Интегрированная среда разработки

для процессоров на базе архитектуры RISC-V 64bit

Руководство пользователя

АННОТАЦИЯ

Настоящий документ содержит руководство пользователя для интегрированной среды разработки.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

1.1 Назначение программы

Интегрированная среда разработки (ИСР) предназначена для создания, выполнения и отладки программного обеспечения для процессоров на основе архитектуры RISC-V.

Режим разработки подразумевает возможность создания новых и модификации существующих программ на уровне исходных кодов на языках C/C++ и на языке ассемблера, компиляцию текста программы, сборку бинарного кода и запуск его на исполнение.

В режиме отладки готовый бинарный код запускается с помощью отладчика на симуляторе или на отладочной плате с целью тестирования и отладки кода программ.

1.2 Условия работы программы

ИСР реализована на архитектурах x86_64 под управлением операционных систем Windows 8 и Ubuntu Linux версии не ниже 16.04 LTS.

Для использования ИСР необходимо установить следующие компоненты:

1) среда выполнения Java SE Runtime Environment 1.8 (поставляется в составе пакета установки для Windows);
2) Microsoft Visual C++ 2013 Redistributable (поставляется в составе пакета установки для Windows).

1.3 Состав программы

В качестве основы для ИСР выбрана Eclipse Oxygen CDT - расширяемая среда разработки (Integrated Development Environment, далее IDE) с открытым исходным кодом (далее OpenSource) с набором расширяемых модулей (плагинов) CDT для разработки и отладки программ на языках C/C++.

Платформа Eclipse в сочетании с JDT включает многие из возможностей, которые входят в коммерческие IDE: редактор с подсветкой синтаксиса, инкрементальная компиляция кода, отладчик, навигатор классов, менеджеры файлов

и проектов, а также интерфейсы к стандартным системам контроля исходных текстов, таким как CVS и ClearCase.

Eclipse представляет собой единую открытую интегрированную платформу разработки приложений, обладающую надежностью, функциональностью и уровнем качества коммерческого продукта. Эта платформа представляет собой основу, имеющую блочную структуру и интегрирующую инструменты разработки ПО различных производителей для создания приложений на любом языке, с использованием любых технологий и для любой программной платформы.

Eclipse предлагает такие возможности, как рефакторинг кода, автоматическое обновление/установка кода (с помощью Менеджера Обновлений), список текущих задач, отладку модулей с помощью JUnit.

Несмотря на большое число стандартных возможностей, Eclipse отличается от традиционных IDE по ряду особенностей. Eclipse полностью независима от платформы и языка. В Eclipse реализована Plug-in архитектура расширений, а также богатый API, предоставляемый модулем PDE, позволяющий расширять возможности Eclipse. Добавление новой функциональности является достаточно простой операцией благодаря грамотно разработанным API и большим строительным блокам, предоставляемым Eclipse.

1.4 Структура файлов и каталогов ИСР

Корневая директории ИСР RISC-V Eclipse содержит следующие папки:

- configuration;
- drivers;
- doc
- features;
- riscv-ide;
- p2;
- plugins.

Для работы больше всего интересны следующие папки:

- *configuration* — в этом каталоге содержатся начальные настройки Eclipse и плагинов. Сюда же помещаются логи работы в случае неудачного запуска среды, отладки или иных операций;
- *doc* — содержит набор документации, необходимой разработчику;
- riscv-*ide* — каталог тулчейна. Для работы в IDE RISC-V Eclipse важны следующие папки:

  1) riscv-*ide/bin* – содержит бинарные файлы запуска, такие как компилятор, линковщик, симулятор, отладчик и прочие;

  2) riscv-*ide/lib* – содержит необходимые для работы библиотеки. В частности, здесь содержатся библиотеки сборки, предназначенной для запуска программы на плате;
- *plugins* – содержит установленные в IDE плагины в виде .jar-архивов или каталогов.

2. УСТАНОВКА И ОБНОВЛЕНИЕ

2.1 Установка программы

Установка ИСР осуществляется с помощью специального исполняемого файла – инсталляционного пакета.

Для ОС Windows файл имеет имя *RISC-V-SDK-<ver>.<link_num>.exe*, для ОС Linux это файл *risc-v_toolchain_installer-<ver>.sh*, где *<ver>* – версия ИСР, *<link_num>* – номер сборки.

Для установки ИСР под ОС Windows необходимо в командной строке или в программе «Проводник» запустить инсталлятор и далее следовать инструкциям инсталлятора.

Для установки ИСР под ОС Linux необходимо запустить в командной строке следующую команду:

sudo risc-v_toolchain_installer-<ver>.sh

2.2 Установка драйверов JTAG для работы с платой

По окончании установки ИСР будет предложено установить драйверы USB для работы с отладочной платой. При активации флажка после закрытия окна установки будет открыта утилита Zadig для выбора драйверов.

В утилите необходимо:

1) В меню *Options* выбрать пункт *List All Devices* (рисунок 2.1)
2) Выбрать устройство *J-Link* (рисунок 2.2)
3) В списке, помеченном на рисунке 2.2 красным овалом, выбрать драйвер *WinUSB* и нажать на кнопку *Reinstall Driver*
4) После успешной установки драйвера закрыть программу Zadig.

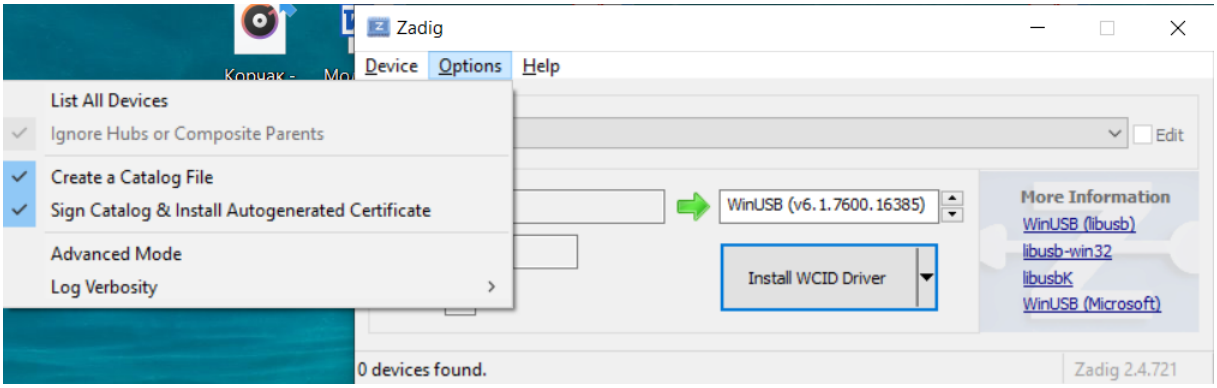

Рисунок 2.1. Выбор расширенного списка устройств

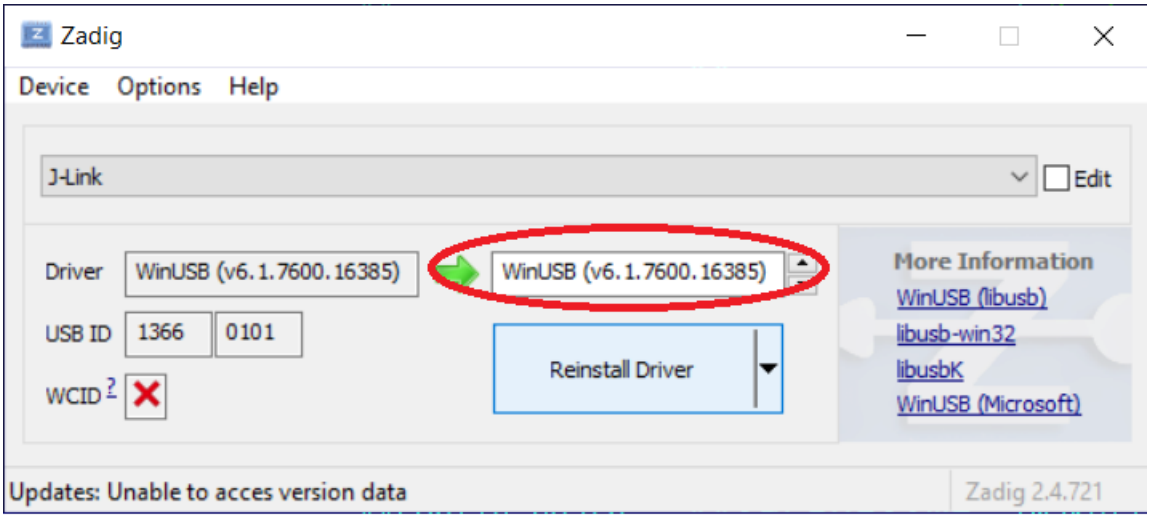

Рисунок 2.2. Выбор устройства и драйвера

3. ЗАПУСК ПРОГРАММЫ НА ИСПОЛНЕНИЕ

После установки программы можно приступить к первому запуску IDE Eclipse.

3.1 Запуск в ОС Windows

Для запуска среды Eclipse необходимо запустить программу *riscv-eclipse.exe* из каталога, в который была установлена программа, либо из меню «Пуск», выбрав следующие пункты:

Пуск → Все программы → NIIME → RISC-V-SDK → RISC-V Eclipse.

3.2 Запуск в ОС Linux

В ОС Linux запуск среды Eclipse можно осуществить из командной строки:

riscv-eclipse либо **/usr/local/bin/riscv-eclipse**

Примечание - для корректной работы отладки на устройстве может потребоваться запуск среды с правами Администратора:

sudo riscv-eclipse либо **sudo /usr/local/bin/riscv-eclipse**

3.3 Загрузка Eclipse

С момента запуска Eclipse действия оператора на любой платформе одинаковы. При запуске на экране появится логотип системы (рисунок 3.1).

Рисунок 3.1. Логотип системы Eclipse

При первом запуске Eclipse перед появлением самой среды выполняется ряд завершающих установочных шагов, например, создание рабочей директории (*workspace)* для хранения файлов проектов (рисунок 3.2).

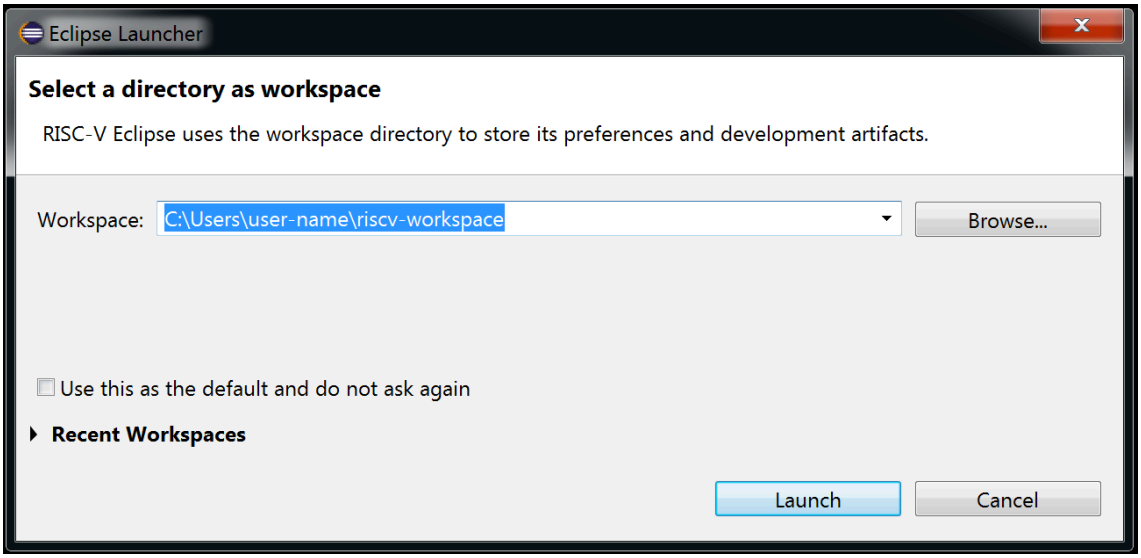

Рисунок 3.2. Создание рабочей директории для хранения файлов проектов

3.4 Оконный интерфейс Eclipse

Окно Eclipse после первого запуска будет аналогично рисунку 3.3.

