................... 74
  Disassembly Viewing ......................................................................................................................... 74
  Variable Viewing ............................................................................................................................... 74
  Register Viewing ............................................................................................................................... 76
  Memory Viewing ............................................................................................................................... 77
Chapter 1. Introduction to Arriba

Viosoft Corporation's Arriba is a complete Eclipse-based Integrated Development Environment (IDE) for the PIC® family of microcontrollers (MCUs). Based on the Eclipse environment, Arriba is fully integrated with Microchip's MPLAB® XC C compilers, hardware debug tools, and simulator.

A basic overview of the components of Arriba is shown in Figure 1. The version of each of the above components is as follows:

- Eclipse IDE
- GDB
- MPLAB XC C Compiler
- Arriba Connect
- Microchip HW tools / simulator
- Microchip devices
Viosoft Corporation has modified various components in order to integrate them, improve the overall development flow, and provide new features such as RTOS aware debugging and managed build.

Also notice that Arriba does not rely on Cygwin for its operation, which makes installation and maintenance much easier.

Furthermore, Arriba is inherently setup to allow any number of new plug-ins, including but not limited to new HW debug tools and RTOS. This unparalleled versatility makes Arriba the ideal embedded development solution for PIC MCU users.

**Arriba Requirements**

Viosoft Corporation’s Arriba runs on Windows 32/64-bit system, Linux 32/64-bit system, or MacOS 64-bit system. The host system should have a minimum of 1GB of RAM, 20 GB of available hard-disk space, and a minimum display of 1024x768 with 256 colors.

**Arriba Constraints**

The Arriba development environment has no real constraints—it is flexible enough to handle the smallest project and powerful enough to handle the largest projects. However, Arriba is also integrated with several other third-parties software components, as seen above, and is constrained by the capabilities of these software components.

**Arriba Overview**

The Arriba development environment is described in greater detail in subsequent chapters, but the following is a brief overview. All editing, building, executing, and debugging operations are done from the main Arriba desktop.

Editing and building is done through the “Project Explorer” and source windows as shown in Figure 2:
In this figure, the “src” directory is selected and its contents are displayed in the Project Explorer view, which is where all editing of the source code takes place. Important buttons associated with building the program under Arriba are as follows:

- All new items can be created through selection of this button, including new projects, files.
- Editing changes are saved via the selection of this button (or Ctrl+S).
- The current source file being viewed is printed via selection of this button (or Ctrl+P).
- Selecting this button (or Ctrl+B) builds all the projects in the workspace.
- Selecting this button builds the current project.
- The Debug or Release configuration is determined by selecting this button.
- Searching is accomplished by selecting this button (or Ctrl-H).
- Whitespace characters are displayed by selecting this button.
Once the program is successfully built, program execution and debug is accomplished by the resources in the “Debug,” “Variables,” “Breakpoints,” “Registers,” “Memory,” and “Disassembly” windows. Important buttons associated with executing and debugging the program under Arriba are:

- Establish a target debug connection.
- Selecting this button resumes execution of the program.
- Selecting this button suspends execution of the program.
- Assembly-level instruction stepping is toggled by selection of this button.
- Single-stepping INTO functions is selected by this button.
- Single-stepping OVER functions is selected by this button.
- The current connection is terminated and removed by selecting this button.
- Selecting this button reloads the current program.

As mentioned previously, the following chapters of this guide will fully detail the operation of Arriba. In particular, Chapter 3 shows how to build and execute the “PIC32” demonstration in a matter of minutes!
Chapter 2. Installation of Arriba
This chapter contains a description of various issues related to installation and setup of the Arriba development tool.

Product Distribution
Arriba is delivered as an executable, self-installing image, or as an Eclipse plugin. The Arriba package also contains example pre-built project source code that should serve as a good starting point for new Arriba users. The release notes associated with each new Arriba release can be found in the file readme.txt. Please review this file to see what has changed between successive Arriba releases.

Arriba Installation Directory
We recommend that Arriba be installed in the default directory, which varies depending on the host computer Operating System. On Windows, Arriba is installed under:

\Program Files\Arriba4PIC

Subsequent versions of Arriba will install in a version specific directory as well—allowing multiple versions of Arriba to co-exist on the host platform.

