Быстрый старт с микроконтроллерами К1921ВГ015, К1921ВГ1Т, К1921ВГ3Т, К1921ВГ5Т, К1921ВГ7Т

(версия от 17.04.2026г.)

Содержание

1 Описание микроконтроллеров

Микросхемы К1921ВГ015, К1921ВГ1Т, К1921ВГ3Т, К1921ВГ5Т, К1921ВГ7Т представляют собой СБИС 32-разрядного микроконтроллера на базе ядра RISC-V, предназначенного для промышленных и потребительских приложений.

1.1 Состав микроконтроллера К1921ВГ015

В состав микроконтроллера входит 32-разрядное ядро BM-310 архитектуры RISC-V с поддержкой системы команд RV32IMFCN_ZBA_ZBB_ZBC_ZBS (Zicsr), набора команд умножения, арифметических и логических команд, встроенным модулем обработки команд с плавающей запятой с одинарной точностью FPU, кэшем команд и поддержкой отладочного интерфейса JTAG;

1.2 Состав микроконтроллера К1921ВГ1Т

В состав микроконтроллера двухъядерное 32-разрядное ЦПУ SCR5 архитектуры RISC-V с поддержкой системы команд RV32IMAFDCP (Zicsr), поддержкой DSP инструкций, поддержкой набора одноцикловых команд умножения с накоплением, команд централизованного управления потоком данных, арифметических и логических команд, встроенным модулем обработки команд с плавающей запятой с одинарной точностью FPU, поддержкой отладочного интерфейса JTAG;

1.3 Состав микроконтроллера К1921ВГ3Т

В состав микроконтроллера входит 32-разрядное ядро SCR4 архитектуры RISC-V с поддержкой системы команд RV32IMFDC (Zicsr), поддержкой набора одноцикловых команд умножения с накоплением, команд централизованного управления потоком данных, арифметических и логических команд, встроенным модулем обработки команд с плавающей запятой с одинарной точностью FPU, поддержкой отладочного интерфейса JTAG;

1.4 Состав микроконтроллера К1921ВГ5Т

В состав микроконтроллера входит 32-разрядное ядро SCR4 архитектуры RISC-V с поддержкой системы команд RV32IMFC (Zicsr), с поддержкой набора одноцикловых команд умножения с накоплением, команд централизованного управления потоком данных, арифметических и логических команд, встроенным модулем обработки команд с плавающей запятой с одинарной точностью FPU, поддержкой отладочного интерфейса JTAG;

1.5 Состав микроконтроллера К1921ВГ7Т

В состав микроконтроллера входит 32-разрядное ядро SCR4 архитектуры RISC-V с поддержкой системы команд RV32IMFC (Zicsr), поддержкой набора одноцикловых команд умножения с накоплением, команд централизованного управления потоком данных, арифметических и логических команд, встроенным модулем обработки команд с плавающей запятой с одинарной точностью FPU, поддержкой отладочного интерфейса JTAG;

2 Подготовка к работе

Для запуска Bare Metal кода на RISC-V понадобится:
• RISC-V toolchain: xpack-riscv-none-elf-gcc;
• Система сборки: make, cmake, platformio, Eclipse и тп.;
• startup файл;
• Скрипт компоновщика (.ld).

2.1 RISC-V toolchain

Флаг –march

Флаг **–march** определяет набор расширений, поддерживаемый текущим процессором.

Существующие расширения:

• **RV32I**: Стандартный набор целочисленных инструкций. Содержит набор из 32 регистров 32-бит.

• **RV32E**: ISA для встраиваемых систем. Совпадает с RV32I, но содержит только 16 регистров.

• **RV64I**: 64-бит версия RV32I.

• **M** - целочисленное умножение/деление.

• **A** - атомарные операции с памятью.

• **F/D** - вычисления с плавающей точкой одинарной/двойной точности.

• **C** - сжатый формат команд 16-бит.

• **P** - расширение для DSP/SIMD инструкции.

• **Zicsr** – Инструкции доступа к Control and Status Register (CSR).

• **zb*** - битовые операции.

Флаги **–march, –mabi** для К1921ВГ015, К1921ВГ1Т, К1921ВГ3Т, К1921ВГ5Т, К1921ВГ7Т приведены в таблице 2.1.

Таблица 2.1 – Флаги –march, –mabi

Микроконтроллер	Флаг **–march**	Флаг **–mabi**
К1921ВГ015	–march=rv32imfc_zba_zbb_zbc_zbs_zicsr	–mabi=ilp32f
К1921ВГ1Т	–march=rv32imafdcp_zicsr	–mabi=ilp32d
К1921ВГ3Т	–march=rv32imfdc_zicsr	–mabi=ilp32d
К1921ВГ5Т	–march=rv32imfc_zicsr	–mabi=ilp32f
К1921ВГ7Т	–march=rv32imfc_zicsr	–mabi=ilp32f

Флаг –mabi

Флаг **–mabi** определяет используемый ABI:

• **ilp32** - int, long, и указатели имеют длину 32-бит. long long длину 64-бит, char длину 8-бит, и short длину 16-бит.

• **Lp64** - long и указатели имеют длину 64-бит, но int длину 32-бит. Остальные типы такие же как в ilp32.

• **””** (пустая) – Целочисленные аргументы функций передаются в регистрах, с плавающей точкой через стек.

• **f**: 32-бит и меньше аргументы с плавающей точкой передаются через регистры FPU. Данная ABI требует поддержку F расширения.

• **d**: 64-бит и меньше аргументы с плавающей точкой передаются через регистры FPU. Данная ABI требует поддержку D расширения

На рисунке ниже приведены установки флагов –**march** и **–mabi** в среде Syntacore IDE.