Eclipse имеет стандартное меню (рисунок 3.4) и панель инструментов (рисунок 3.5).

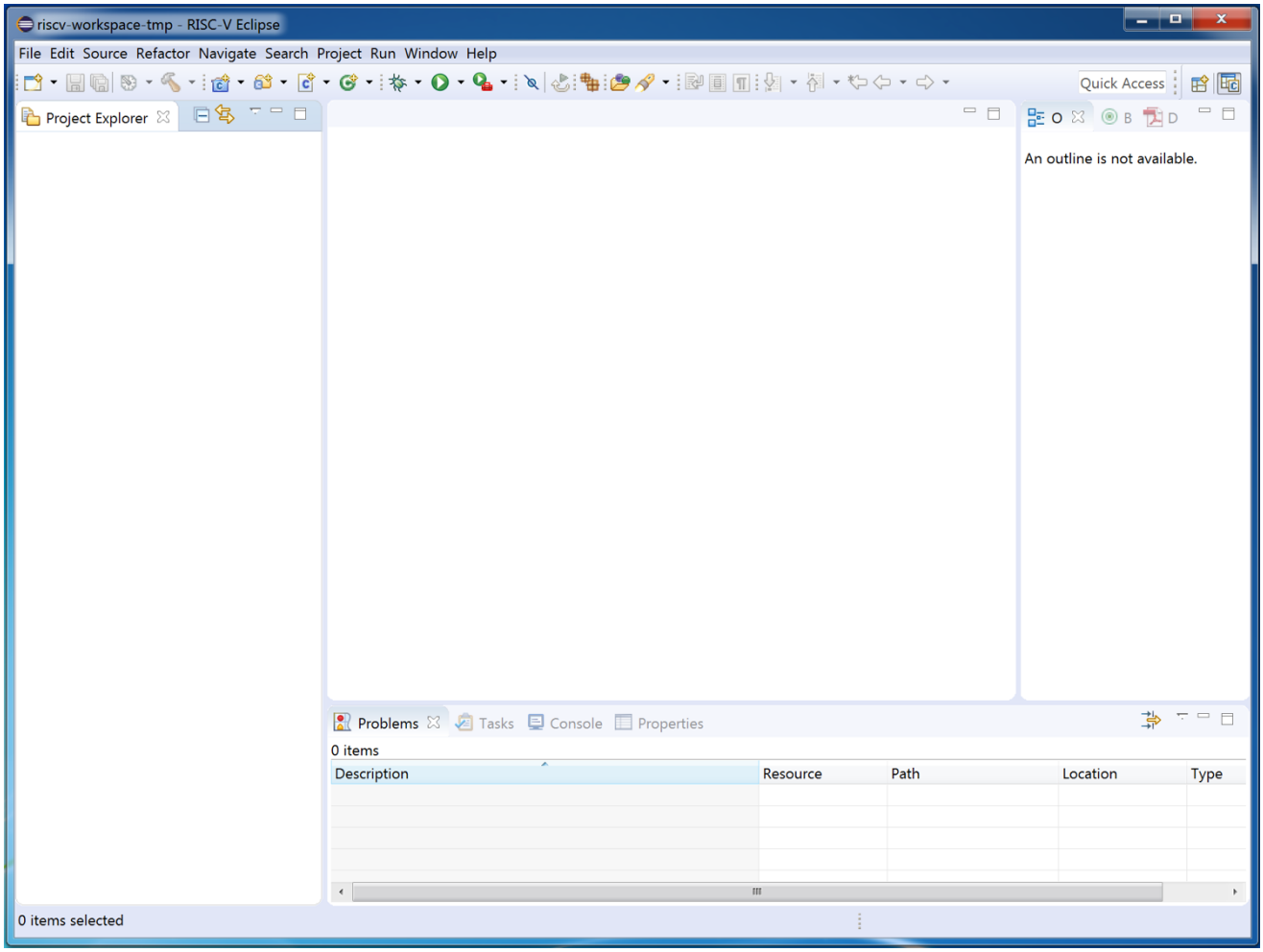

Рисунок 3.3. Первоначальный вид окна Eclipse

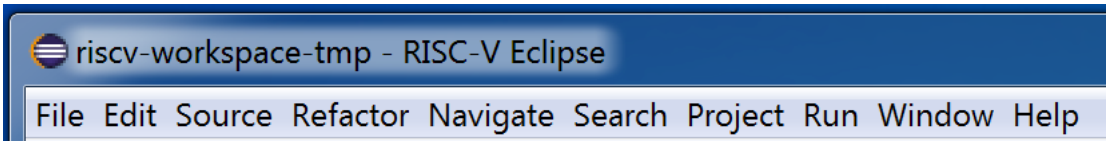

Рисунок 3.4. Меню системы Eclipse

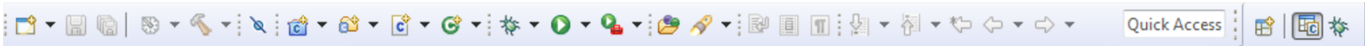

Рисунок 3.5. Панель инструментов системы Eclipse

Основными рабочими элементами среды являются:

– страница навигатора (рисунок 3.6);
– многостраничный редактор кода в центральной части экрана;
– панель уведомлений (рисунок 3.7).

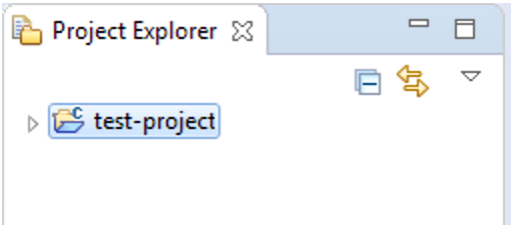

Рисунок 3.6. Страница навигатора системы Eclipse

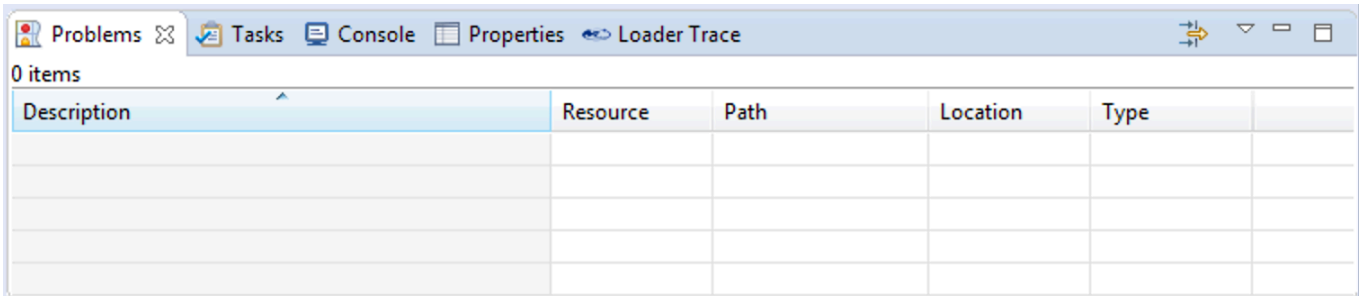

Рисунок 3.7. Панель уведомлений системы Eclipse

3.5 Создание и сборка проектов на языках C/C++

3.5.1 Создание нового проекта

Для создания нового проекта на языке C необходимо выбрать в меню *File→ New→ C/C++ Project*, затем выбрать *C Managed Build* (рисунок 3.8.1), либо *File→ New→ Project...*, а в появившемся окне выбрать *C/C++→ C Project* (рисунок 3.8.2). Для создания C++-проекта необходимо проделать те же действия, вместо *C Project* выбрав *C++ Project*.

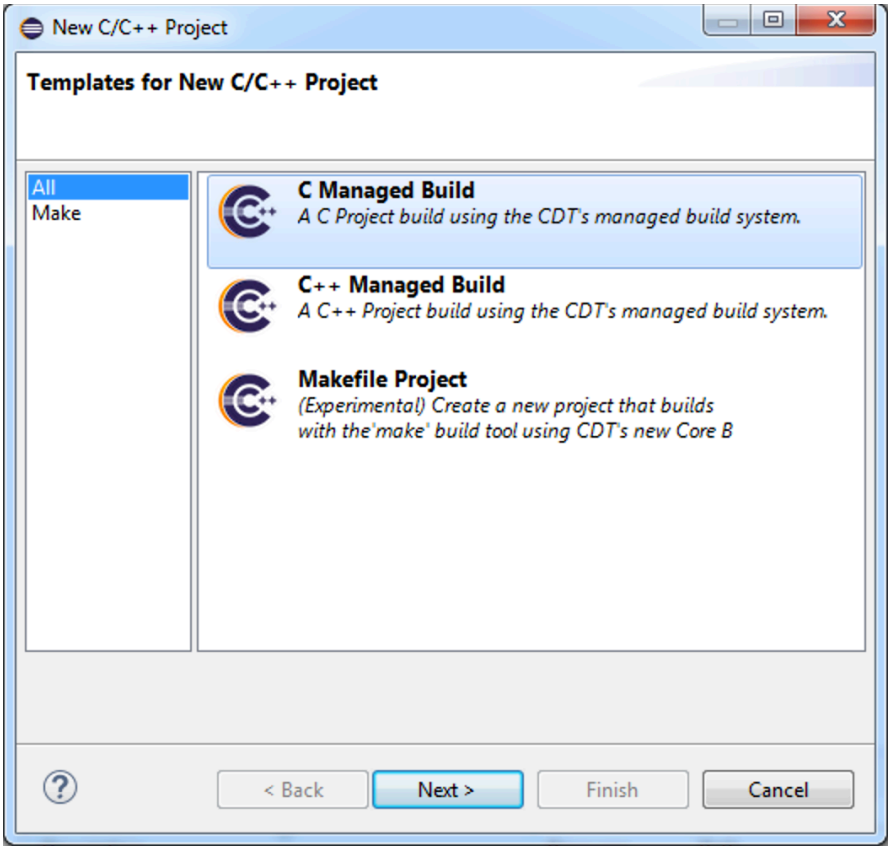

Рисунок 3.8.1. Создание нового С-проекта, вариант 1

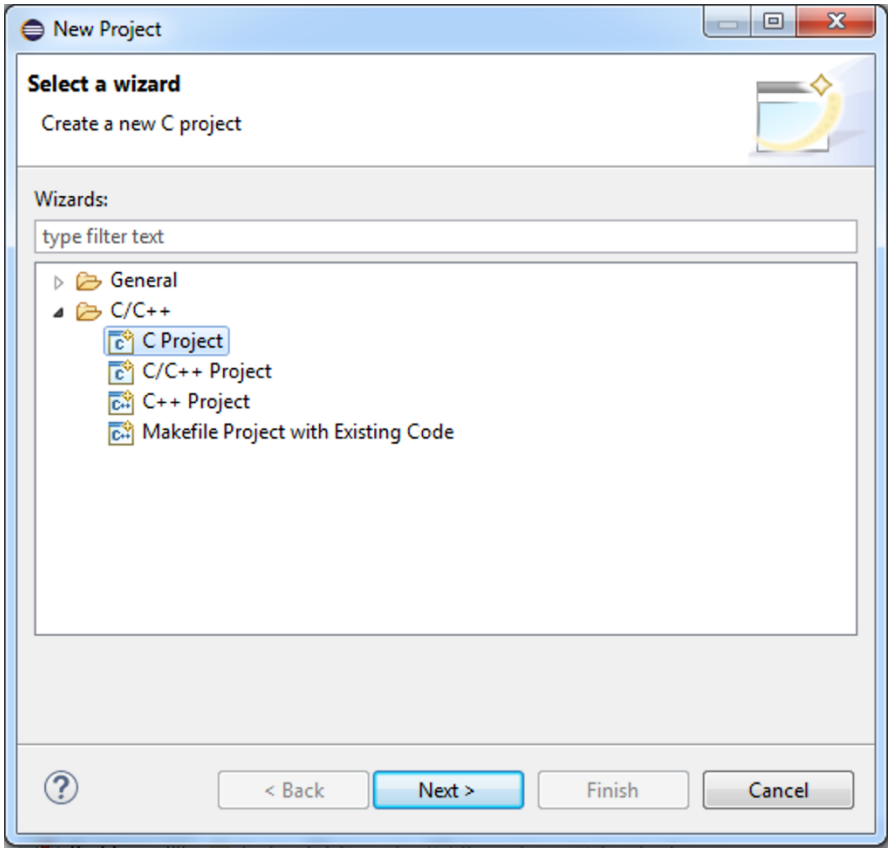

Рисунок 3.8.2. Создание нового С-проекта, вариант 2

В появившемся окне (рисунок 3.9) выбрать *Executable→ Empty Project*, задать имя проекта и выбрать в качестве используемого тулчейна *RISC-V Cross GCC*. Затем нажать *Next→ Next*.