Arriba Install Flow
Arriba utilizes izPack for installing Arriba on the host machine. To install Arriba, run the executable installer, either from the command line, or by double-clicking on it from the desktop. The main Arriba installer screen is shown in Figure 4:
Figure 4 - Main Installer Screen

Select the "Next" button to proceed with the installation. The Arriba release note is shown in Figure 5. Please read over to verify if the contents meet your expectation,
and select the “Next” to proceed:

![Figure 5 - Arriba Release Notes](image)

Once “Next” is selected, you shall be presented with the Arriba End User License Agreement, shown in Figure 6. Review this agreement carefully, select “I accept the terms of this license agreement” and “Next” to continue or “Quit” to exit.
Figure 6 - Arriba License Agreement

Once "Next" is selected, Arriba presents the packages to install in Figure 7. It is recommended that you simply select "Next" to proceed.
Once "Next" is selected, Arriba installer presents the installation path in Figure 9. You can choose to keep the recommended installation path, or enter an alternative. Select “Next” to continue.

If Arriba has previously been installed in the selected path, you will be asked whether or not to overwrite the existing installation as shown in Figure 8. It is strongly recommended that you select an alternate path in this case.
The Arriba installer presents a summary of packages to be installed in Figure 10. Simply select “Next” to proceed.
Once "Next" is selected, Arriba shall begin to copy the selected packages into the installation directory, shown in Figure 11.
After all the Arriba files have been copied, one final dialog is presented in Figure 12. Simply select the “Finish” button to complete the installation of Arriba. You are now ready to use Arriba for PIC MCUs!
Installation Finished
Step 9 of 9

Installation has completed successfully.
An uninstaller program has been created in:
/Applications/Arriba4PIC/Uninstaller

Figure 12 - Installation Finished
Chapter 3. A Simple Example Project

Using Arriba is practically as easy as opening a workspace and selecting the build button! This chapter discusses building and running an entire example project from start to finish. Arriba comes complete with a variety of examples. For the sake of this brief introduction, we will focus on the example of a standalone PIC32 application.

Getting Started

Simply run Arriba via the “Start” button or via the Arriba icon on the desktop: At this point you will observe the Arriba Workspace Launcher dialog in Figure 13. Select a directory on your host computer that Arriba can use to store projects and the workspace. Ensure that you have write permission for this directory. More on Eclipse projects and the workspace can be found in the Eclipse documentation: http://help.eclipse.org/kepler/index.jsp

![Workspace Launcher](image)

If you have selected a new workspace, Arriba will start at the welcome screen, shown in Figure 14:
Creating a Project
From the Arriba workbench, you can begin to create a project for a PIC MCU. Start by selecting **File->New->Project** to show the New Project Dialog in Figure 15:
Select “C Project” and click “Next” to continue.
Arriba will now present a list of project types to choose from in Figure 16. You may need to check/uncheck the “Show project types and toolchains...” checkbox to show all-available project types for PIC MCUs. Select “PIC32 Sample” for this exercise and enter “PIC32” for Project name. Click “Next” to move to the C Project Dialog, shown in Figure 17.
Arriba User Guide

Figure 17 - C Project Dialog

Arriba presents a list of configurations to create for this project. Select “Next” to continue.
Arriba will now require that you specify some information about the PIC MCU target on which the resulting code generated by the project will execute. By default, no information is specified for the project and you’re required to enter the device family, device, and hardware tools associated with the newly created project, shown in Figure 18.

Note that you can always modify these choices after the project has been created. For this exercise, click on “Configure” to select these data and Arriba will open the Device and Hardware Tools Configuration Dialog shown in Figure 19.
Again for this exercise, we recommend that you keep the default options specified in this dialog, and simply click “OK” to close the Hardware Tools Configuration Dialog. Then click on “Finish” to create the project. Arriba will offer to open the “C” perspective and you shall accept, as shown in Figure 20.

Building the Project
You should see the following Arriba desktop in Figure 21.
Note that:
- You can select “C/C++” button to view the C perspective at all time
- Edit/view source files by double-clicking on it under the Project Explorer
- Arriba Connect status indicator should show “active” in the lower left hand corner. If not, your Arriba installation is not properly running; contact Viosoft support.

At this point, you’re ready to build the PIC32 project. Note that you must install the MPLAB XC32 C compiler toolchain prior to using Arriba in order to build this PIC32 application. (For other PIC MCUs, you may need to select a different MPLAB XC C compiler.)