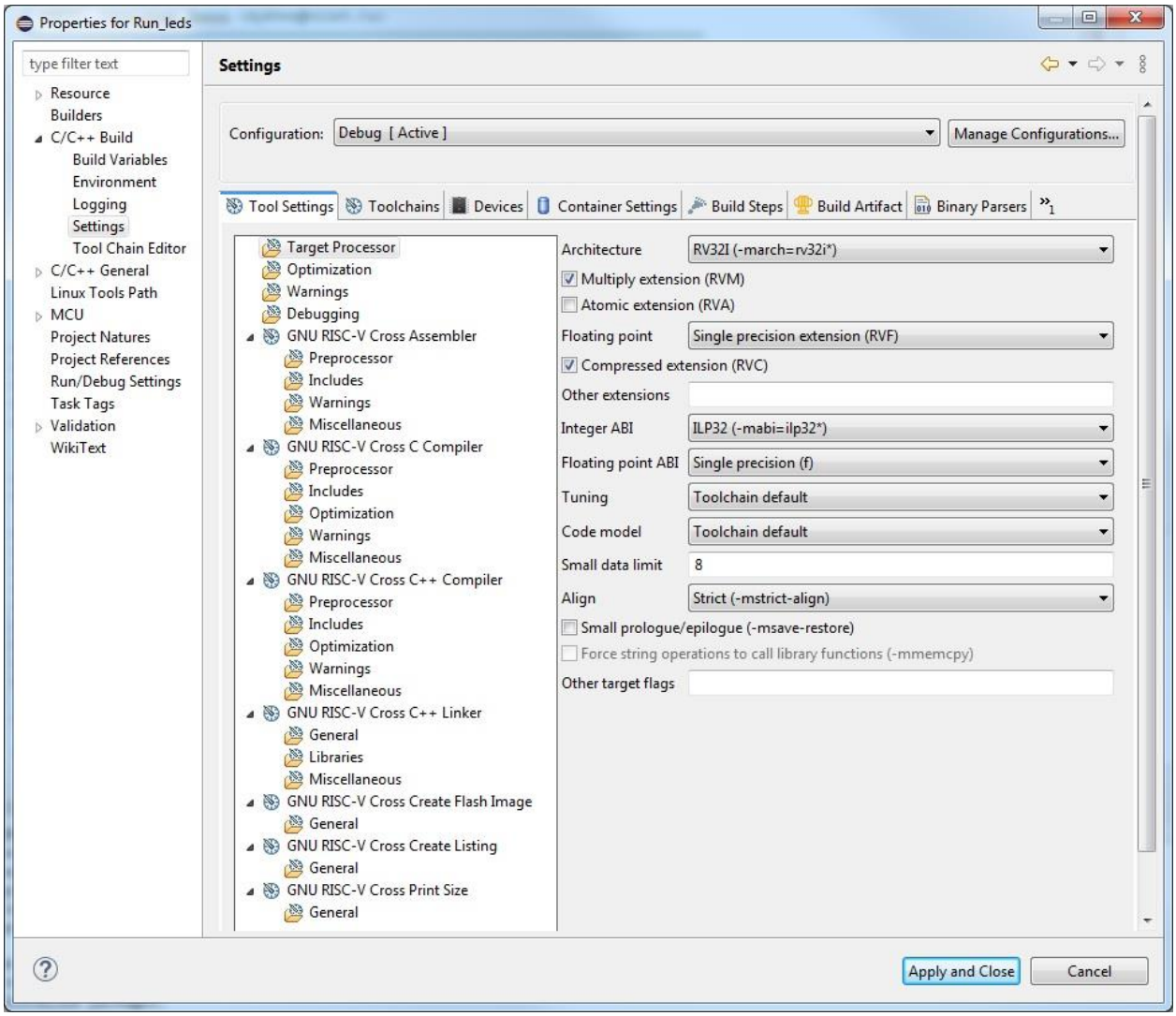

Рисунок 2.1 – Установки флагов –march и –mabi для К1921ВГ015

2.2 Аппаратные отладчики JTAG

Для обеспечения взаимодействия между интегрированной средой разработки Eclipse (с GCC компилятором для RISC-V и системой отладки OpenOCD), установленной на персональном компьютере, и отладочными ресурсами, встроенными в микроконтроллер, можно использовать адаптер Olimex ARM-USB-OCD-H, J-Link, cmsis-dap или отладчик, построенный на ИС FT2232.

Для отладки ИС К1921ВГ015 также можно использовать отладчики BlueProg или sipeed-rv debugger. Данные отладчики ведут обмен по JTAG на частоте TCK 2400 КГц и не реагируют на команды изменения скорости «adapter speed» OpenOCD, поэтому не подходят для ИС К1921ВГ1Т, К1921ВГ3Т, К1921ВГ5Т, К1921ВГ7Т. Для подключения к ИС К1921ВГ1Т, К1921ВГ3Т, К1921ВГ5Т, К1921ВГ7Т необходимо использовать частоту TCK не более (200 – 250) КГц.

Также можно использовать любой отладчик поддерживающий интерфейс JTAG и имеющий драйвер для OpenOCD.

3 Настройка среды Syntacore IDE

Для разработки проектов, а также запуска примеров в среде Syntacore IDE потребуются:

– Набор инструментов для разработки Syntacore Development Toolkit (можно скачать по ссылке: https://syntacore.com/page/products/sw-tools). Имеются версии под Windows и Linux.

– Компилятор, содержащий программный обход (добавление команды «NOP») для обхода п.5 Errata.

– Набор программных средств разработки для микроконтроллеров RISC-V НИИЭТ NIIET_RISCV SDK (расположены по ссылке: https://gitflic.ru/project/niiet/niiet_riscv_sdk). Скачать можно архивом, или же с помощью git-bush используя ссылку https://gitflic.ru/project/niiet/k1921vkx_sdk.git. Необходимая ветка – master.

– Инструмент отладки OpenOCD с драйвером Flash К1921VG015 (расположен в релизах репозитория: https://gitflic.ru/project/niiet/openocd/release).

3.1 Подготовка среды Syntacore IDE для работы

Ниже приведены шаги по подготовке среды на примере К1921ВГ015 и они также справедливы для любого из микроконтроллеров: К1921ВГ1Т, К1921ВГ3Т, К1921ВГ5Т, К1921ВГ7Т.

Шаг 1 – Установка Syntacore Development Toolkit

Переходим по ссылке https://syntacore.com/page/products/sw-tools, скачиваем архив **sc-dt-2025.06-win.zip** (для Windows) или **sc-dt-2025.06.tar.gz** (для Linux) и распаковываем в любом удобном для работы месте.

Шаг 2 – Скачивание репозиториев SDK

Скачивание осуществляется с помощью git-bash. Консольной командой **cd** выбираем директорию, куда будет скачан пакет SDK.

Далее для скачивания в консоли git-bush вводим команду: **git clone -b master https://gitflic.ru/project/niiet/niiet_riscv_sdk.git**

Набор программных средств разработки для микроконтроллеров RISC-V НИИЭТ NIIET_RISCV SDK будет скачан в выбранную ранее директорию.

Репозиторий пример был разделен. Теперь простые примеры, демонстрирующие работу периферийных блоков микроконтроллеров, размещаются в отдельных репозиториях для каждого микроконтроллера:

- https://gitflic.ru/project/niiet/k1921vg015_sdk - содержит примеры программ для К1921ВГ015;
- https://gitflic.ru/project/niiet/k1921vg3t_sdk - содержит примеры программ для К1921ВГ3Т;
- https://gitflic.ru/project/niiet/k1921vg5t_sdk - содержит примеры программ для К1921ВГ5Т;
- https://gitflic.ru/project/niiet/k1921vg7t_sdk - содержит примеры программ для К1921ВГ7Т.

Структура каталогов NIIET_RISCV SDK:

– **platform**: Общие библиотеки.
  - Device: Заголовочные файлы микроконтроллера, файлы startup и скрипты линкера.
  - middleware: Промежуточное ПО: протоколы, интерфейсы.
  - plib015: Библиотека периферии К1921ВГ015.

– **projects**: Примеры проектов.
  - NIIET-DEMO-K1921VG3T: Проекты для отладочной платы NIIET-DEMO-K1921VG3T с управлением через регистры МК.
  - NIIET-DEV-K1921VG015: Проекты для отладочной платы NIIET-DEV-K1921VG015 с управлением через регистры МК.
  - plib015: Проекты для отладочной платы NIIET-DEV-K1921VG015 с использованием библиотеки PLIB015.

– **templates**: Шаблоны проектов.
  - k1921vg015-bare : Шаблон проекта для К1921ВГ015.
  - k1921vg015-plib015 : Шаблон проекта для К1921ВГ015 с использованием библиотеки PLIB015.

– **tools**: Вспомогательный инструментарий.
  - openocd : Файлы для осуществления отладки МК.
  - svd : SVD файлы микроконтроллеров.
  - niiet-aspect-1.0.1.vsix : Расширение для VSCode.
  - sc-dt_Patch_Niiet_Linux.zip : Архив для поддержки микроконтроллеров К1921ВГ3Т, К1921ВГ015 в Syntacore Development Toolkit (ОС Linux).
  - sc-dt_Patch_Niiet_Win32.zip : Архив для поддержки микроконтроллеров К1921ВГ3Т, К1921ВГ015 в Syntacore Development Toolkit (ОС Windows).

Шаг 3 – Интеграция поддержки МК средой Syntacore IDE

Для ОС Windows нам понадобится архив **sc-dt_Patch_Niiet_Win32.zip** из релизов репозитория https://gitflic.ru/project/niiet/openocd/release. Прямая ссылка на момент написания руководства: https://gitflic.ru/project/niiet/openocd/release/4a4e7431-0990-455c-9a8b-487bc98f72f8/de378fc0-4433-4b83-a17e-2077783063da/download. Его необходимо распаковать в каталог **sc-dt** ранее скачанной среды Syntacore IDE (с подтверждением замены файлов). Также после распаковки необходимо проверить наличие файла **k1921vg015.cfg** в каталоге: \sc-dt\tools\share\openocd\scripts\target.

Шаг 4 – Запуск и настройка среды Syntacore IDE

Запуск среды осуществляется с помощью скрипта **start-scr-ide.cmd** (для ОС Windows) находящегося в каталоге **sc-dt**. При первом запуске появится окно выбора рабочей области workspace.