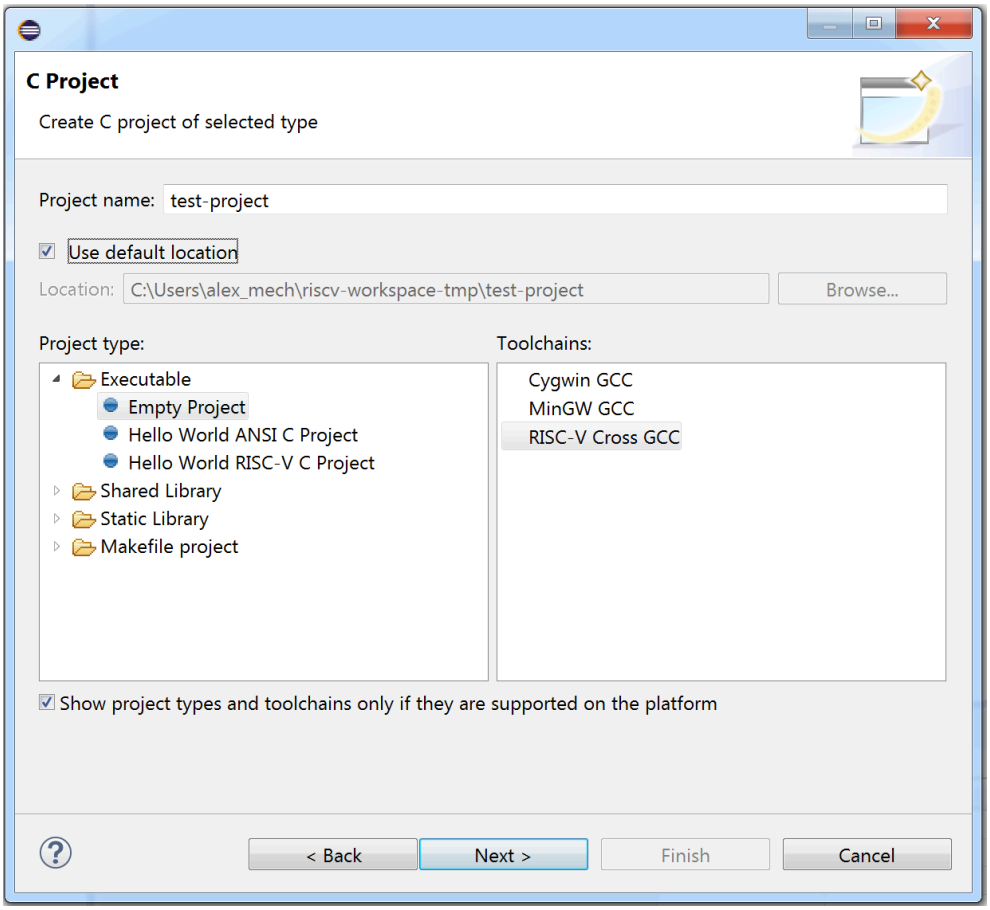

Рисунок 3.9. Окно создания нового проекта

В следующем окне требуется задать тип (имя) и директорию расположения тулчейна (рисунок 3.10). Тип (имя) тулчейна следует оставить значением по умолчанию, *RISC-V GCC/Newlib (riscv64-unknown-elf-gcc)*. Поле директории расположения тулчейна заполняется автоматически при условии установки RISC-V SDK в стандартный путь (например, *C:\Program Files\NIIME\RISC-V-SDK*), в таком случае расположение тулчейна будет следующим: *C:\Program Files\NIIME\RISC-V-SDK\riscv-ide\bin*. Обратите внимание, что путь установки тулчейна всегда должен оканчиваться на “*riscv-ide\bin*”.

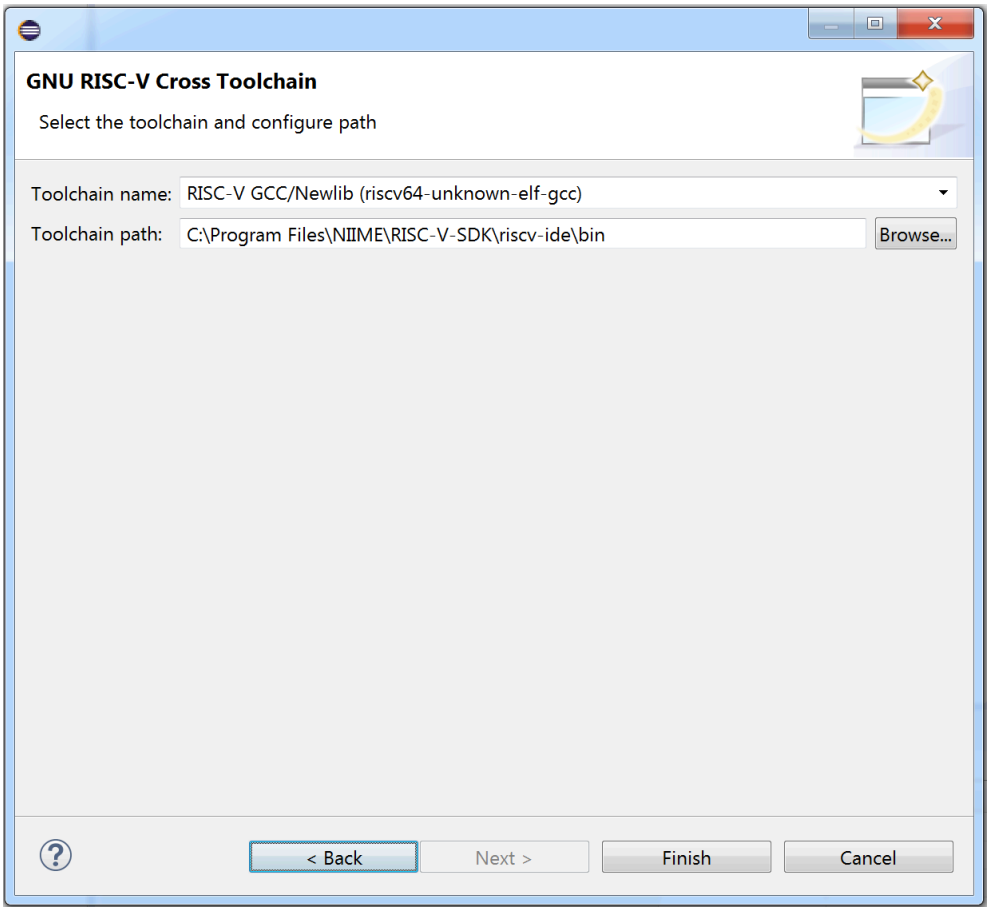

Рисунок 3.10. Окно типа и расположения тулчейна

После заполнения пути нажать кнопку *Finish* для завершения создания проекта. На запрос о смене текущей перспективы ответить *Yes*.

3.5.2 Импорт существующего проекта

Для добавления уже существующего проекта в текущий workspace необходимо выбрать в меню *File→ Import…*, затем выбрать *General→ Existing Projects into Workspace* (рисунок 3.9.1), затем нажать *Next.*

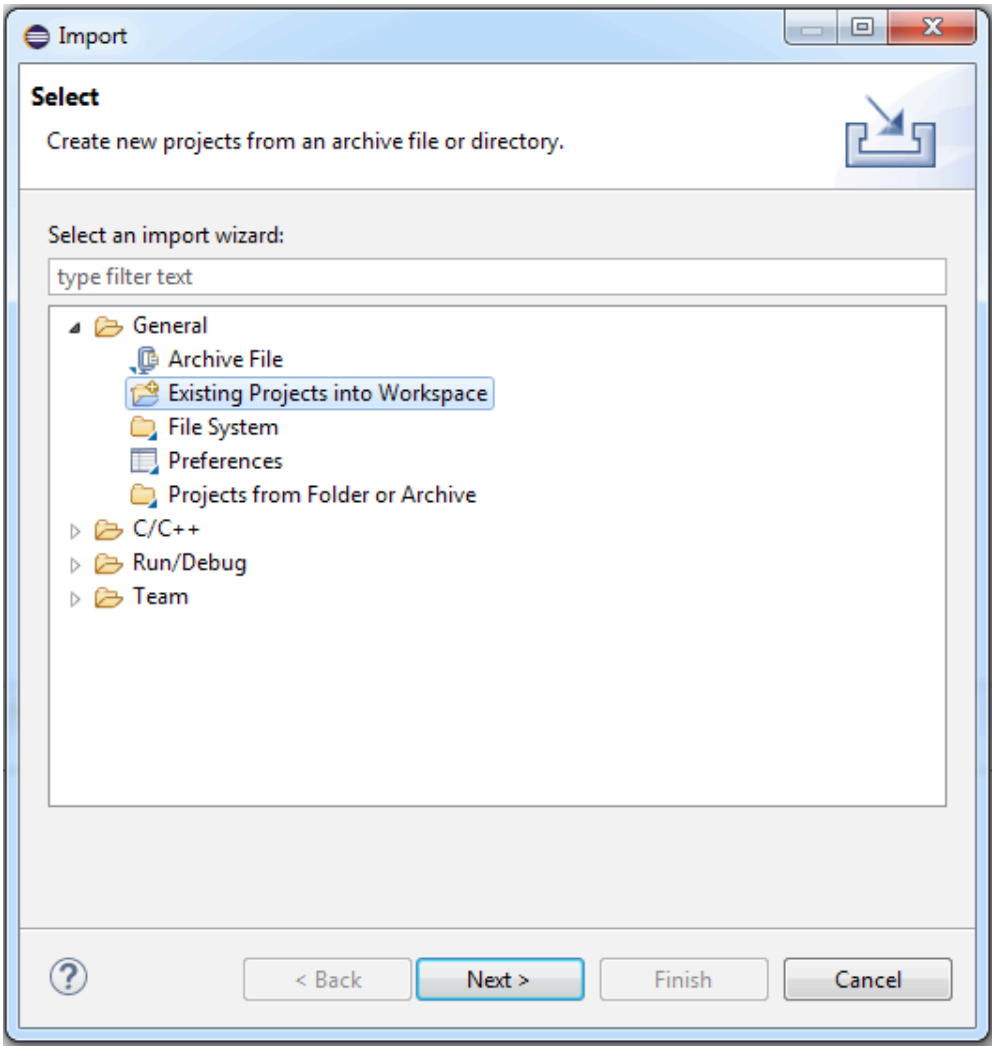

Рисунок 3.9.1. Импорт существующего проекта

В следующем окне введите путь к директории, где содержится папка с проектом, нажмите *Enter*. Затем в списке ниже выберите проект(ы) для импорта (при необходимости создать свою копию проекта в текущем workspace поставьте флажок *Copy projects into workspace* и нажмите *Finish* (см. рисунок 3.9.2).