Building is simply done by clicking on the build button in the toolbar, shown in Figure 22.
You should see the following messages in the Console view of Figure 23.

**Debugging the Project**
Before you debug, it is recommended that you select the PIC32 entry in the Project Explorer, as shown in Figure 24, before opening the Debug Configuration Dialog. Doing so will enable eclipse to automatically fill out the necessary debug configuration information.
Now you can debug the “PIC32” application by selecting Run->Debug Configurations, or by clicking on the tiny down arrow next to the cockroach and select “Debug Configurations” as shown in Figure 25.

From the Debug Configuration Dialog:

- Select Microchip Config in the launch list
- Double click on the plus icon in the toolbar to create a new debug configuration for the PIC32 project

You should see the Main Debug Tab shown in Figure 26:
At this point, Arriba should offer the Debug perspective and you shall accept, as shown in Figure 27.
Figure 27 - Switching to Debug Perspective

From the Debug perspective shown in Figure 28, you can press "Resume" button to resume execution, or "Halt" to suspend execution. Notice that the debug view should annotate your current program location within the execution flow. In addition, breakpoints, register, and memory viewing are all available with just a few mouse clicks!
The project building and execution facilities of Arriba are discussed in greater detail in subsequent chapters of this guide. Hopefully, this chapter illustrates just how easy it is to get a project up and running with Arriba!
Chapter 4. The Arriba Perspective

Many Eclipse implementations have two main IDE perspectives, namely \textit{C/C++} and \textit{Debug} perspectives. Previous versions of Arriba have integrated these two perspectives into a unified perspective called the Arriba perspective. While this approach has simplified the adoption of Arriba by new users of Eclipse, it can also cause confusions for experienced Eclipse and CDT users.

Moving forward, Arriba will revert to having separate \textit{C/C++} and Debug perspective in keeping with the consistency of Eclipse and CDT.

The Arriba C/C++ Perspective

The Arriba perspective provides all the tools necessary to create, edit, assemble, compile, and link an embedded program. Below is the \textit{Arriba C/C++} perspective shown in Figure 29 for an empty workspace (a workspace without any projects).

![Figure 29 - Arriba C/C++ Perspective](image)

The Arriba Debug Perspective

The Arriba Debug perspective provides all the tools necessary to debug and run an embedded program. Below is the Arriba Debug perspective shown in Figure 30 for the PIC32 project:
 resets Perspective
The perspective can be reset by selecting “Window -> Reset Perspective” while that perspective is selected.
Chapter 5. The Arriba Workspace

This chapter contains a description of various issues related to application development workspaces in Arriba.

What Is a Workspace?
Arriba workspace is effectively a sub-directory that contains one or more Arriba projects. Arriba workspaces are fully compatible with Eclipse workspaces, meaning that a workspace created by Arriba can be opened and viewed by a compatible Eclipse IDE.

A workspace can contain one or more project(s).

Selecting a Workspace
There are basically two methods to select an existing Arriba workspace. The first method is through the Workspace Launcher, which is the first dialog presented after Arriba is started, shown in Figure 31.

![Workspace Launcher Dialog](image)

Figure 31 - Workspace Launcher Dialog

Arriba also provides the ability to switch workspaces. This is accomplished from the File menu as shown in Figure 32:
Selecting “Switch Workspace” results in the workspace selection dialog to reappear, thereby allowing a new workspace to be specified.

Creating a New Workspace
Creating a new workspace in Arriba is very easy and amounts to simply selecting an empty or non-existent directory. If the directory does not exist, Arriba will create it and in either case prepare the workspace for importing and/or creating new projects.

Note that if you don’t specify the complete root directory path, Arriba will create the workspace directory relative to the Arriba installation directory.

Upon the opening of a newly created workspace, Arriba shows the Welcome screen and creates storage for the workspace in the specified location on the host computer and prepares it to import and/or create one or more projects. Welcome screen for the new empty workspace is shown in Figure 33:
Selecting the “Go to the workbench” button enters the newly created, empty workspace. An empty workspace simply has no projects and as a result the Project Explorer section is blank.

