Глава 1. Платформа

# Введ

[Введение 1](#_Toc150442188)

[ение 1](#_Toc150442189)

[Глава Обзор(Видео) 1](#_Toc150442190)

[Цели обучения 2](#_Toc150442191)

[Платформа разработки 2](#_Toc150442192)

[Приступая к работе 2](#_Toc150442193)

[Плата RISC-V 3](#_Toc150442194)

[Среда разработки 5](#_Toc150442195)

[Драйвер J-Link USB 7](#_Toc150442196)

[Начало работы с Freedom Studio 7](#_Toc150442197)

[Создание проекта 10](#_Toc150442198)

[Перспективы Eclipse 14](#_Toc150442199)

[Создание вашего приложения 16](#_Toc150442200)

[Загрузка вашего приложения на плату 19](#_Toc150442201)

[Hello world Демо (Видео) 22](#_Toc150442202)

[Отладка 23](#_Toc150442203)

[Выполнение вашего кода в отладчике 26](#_Toc150442204)

[Демонстрационная версия отладки Hello World (Видео) 31](#_Toc150442205)

[Подведение итогов 31](#_Toc150442206)

# ение

## Глава Обзор(Видео)

[Эдуардо Корпеньо] Давайте познакомимся с платформой разработки.

В этой главе мы поговорим об аппаратном и программном обеспечении.

Что касается аппаратного обеспечения, то мы познакомимся с Red-V Thing Plus.

И, что касается программного обеспечения, мы узнаем, как использовать Freedom Studio, которая является программной средой от SiFive.

Мы узнаем, как создать проект, как загрузить ваши приложения на плате и как отлаживать ваши приложения.

Мы будем делать это шаг за шагом, чтобы вы могли легко ознакомиться с этими инструментами.

Итак, приступим.

## Цели обучения

К концу этой главы вы должны:

Иметь представление об аппаратном и программном обеспечении, с которым мы будем работать на протяжении всего курса.

Уметь создавать проект для платы Red-V Thing Plus в Freedom Studio.

Уметь загружать приложения на свою плату Red-V Thing Plus.

Уметь отлаживать свой код с помощью Freedom Studio.

# Платформа разработки

## Приступая к работе

В этой последовательности вы познакомитесь с аппаратной и программной платформой, которую мы будем использовать на протяжении всего курса. SiFive Freedom Studio, отличная среда IDE, поддерживающая микроконтроллер Red-V Thing Plus.

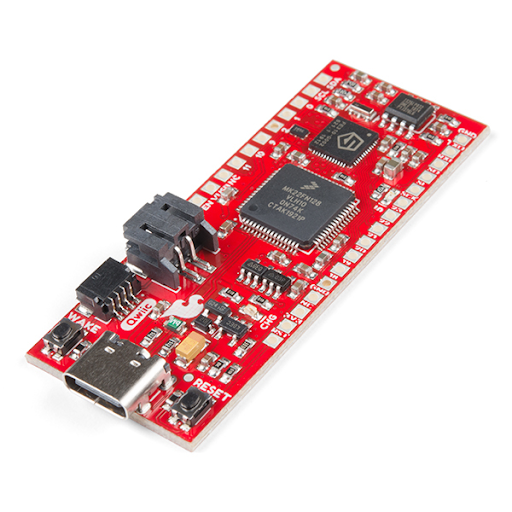
Если вы решили пройти этот курс, не изучая вместе с кодом на физической плате, вы можете расслабиться и плавно пройти материал курса. Пожалуйста, не стесняйтесь бегло просматривать все предстоящие практические разделы, всегда обращая особое внимание на то, как и почему мы предпринимаем каждый шаг.

Если вы следите за ходом работы, не стесняйтесь вносить изменения в код и экспериментировать со своими идеями. Если вы не уверены в какой-то идее или вам нужен совет, пожалуйста, свяжитесь с форумом курса.

Пожалуйста, просмотрите ссылки в каждом разделе и добавьте полезные из них в закладки для дальнейшего использования. Вы также можете найти ссылки и ресурсы в загружаемом PDF-документе [здесь](https://courses.edx.org/asset-v1:LinuxFoundationX+LFD115x+3T2022+type@asset+block@LFD115x_Links_and_Other_Resources.pdf), а также в разделе раздаточных материалов курса.

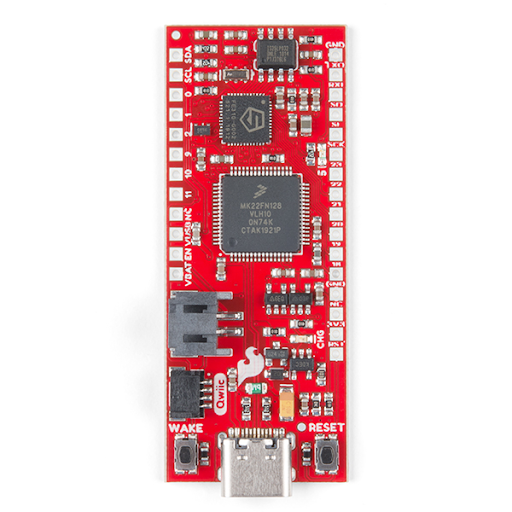
## Плата RISC-V

Первый элемент, который нам нужно обсудить, - это плата разработки Red-V Thing Plus, оснащенная микроконтроллером SiFive, или MCU. Вот подробное изображение Red-V Thing Plus.

  
  
**The Red-V Thing Plus**(Взято у SparkFun Electronics, предоставляемая в рамках CC BY 2.0)

Одной из главных особенностей этой платы является то, что она не содержит большого количества аппаратных средств вокруг микроконтроллера, поэтому вы можете сосредоточиться на создании собственных приложений, используя то оборудование, которое вы выберете для подключения к плате. Это идеальный тип микроконтроллерной платформы для обучения. Ознакомьтесь с принципиальной схемой чтобы увидеть, насколько проста эта плата.

Давайте посмотрим, что на ней находится:

  
**Red-V Thing Plus вид сверху.**(Взято с SparkFun Electronics, в рамках [CC BY 2.0](https://creativecommons.org/licenses/by/2.0/))

Самый большой чип на плате - это не микроконтроллер RISC-V, а микроконтроллер NXP Kinetis ARM Cortex-M4 MCU, используемый в качестве интерфейса программирования для обеспечения бесшовного интерфейса USB для передачи ваших приложений из IDE в микроконтроллер RISC-V.

Микроконтроллер RISC-V - второй по величине чип на плате, с буквой S внутри пятиугольника (это логотип SiFive). Это Freedom E310-G002, который основан на очень популярном процессорном ядре RV32IMAC.

Микросхема в самом верху рисунка - это флэш-память QSPI с 4 Мбайт для хранения ваших программ. Такого объема памяти достаточно даже для больших приложений, работающих на микроконтроллере.

На плате есть две кнопки специального назначения: одна для сброса настроек микроконтроллера и одна для вывода микроконтроллера из режима глубокого сна с низким энергопотреблением.

Также, прямо над кнопкой **Запуска**, на плате есть разъем qwiic, позволяющий легко подключать ваши qwiic-устройства. Qwiic - это соединительная система, разработанная SparkFun Electronics для создания широкого спектра устройств I2C, таких как датчики и исполнительные механизмы, что устраняет необходимость пайки.

Теперь давайте посмотрим на плату снизу:

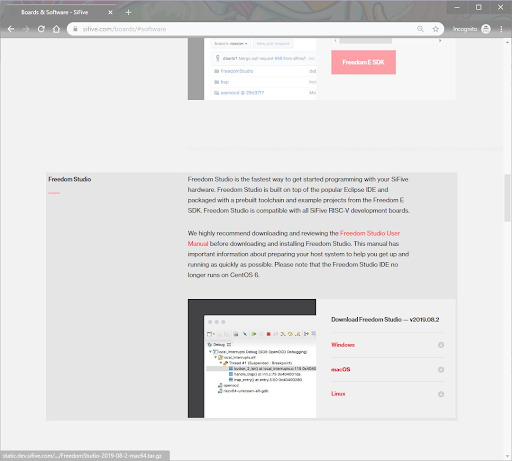
  
**Red-V Thing Plus вид снизу, показывающий распиновку панели**  
(Взято с SparkFun Electronics, в рамках [CC BY 2.0](https://creativecommons.org/licenses/by/2.0/))

На этой плате имеется 19 выводов ввода-вывода, расположенных в 2 рядах прокладок со сквозными отверстиями, к которым, при желании, можно припаять разъемы.

## Среда разработки

IDE, которую мы будем использовать в этом курсе, - это SiFive Freedom Studio. Вы можете загрузить ее на официальном сайте. Просто поищите Freedom Studio (не Freedom E SDK).

На момент создания этого курса последней версией была 2021.04.1. Это программное обеспечение доступно для Windows, macOS и Linux.

  
**Страница загрузки Freedom Studio**(Взято с SparkFun Electronics в рамках [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

В остальной части курса будут показаны скриншоты и процедуры для настройки Windows 10. Пожалуйста, имейте в виду, что Freedom Studio может иметь незначительные изменения в macOS и Linux, поэтому, пожалуйста, свяжитесь с нами на форуме курса, если у вас возникнут какие-либо трудности.

Установка Freedom Studio проста, но SiFive рекомендует вам ознакомиться с руководством по установке в [руководстве Freedom Studio](https://static.dev.sifive.com/dev-tools/FreedomStudio/2021.04/freedom-studio-manual-2021-04.pdf?_ga=2.255799240.1588695809.1653950259-754836148.1653950259). Там вы найдете важные шаги по правильной установке IDE.

Например, если вы используете Windows, загружаемый файл представляет собой большой zip-архив со всеми файлами, необходимыми для запуска Freedom Studio, поэтому классического установщика для Windows не существует. Однако, программное обеспечение содержит файлы с длинными глубокими путями, которые могут превышать максимальное количество символов, поддерживаемых Windows. Рекомендуемый обходной путь - извлечь zip-архив не в папку Program Files, а в короткий путь без пробелов.

Мы использовали рекомендуемый путь для Windows:  
**C:/FreedomStudio/FreedomStudio-2021.04/**  
Вам следует следовать процедуре, рекомендованной для вашей операционной системы.