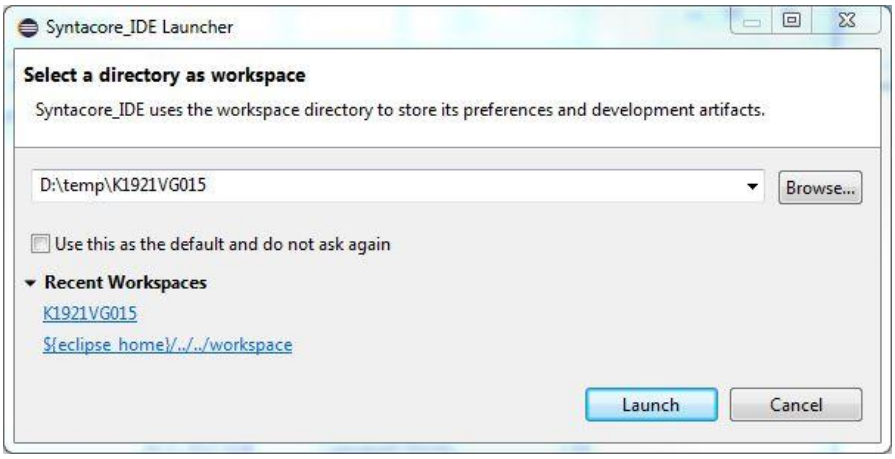

Рисунок 3.1 – Настройка workspace

В окне настройки рабочей области (рисунок 3.1) нужно выбрать и указать путь к рабочему пространству, где будут находиться проекты по работе с микроконтроллером К1921ВГ015. Выбираем каталог «projects/NIIET-DEV-K1921VG015» скачанного репозитория https://gitflic.ru/project/niiet/k1921vg015_sdk.

Далее, для импорта проектов заходим в раздел File/Import и выбираем Projects from Folder of Archive.

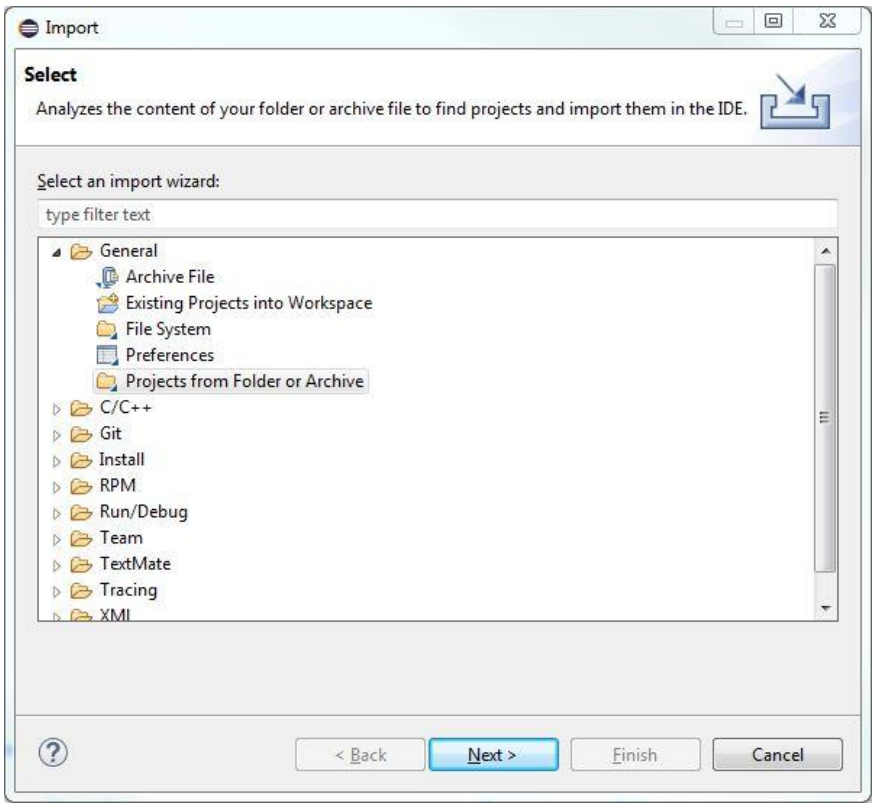

Рисунок 3.2 – Импорт проектов в рабочую область

Далее выбираем необходимые для импорта проекты (см. рисунок 3.3).

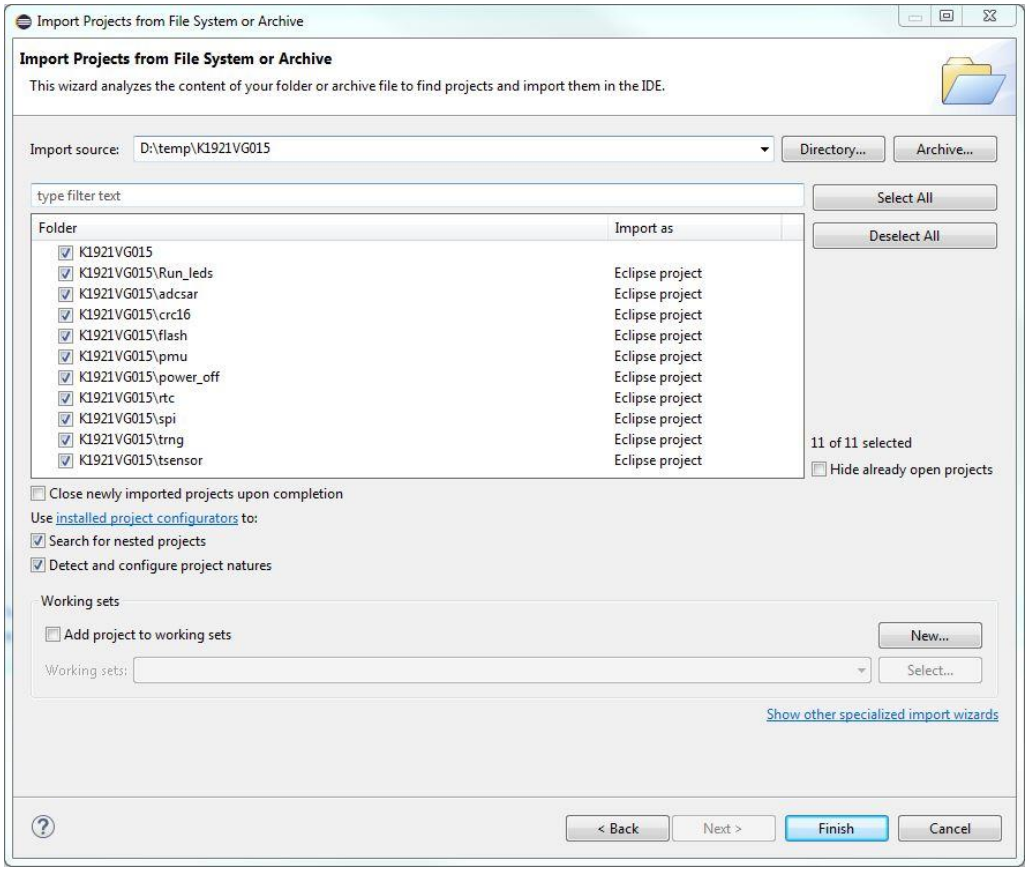

Рисунок 3.3 – Выбор проектов для импорта в рабочую область

После импортирования выбранные проекты появятся в окне Project Explorer (см. рисунок 3.4).

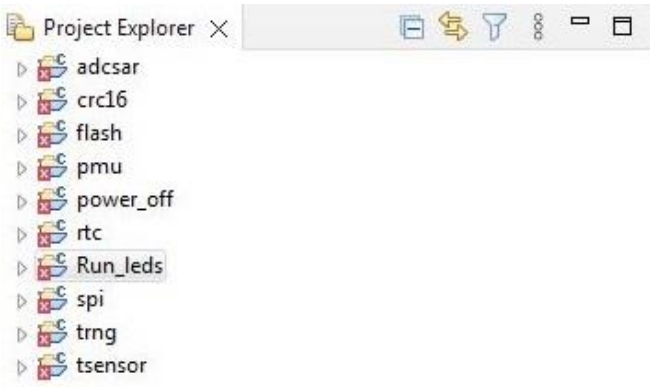

Рисунок 3.4 – Окно с импортированными проектами

Шаг 5 – Сборка проекта

Для сборки проекта необходимо вызвать контекстное меню нажатием правой кнопки мыши по названию проекта в окне Project Explorer и выбрать во всплывающем окне пункт «Build Project». О результате сборки можно узнать из окна консоли, пример указан на рисунке 3.5.