Deleting a Workspace
Deleting a workspace in Arriba is done by simply deleting the sub-directory of the workspace. Doing so will also delete any projects contained within the workspace, since they are all maintained in a workspace-relative fashion.
Chapter 6. Arriba Projects

This chapter contains a description of projects in Arriba. A project is a logical construct that facilitates building a collection of C and/or assembly files into an executable image.

Creating Projects

There are principally two types of projects in Arriba: standalone executable projects or RTOS projects.

A standalone executable is a program that does not require the services of an RTOS, whereas an RTOS executable is one that relies on the API published by the respective Real-Time Operating System.

Creating a new project in Arriba is easy; simply select “File -> New -> Project” as demonstrated in the Simple Example Project in Chapter 3.
Creating Source Files
Creating new source files for a library project is straightforward; simply follow the file creation instructions found in Chapter 7. The examples that follow in this chapter are focused on importing existing files, which is typically the more common operation.
Importing Source Files

Importing files into the newly created project is straightforward; simply drag and drop the desired files into the project. Files may also be imported to the project through menu selections. To explicitly import one or more files, "right-click" on the project as shown and select the **Import** menu selection as shown in Figure 35.

![Figure 35 - Import Menu Selection](image)

Once **Import** is selected, Arriba displays the **Import** dialog, showing many options for the source of the files to import. Generally, the **File System** option is the most used source for importing. The figure below shows the **Import** dialog, the **File System** selection and the mouse positioned on the "Next" button ready to proceed. The **File System Import** dialog is now displayed. Next, expand the "General" folder, select "File System," and select the "Next" button. This results in the file browse dialog shown in Figure 36:
Simply use the **Browse** button to navigate to the specific folder containing the source files to import, select the files to import within that directory, and select **“Finish.”** This is show below in Figure 37.
Building the Executable

To build the executable, first ensure that the Console tab is selected. If not, the Console window can be enabled by selecting “Window -> Show View -> Console.” The Console window displays the progress of the library build – including the actual MPLAB XC C compiler invocation for each source file in the library. Next, simply select the project Build button to build the executable as shown in Figure 38.
Note that the default project settings are for the device and hw tools selected at project creation time. For other settings, please refer to the subsequent section in this chapter titled **Modifying Projects**.

Once selected, observe compilation commands in the **Console** window as shown below in Figure 39. This project produces a standalone executable by the name of **PIC32.elf**:

```
Building file: ../src/user.c
Invoking: xc32 C Compiler
xc32-gcc -mprocessor=32MX360F512L -g -c -MMD -MP -MF"../src/user.d" -MT"../src/user.d" -o "src/user.o" "../src/user.c"
Finished building: ../src/user.c

Building target: PIC32.elf
Invoking: xc32 C Linker
xc32-gcc -mprocessor=32MX360F512L -o "PIC32.elf" ../*.c ../*.h ../*.o
Finished building target: PIC32.elf
```

Creating an RTOS Project

Creating an RTOS project is very similar to creating a standalone project – the biggest difference is the project type selection in the creation process. RTOS projects come with source files from the respective RTOS to enable successful generation of an executable utilizing the API from said RTOS.

Below is the new project wizard dialog, which has selections for both types of projects. You can see that FreeRTOS, and Micrium are just a few of the RTOS project types supported by Arriba in Figure 40.
Note the **Debug** and **Release** options in the dialog. These options allow Arriba to manage two different configurations of the same project. The **Debug** configuration typically contains settings appropriate for developing and debugging the code in the project, while the **Release** configuration typically has settings appropriate for finished production code—the main difference being the lack of debug symbols and perhaps higher optimization settings in the **Release** configuration. The examples in this guide use the **Debug** configuration.
Creating Source Files
Creating new source files for an executable project is straightforward; simply follow the file creation instructions found in Chapter 7. The examples that follow in this chapter are focused on importing existing files, which is typically the more common operation.

Importing Source Files
Importing files into newly created project is straightforward; simply drag and drop the desired files into the project. Files may also be imported to the project through
menu selections. To explicitly import one or more files, "right-click" on the project as shown below and select the **Import** menu selection as shown in Figure 41.

![Figure 41 - Import Menu Selection](image)

Once Import is selected, Arriba displays the Import dialog, showing many options for the source of the files to import. Generally, the "**File System**" option is the most used source for importing. Select this option to import one or more files from the file system into your project.