## Драйвер J-Link USB

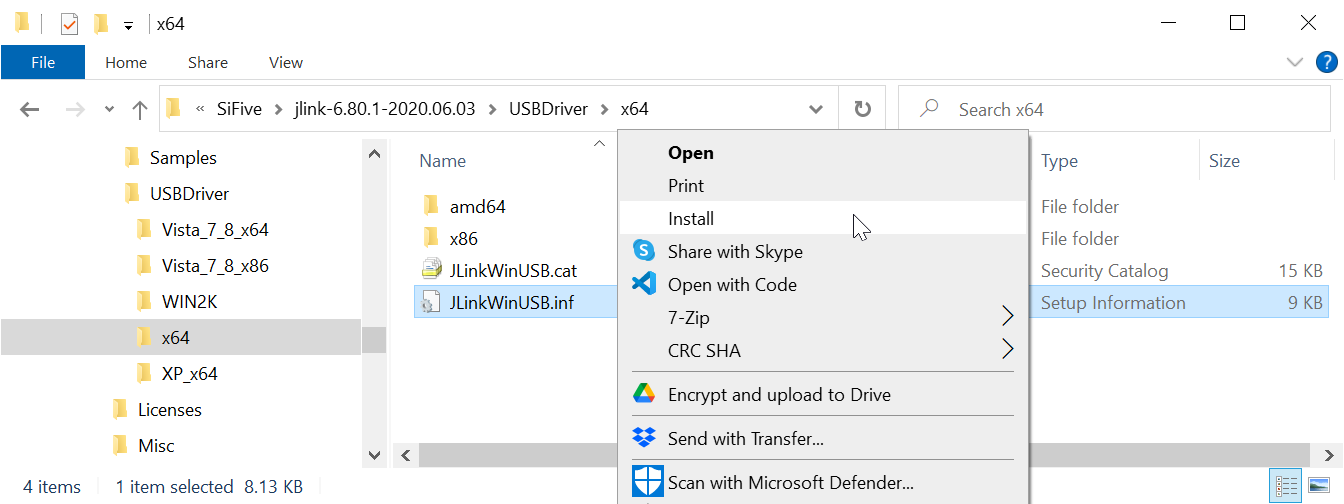
Чтобы запрограммировать Red-V Thing Plus через интерфейс USB J-Link, вам понадобится драйвер USB-устройства J-Link.

Для Linux драйверы включены в дистрибутив Freedom Studio. То же относится и к ОС macOS Catalina и позже.

Для macOS Mojave и более ранних версий, инструкции см. в руководстве Freedom Studio.

Для Windows вы можете найти несколько установщиков в следующей папке:  
**<папка установки>/SiFive/jlink/jlink<versioninfo>/USBDriver/**

Существует несколько вложенных папок для разных версий Windows. Убедитесь, что установлен правильный драйвер для ваших настроек Windows. Это установщик драйверов для 64-разрядной версии Windows 10:

  
**Установщик драйверов JLink для Windows 10, 64-разрядная**

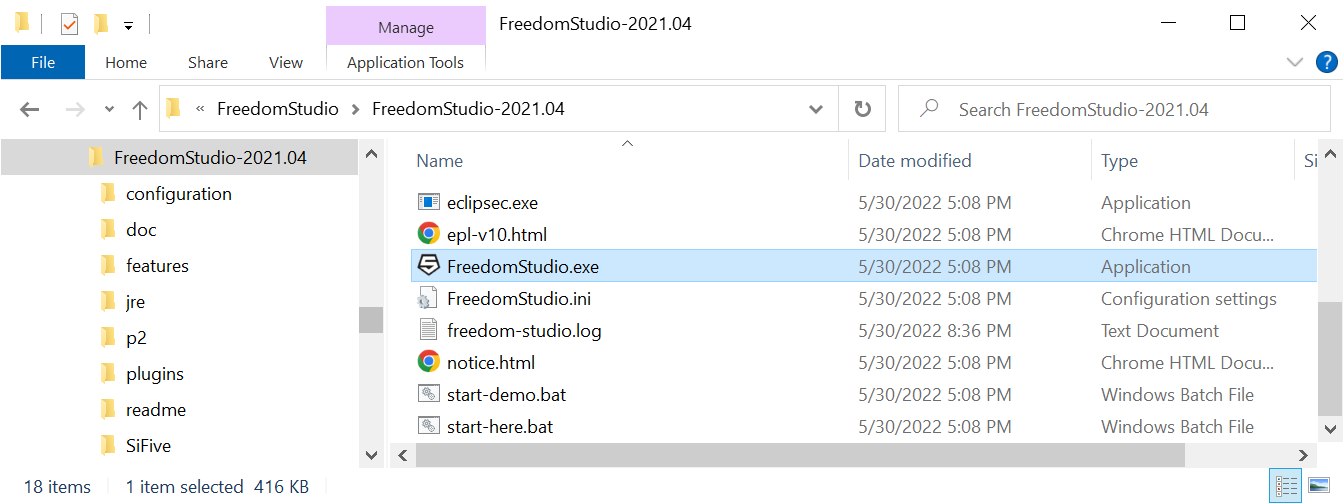
Если найти свою версию Windows среди этих папок слишком сложно, вы всегда можете загрузить пакет программного обеспечения и документации J-Link, который содержит последние версии драйверов. Вы можете скачать его с [веб-сайта SEGGER](https://www.segger.com/downloads/jlink/).

## Начало работы с Freedom Studio

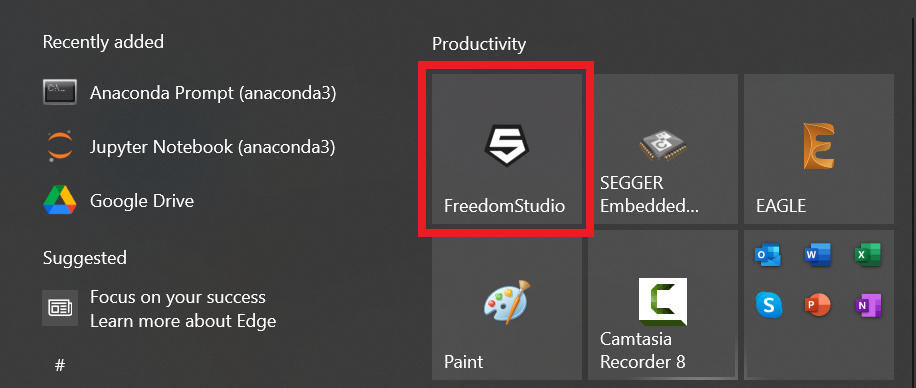
Теперь, когда у вас есть плата микроконтроллера, Freedom Studio и USB-драйверы, вы готовы попробовать hello world!

Оказывается, Freedom Studio поставляется с шаблонами проектов для нескольких поддерживаемых платформ, таких как [HiFive1 Rev B](https://www.sifive.com/boards/hifive1-rev-b), которая использует тот же целевой микроконтроллер, что и Red-V Thing Plus. Фактически, мы будем создавать наши проекты для Red-V Thing Plus, используя шаблон проекта HiFive1 Rev B.

Первым шагом является запуск Freedom Studio. Вы можете найти исполняемый файл в только что созданной папке установки. Вот исполняемый файл Windows:

  
**Исполняемый файл Freedom Studio в Windows 10**

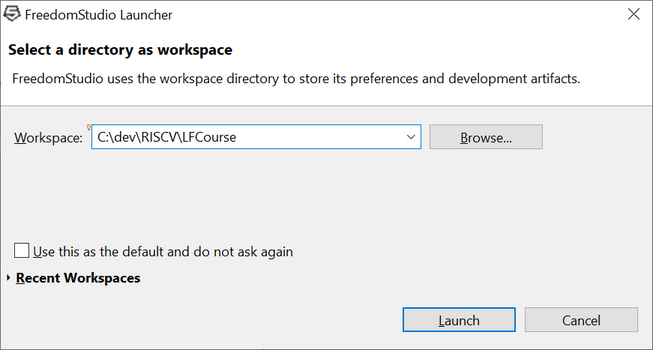
Рекомендуется создать ярлык для легкого доступа на вашем рабочем столе. Ниже вы найдете пример этого ярлыка:

  
**Ярлык Freedom Studio в Windows 10**

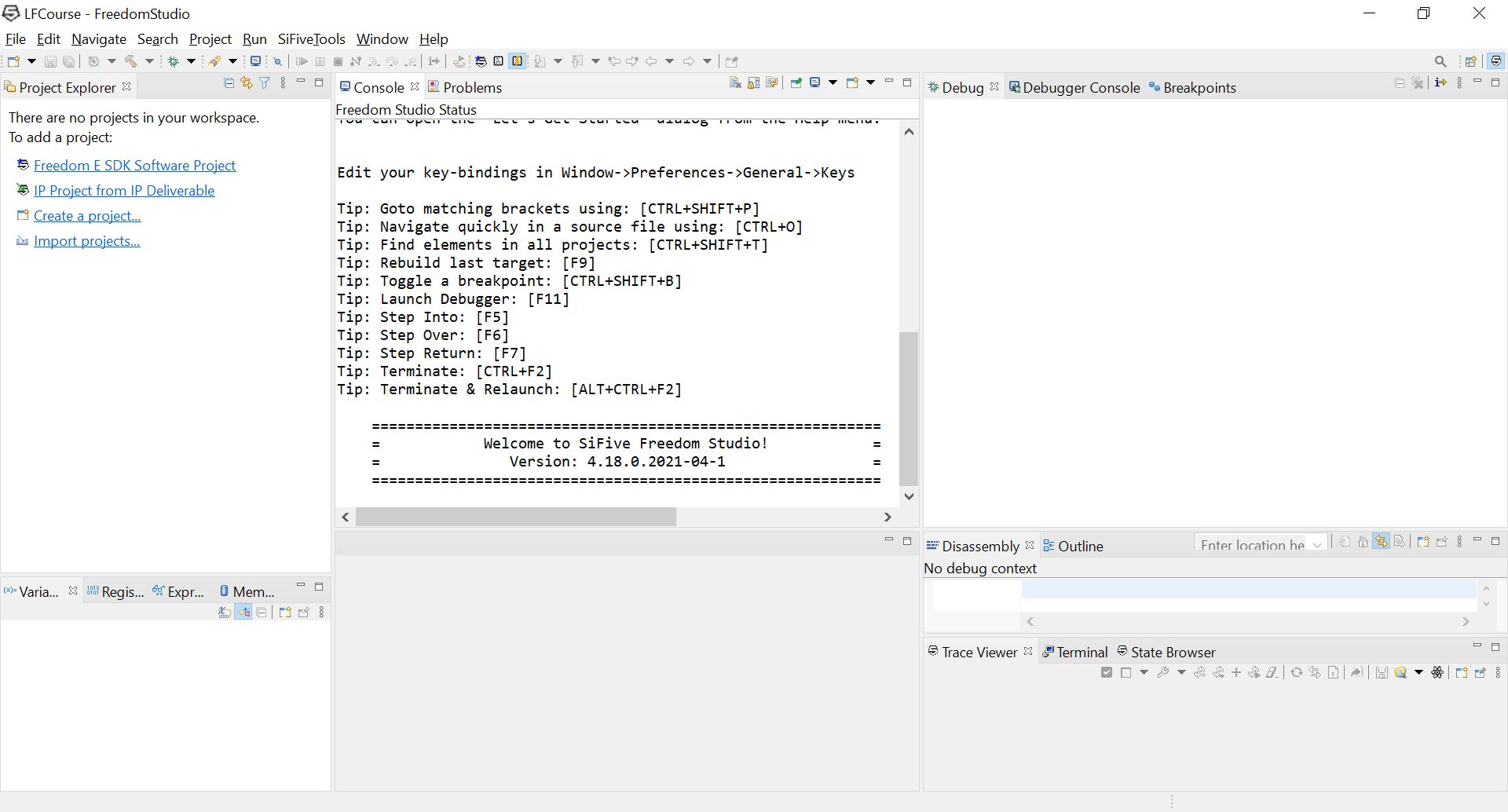
Freedom Studio - это IDE на базе Eclipse. Если вы не знакомы с Eclipse, пусть это вас не останавливает. Вы узнаете кое-что из этого курса, поскольку вас проведут по его основным элементам. Однако, если вы столкнетесь с трудностями при работе с некоторыми терминами, такими как рабочие области, перспективы, проекты и конфигурации отладки, вы можете поискать в Интернете учебное пособие, соответствующее вашим потребностям. Чтобы официально ознакомиться с eclipse и стать опытным пользователем, вы всегда можете обратиться к [странице "Начало работы](https://www.eclipse.org/getting_started/)" на веб-сайте Eclipse Foundation.