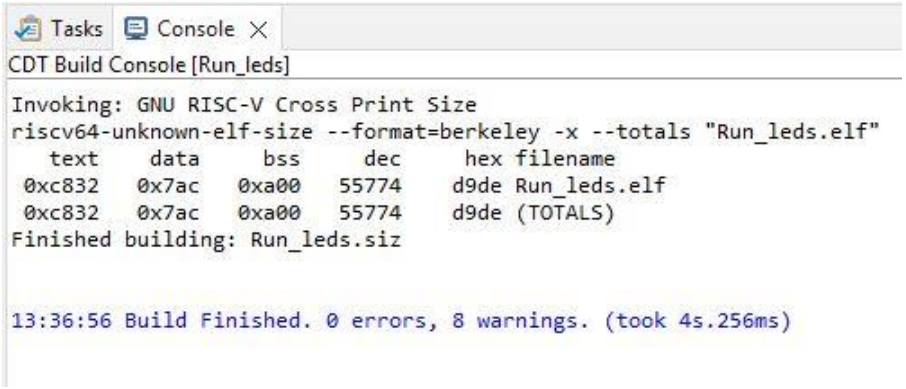

Рисунок 3.5 – Результаты сборки проекта

Шаг 6 – Установка драйверов JTAG-эмулятора под Widnows

Данный шаг необходим при подготовке среды под ОС Widnows. Для ОС Linux данный шаг пропускается, поскольку она уже содержит драйвер libusb, с помощью которого OpenOCD взаимодействует с выбранным отладчиком.

Чтобы установить драйверы для JTAG-эмулятора (таких как «J-Link» или отладчик, построенный на ИС FT2232), совместимых с используемым в Syntacore IDE отладчиком «OpenOCD» необходима программа «Zadig». Программа «Zadig» находится в каталоге «sc-dt\tools\bin». Скачать другие версии программы «Zadig» можно по ссылке: https://zadig.akeo.ie.

Подключите JTAG-адаптер к USB-порту вашего ПК. Дождитесь, пока закончится процедура автоматической установки драйверов средствами Windows. Независимо от успешности её результата по окончании запустите программу «Zadig» (см. рисунок 3.6).

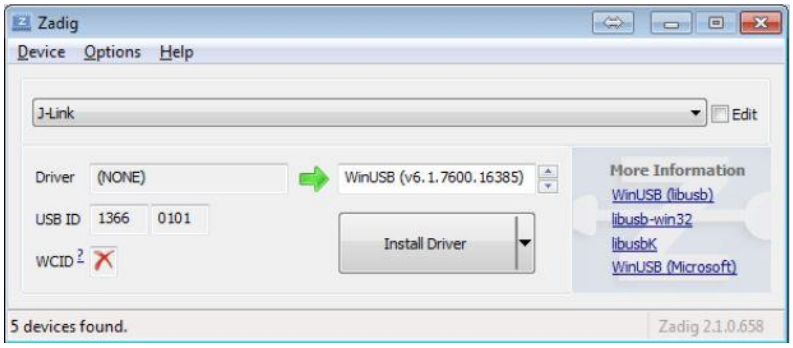

Рисунок 3.6 – Главное окно программы Zadig

В меню «Options» поставьте галочку «List all devices». В выпадающем списке выберите используемое устройство (в нашем примере «J-Link») или неизвестное устройство (убедитесь в таком случае, что это именно jtag подключая/отключая USB кабель).

В ОС Windows JTAG-эмулятор «J-Link» определяется как два интерфейса:

– "CDC (Interface 0)" можно использовать в качестве USB-UART преобразователя.

– "BULK Interface (Interface 2)" используется в качестве JTAG отладчика.

В ОС Windows необходимо сделать подмену драйвера на WinUSB с помощью программы zadig.

В ОС Windows ИС FT2232 (FTDI), установленная на плате К1921ВГ015 (КФДЛ.441461.029) определяется как два интерфейса:

– "Dual RS232-HS (Interface 0)" на плате используется в качестве USB-UART преобразователя, подключенного к UART0 и должен использовать оригинальные драйвера "FTDI".

– "Dual RS232-HS (Interface 1)" на плате используется в качестве JTAG отладчика. В ОС Windows необходимо сделать подмену драйвера на WinUSB с помощью программы zadig.

В строке справа от стрелки выберите драйвер «WinUSB (v.xxxx)». Нажмите на кнопку «Install Driver» (если кнопка называется по-другому, значит у вас установлены другие драйвера и нужно нажать «Replace Driver»). Драйвера установлены. Однако по опыту эксплуатации рекомендуется вынуть/вставить JTAG из USB компьютера – без этого драйвер может не заработать. В некоторых случаях требуется перезагрузка. В диспетчере устройств на вкладке с устройством JTAG по кнопке «сведения» должно открываться окно с похожим содержимым, указанном на рисунке 3.7.

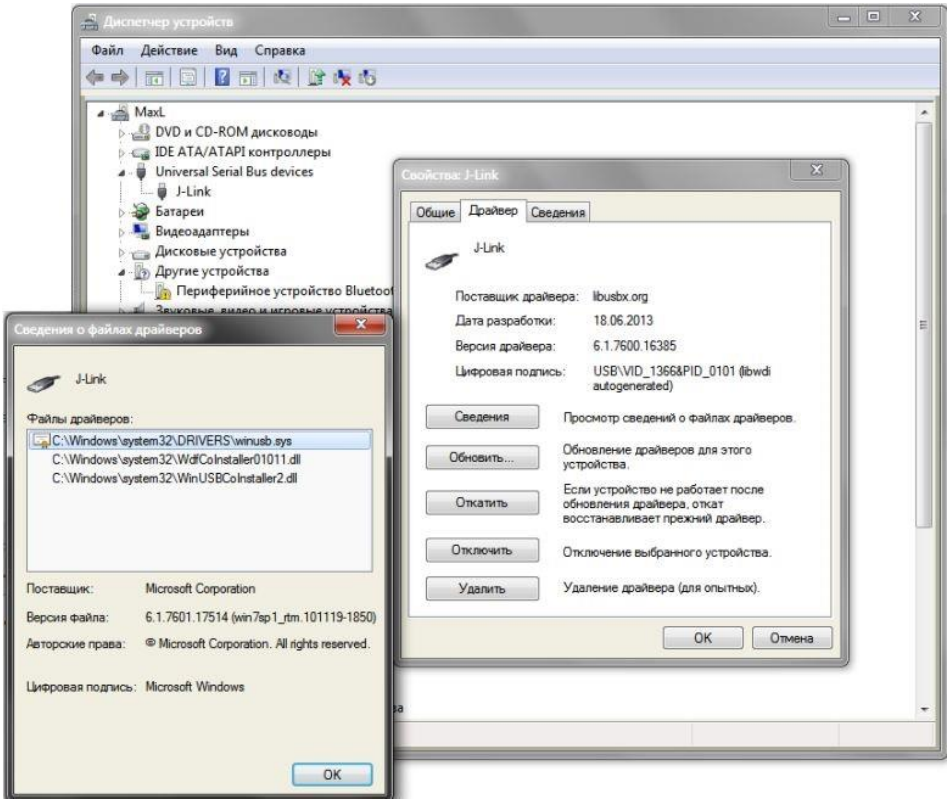

Рисунок 3.7 – Сведения о файлах драйверов

Из рисунка видно, что использован драйвер «WinUSB». Если драйвер не установился или не работает, можно попробовать нажать на кнопку «Install Driver» несколько раз (если «Zadig» сообщает, что установка не удалась), вынуть/вставить JTAG из USB, перезагрузить компьютер, отключить антивирус, запустить программу с правами администратора, включить службу "Центр обновления Windows" (если она отключена), проверить состояние устройства в диспетчере устройств Windows. Чтобы вернуться к стандартным драйверам JTAG для работы с другой средой, в диспетчере устройств Windows найдите свое устройство (например J-Link), правой кнопкой → Обновить драйверы... → Выполнить поиск драйверов на этом компьютере → Выбрать драйвер из списка уже установленных" и выбрать, например, "J-link driver".