The **File System Import** dialog is now displayed. Next, expand the "**General**" folder, select "**File System,**" and select the "**Next**" button. Simply use the **Browse** button to navigate to the specific folder containing the source files to import, select the files to import within that directory, and select "**Finish.**" You will now observe the selected files being imported into the selected project.

At this point, all the necessary files for building an RTOS executable for PIC MCUs are contained in the project. *Note that additional files might be required for executable projects and are specified in the linker settings dialogs. Such files include linker control files and the desired library files to be linked with the executable project.*

### Building the Executable

To build the executable, first ensure that the **Console** tab is selected. This makes it possible to view the progress of the executable build since the actual MPLAB XC C compiler invocation for each source file and the linker invocation is displayed in the **Console** view. Next, simply select the **Build** button to build the executable.

Note that the default project settings are for a specific combination of PIC MCU device architecture, device, and hardware debug tools. To select a different
combination, please refer to the subsequent section in this chapter called *Modifying Projects*. Once selected, observe compilation and linkage commands in the *Console*. At this point, the RTOS executable is ready for program, execution, and debug. Chapter 7 of this guide contains details of the program and debug processes.

**Closing Projects**

If a specific project is not currently needed, it is easily closed by right-clicking on the desired project to close and then clicking on the “Close Project” selection, as shown below in Figure 42:

![Figure 42 - Closing Project Menu Selection](image)

Below is the *C/C++* perspective with the project in a closed state.
There are many build options available for Arriba for PIC MCUs developers. The options are available through the **Properties** dialog. To bring up the **Properties** dialog, right click on the project name and then select “**Properties**,” as shown below in Figure 44.
Selecting “Properties” results in Arriba bringing up the Properties dialog. This dialog has many configuration options, however, the most typically used options are found within the “C/C++ Build -> Settings” selection, shown here in Figure 45.
Device and Tool Settings

The Device and Tool Settings show the current device family, device, and hardware tools associated with the selected project. To modify these parameters, click on “Configure...” to open the Device and Tools Configuration Dialog, shown here in Figure 46.
From this dialog, you can select a different device family, device within that family, and associated hardware debug tools for debugging purpose. You can also configure various parameters of the hardware tools by selecting from the different options in the category combo menu.

**MPLAB XC C Compiler Settings**

The high-level overview of the current *C Compiler* settings is shown below in Figure 47.
Figure 47 - C Compiler High Level Overview

The “Command” and “All options” entries show exactly how the compiler is going to be invoked for this project. All files with a “.c” extension will be built, by default, with these options.

**General Settings**

The **General** settings dialog contains options enabling you to control various aspect of code and debug symbol generation by the compiler. You can also define additional pre-processor symbols, as well as include directory search path. An overview of the general setting dialog is shown below in Figure 48.
Errors and Warning Settings
This dialog allows you to control various error and warning reporting aspect of the compiler, as shown below in Figure 49.
Optimization Settings
The optimization settings dialog allows you to change the optimization level of code generated by the compiler. Note that you will need to upgrade to a paid version of the compiler in order to access the full optimization settings of the compiler.

MPLAB XC Linker Settings
The high-level overview of the current C Linker settings is shown below in Figure 50.
The "Command" and "All options" entries show exactly how the linker is going to be invoked for this project. All executables, by default, will be linked with these options.

**General Options**
The General Options allows you to control the heap and stack size, as well as to write in additional options not covered by the dialog. A view of the General Option dialog is shown below in Figure 51.
Figure 51 - Linker General Options

Symbols and Macros
This option group lets you introduce additional linker symbols, as well as preprocessor macro definition used during the linking process, as shown in Figure 52.
Figure 52 - Symbols and Macros Settings

Libraries Options
These options, shown in Figure 53, let you specify one or more standard libraries to link with the project executable. In the event that one or more libraries are not part of the MPLAB XC toolchain (i.e. they reside in a different directory), you can also specify one or more search paths for the libraries. Other options include the ability to exclude standard libraries, start-up code, or floating point library.