Как и в любой среде IDE на базе Eclipse, при запуске Freedom Studio вы увидите диалоговое окно с запросом рабочей области для работы. Рабочие области - это папки, в которых будут храниться ваши проекты. Eclipse будет показывать вам проекты только внутри одной рабочей области одновременно. Идея состоит в том, чтобы сосредоточиться на связанных проектах. Рабочее пространство также сохраняет свои собственные настройки и конфигурации, которые должны быть общими для всех проектов в рабочем пространстве.

Программа запуска Freedom Studio, вероятно, покажет вам вложенную папку в качестве рабочей области по умолчанию в папке установки Freedom Studio. Однако при работе с наборами инструментов компилятора C и IDE часто рекомендуется использовать пути с короткими именами папок без пробелов или специальных символов, включая тире. Вот почему мы решили создать другое рабочее пространство для этого курса и назвали его **C:/dev/RISCV/LFCourse**

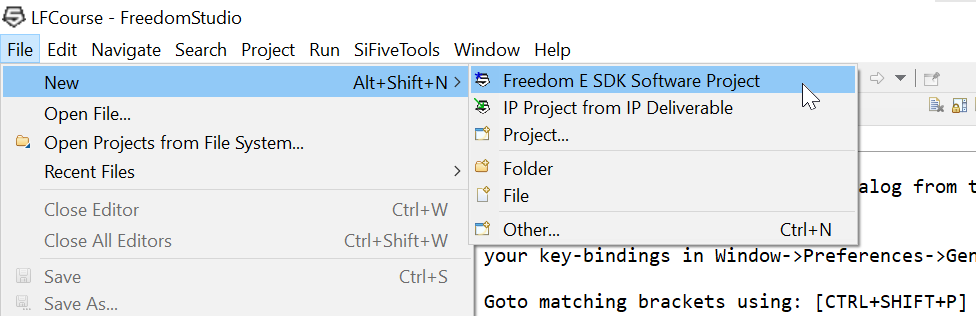
**  
Диалоговое окно рабочей области Freedom Studio**

После нажатия на кнопку запуска вы можете получить экран приветствия или какое-либо другое диалоговое окно. Пожалуйста, отклоняйте их до тех пор, пока не увидите IDE, показывающую перспективу SiFive. Что-то вроде этого:

  
**Freedom Studio в Windows 10**

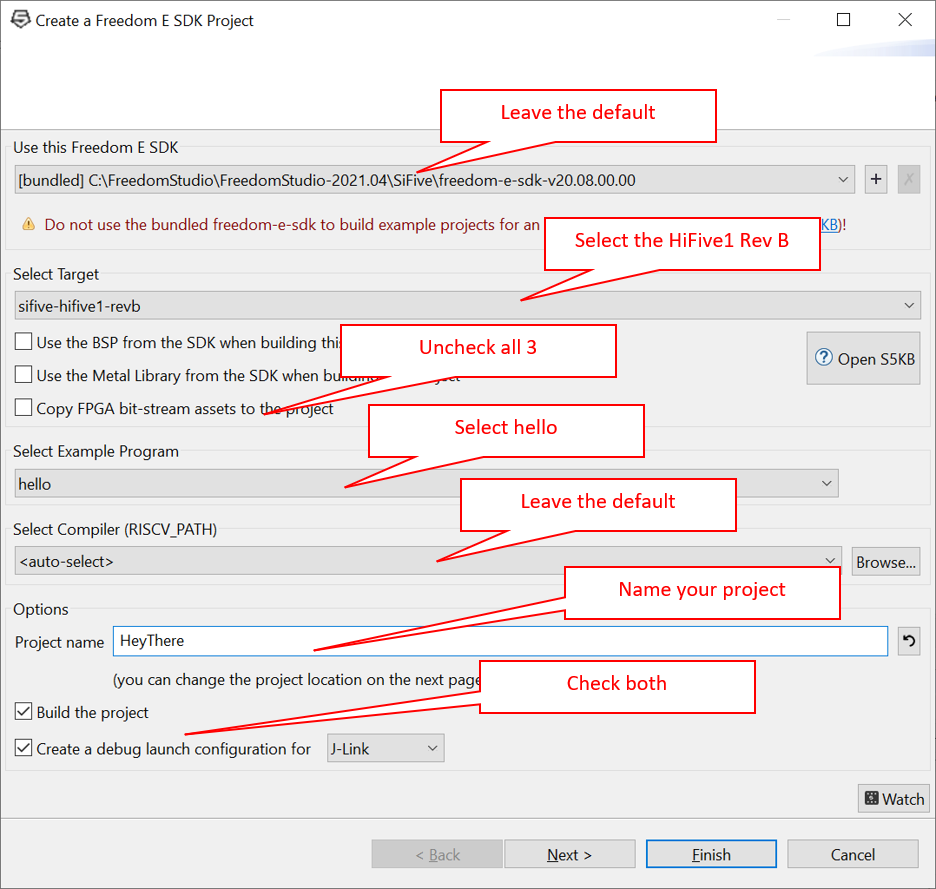
## Создание проекта

Теперь мы создадим проект. Нажмите на **Файл → Создать → Проект программного обеспечения Freedom E SDK**. Это позволит создать проект из Freedom E SDK, хранилища демонстрационных программ и пакетов поддержки платы. Это то, откуда мы будем брать образцы проектов HiFive1 REV.

  
**Создание нового проекта в Freedom Studio**

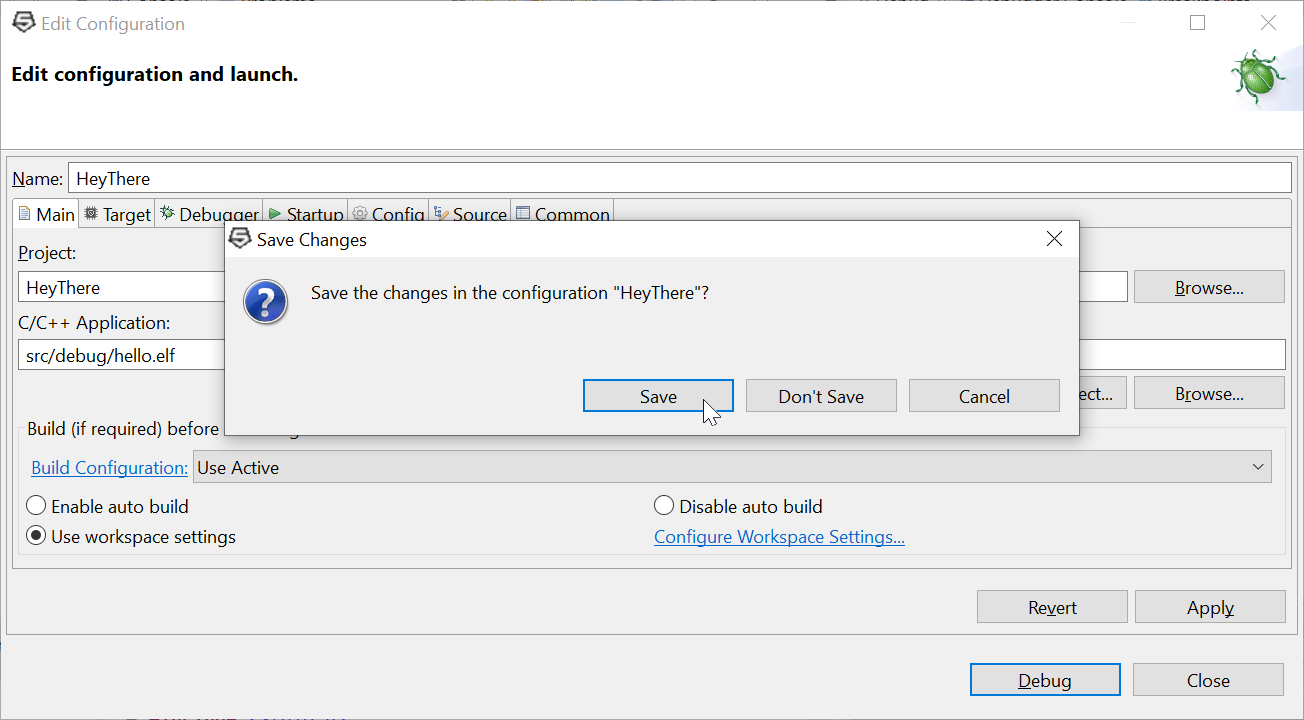
В следующем диалоговом окне выберите HiFive1 Rev B в качестве целевого (это должна быть версия B, поэтому следите за другими доступными HiFive 1).

Также обратите внимание, что вы можете выбрать пример проекта для начала работы. Не стесняйтесь начинать свои проекты с проекта Hello World example. Попробуйте и выполните следующие настройки:

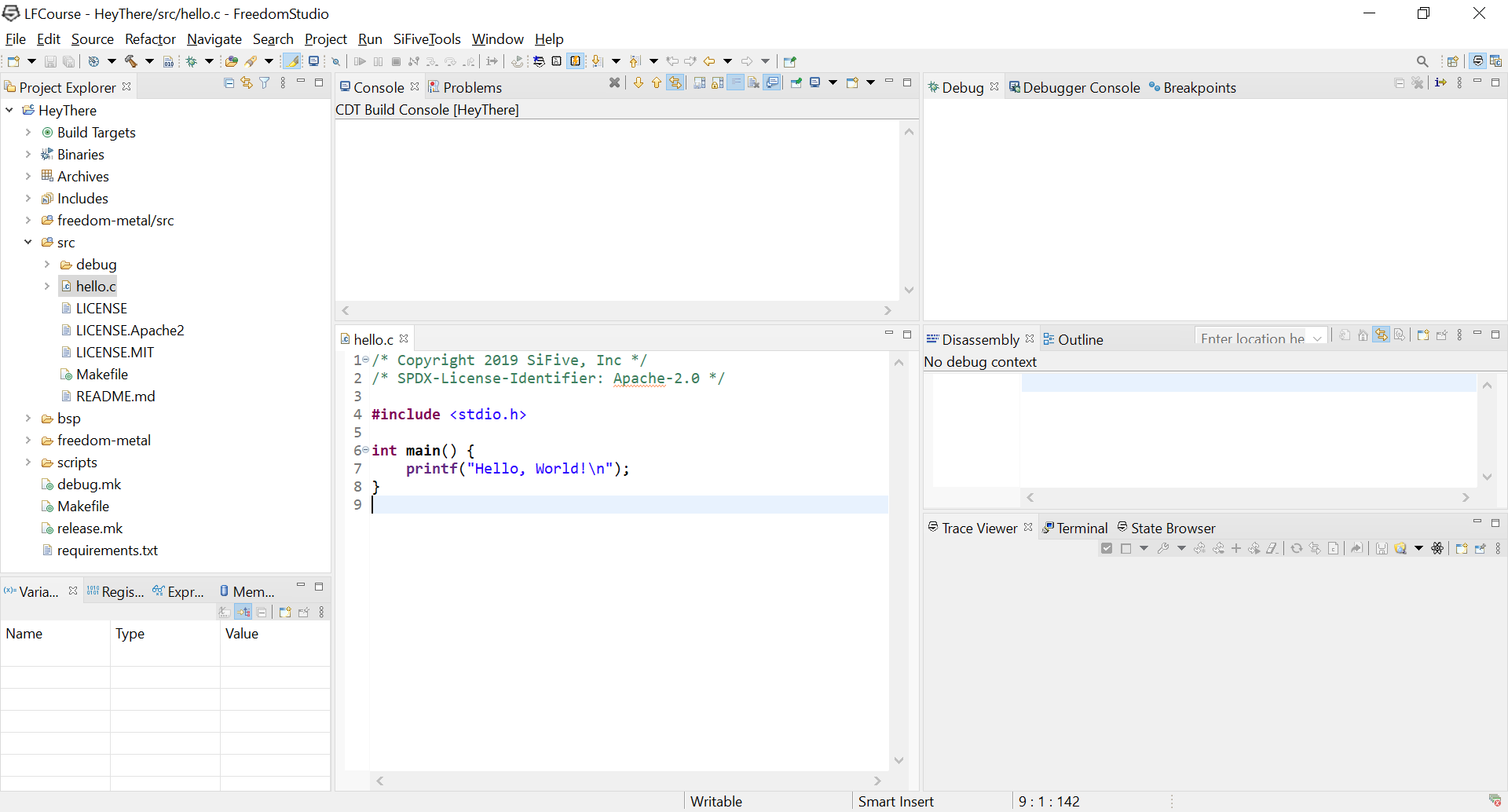
  
**Мастер нового проекта Freedom Studio**

Вы можете нажать на **Finish**. В качестве альтернативы вы можете нажать на **Next**, и вас снова попросят ввести название проекта и другой путь.

Через некоторое время ваш проект будет создан. На компьютере с процессором Intel i7 это занимает около 2 минут, поэтому, пожалуйста, наберитесь терпения. Поскольку мы решили создать проект, вы увидите диалоговое окно для редактирования конфигурации отладки USB J-Link и запуска ее. Мы пока не хотим запускать его, но мы хотим его настроить, поэтому пока просто нажмите "Cancel", но обязательно сохраните конфигурацию. Позже мы увидим, как запустить эту конфигурацию отладки.

  
**Диалоговое окно настройки отладки Freedom Studio**

Теперь вы увидите проект, который вы только что создали. Как вы можете видеть, он показывает перспективу SiFive с открытым файлом **hello.c**. Уведомление (в файле структуры в левой), что этот файл находится в папке **HeyThere/src/**.

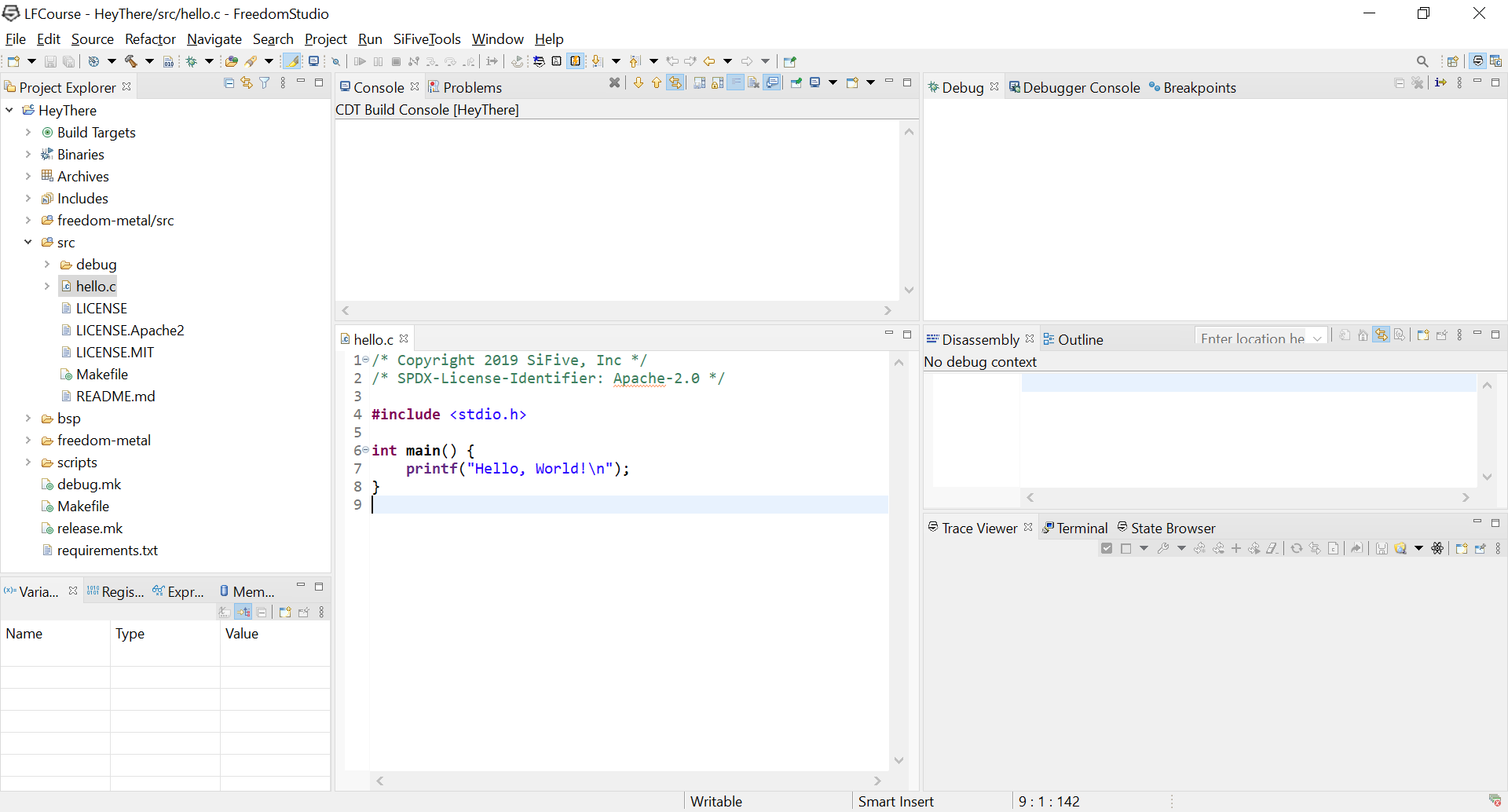
  
 **Freedom Studio отображает новый проект HeyThere**

Обратите внимание, что программа просто печатает “Hello, world!” и завершает работу. Она использует интерфейс последовательного порта Red-V Thing Plus для последовательного вывода данных с устройства UART со скоростью 115 200 бод.

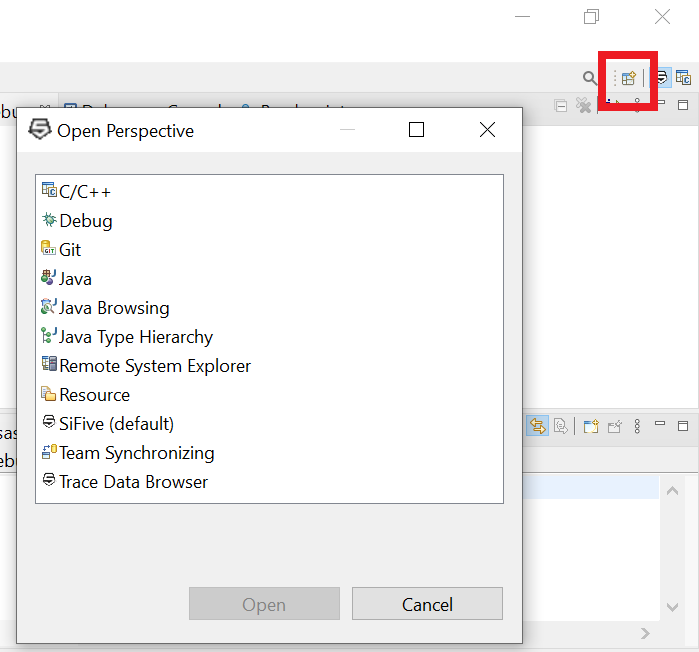
## Перспективы Eclipse

В Eclipse перспективы - это различные коллекции и макеты окон в IDE. Некоторые перспективы подходят для написания кода, другие - для отладки и т.д.

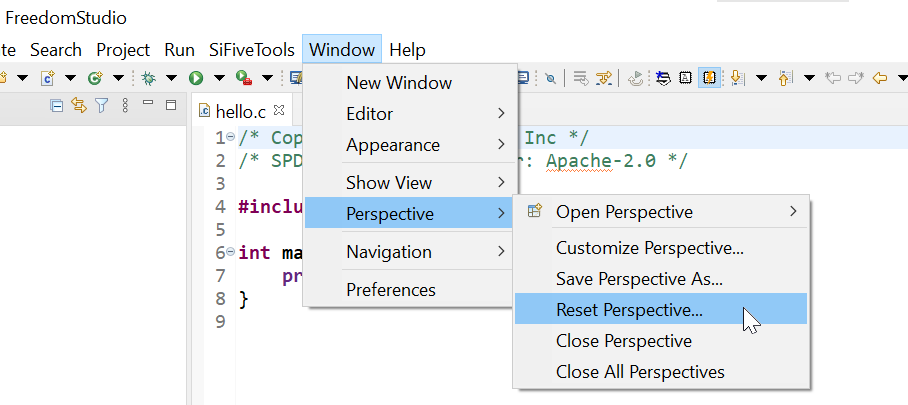
Проект, который мы только что создали, показан в SiFive перспективе. Перспектива SiFive показывает краткое описание проекта с окнами, которые вам, вероятно, понадобятся для базового написания кода и отладки. На лексиконе Eclipse эти окна (или вкладки) известны как ***виды***.

  
**Freedom Studio отображает новый проект HeyThere в SiFive**

Однако Eclipse предлагает множество перспектив, которые вы можете использовать в зависимости от того, над чем вы работаете. Например, вы можете использовать перспективу C / C ++ для написания кода, тогда как для отладки существует перспектива Debug. Продолжайте изучать различные перспективы, нажав на кнопку **New Perspective** в правом верхнем углу.

  
**Диалоговое окно "Open perspective"**

При работе с разными окнами в любой перспективе не беспокойтесь, если вы случайно переместите окно в неудобное место или, возможно, закроете его и не сможете найти. Не паникуйте. Вы всегда можете перейти к **Window → Perspective → Reset perspective**, и макет окон для этой перспективы вернется в исходное состояние.