После замены драйвера в главном окне программы zadig мы также можем убедиться, что драйвер заменился. Так, при выборе используемого интерфейса, напротив надписи "Driver" должен отобразиться драйвер "WinUSB".

При использовании ОС Windows10 было замечено, что при работе под учетной записью пользователя с правами администратора драйвер успешно подменяется, однако продолжает работать до завершения работы или перезагрузки ОС. После перезагрузки необходимо снова подменить драйвер с помощью программы Zadig. При работе под учетной записью пользователя без прав администратора, замену драйвера необходимо проводить в режиме отключенной проверки цифровой подписи драйверов (см. подраздел «Отключение проверки подписи драйвера с помощью параметров загрузки»).

Примечание – При замене драйверов на Windows 10 с помощью программы «Zadig» следует соблюдать осторожность, поскольку можно заменить общий (generic) драйвер, который Windows может автоматически использовать при подключении нового устройства. Generic-драйвер может одновременно использоваться несколькими USB-устройствами. После такой процедуры остальные устройства перестанут нормально функционировать.

Отключение проверки подписи драйвера с помощью параметров загрузки

Способ, отключающий проверку цифровой подписи единожды, при перезагрузке системы и до следующей перезагрузки – использование параметров загрузки Windows 10.

1) Для того, чтобы воспользоваться способом, зайдите в «Параметры» — «Обновление и безопасность» — «Восстановление». Затем, в разделе «Особые варианты загрузки» нажмите «Перезагрузить сейчас».

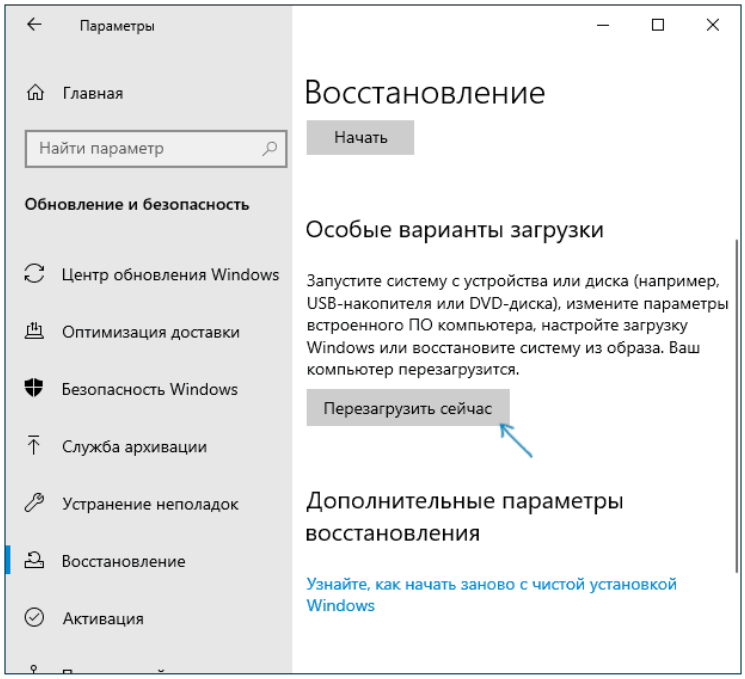

Рисунок 3.8 – Запуск особых вариантов загрузки Windows 10

2) После перезагрузки, пройдите по следующему пути: «Поиск и устранение неисправностей» (или «Диагностика») — «Дополнительные параметры» — «Параметры загрузки» и нажмите кнопку «Перезагрузить».

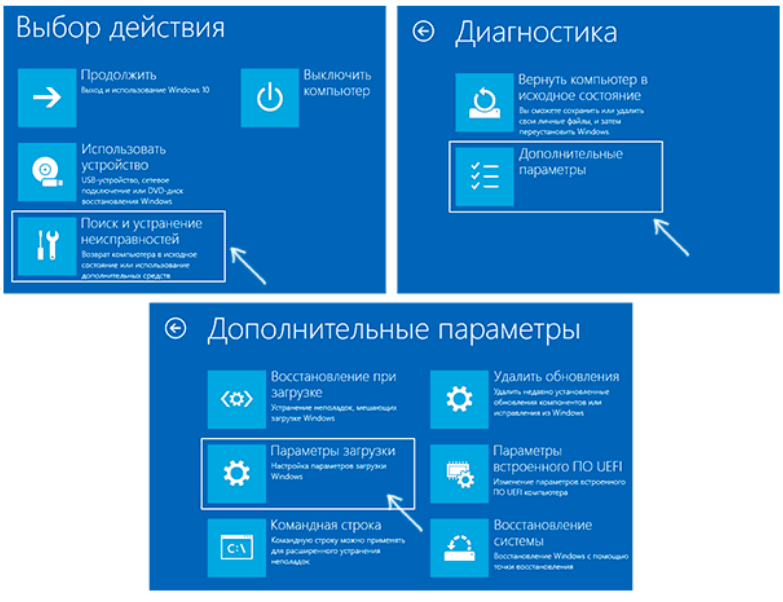

Рисунок 3.9 – Меню особых вариантов загрузки Windows 10

3) После перезагрузки появится меню выбора параметров, которые будут использоваться в этот раз в Windows 10.

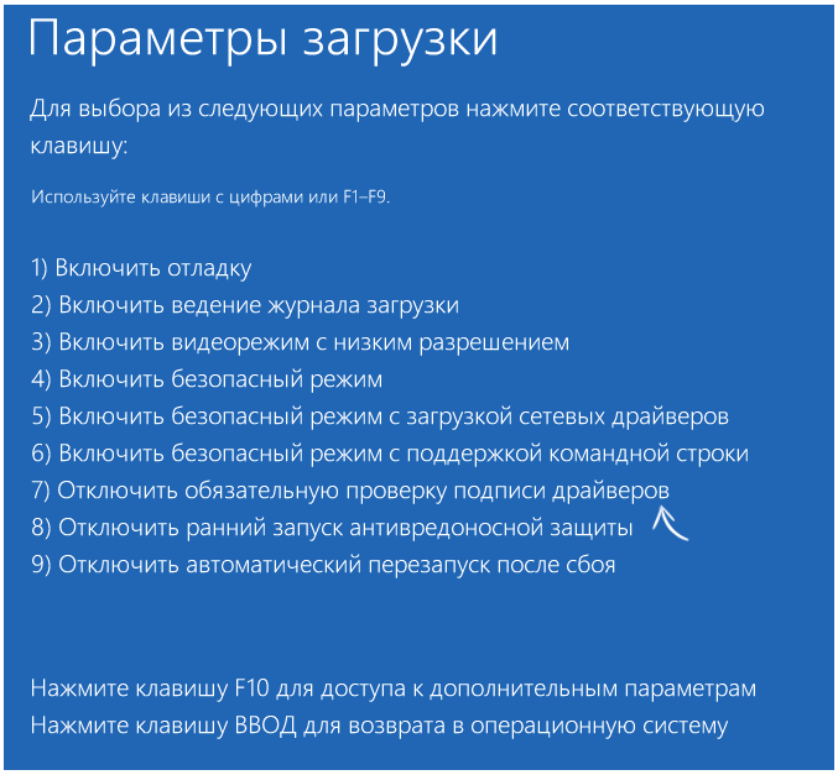

Рисунок 3.10 – Запуск без проверки цифровой подписи драйвера

4) Для того, чтобы отключить проверку цифровой подписи драйверов, выберите соответствующий пункт, нажав клавишу 7 или F7 (или Fn+F7 на некоторых ноутбуках).

Готово, после перезагрузки Windows 10 запустится с отключенной проверкой подписи драйверов и вы сможете установить свой драйвер без цифровой подписи.