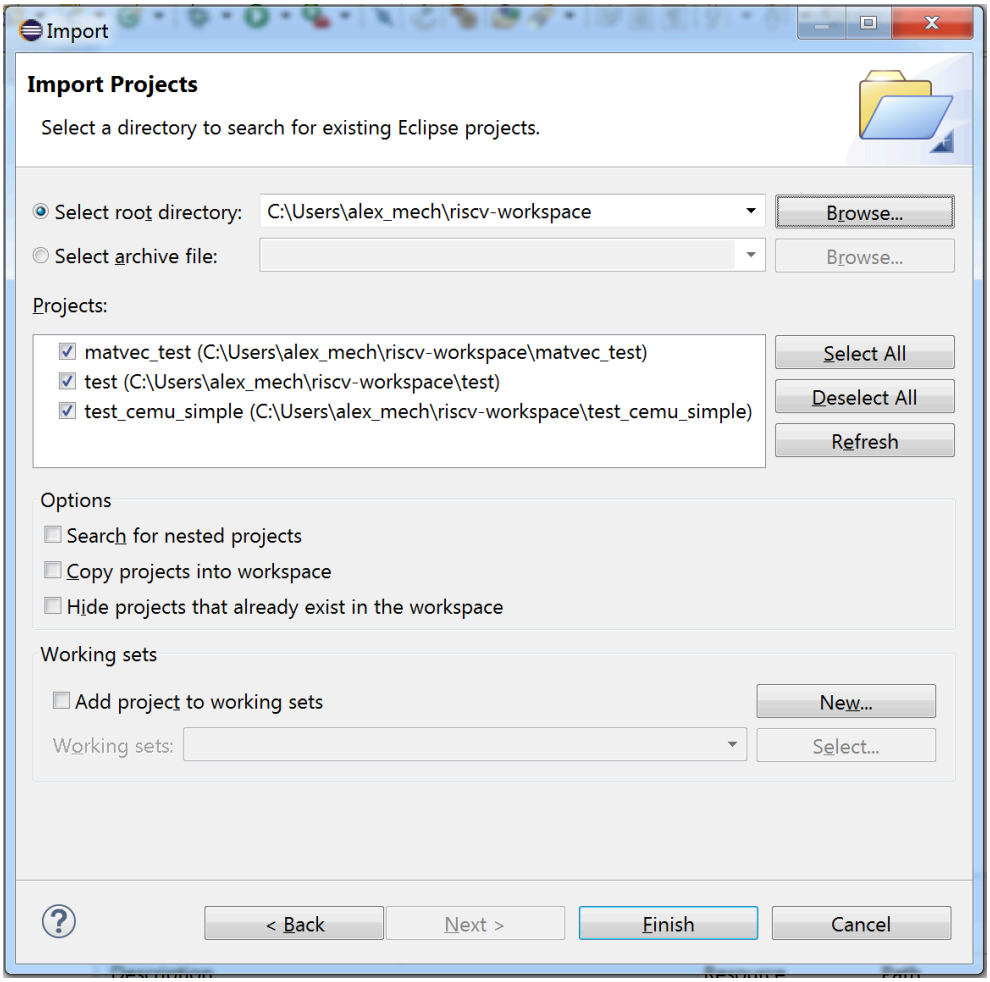

Рисунок 3.9.2. Поиск проектов для импорта

3.5.3 Добавление файлов в проект

Добавить файл исходного кода в проект можно несколькими способами.

Простейшим способом добавления нового файла в проект является выбор проекта правой кнопкой мыши (далее – ПКМ), затем выбор *New→ File / Source File / Header File*, в зависимости от назначения (см. рисунок 3.10).

Добавить уже существующий файл в проект можно либо перетащив его мышкой из любого файлового браузера, например, из Проводника Windows, либо импорт с помощью *ПКМ→ Import→ File System*.

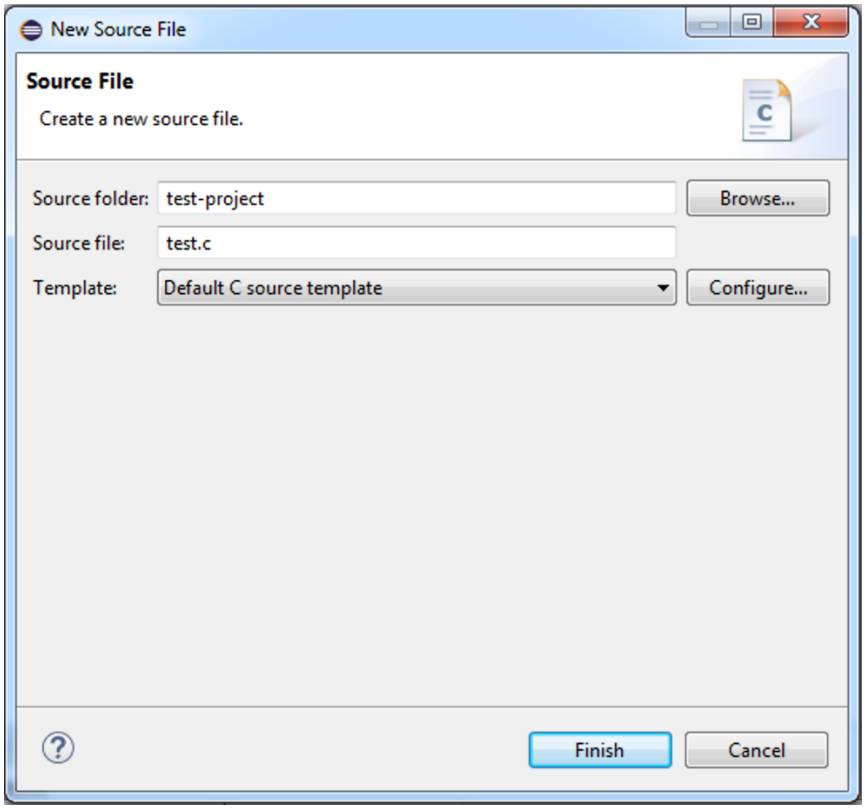

Рисунок 3.10. Добавление нового файла в проект

Для создания нового файла введите его имя в строке имени. Следует добавить расширение **.c** к имени файла в случае C проекта, либо **.cpp** в случае C++ проекта.

3.5.4 Сборка проекта

Для каждого нового проекта автоматически создаются две конфигурации сборки:

– *Debug* (сборка для отладки);

– *Release* (сборка для запуска без отладки);

Выбрать конфигурацию сборки можно следующим образом: выбрать проект в *Project Explorer*, затем в главном меню выбрать *Build Configurations*→ *Set Active*→ *<имя_сборки>*. Альтернативно, сборку нужной конфигурации можно выбрать с помощью выпадающего меню кнопки Build, изображенной на рисунке 3.11.

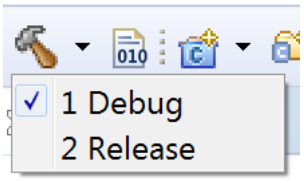

Рисунок 3.11. Доступные конфигурации проекта для сборки

Для непосредственной сборки нужно выбрать проект в *Project Explorer* и затем в главном меню выбрать *Project→Build Project*.

После запуска сборки информация о состоянии появляется во вкладке *Console* (рисунок 3.12). В случае удачной сборки в консоли появится сообщение *Build Finished*.

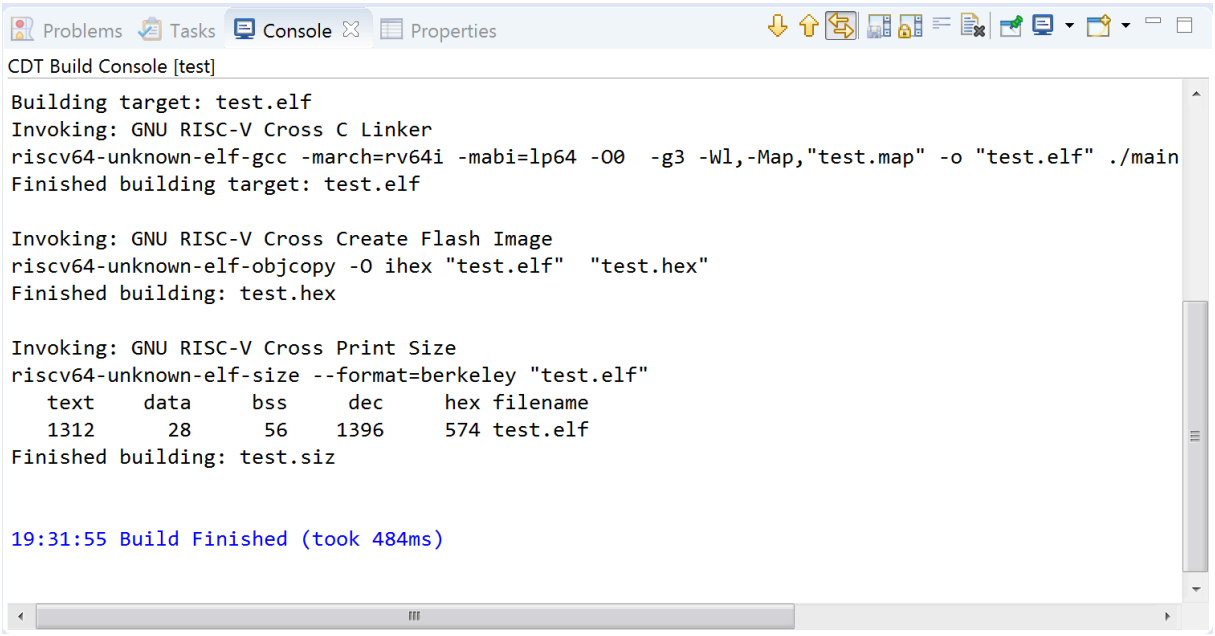

Рисунок 3.12. Окно консоли после успешной сборки проекта

Кроме того, в окне *Project Explorer* в папке проекта должны появиться бинарные файлы (группа *Binaries* с файлами внутри), а также каталог с именем выбранной конфигурации сборки (рисунок 3.13).

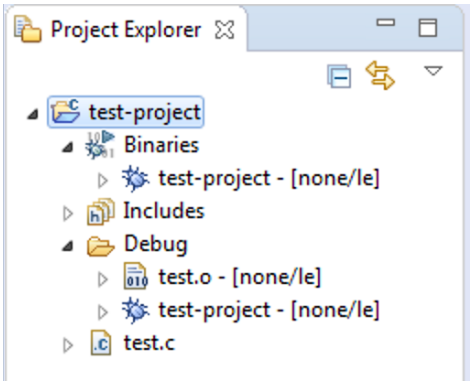

Рисунок 3.13. Окно Project Explorer после успешной сборки проекта

3.5.5 Настройка параметров сборки

Пользователь может настроить отдельные параметры сборки для каждой конфигурации. Изменения включают добавление различных ключей на вход компонент тулчейна, подключение библиотек, оптимизационных флагов, выбор стандарта языка и многое другое.

Для настройки параметров сборки выбрать во вкладке *Project Explorer* проект, затем правая кнопка мыши→ *Properties*→ *C/C++ Build*→ *Settings*.