*Note that the selections in the Libraries dialog specify the “c” and “gcc” libraries. It is the GNU linker that prepends the “lib” modifier to the library names specified and thus expects to find “libc.a” and “libgcc.a” library files in one of the specified library paths. Arriba also makes it possible to specify the path and actual name of library files. The “Additional Inputs (*.o/*.lib)” dialog is used to do this. Please note that references to actual path on your host computer will make your project/workspace not portable to other host systems.*
Diagnostic Options
These options, shown in Figure 54, let you generate extra reports about the executable, including memory usages, files crossed referenced, and one or more trace symbol(s).
MPLAB XC32 Assembler Settings

These settings let you control various aspects of the assembler. A general view is shown below in Figure 55.
As with compiler and linker settings, the "Command" and "All Options" show exactly how the assembler is to be invoked for files ending with "s" extension in the selected project.

**General Settings**

The General settings let you define preprocessor macros, as well as assembler symbols and search paths, as shown below in Figure 56.
Figure 56 - Assembler General Settings

**Other Options**
The Other options let you tune various parameters of the assembler, as shown in Figure 57.
Deleting Projects
Deleting a project is done by right clicking on the project and selecting “Delete.” Once “Delete” is selected, Arriba presents a dialog that specifies whether or not everything in the project directory is to be deleted. If the contents are not deleted, they may be used in a subsequent creation of another project.
Chapter 7. Arriba File Creating and Editing

This chapter contains a description of how to create, edit and save source files in the Arriba environment. Arriba provides a rich C/C++ language editor that shows color coding for various language constructs. The following sections describe the file creating and editing features in greater detail.

Creating a New File

Creating a new file within a project is accomplished by selecting the “File -> New -> Other” menu selection as shown in Figure 58.

Once selected, the new resource wizard dialog is displayed in Figure 59, from which “File” should be selected followed by “Next” as shown:
Figure 59 - Resource Wizard Dialog
The resulting dialog in shows an example of creating a file named “new_file.c” in the selected project. After finish is selected, the file “new_file.c” is now part of the selected project, as shown below in Figure 60 and Figure 61.
Deleting a File
Deleting a project file is easy; simply select the file and the “Delete Key.” This will bring the delete file dialog from which simply select “Yes” to delete the file.

Editing an Existing File
Selecting a file to edit is easy; simply double click on the file name in the “Project Explorer” view. Once selected, the file is presented in the source view. Notice that comments are colored a light green, language keywords are colored magenta, and language statements are colored black. These color codings make it easier to write code in the C/C++ language.
Editing is free-form, simply click on the desired location to edit and start typing.

Saving Changes
Saving editing changes to a file is done simply by clicking on the “Save” button. The corresponding project can then be rebuilt to incorporate the latest source code changes.
If changes are being made during execution and debug, it is important to rebuild the image and download it again in order for the Arriba debugger to accurately perform source level debugging.
Chapter 8. Running and Debugging Executable Images

This chapter contains a description of how to execute and debug projects built by Arriba. The examples in this chapter rely on the **PIC32** project that was created in Chapter 6. In Arriba, execution and debug is done within Debug perspective.

**Debug Configurations**

The Debug Configuration specifies the executable image is to be programmed on the device, executed and debugged. In addition, you can specify additional GDB commands to be executed at connect time as shown below in Figure 62.

![Figure 62 - Startup Tab in Debug Configuration Dialog](image)

By default, Arriba initialize the debug configuration with values appropriate for the selected project when you create a new debug configuration. If a project is not selected, or if the selected project has more than one active build configuration, you will need to explicitly specify the project executable to be used at debug time.
Launching The New Connection
Launching an existing connection is very simple; just click on the **Debug** button as shown below in Figure 63.

![Figure 63 - Launching Debugger](image)

At this point the connection is ready for execution and debugging of the specified executable image.

Terminating a Connection
When execution and debug of an image is complete and the connection is no longer needed, terminating a connection is very simple; just click on the **Terminate and Remove** button as shown below in Figure 64.

![Figure 64 - Terminating Debugger](image)
**Program Download**
The processing for making a target connection automatically program the executable unless the **“Connect Only”** option under the **“Debugger”** tab is selected. If not selected, establishing a connection will perform all initialization on the target and then download the executable image. The image downloaded can also be verified (read back and compared) to make sure the download was successful. This is accomplished by selecting the **“Verify downloaded image”** option under the **“Debugger”** tab. If the verification fails, it is likely that the image is being loaded to an invalid memory address or the board initialization is incorrect.