Шаг 7 – Настройка и запуск отладочной сессии

Запуск отладочной сессии с файлами *.launch

Изначально в каталогах sc-dt всех примеров проектов отсутствуют файлы запуска сессии отладки *.launch. Для их генерации в каталоге SDK «tools» расположены скрипты prepare_launch_*.bat. Каждый скрипт в имени файла содержит название отладчика и предназначен для генерации файлов запуска сессии отладки *.launch с настройками соответствующего отладчика. При использовании отладчика для К1921ВГ015 на плате КФДЛ.441461.029 необходимо выполнить скрипт prepare_launch_Onboard_FTDI.bat, при использовании JLink - выполнить скрипт prepare_launch_JLink.bat.

После выполнения скрипта в каждом примере проекта в каталоге sc-dt появится файл запуска сессии отладки *.launch с именем проекта и названием отладчика. В менеджере проекта Syntacore_IDE эти файлы также отображаются в структуре каждого проекта.

Для запуска сессии отладки достаточно в правом списке выбрать необходимый проект для отладки, в левом списке выбрать Debug - для запуска проекта и останова на функции main() или Run - для запуска проекта (см. рисунок 3.11). Далее нажать кнопку  для запуска в режиме «Debug» или кнопку  - в режиме «Run».

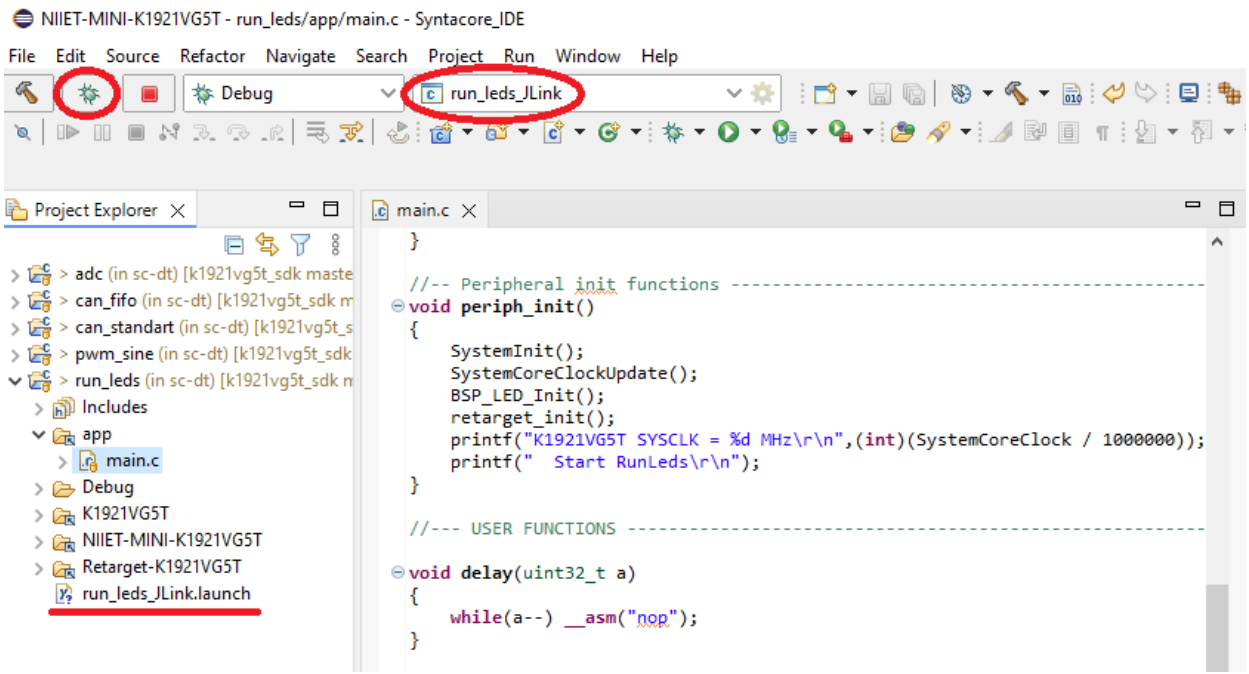

Рисунок 3.11 – Выбор сессии отладки

Настройка отладочной сессии “Debug Configurations”

Для настройки отладочной сессии для работы с микроконтроллером К1921ВГ015 сначала нужно включить режим перспективы “Debug” кнопкой “Open Perspective” в правом верхнем углу рабочего окна Syntacore IDE (см. рисунок 3.12).

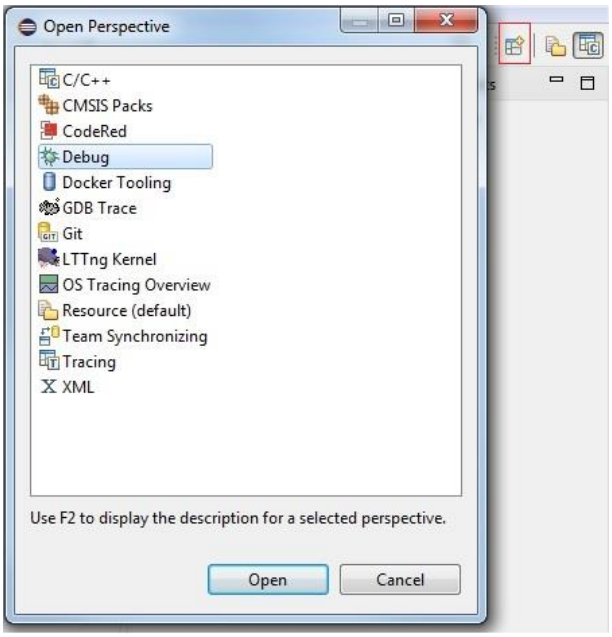

Рисунок 3.12 – Включение режима Debug

Затем нужно зайти в настройки отладки выбранного проекта (см. рисунок 3.13).

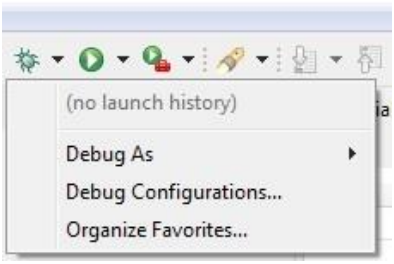

Рисунок 3.13 – Настройка конфигурации отладки

При открытии окна Debug Configurations двойным щелчком мыши по вкладке “GDB OpenOCD Debugging” создаем отладочную конфигурацию выбранного проекта (см. рисунок 3.14).