Настройки сборки расположены во вкладке *Tool Settings*. Выставляя нужные настройки, можно изменить параметры сборки, среди них: *GNU RISC-V Cross C/C++ Compiler, GNU RISC-V Cross Assembler, GNU RISC-V Cross C/C++ Linker* (рисунок 3.14.1).

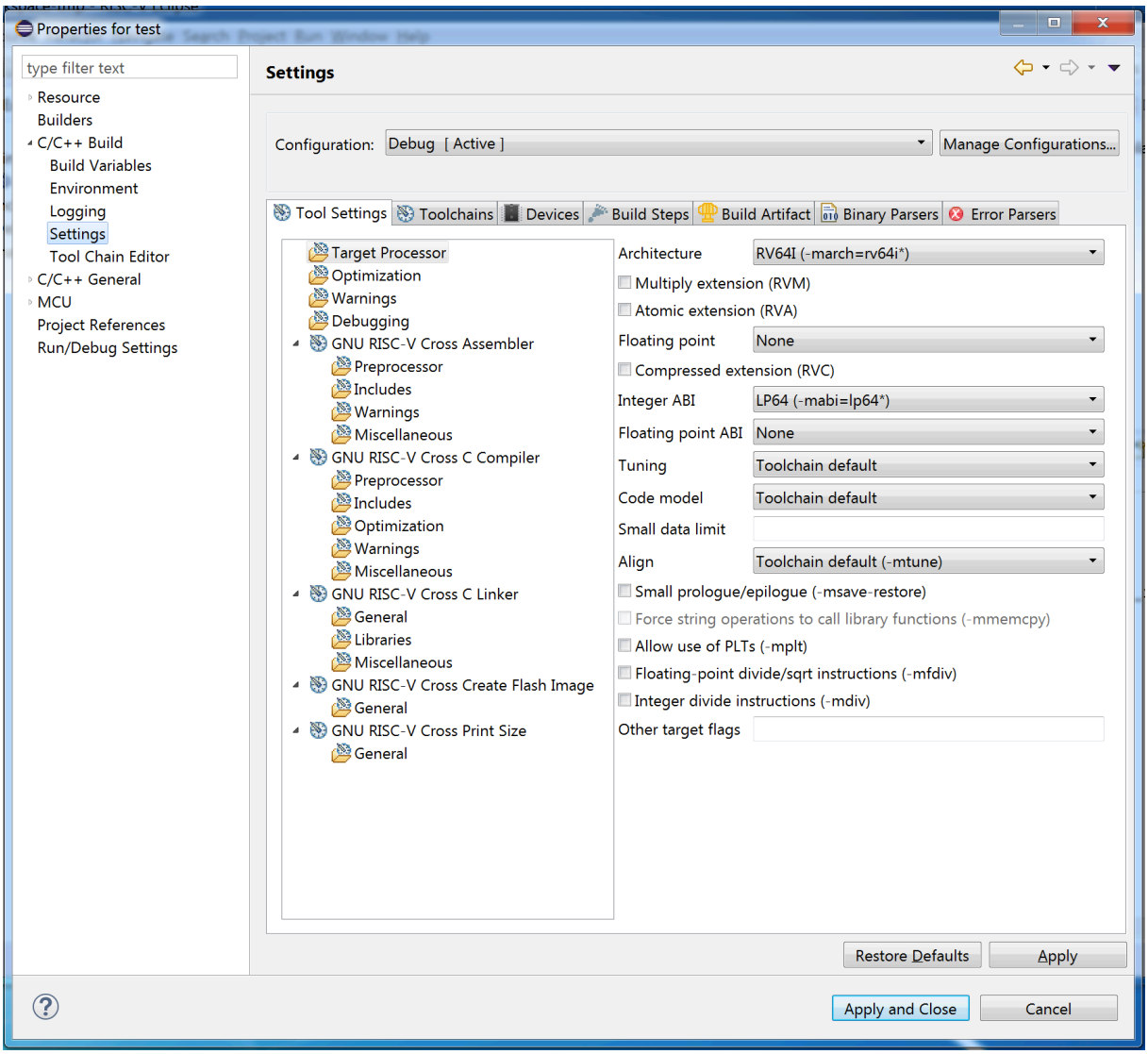

Рисунок 3.14.1. Настройки параметров сборки проекта

На рисунке приведены параметры для C-проекта. Для C++-проекта окна имеют аналогичный вид.

В этой вкладке расположены настройки параметров линкера, компилятора и ассемблера. Сразу заметим, что во вкладках *GNU RISC-V Cross C/C++ Linker, GNU RISC-V Cross C/C++ Compiler, GNU RISC-V Cross Assembler* находится строка запускаемой команды (изменяемое поле *Command*), неизменяемое поле *All options* (в нее помещаются все опции, которые подаются на вход компоненты) и поле *Command line pattern*, в которой прописан шаблон формирования запускаемой команды.

Кроме того, стоит обратить внимание на содержимое вкладки *Properties→ C/C++ Build→ Environment*, где расположен набор переменных окружения, необходимых для сборки и запуска проектов (рисунок 3.14.2).

На этой вкладке можно изменять существующие, а также добавлять собственные переменные, которые будут доступны практически из всех полей проекта (настройки сборки, конфигурации отладки и запуска, исходный код и т.д.).

Настоятельно не рекомендуется изменять значения существующих переменных (*GCC_EXEC_PREFIX, PATH*), т.к. их изменение или удаление может повлечь за собой ошибки при сборке и запуске проектов.

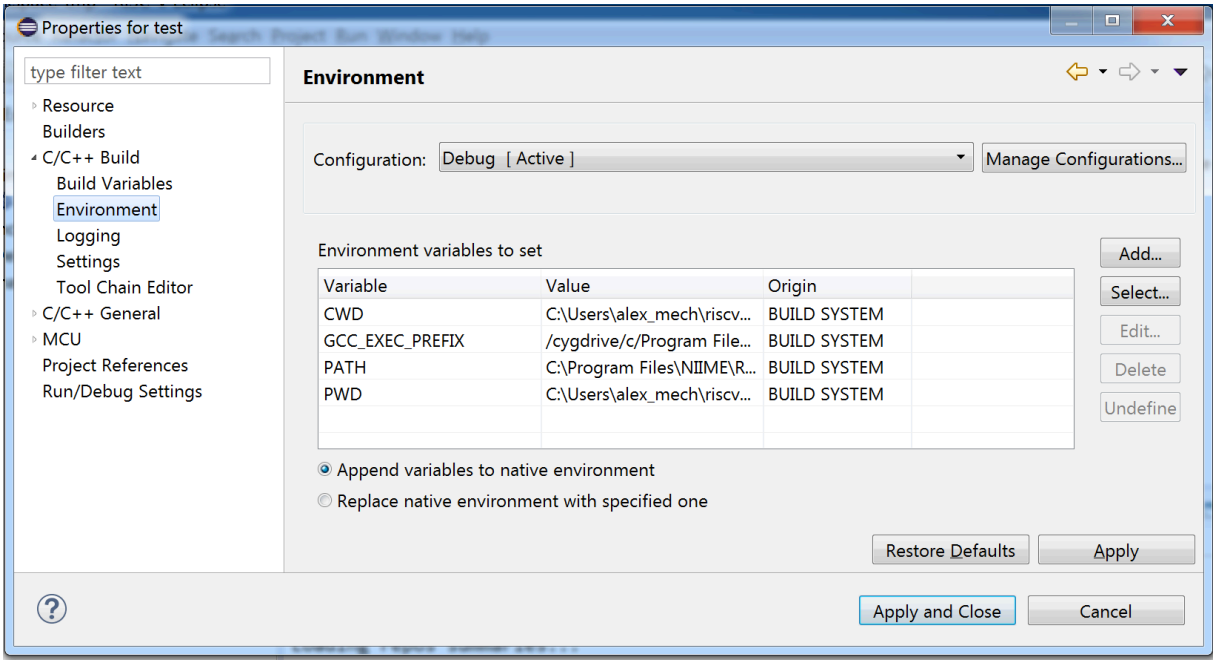

Рисунок 3.14.2. Настройка переменных окружения проекта

3.5.6 Экспорт проекта

Существует несколько способов осуществлять экспорт проектов Eclipse для обмена ими с другими разработчиками.

Первый способ заключается в компоновке проектов в архив или внешнюю папку для последующей их удобной передачи. Для этого необходимо:

1. В главном меню выбрать *File→ Export…*
2. В появившемся окне выбрать *General→ Archive File* (если нужно создать архив) или *General→ File System* (если нужно выполнить экспорт в папку)

3. Выбрать проекты и файлы для экспорта
4. Настроить прочие параметры экспорта (тип архива, каталог экспорта и проч.)
5. Нажать кнопку *Finish*.

Второй способ подходит в случае использования систем контроля версий (Git/SVN) для обмена проектами. В таком случае допустимо сохранять в репозитории рабочий каталог workspace, но необходимо исключить из отслеживания ненужные файлы, среди них:

- Каталог .**metadata** (временные файлы workspace конкретной машины)
- Каталог **.settings** (локальные настройки проектов, не относящиеся к сборке)
- Файлы сборки проектов (каталоги конфигураций сборок)

Таким образом, файлы, входящие в коммиты, могут включать:

- Файлы исходного кода проектов
- Файлы-описания проектов **.project**
- Файлы настроек сборки проектов **.cproject**
- Файлы конфигураций **.classpath**
- Файлы конфигураций запуска **XXX.launch** (в случае сохранения их во внешний файл)

3.6 Отладка проекта с помощью симулятора Ergochip

3.6.1 Запуск отладки

Для запуска отладки необходимо создать конфигурацию отладки. Для этого в главном меню необходимо выбрать *Run*→*Debug Configurations…*(рисунок 3.15).

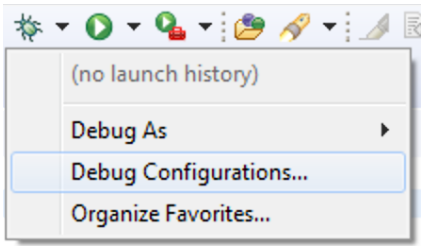

Рисунок 3.15. Вызов окна конфигураций отладки

В открывшемся окне нужно создать новую конфигурацию. Для отладки в симуляторе нужно создать конфигурацию типа *GDB QEMU-TLM Debugging*, для отладки на плате — *GDB OpenOCD Debugging* (рисунок 3.16). Для создания требуемой конфигурации кликните на нее два раза левой кнопкой мыши либо нажмите на нее правой кнопкой мыши, а затем выберите *New*.

Перед запуском сеанса отладки необходимо проверить настройку конфигурации – в поле *C/C++ Application* вкладки *Main* проверить корректность имени исполняемого файла, а также проверить содержимое полей на вкладке *Debugger*, при необходимости включить дополнительные опции симуляции.

Для запуска отладки нужно нажать на кнопку *Debug*. После этого начнется сеанс отладки.