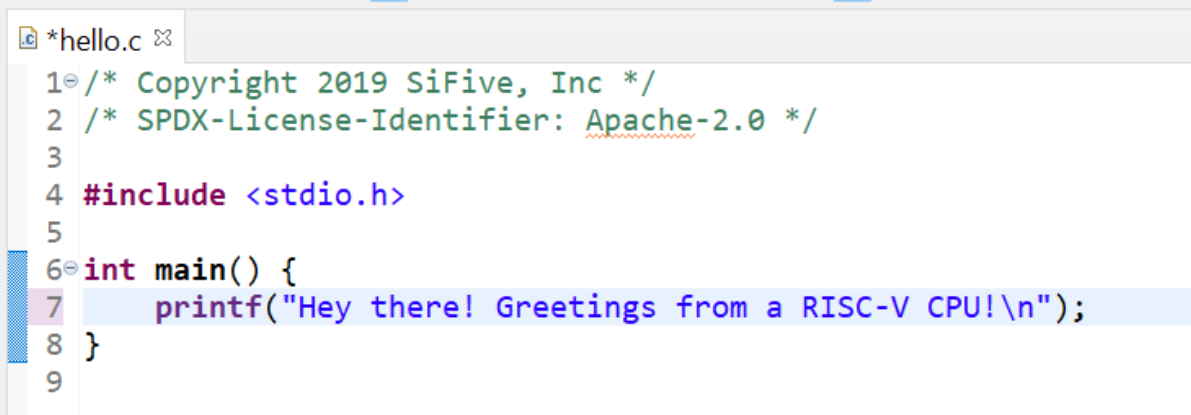
  
**Сбросить настройки перспективы**

Перспектива SiFive содержит все, что вам нужно для написания и отладки вашего кода. SiFive проделала отличную работу по созданию этой перспективы, и рекомендуется использовать ее для всего.

## Создание вашего приложения

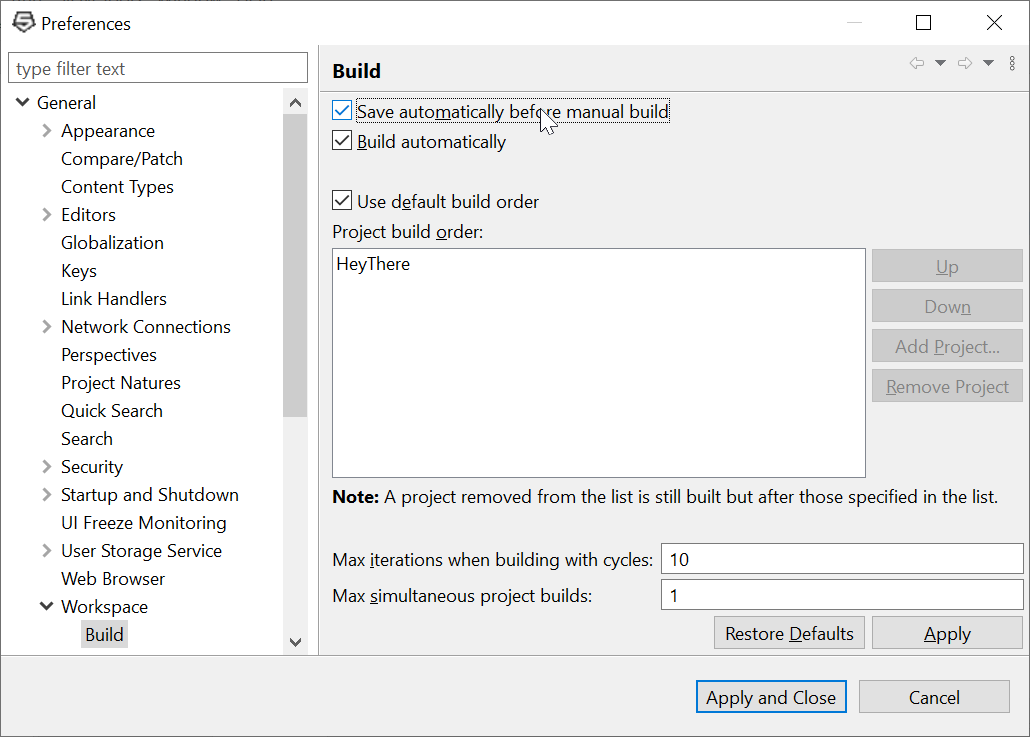
Перед загрузкой вашего приложения на плату, вам нужно будет его создать. Существует несколько типов выходных файлов, которые может создавать компилятор. Ваш Red-V Thing Plus определяется как внешний накопитель, когда вы подключаете его к USB-порту вашего компьютера. Это не настоящий диск, а популярный механизм для загрузки **hex.** файла в память QSPI в RED-V Thing Plus.

Перед созданием приложения давайте убедимся, что наш собственный код попадет в конечное приложение, поэтому давайте изменим строку hello world на что-нибудь другое:

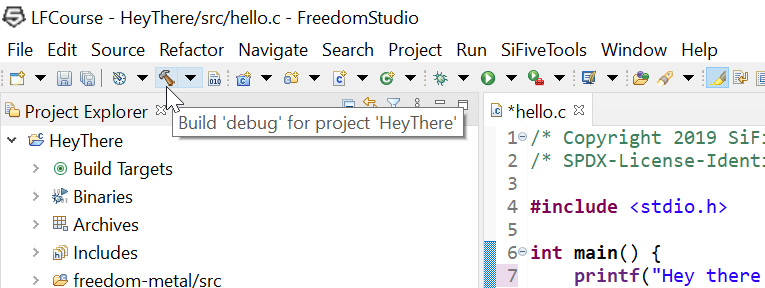
  
**Исходный код Hello world с нашим пользовательским сообщением**

К сожалению, если вы нажмете на кнопку **Запуск** перед сохранением файлаhello.c, по умолчанию будет использоваться несохраненная версия этого файла. Итак, у вас есть два варианта: Вариант A - всегда сохранять исходные файлы перед сборкой, а вариант B - указать Freedom Studio автоматически сохранять исходные файлы при нажатии кнопки запуска.

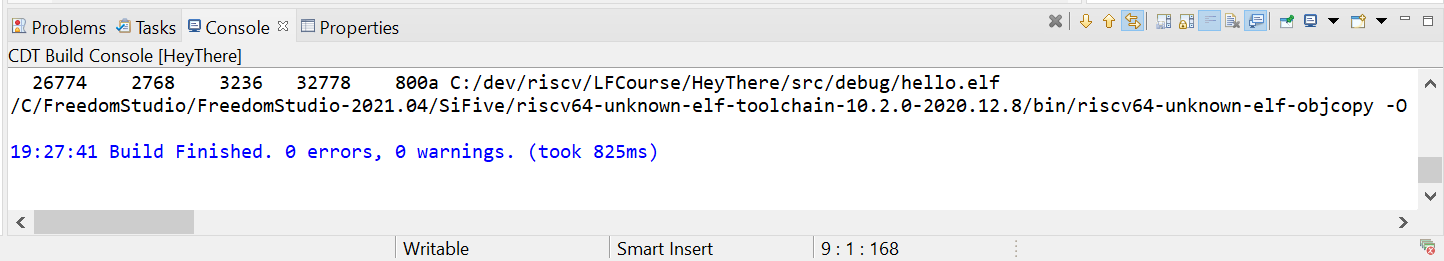
Если вы выберете вариант B, перейдите в **Window→ Preferences**, перейдите в **General → Workspace → Build**, и установите флажок “Save automatically before manual build”, наконец, нажмите “Apply and close”.

  
**Диалоговое окно настроек для автоматического сохранения перед ручной сборкой**

Теперь мы готовы к созданию нашего приложения. Просто нажмите на кнопку **Запуск** на панели инструментов (обратите внимание, что на следующем рисунке показана перспектива C / C ++).

  
**Кнопка сборки Freedom Studio**

Надеюсь, консоль сообщит, что сборка прошла успешно. Если есть ошибки, вы можете найти их на вкладке **Проблемы**.

  
**Окно консоли Freedom Studio, сообщающее об ошибках и предупреждениях**

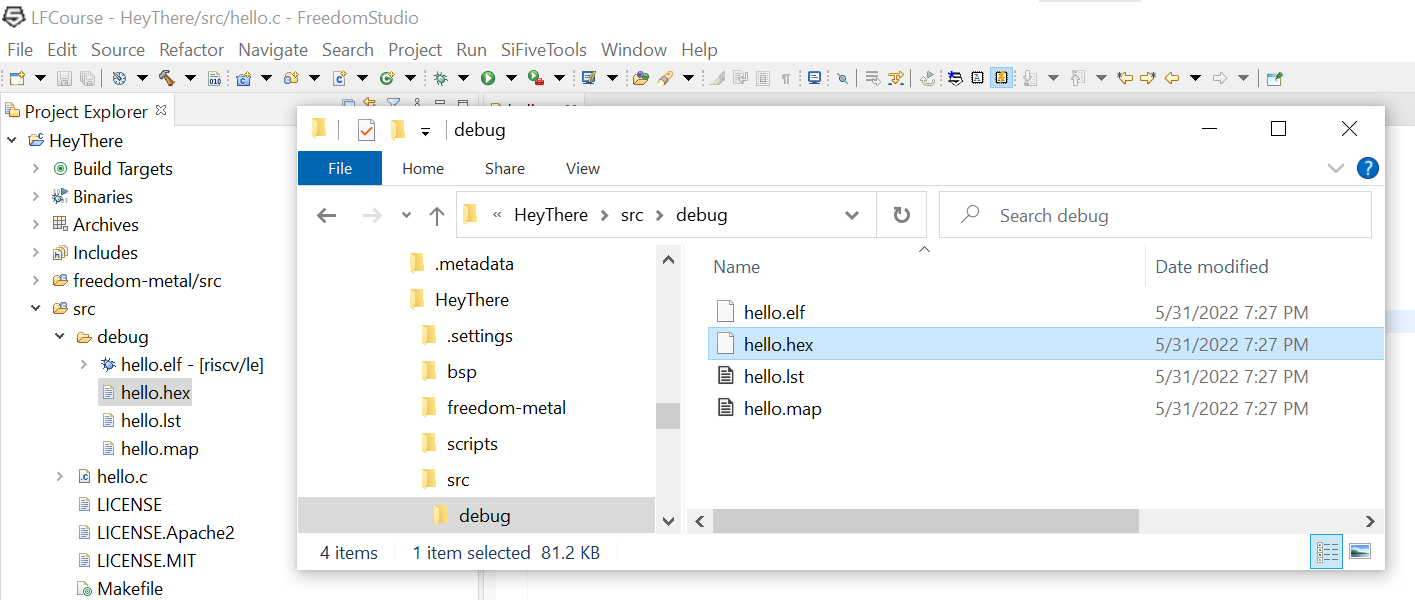
## Загрузка вашего приложения на плату

Теперь, когда у нас есть готовый проект, мы можем загрузить приложение на плату. Для этого у вас есть два варианта:

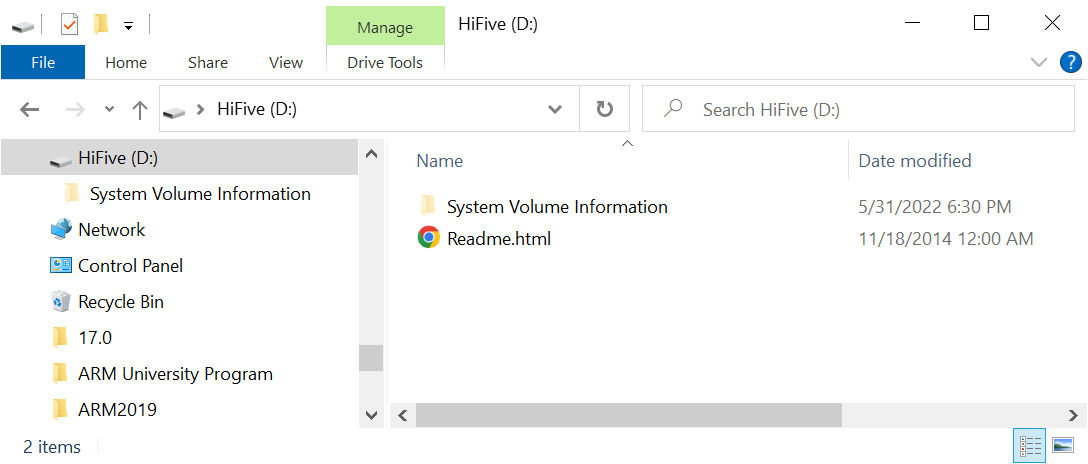
**Вариант A: *Загрузите созданный hex.файл на плату***  
Этот файл выдержит циклы отключения и будет запускаться каждый раз, когда вы нажимаете кнопку **Сброса** на плате. Это автономная процедура, означающая, что плата будет запускать вашу программу, не передавая отладочные данные на компьютер.