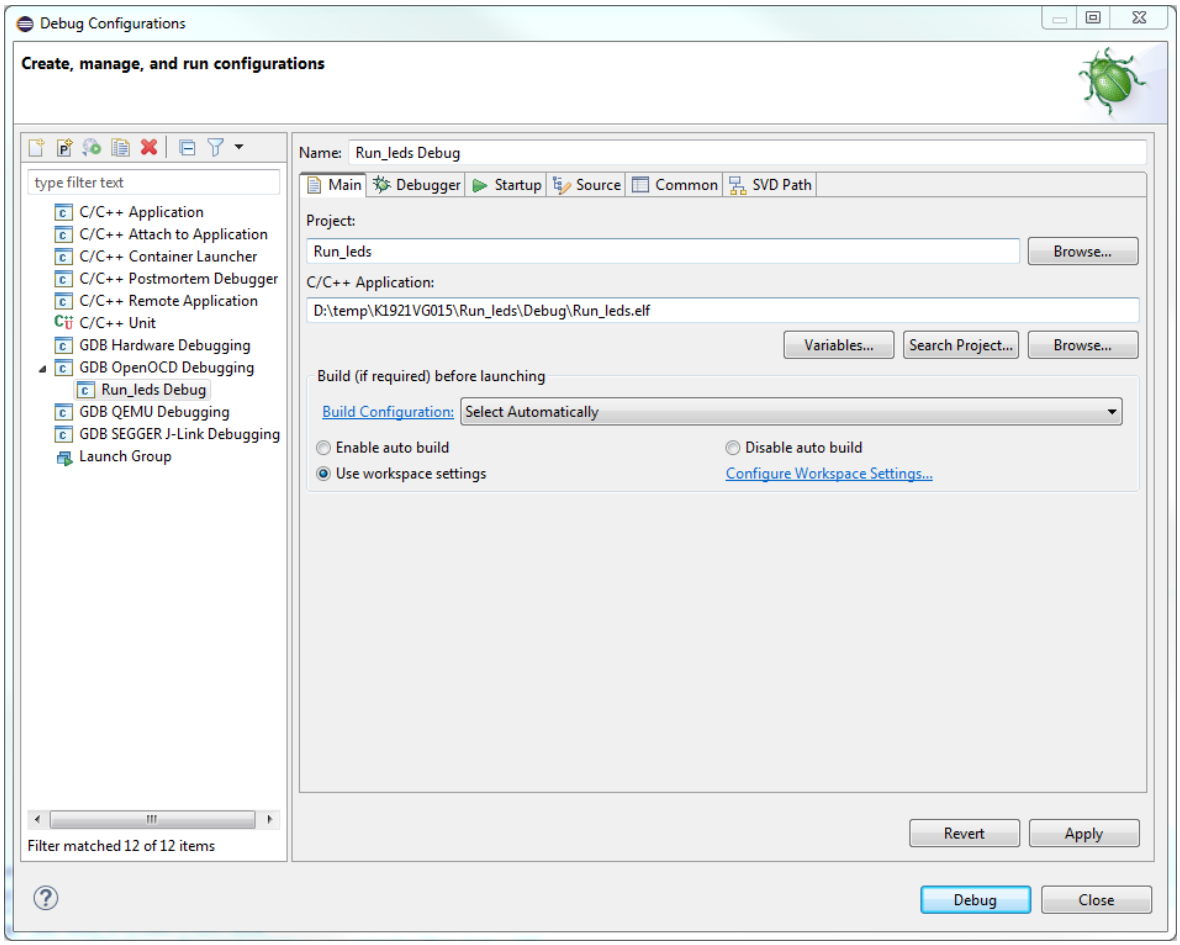

Рисунок 3.14 – Создание отладочной конфигурации проекта

В строке “C/C++ Application” нужно указать путь к **.elf** файлу, созданному при сборке проекта. Набор опций конфигурации отладчика находится на вкладке “Debugger” в разделе “Config options” (см. рисунок 3.15).

Состав набора опций конфигурации:

-s ${eclipse_home}/../tools/share/openocd/scripts
-s ${eclipse_home}/../tools/share/openocd/scripts/interface/ftdi
-s ${eclipse_home}/../tools/share/openocd/scripts/interface
-s ${eclipse_home}/../tools/share/openocd/scripts/target
-f jlink.cfg

-f k1921vg015.cfg
-c "init;halt"

Если используется отладчик, отличный от jlink, тогда в строке -f jlink.cfg вместо jlink.cfg необходимо указать конфигурационный файл используемого отладчика. Путь к местонахождению конфигурационных файлов: sc-dt/tools/share/openocd/scripts/interface.

При использовании макетно-отладочной платы VG015 DK разработки ДЦЭ Восток, с установленной ИС FTDI на плате, необходимо использовать конфигурационный файл vg015_dev_onboard_ftdi.cfg. В таком случае заменяем строку «-f jlink.cfg» на «-f vg015_dev_onboard_ftdi.cfg».

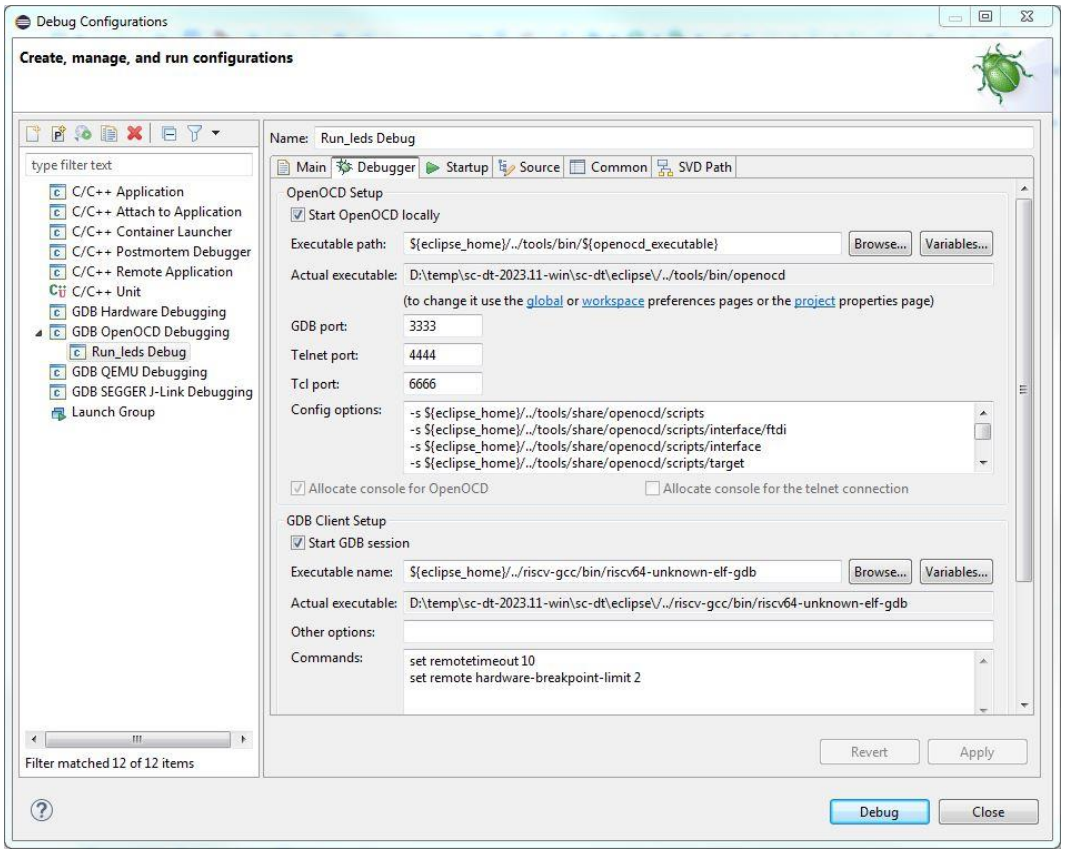

Рисунок 3.15 – Набор конфигурации отладчика

Теперь при нажатии кнопки Debug собранный проект будет записан в микроконтроллер и запущена сессия отладки. В дальнейшем отладку можно будет запускать быстрее, используя историю запусков (см. рисунок 3.16).