**Program Control**
Arriba automatically places a breakpoint and launches the downloaded image, if the **“Stop in main”** option is selected under the **“Debugger”** tab. If this is not selected, Arriba simply positions the program counter at the program entry point, which in GNU-built executable is at the label `_start_`.

**Resume**
Execution of a suspended (or just downloaded) program is resumed via the **“Resume”** button in the **Debug** dialog shown in Figure 65.

Once selected, the Arriba debugger shows the system running. The program is easily halted via the **“Suspend”** button shown in Figure 66.
Once selected, the Arriba debugger shows where the system was halted. The program is easily resumed again via the “Resume” button.

Figure 66 - Suspend Execution Button
**Breakpoint**
Stopping the program at certain locations in the source code is a necessity and is easily accomplished in Arriba via the breakpoint. Breakpoints are set and cleared by double-clicking on the left margin of the source window, as shown in Figure 67. When the program hits the breakpoint it is halted and all the pertinent register and data views are updated.

![Figure 67 - Setting Breakpoints](image)

Selecting the “Resume” button again will cause the demonstration program to run again. As mentioned previously, breakpoints can be removed simply by double-clicking on them. Alternatively, they can be removed or disabled via the “Breakpoints” dialog. Right-clicking on the breakpoint in this dialog provides selections for the specific breakpoint, shown in Figure 68.
Single-Step Mode
Arriba supports single-stepping by assembly instruction or by C source level statements. The type of single stepping is determined by the “Instruction Stepping Mode” button, shown in Figure 69. By default, the stepping mode is for stepping C source level statements; selecting this button will toggle the mode to assembly level stepping. Another selection of this button will change the mode back to C source level statement stepping.

Single Step Into
Single-stepping allows the developer to execute one statement of code at a time. If a function call is present, selecting the “Step Into” button will step into the function call.
**Single Step Over**
The “Step Over” button, shown in Figure 70, operates in the same way as the “Step Into” button except when a function call is present. Instead of stepping into the called function, the “Step Over” button steps to the statement following the called function, i.e., “over the function.”

![Figure 70 - Step Over Button]

Once selected, the “Step Over” button will cause the program to run to the next instruction after the function call.

**Step Return**
When selected, the “Step Return” button, shown in Figure 71, steps back to the caller of the current function.

![Figure 71 - Step Return Button]

Selecting the “Step Return” button causes the program to return to the caller.

**Program Viewing**
Arriba provides extensive viewing capabilities into the target program, including views of source, disassembly, variables, registers, memory, and ThreadX objects. The following sections describe each of these Arriba views.
Program Code Viewing
Viewing program source code in Arriba is easy; simply double-click on the file to view in the “Project Explorer” dialog. Doing so opens the source viewing window with the contents of the selected file.

Disassembly Viewing
The disassembly view is opened by simply selecting the “Disassembly” tab, or Windows->Show Views->Disassembly as shown in Figure 72.

Variable Viewing
Viewing variables in Arriba is easy; simply select the Windows->Show Views->Variables as shown in Figure 73.
Global variables can be added by simply right-clicking on any valid entry in the “Variables” display area and selecting the “Add Global Variables” option, as shown in Figure 74.
This selection results in the “**Global Variables**” dialog to be presented. Simply select from this dialog as shown to add global variables to the “**Variables**” display.

**Register Viewing**

The target processor registers are also available for viewing, simply select the “**Registers**” tab, or **Windows->Show Views->Registers** as shown in Figure 75.

Once selected, the “**Registers**” window is displayed. By default, the PIC MCU registers are shown in the “**Registers**” window displayed in decimal format. Clicking on the register shows its value in different formats, as shown in Figure 76.
**Memory Viewing**
The target memory is easily viewed; simply select the “Memory” tab, or Windows- >Show Views->Memory as shown in Figure 77.

![Figure 77 - Viewing Memory Menu Selection](image)

Once selected, the “Memory” window is displayed, as shown in Figure 78.

![Figure 78 - Memory View (Empty)](image)

To add an address to display in the “Memory” view, simply click the plus icon to bring up the address dialog as shown in Figure 79.
Figure 79 - Address Dialog Box

Entering "\&initApp" causes the address of this variable in the demonstration system to be displayed, as shown in Figure 80.

Figure 80 - Memory View (Populated)