**Вариант B: *Загрузите приложение с помощью отладчика***  
Это более практично, потому что вы можете сделать это, не выходя из IDE. Этот параметр подчиняется отладчику Freedom Studio, но он также выдерживает циклы включения питания и выполняется самостоятельно при нажатии кнопки.

Давайте узнаем, как загрузить созданный **hex.**файл на доску. Файл находится в папке **./src/debug**  Вы можете увидеть это в Freedom Studio и в проводнике Windows:

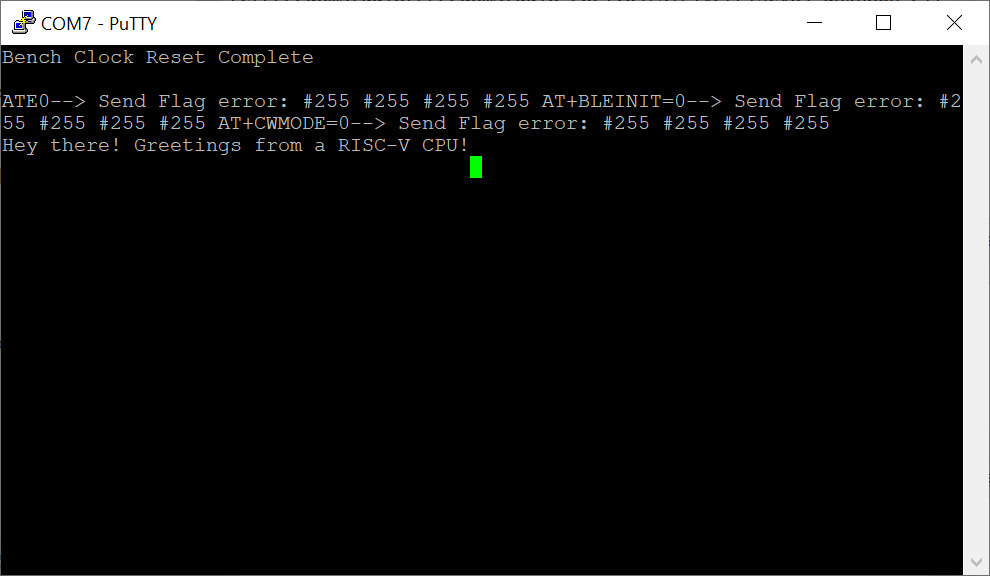
**  
Вывести hex. файл как в Freedom Studio, так и в проводнике Windows**

Вы можете перетащить hex-файл с любого из них на USB-накопитель, созданный Red-V Thing Plus, когда вы подключили его к USB-порту.

  
**Фиктивный USB-накопитель, созданный с помощью платы Red-V Thing Plus**

После того, как копирование файла выполнено, вы можете заметить, что USB-устройство перезагружается. Это связано с тем, что Red-V Thing Plus запустил недавно сохраненную программу во флэш-памяти QSPI.

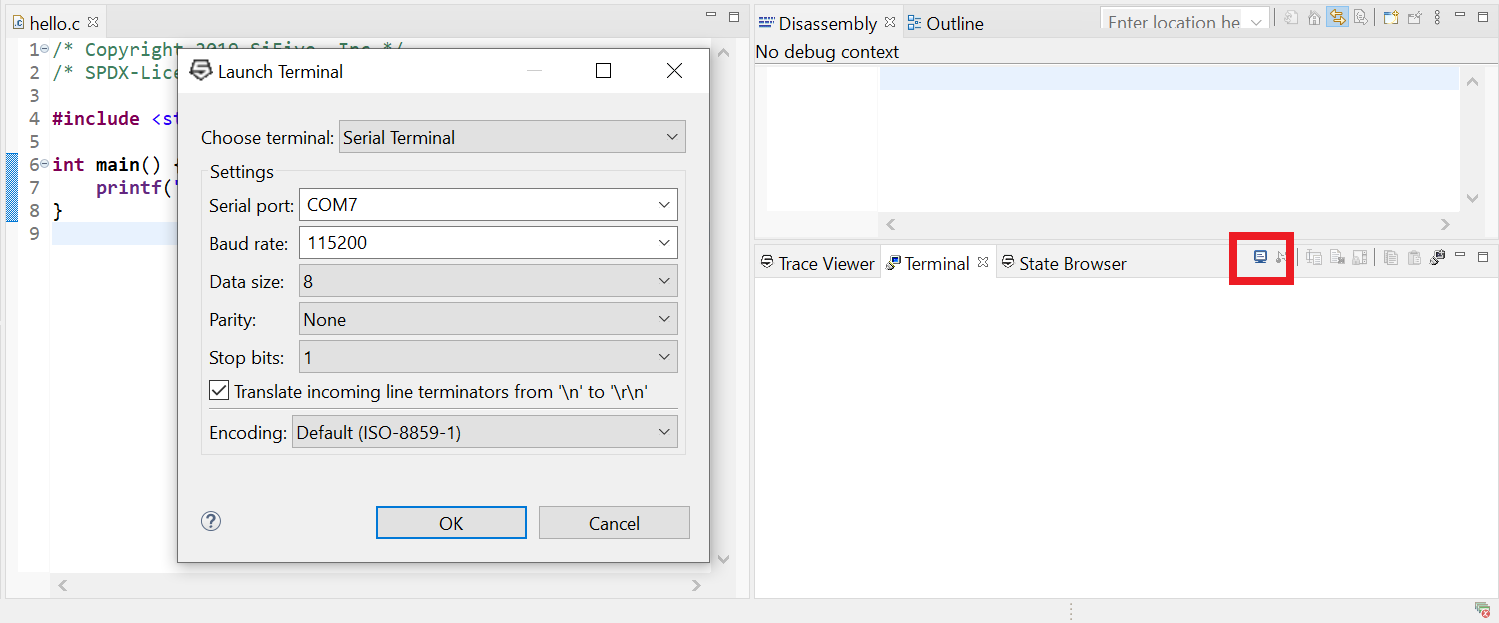
Чтобы просмотреть сообщение Hello World, вы можете использовать свой любимый эмулятор терминала, такой как PuTTY или Tera Term. Конфигурация последовательного порта для этого приложения составляет 115 200 бод, 8 битов данных, без четности и 1 стоповый бит. После нажатия кнопки **Reset** на плате, вот что отображается в PuTTY:

  
**Окно терминала PuTTY, показывающее некоторые диагностические сообщения, а затем выводимое сообщение в нашем приложении hello world**

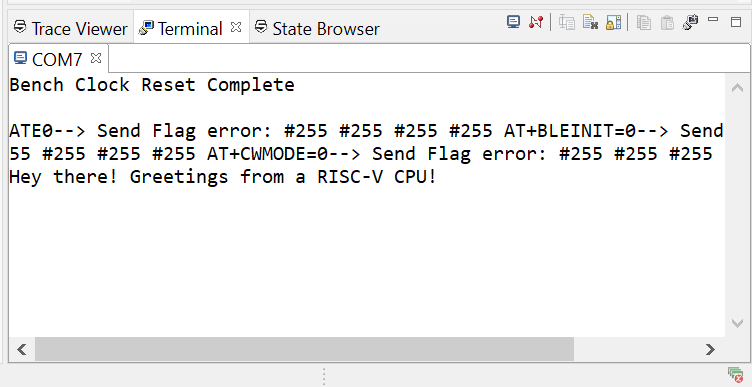
Как вы можете видеть, плата отправляет некоторые сообщения при загрузке и печатает наше сообщение hello world.

Вы также можете использовать эмулятор терминала, входящий в состав Freedom Studio. Он расположен в правом нижнем углу в просеивающей перспективе. Это вкладка под названием ***Терминал***.

Прежде чем попробовать это, убедитесь, что вы не используете последовательный порт в другом месте (закройте PuTTY, если вы им пользовались). Нажмите на кнопку со значком монитора, чтобы настроить его.

  
**Настройка эмулятора терминала в Freedom Studio**

Вот выходные данные платы в терминале Freedom Studio после нажатия **кнопки**:

  
**Эмулятор терминала Freedom Studio, показывающий выходные сообщения платы**

## Hello world Демо (Видео)

[Эдуардо Корпеньо] Итак, давайте посмотрим, как мы можем загружать наши приложения на плату, используя фиктивный USB-накопитель. Итак, здесь у нас есть hello.hex. Обратите внимание, что плата Red-V сообщила Windows о себе как о USB-накопителе под названием HiFive. Теперь позвольте мне перетащить hello.hex файл на диск. И пока Windows думает, что копирует этот файл на USB-накопитель, Red-V Thing Plus копирует только что созданную нами прошивку во флэш-память QSPI. Теперь позвольте мне настроить терминал в Freedom Studio и теперь я нажму кнопку сброса на плате. И вот оно. Это сообщение о загрузке. И внизу у нас есть наша строка.

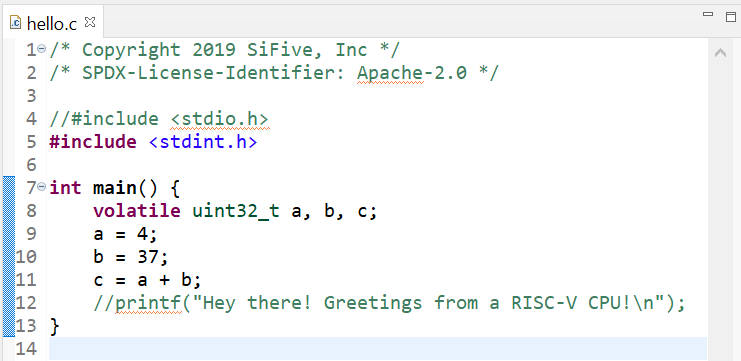
Привет всем! Привет от процессора RISC-V! Позвольте мне снова нажать кнопку сброса. И вот еще одно послание. Теперь обратите внимание, что это необычное приложение для встроенной системы, потому что оно завершает основную функцию. Это не обычный случай для встроенного приложения, потому что предполагается, что все эти приложения имеют бесконечный цикл внутри функции main. Итак, если вам это интересно, вы можете запустить отладчик и выполнять пошагово, включая момент, когда выполнение завершает основную функцию. Вы увидите, что в конечном итоге это приводит к бесконечному циклу на языке ассемблера.