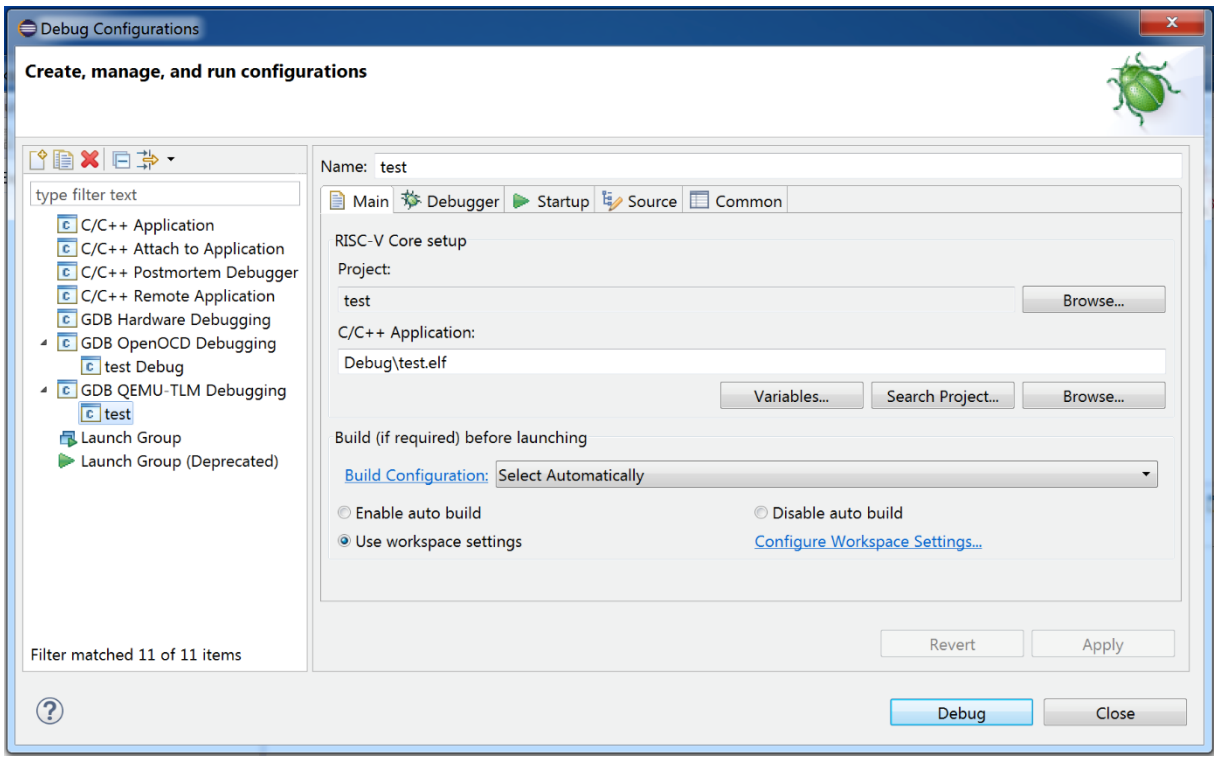

Рисунок 3.16. Создание новой конфигурации отладки

3.6.2 Запуск без отладки

Для запуска программы без отладки (например, при сборке в конфигурации *Release*) необходимо создать конфигурацию запуска. Для этого в главном меню необходимо выбрать *Run*→*Run Configurations*… (рисунок 3.17).

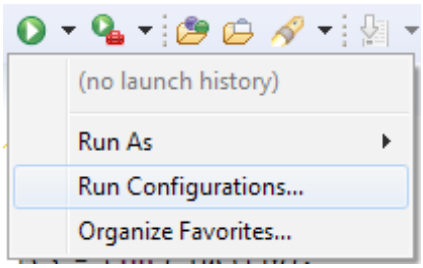

Рисунок 3.17. Вызов окна конфигураций запуска

В открывшемся окне нужно создать новую конфигурацию. Для создания требуемой конфигурации кликните на нее два раза левой кнопкой мыши либо нажмите на нее правой кнопкой мыши, а затем выберите *New*.

Перед запуском программы необходимо проверить настройку конфигурации – в поле *C/C++ Application* вкладки *Main* проверить корректность имени исполняемого файла.

Для запуска нужно нажать на кнопку *Run*. После этого будет произведен запуск программы на симуляторе.

Подробнее о способе вывода информации программы на экран см. п. 3.6.3.

3.6.3 Отладка

После нажатия на кнопку *Debug* в окне *Debug Configurations* может появиться окно с предложением изменить текущую перспективу на *Debug perspective* (рисунок 3.19). Нужно ответить *Yes*.

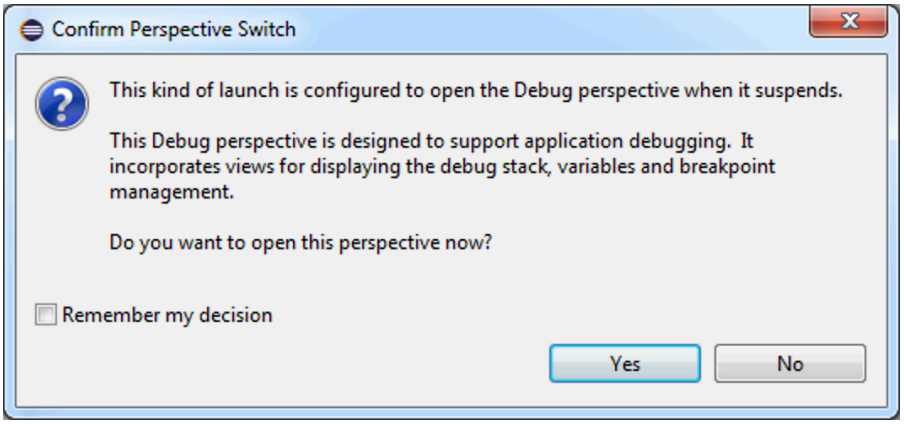

Рисунок 3.19. Смена перспективы при запуске отладки

Существует две возможности проведения сеанса отладки. Первая – это проход по инструкциям языка С. Вторая - по инструкциям ассемблера в окне дизассемблера. По умолчанию отладка осуществляется первым методом.

Для отладки можно проставить контрольные точки в окне с именем рассматриваемого файла во вкладке Debug, дважды кликнув в начале нужной строчки кода, а затем нажимать на кнопку *Resume* для прохода по контрольным точкам (рисунок 3.20). Контрольные точки могут быть выставлены и в перспективе C/C++.

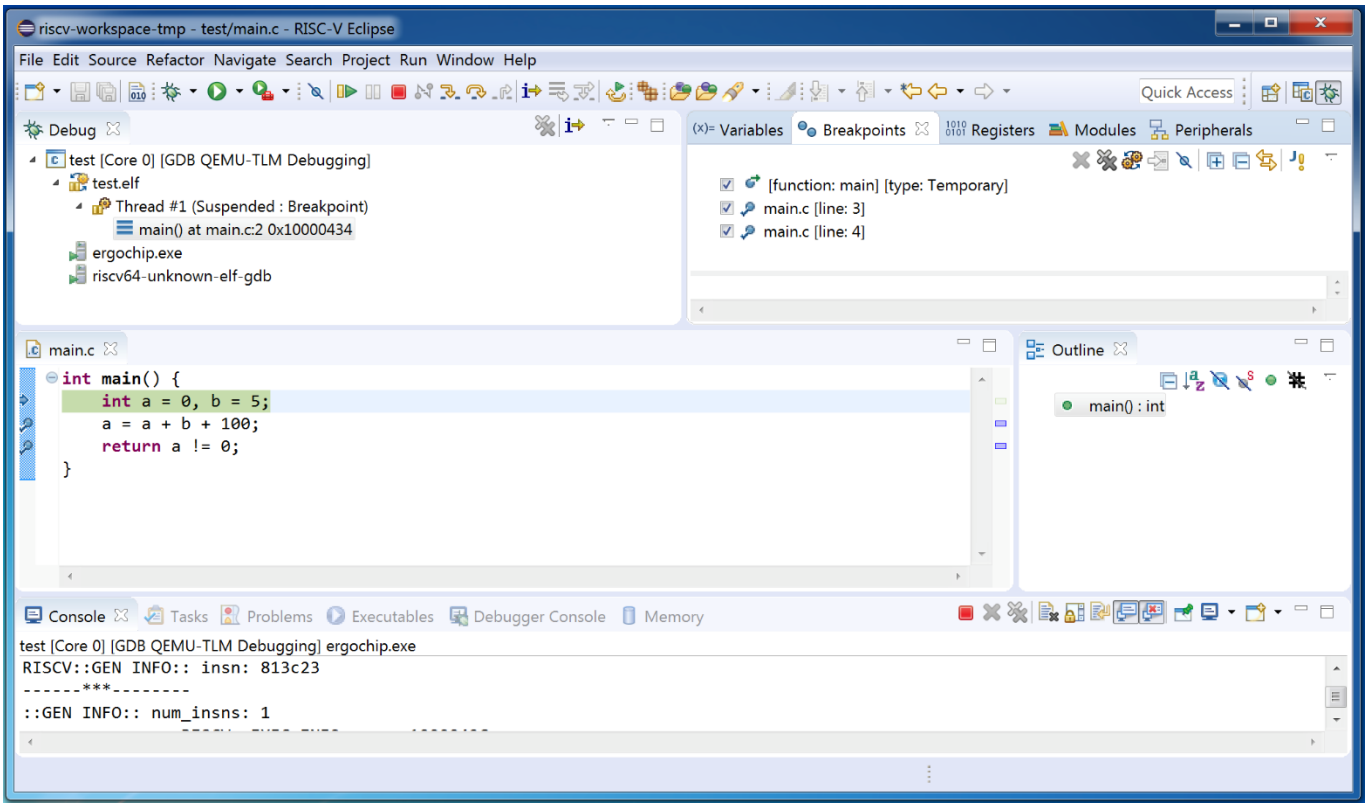

Рисунок 3.20. Программа с контрольными точками

Во вкладке *Breakpoints* отображаются номера строк или функций, в которых установлены контрольные точки (рисунок 3.21).

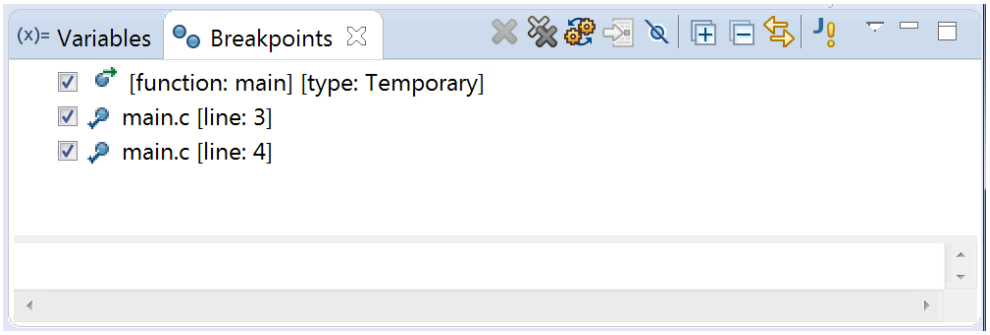

Рисунок 3.21. Список контрольных точек

Если требуется удалить контрольную точку, следует нажать на рассматриваемую контрольную точку во вкладке *Breakpoints*, а после нажать на кнопку *Remove Selected Breakpoint* ( ), либо дважды щелкнуть на нужную контрольную точку слева от нужной строчки программы.

После того, как контрольные точки выбраны, нажать на кнопку *Resume*. В результате произойдет переход на первую контрольную точку. С каждым нажатием кнопки *Resume* будет осуществляться переход на следующую контрольную точку вплоть до завершения программы.

Понять, на какой контрольной точке вы находитесь, можно из окна *Debugger Console*.

Также можно осуществлять проход по инструкциям ассемблера, для этого контрольные точки нужно выставлять в окне дизассемблера.

Примечание - вкладка Disassembly может быть отключена по умолчанию, для ее отображения следует выбрать в главном меню Windows→ Show View→ Disassembly.

Для прохода по инструкциям дизассемблера нажать на значок Instruction Stepping Mode ( ) и в появившемся окне дизассемблера поставить контрольные

точки двойным щелчком по нужной строчке. Процесс отладки осуществляется так же, как и в случае отладки по С-инструкциям (рисунок 3.22).