## Отладка

Теперь перейдем к варианту B: вы также можете отлаживать свой код, используя интерфейс J-Link и хорошо известный отладчик GDB в качестве серверной части. Внешний интерфейс - это отладчик Eclipse.

Это самый простой вариант быстрого размещения вашего приложения на плате, которым вы, вероятно, будете пользоваться чаще всего.

Поскольку наша программа имеет только вызов функции printf, отладка была бы не такой заметной. По этой причине давайте изменим код:

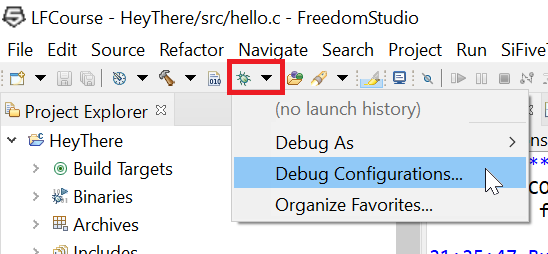
  
**Модифицированный код для отладки**

Теперь мы ожидаем увидеть 3 целых числа без знака с двумя непосредственными присвоениями и дополнением. Обратите внимание, что строки include и printf в исходном коде теперь закомментированы.

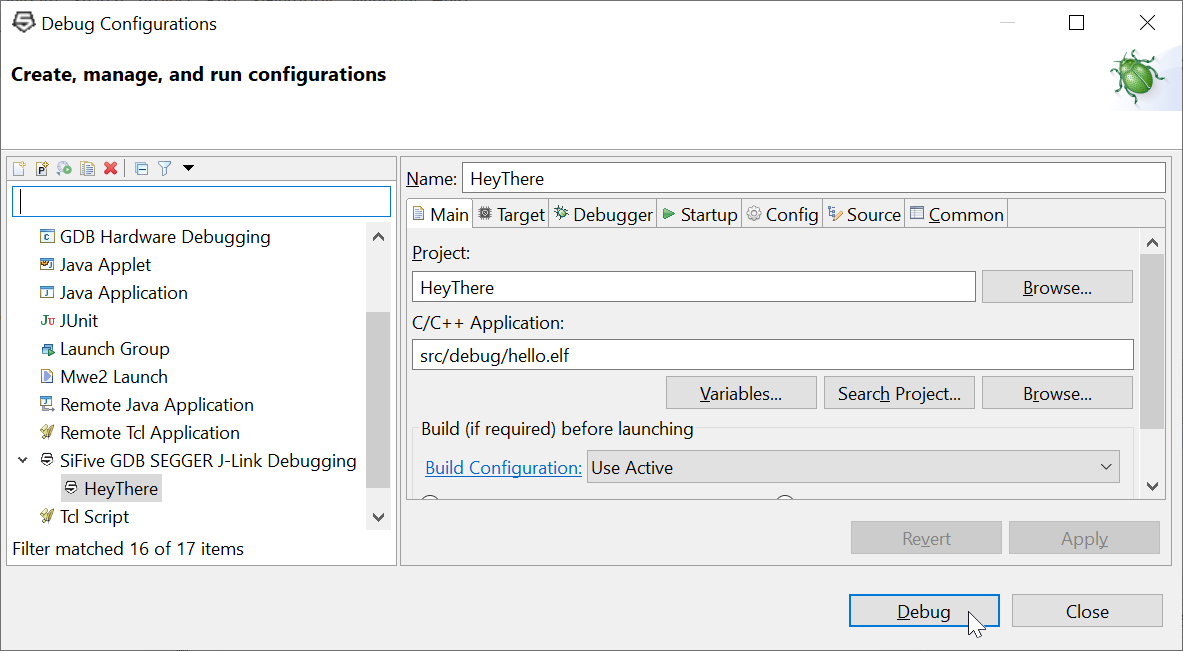
Перед началом процесса отладки убедитесь, что плата подключена к порту USB.

**Предупреждение:** Возможно, вам придется повторить эту процедуру позже, поэтому будьте внимательны!

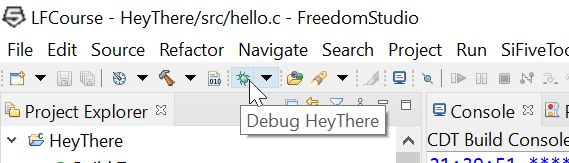
Поскольку мы не запускали процесс отладки при создании проекта, кнопка **Debug** не будет работать. Сначала нам нужно указать конфигурацию отладки. Это просто: это та конфигурация, которую мы создали несколько шагов назад. Просто выберите его, нажав на стрелку раскрывающегося списка кнопки **Debug**, а затем **Отладочные конфигурации ...**:

  
**Кнопка отладки**

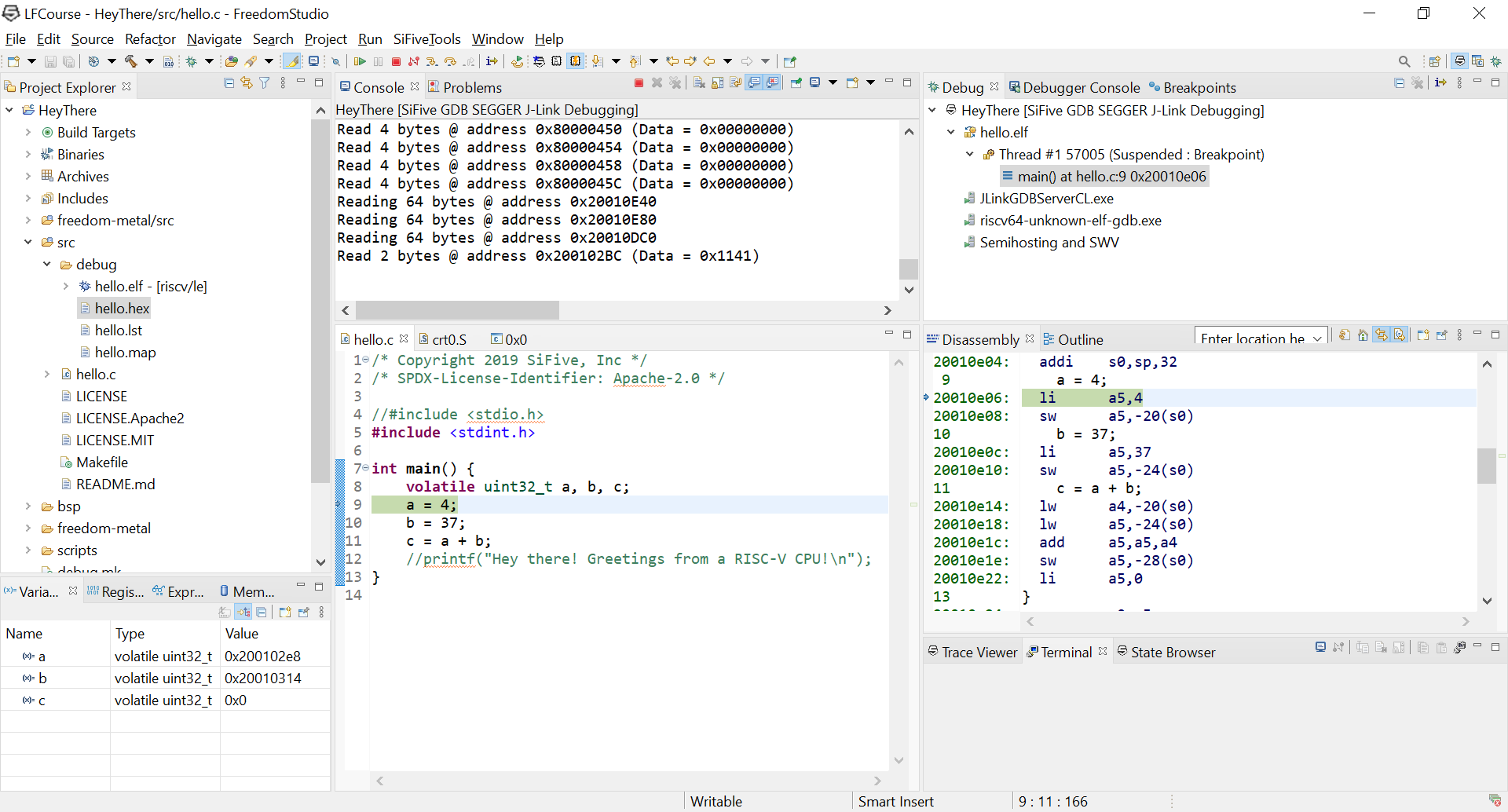
Вы найдете нужную конфигурацию в разделе “Debug J-LinkSiFive GDB SEGGER”. Она называется как ваш проект (HeyThere). Просто нажмите на **Debug**:

  
**Диалоговое окно конфигурации отладки Freedom Studio**

В следующий раз, когда вы захотите выполнить отладку, вы можете просто нажать кнопку **Debug** на панели инструментов, которая будет использовать последнюю конфигурацию. Здесь мы можем увидеть всплывающую подсказку, подтверждающую это:

  
**Кнопка Отладки со всплывающей подсказкой, указывающей, что будет использоваться конфигурация HeyThere**

Это то, что было отображено после запуска отладчика:

  
**Перспектива SiFive от Freedom Studio, показывающая сеанс отладки**

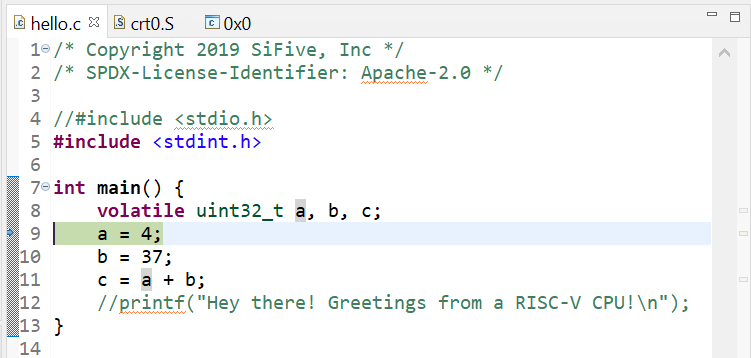
## Выполнение вашего кода в отладчике

Перспектива SiFive содержит все, что вам нужно для отладки вашего кода. Еще раз, это рекомендуемый подход ко всему в этом курсе.