Вывод на экран информации из программы осуществляется в консоль среды, которое открывается во время отладки (рисунок 3.23).

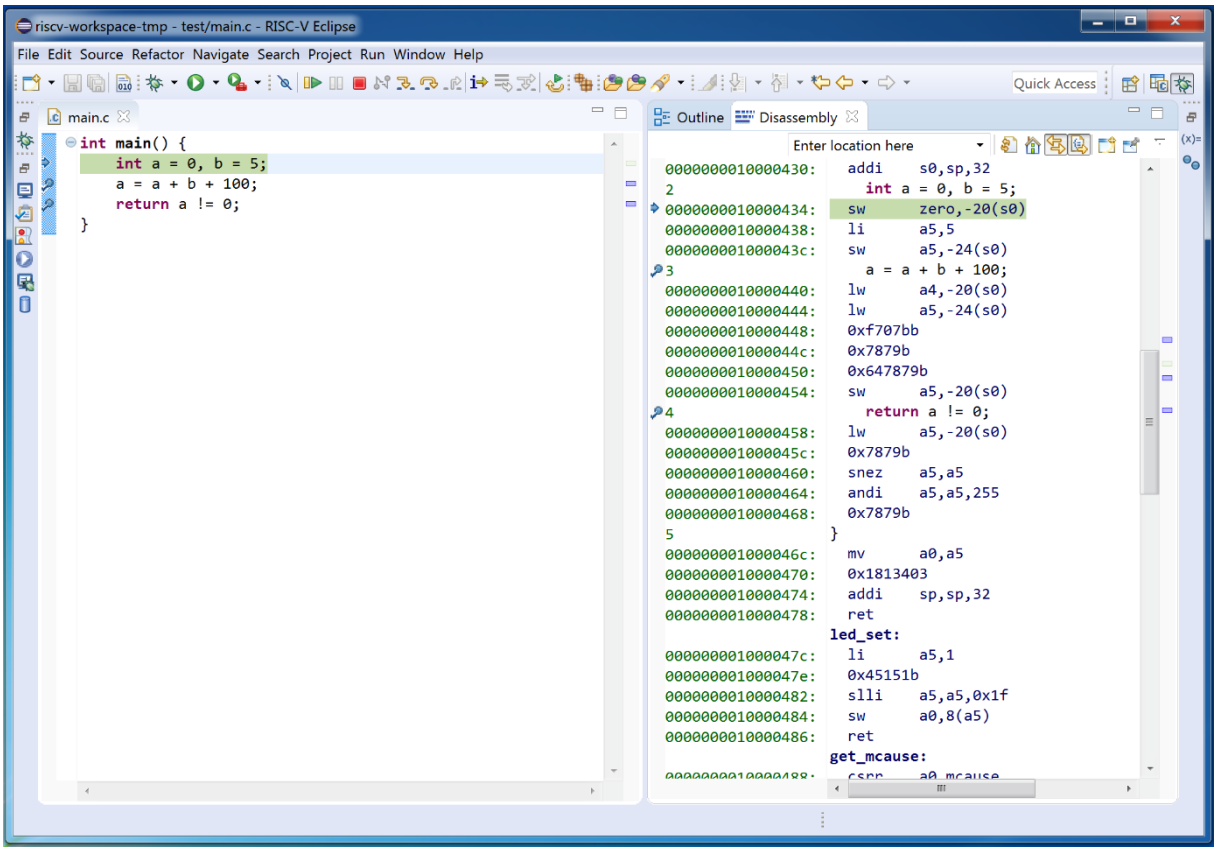

Рисунок 3.22. Отладка программы на ассемблере

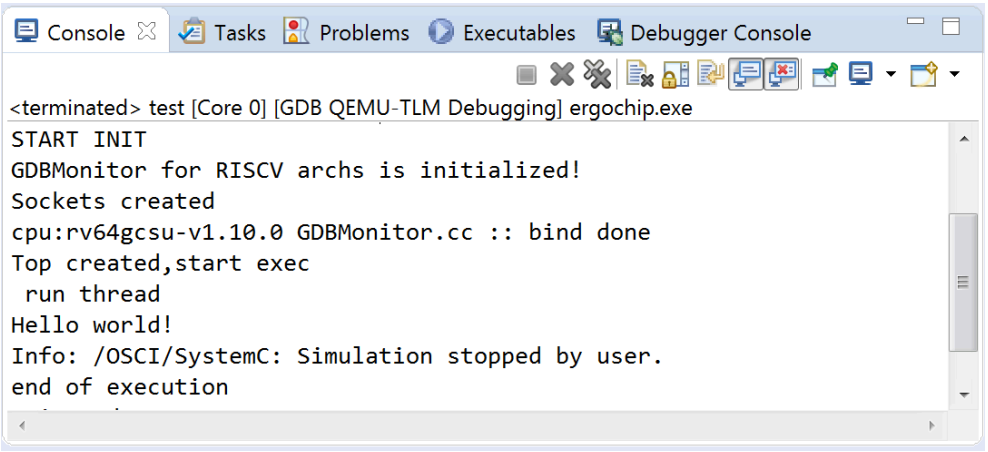

Рисунок 3.23. Консоль Eclipse с выводом Ergochip

3.7 Отладка проекта с помощью отладочной платы

Процесс запуска отладки на плате отличается от отладки на симуляторе тем, какая конфигурация отладки необходима.

Перед началом отладки необходим подключить аппаратный отладчик J-Link. J-Link имеет два интерфейса: USB для подключения к ПК и JTAG для подключения к плате. На плате имеется разъем JTag, к которому подключается отладочная плата. Кроме J-Tag отладчика к плате рекомендуется подключить кабель USB-Micro-USB (USB – к ПК, MicroUSB – к плате в разъем, обозначенный как UART). В случае отсутствие кабеля к плате необходимо подключить питание (поставляется в комплекте), а также переставить разъем USB Power… После подключения питания необходимо включить питание, переключив переключатель в положение On.

Выбрав в главном меню *Run→ Debug Configurations…,* нужно создать конфигурацию *GDB OpenOCD Debugging*.

Во вкладке *Main* нужно выбрать запускаемый проект и запускаемую программу. Во вкладке *Debugger* можно настроить дополнительные параметры запуска (рисунок 3.24). После этого можно начать процесс отладки (кнопка *Debug*).

Во всем остальном процесс отладки на плате совпадает с процессом отладки на симуляторе.

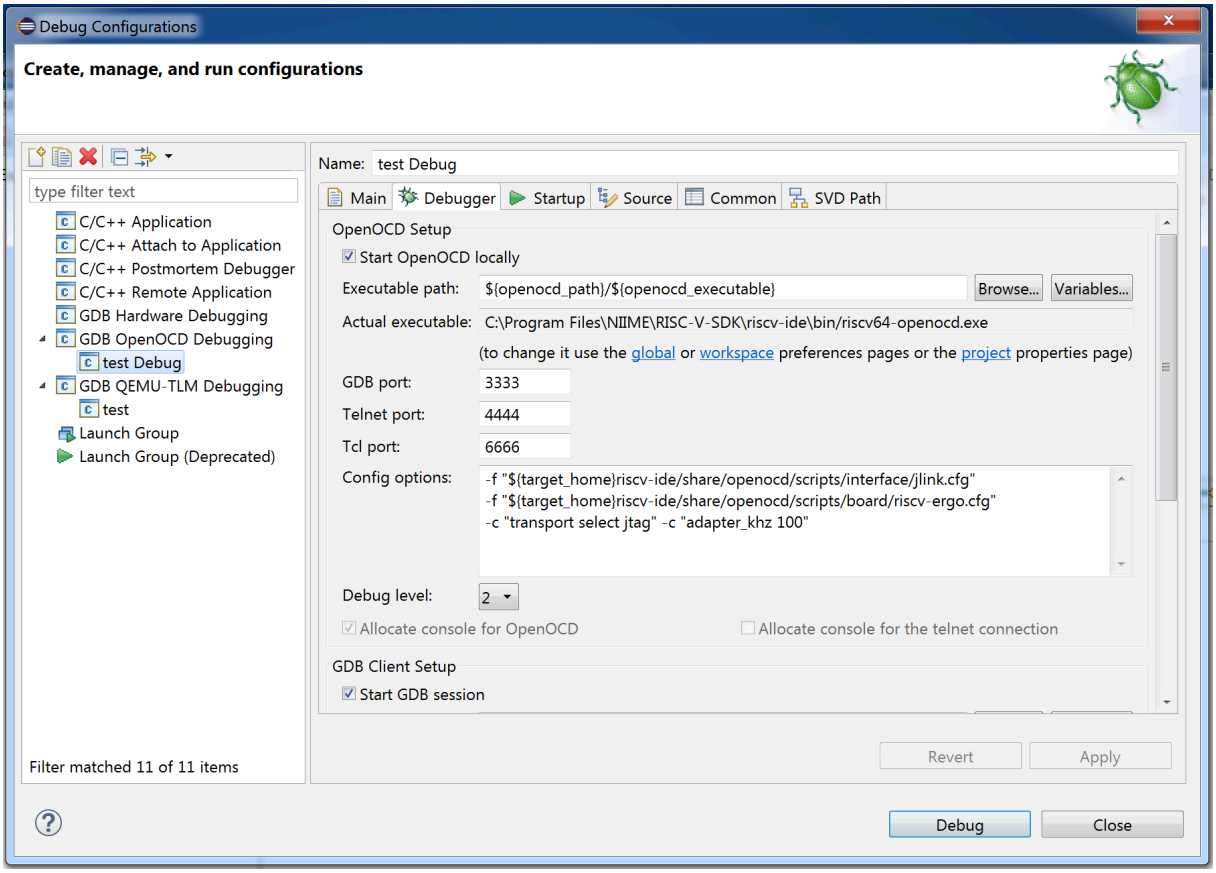

Рисунок 3.24. Конфигурация отладки для работы с платой

3.8 Работа с компонентами тулчейна из консоли Windows

В некоторых случаях необходимо работать с компонентами тулчейна (компилятором, отладчиком и прочими утилитами) не из Eclipse CDT, а из консоли. В случае работы в ОС Windows требуется запускать специализированную консоль *RISC-V-SDK Command Prompt*.

Для запуска необходимо открыть

Пуск → Все программы → NIIME → RISC-V-SDK → RISC-V-SDK Command Prompt.

В открывшейся консоли разработчика будут проинициализированы все переменные и назначены все пути, необходимые для корректной работы с компонентами тулчейна.

Если требуется запустить какую-либо утилиту или компонент, то это необходимо делать в данной консоли. Консоль предназначена для запуска исключительно из меню Пуск.

Если требуется произвести сборку проекта, используя Makefile, то вышеуказанных манипуляций не потребуется, если в конфигурацию Makefile внести следующую строку:

include $(RISCV_PATH)\riscv-ide\include\riscvvars_win.mk

где *$(RISCV_PATH)* – путь к корню установки RISC-V SDK, заполняется вручную.

3.9 Работа с компонентами тулчейна из терминала Linux

Если требуется осуществлять работу с компонентами в ОС Linux, то никаких дополнительных назначений в путях не требуется, поскольку расположение тулчейна в Linux фиксировано и не вызывает коллизий с прочими утилитами системы.

3.10 Сборка проекта Eclipse из консоли

Помимо сборки непосредственно из графической среды Eclipse имеется также возможность осуществить сборку или очистку проекта из консоли.

Для этого необходимо:

1. Запустить терминал:

a. ОС Windows: *Пуск → Все программы → NIIME → RISC-V-SDK → RISC-V-SDK Command Prompt*
b. ОС Linux: запустить обычный терминал

2. Запустить **eclipsec.exe** (Windows) или **riscv-eclipse** (Linux) с необходимыми параметрами:

Аргумент	**Описание**
-nosplash	Отключение показа экрана загрузки
--launcher.suppressErrors	Направление вывода ошибок в консоль вместо показа графических окон
-application org.eclipse.cdt.managedbuilder.core.headlessbuild	Обязательный аргумент для активации сборки без загрузки IDE
-data {workspaceDir}	Путь к workspace
-import {projectDir}	Импорт проекта по указанному пути в указанный workspace. Можно использовать несколько раз для импорта нескольких проектов
-build {projectName[/configName] | all}	Сборка проекта с указанным именем, (*projectName*) либо всех проектов в указанном workspace (*all*). По умолчанию, этот параметр собирает каждую конфигурацию каждого проекта. Можно ограничиться сборкой одной конкретной конфигурации (напр., *Debug* или *Release*), указав имя конфигурации после имени проекта, разделив их через ‘/’. Можно использовать несколько раз для сборки нескольких проектов
-cleanBuild {projectName[/configName] | all}	Опция, аналогичная предыдущей, за исключением того, что перед сборкой будет произведена очистка проектов

Примечание: для запуска сборки из произвольной папки необходимо указать полный путь к **eclipsec.exe** (Windows) или **riscv-eclipse** (Linux).

Примечание: если пути, используемые в аргументах сборки, содержат пробелы, необходимо окружать путь кавычками (" ") (только для Windows).

Примеры использования (Windows):

- Вывод списка достпных опций:
  eclipsec.exe -nosplash -application org.eclipse.cdt.managedbuilder.core.headlessbuild
- Очистка и сборка всех проектов в workspace расположенном в *C:\workspace*:
  eclipsec.exe -nosplash -application org.eclipse.cdt.managedbuilder.core.headlessbuild -data "C:\workspace" -cleanBuild
- Сборка проекта *MyProject* в конфигурации *Release* в workspace, расположенном в *C:\workspace*:
  eclipsec.exe -nosplash -application org.eclipse.cdt.managedbuilder.core.headlessbuild -data C:\workspace-build -build MyProject/Release

Также, для простоты сборки одного проекта из консоли Windows имеется скрипт *eclipsec.bat*, который может быть вызван из любой папки. Для этого введите в консоли следующую команду:

eclipsec [-build|-cleanBuild] <workspace> <project> <rev>

Скрипт установит необходимые переменные окружения и вызовет сборку <*rev*> проекта <*project*>, расположенного по адресу <*workspace*>.

Для получения дополнительной информации введите команду **eclipsec [-h|-help]**.

5 СООБЩЕНИЯ

В процессе работы в ИСР Eclipse среда может выдавать сообщения несколькими способами:

5.1 Сообщения вкладки *Problems*

На вкладке *Problems* среда может выдавать сообщения об ошибках или предупреждения, возникающие во время написания исходного кода программ. Пример такого сообщения показан на рисунке 5.1;

5.2 Сообщения вкладки *Console*

На вкладке *Console* среда выводит информацию, относящуюся к сборке программ (компиляция, очистка, перестроение проектов и проч.). При возникновении ошибок сборки сообщения об ошибке выделяются красным цветом, предупреждения - желтым (рисунок 5.2). При отсутствии ошибок сообщение сборки имеет стандартный вид (рисунок 5.3).

5.3 Сообщения вкладки *Debugger Console*

На вкладке *Debugger Console* отображается информация, полученная от отладчика *GDB* в процессе отладки программы на симуляторе или на плате. Пример таких сообщений приведен на рисунке 5.4.

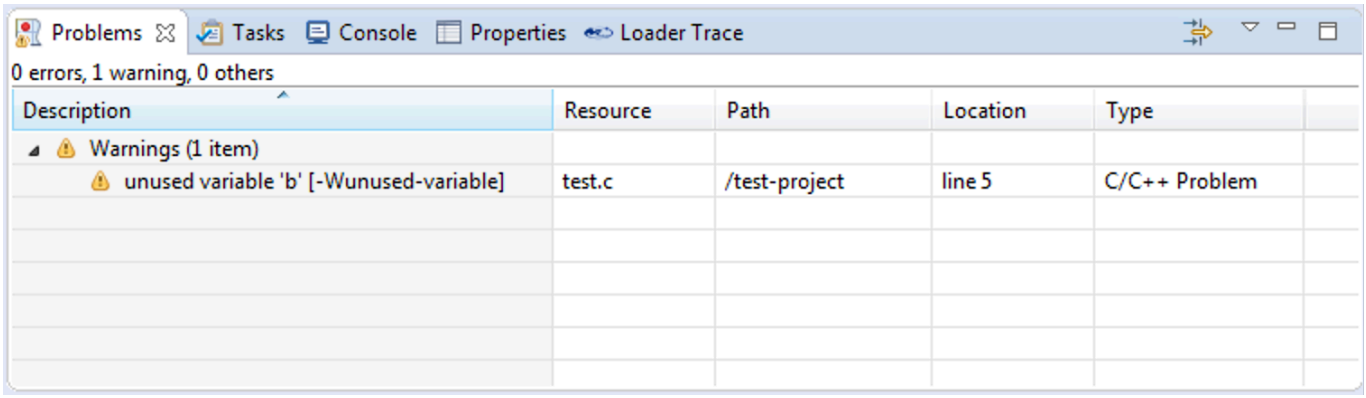

Рисунок 5.1. Предупреждения на вкладке Problems

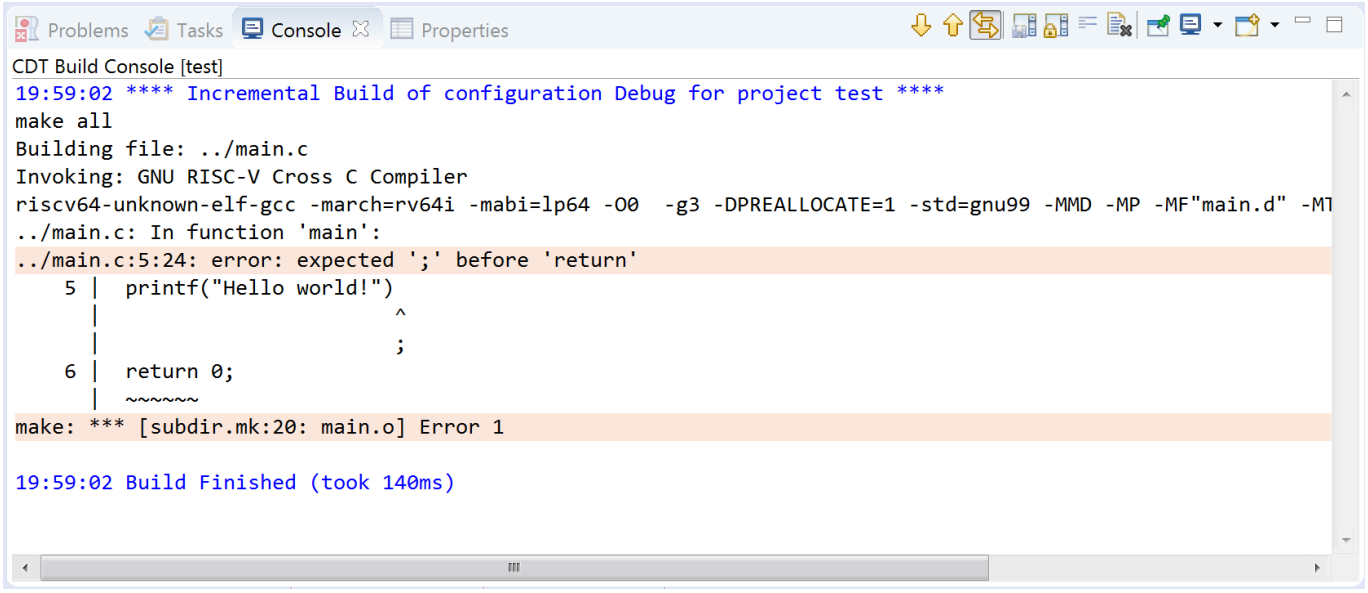

Рисунок 5.2. Сообщения об ошибках сборки на вкладке Console

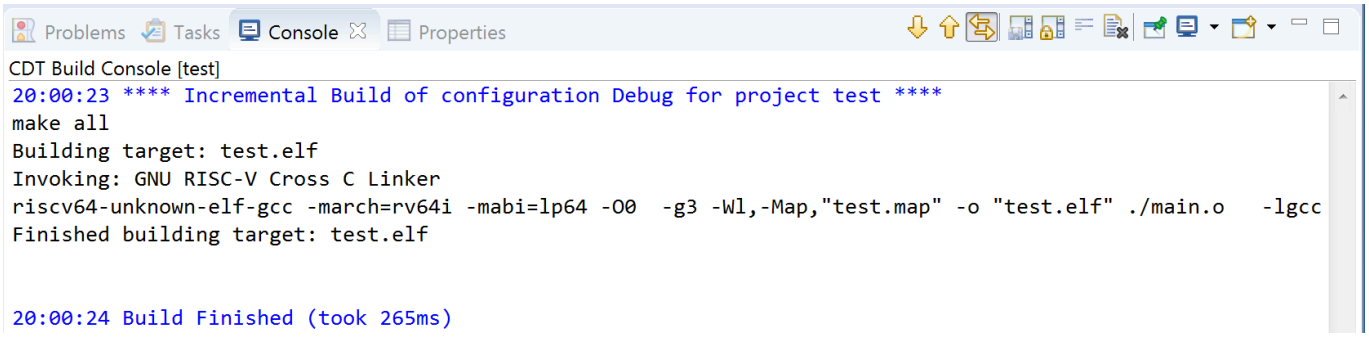

Рисунок 5.3. Сообщение об успешной сборке на вкладке Console

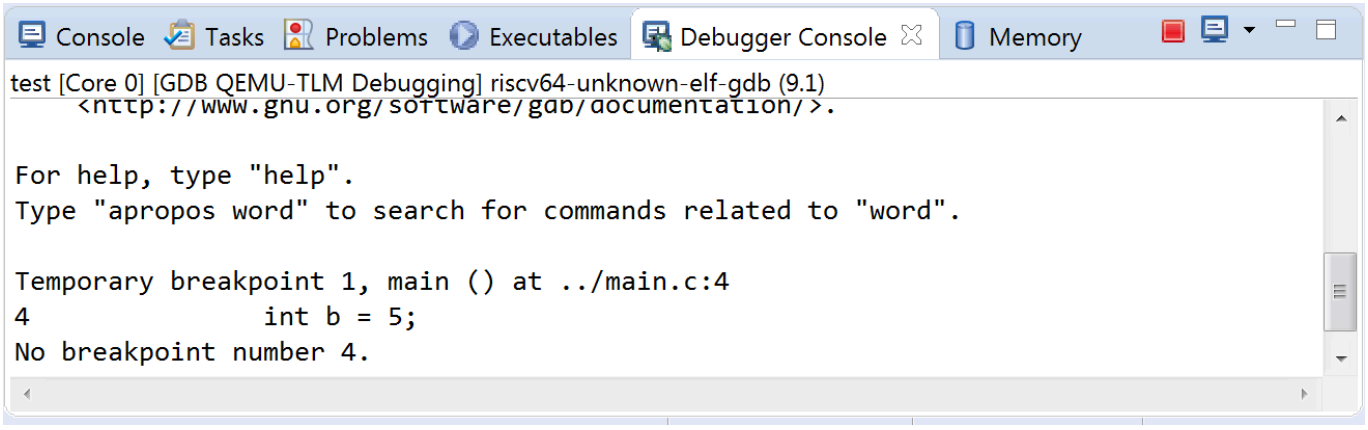

Рисунок 5.4. Сообщения отладчика на вкладке Debugger Console