Во-первых, у нас есть кнопки выполнения кода:

Code execution buttons including step execution  
  
**Возобновление, Приостановка, Завершение, Разъединение, Переход В режим, Перешагивание, Возврат к шагу, Пошаговый режим выполнения инструкций**

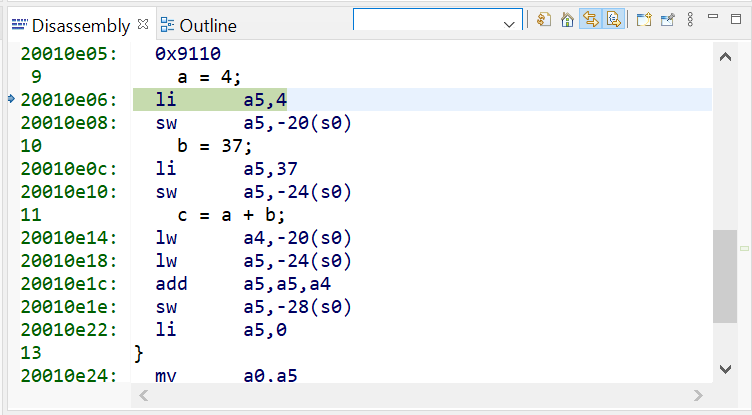
В окне Источник вы можете увидеть следующую строку кода для выполнения, выделенную зеленым цветом и отмеченную крошечной синей стрелкой в желобе слева:

  
**Исходный код C во время выполнения**

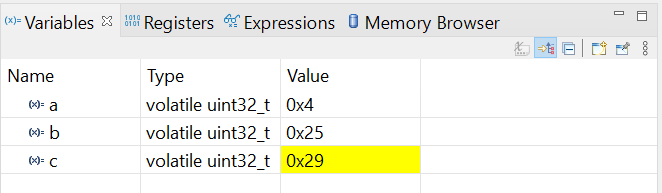
В любой момент времени вы можете переключить точки остановок, чтобы приостановить выполнение при достижении строки кода. Чтобы переключить точку остановки в строке кода, дважды щелкните по номеру строки в желобе, вертикальной черте слева от кода.

Обратите внимание, что не все строки помечены синим цветом в желобе. Эти синие метки указывают на строки кода, которые попали в машинный код. Некоторые строки исходного кода C оптимизированы компилятором и, следовательно, не имеют места для выполнения.

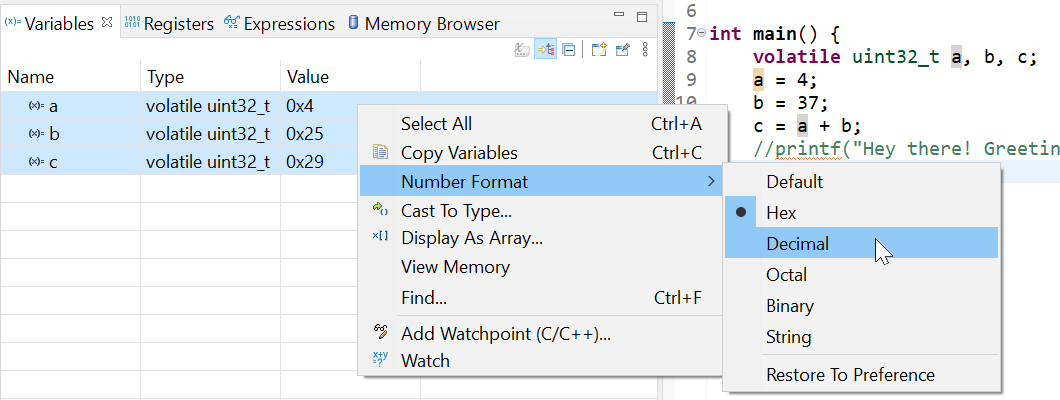
Если вы хотите выполнять пошагово код на ассемблере, показанный в окне ***дизассемблирования***, вы можете сделать это, переключив Режим пошагового выполнения инструкций справа от кнопок выполнения кода. Когда этот режим включен, кнопки выполнения шагов будут работать с уровнем детализации инструкции по сборке:

  
**Дизассемблированный код во время выполнения**

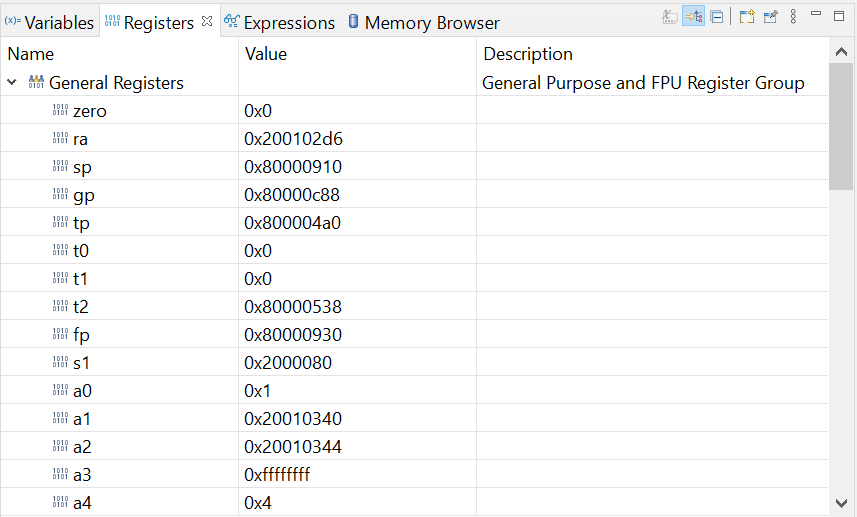
Переменные, регистры, выражения и представления памяти также сгруппированы в разделе внизу слева от SiFive перспективы. Здесь у нас есть вкладка Переменные, на которой отображаются все 3 переменные в шестнадцатеричном формате после выполнения:

  
**Переменные, регистры, выражения и представления памяти**

Вы можете изменить формат чисел, щелкнув правой кнопкой мыши на значении (или группе значений) и выбрав нужный формат из контекстного меню.

  
**Контекстное меню, показывающее, как изменить формат номера**

Здесь у нас есть вкладка registers, на которой показаны регистры процессора RV32I:

  
**Вид регистров процессора**

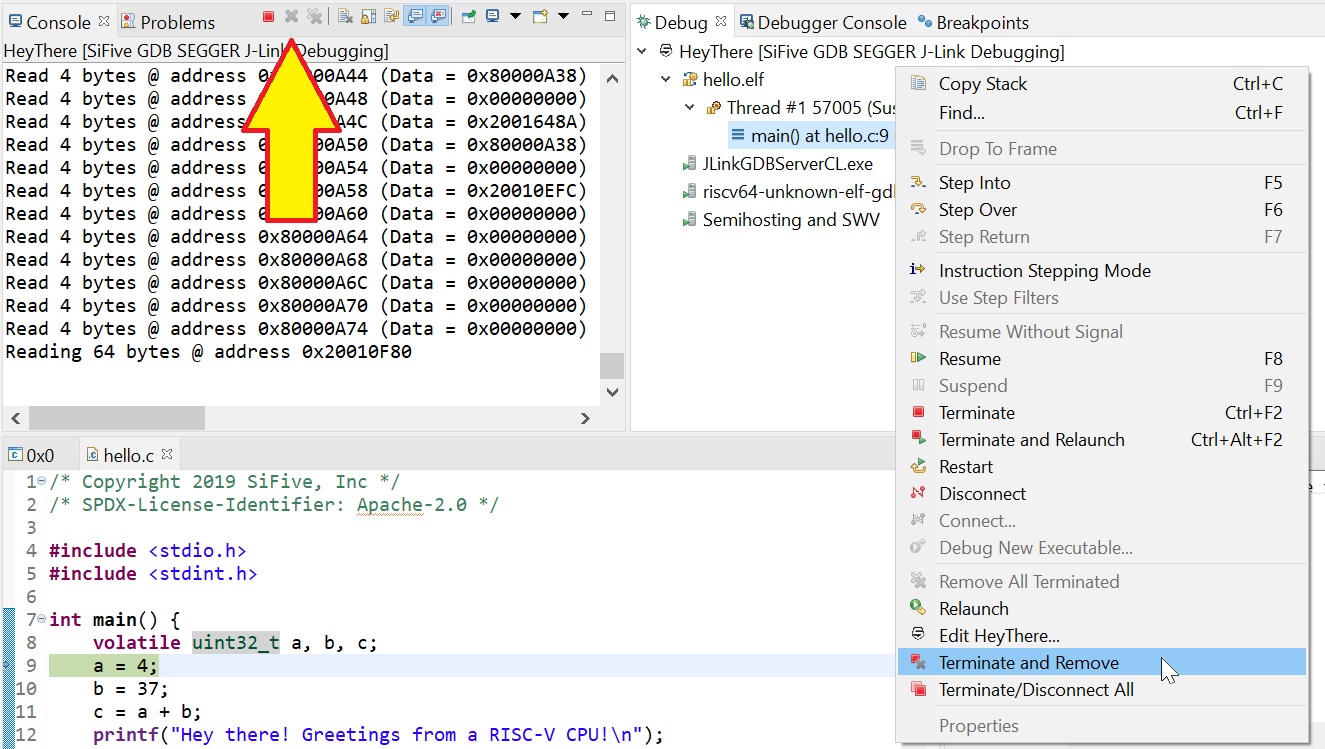
Аналогичным образом, вы можете вычислять выражения или просматривать необработанное содержимое памяти на оставшихся вкладках.

Когда вы закончите отладку, вы должны обязательно завершить и удалить все завершенные запуски, показанные в представлении **Debug** в правом верхнем углу. Если вы не сделаете этого сейчас, вам придется сделать это в следующий раз, когда вы захотите выполнить отладку. У вас есть несколько вариантов для этого:

Щелкните правой кнопкой мыши в представлении Отладки и выберите **Завершить и Удалить**, как показано на следующем рисунке.

Щелкните правой кнопкой мыши в представлении Отладки, выберите **Завершить**, а затем **Удалите все завершенные**.

Нажмите красную квадратную кнопку, показанную желтой стрелкой на следующем рисунке. Это опция **завершения** работы. Затем нажмите R**emove Launch** (одиночный X) или **Remove All Terminated Launches** (двойной X) кнопки справа от этой кнопки.

  
**Контекстное меню в представлении отладки, показывающее, как завершить и удалить запуск**

Это было краткое описание основных элементов Eclipse. Надеюсь, эти знания пригодятся вам в следующий раз, когда вам понадобится использовать Eclipse для других платформ разработки и языков программирования.

## Демонстрационная версия отладки Hello World (Видео)

[Эдуардо Корпеньо] Теперь давайте сделаем то же самое, но на этот раз запустим отладчик. Итак, позвольте мне нажать на кнопку Отладки. И теперь мы готовы к выполнению. Обратите внимание, что выполнение вот-вот запустит строку printf, которая является единственной строкой в основной функции. Мы также можем увидеть его реализацию на языке ассемблера в режиме дизассемблирования. Итак, позвольте мне открыть окно терминала. И теперь мы можем просто запустить код, нажав кнопку Возобновить. Вот оно. Привет, мир! После запуска отладчика всегда не забывайте останавливать и завершать все ваши процессы. Вы можете сделать это, нажав на кнопку Остановить, а затем перейти к завершенному процессу, щелкнуть правой кнопкой мыши и выбрать Удалить все завершенные.

## Подведение итогов

Поздравляю с прохождением этой главы!

Давайте воспользуемся моментом, чтобы оглянуться назад и обобщить некоторые знания, которые вы приобрели:

* Вы познакомились с платой Red-V Thing Plus.
* Вы узнали, как создать проект для платы Red-V Thing Plus в Freedom Studio.
* Вы узнали, как загружать свои приложения на плату, используя ее фиктивный интерфейс USB-накопителя.
* Вы также узнали, как отлаживать свои приложения с помощью отладчика Freedom Studio.

Не стесняйтесь возвращаться к этому материалу, если вам понадобится освежить его в памяти в будущем.