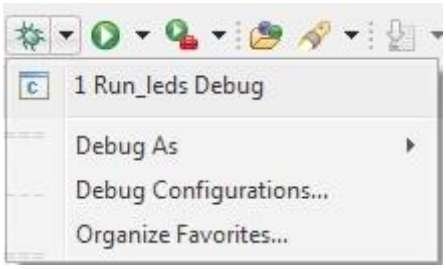

Рисунок 3.16 – Запуск ранее сконфигурированной отладочной сессии

При успешном подключении консоль Syntacore IDE выдаст следующие сообщения (см. рисунок 3.17).

```
Open On-Chip Debugger -00247-g90632ca40-dirty (2025-08-01-14:40)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
DEPRECATED! use 'ftdi layout_init' not 'ftdi_layout_init'
DEPRECATED! use 'adapter speed' not 'adapter_khz'
sw_reset_halt
Info : clock speed 1000 kHz
Info : JTAG tap: K1921VG015.cpu tap/device found: 0x00000d5b (mfg: 0x6ad (CloudBEAR LLC), part: 0x0000, ver: 0x0)
Info : JTAG tap: BSCAN.bscan tap/device found: 0x20c5f44d (mfg: 0x226 (MediaTek), part: 0x0c5f, ver: 0x2)
Info : datacount=2 progbufsize=16
Info : Disabling abstract command reads from CSRs.
Info : Examined RISC-V core; found 1 harts
Info :  hart 0: XLEN=32, misa=0x40103124
Info : starting gdb server for K1921VG015.cpu on 3333
Info : Listening on port 3333 for gdb connections
Started by GNU MCU Eclipse
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : accepting 'gdb' connection on tcp/3333
Info : k1921vg015 detected
Warn : Prefer GDB command "target extended-remote :3333" instead of "target remote :3333"
Info : JTAG tap: K1921VG015.cpu tap/device found: 0x00000d5b (mfg: 0x6ad (CloudBEAR LLC), part: 0x0000, ver: 0x0)
Info : JTAG tap: BSCAN.bscan tap/device found: 0x20c5f44d (mfg: 0x226 (MediaTek), part: 0x0c5f, ver: 0x2)
Info : JTAG tap: K1921VG015.cpu tap/device found: 0x00000d5b (mfg: 0x6ad (CloudBEAR LLC), part: 0x0000, ver: 0x0)
Info : JTAG tap: BSCAN.bscan tap/device found: 0x20c5f44d (mfg: 0x226 (MediaTek), part: 0x0c5f, ver: 0x2)
Info : JTAG tap: K1921VG015.cpu tap/device found: 0x00000d5b (mfg: 0x6ad (CloudBEAR LLC), part: 0x0000, ver: 0x0)
Info : JTAG tap: BSCAN.bscan tap/device found: 0x20c5f44d (mfg: 0x226 (MediaTek), part: 0x0c5f, ver: 0x2)
Info : Erasing sectors from 0 to 0
Info : Padding image section 0 at 0x800000be with 2 bytes
Info : Odd number of words to write, padding with 0xFFFFFFFF
Info : Start block write
Info : buffer_size 1488, count 93
Info : thisrun_count 93, target_address 0x0
Info : Disabling abstract command writes to CSRs.
Info : JTAG tap: K1921VG015.cpu tap/device found: 0x00000d5b (mfg: 0x6ad (CloudBEAR LLC), part: 0x0000, ver: 0x0)
Info : JTAG tap: BSCAN.bscan tap/device found: 0x20c5f44d (mfg: 0x226 (MediaTek), part: 0x0c5f, ver: 0x2)
Info : [K1921VG015.cpu] Found 4 triggers
Info : dropped 'gdb' connection
shutdown command invoked
```

Рисунок 3.17 – Сообщения в консоли Syntacore IDE при успешном запуске отладки

Причины невозможности запуска отладочной сессии

Вероятными причинами невозможности запуска отладочной сессии могут быть:

1) Отсутствие напряжения внутренних LDO (для К1921ВГ015 на выводах 17 VCORE_CAP и 15 VBAT_CAP, напряжение должно соответствовать 0,9–1,2 В).

Причинами могут быть:

– Отсутствие необходимого питания микроконтроллера. (для К1921ВГ015 – отсутствие напряжения питания на выводе 12 VCC1_APC или 14 V_BAT).

– В микроконтроллер зашита программа с использованием режимов пониженного потребления, при котором ядро микроконтроллера находится в спящем режиме (только для К1921ВГ015). В таком случае необходим запуск микроконтроллера в сервисном режиме с полным стиранием flash-памяти.

2) В конфигурационном слове микроконтроллера (CFGWORD) запрещена работа интерфейса JTAG (для К1921ВГ015 см. Руководство пользователя, раздел 7). В данном случае также поможет запуск микроконтроллера в сервисном режиме с полным стиранием flash-памяти.

Сервисное стирание flash-памяти микроконтроллера

Для полного стирания всей flash-памяти микроконтроллера необходимо чтобы при сбросе микроконтроллера вывод SERVEN находился в состоянии логической единицы (подтянут к 3,3 В). Описание сервисного стирания представлено в подразделе «Сервисный сброс всей Flash-памяти» руководства пользователя.

Для удобства использования NIIET_RISCV SDK уже содержит готовые скрипты сервисного стирания под различные отладчики и микроконтроллеры в каталоге tools→openocd.

Например, при работе в ОС Windows для сервисного стирания микроконтроллера К1921ВГ015 в составе платы макетно-отладочной КФДЛ.441461.029 необходимо выполнить скрипт «k1921vg015_onboard_ftdi_service_mode_erase.bat».

Шаг 8 – Создание нового проекта

При создании нового проекта для работы с микроконтроллером К1921ВГ015 наиболее простым способом будет копирование и корректировка одного из примеров.

Для этого нужно взять один из примеров проектов, очистить его (нажатием правой кнопки мыши по названию проекта в окне Project Explorer и выбором во всплывающем окне кнопки Clean Project) и затем скопировать его (нажатием во всплывающем окне кнопки Copy, а затем Paste), и наконец выбрать название для нового проекта. Пример создания копии проекта изображен на рисунке 3.18.

Рисунок 3.18 – Создание копии проекта

В новом созданном проекте большинство настроек будут заимствованы из скопированного (родительского) проекта. Необходимо будет только перед запуском отладки создать отладочную конфигурацию для нового проекта, в строке “C/C++ Application” которой указать путь к **.elf**-файлу, созданному при сборке нового проекта (см. рисунок 3.14 из раздела «Шаг 7…»). Настройки из подраздела “Config options” добавлять не потребуется, поскольку они будут заимствованы из скопированного проекта. Чтобы во время отладки получить доступ к регистрам периферийных блоков микроконтроллера необходимо при конфигурации отладки указать путь к **.svd**-файлу (см. рисунок 3.19). Сам **.svd**-файл можно взять в каталоге: niiet_riscv_sdk\tools\svd.

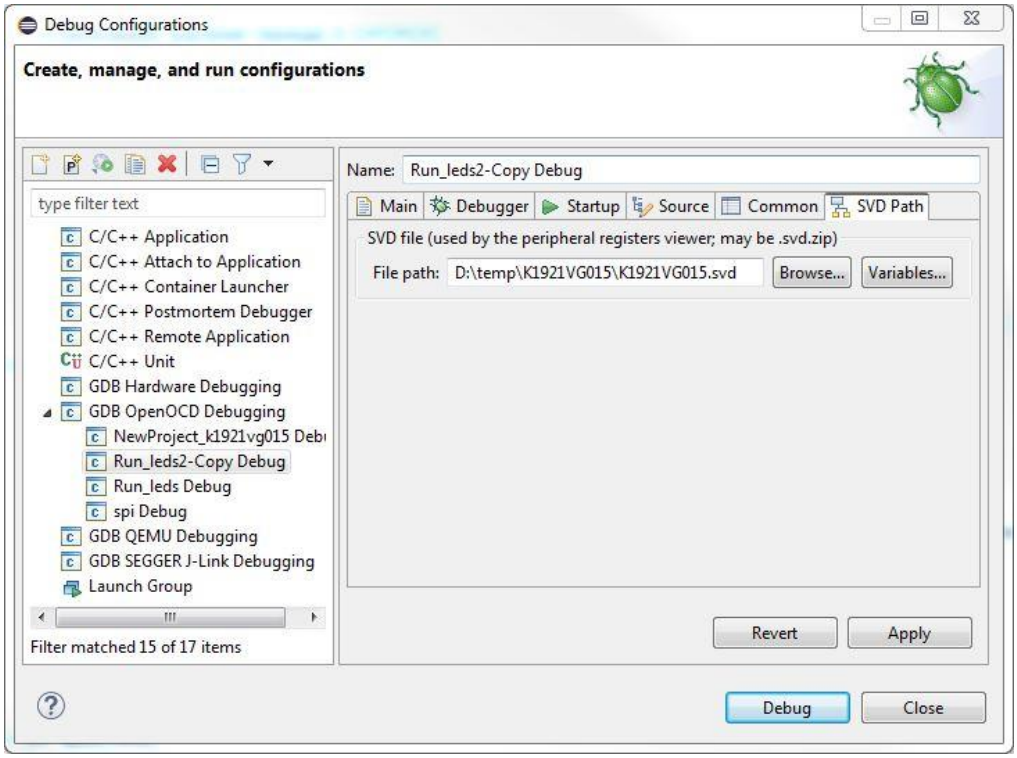

Рисунок 3.19 – Подключение SVD-файла

Создание нового проекта с нуля

Если требуется создать новый проект с нуля потребуется:

1. Создать новый проект, используя набор инструментов RISC-V Cross GCC (см. рисунок 3.20).

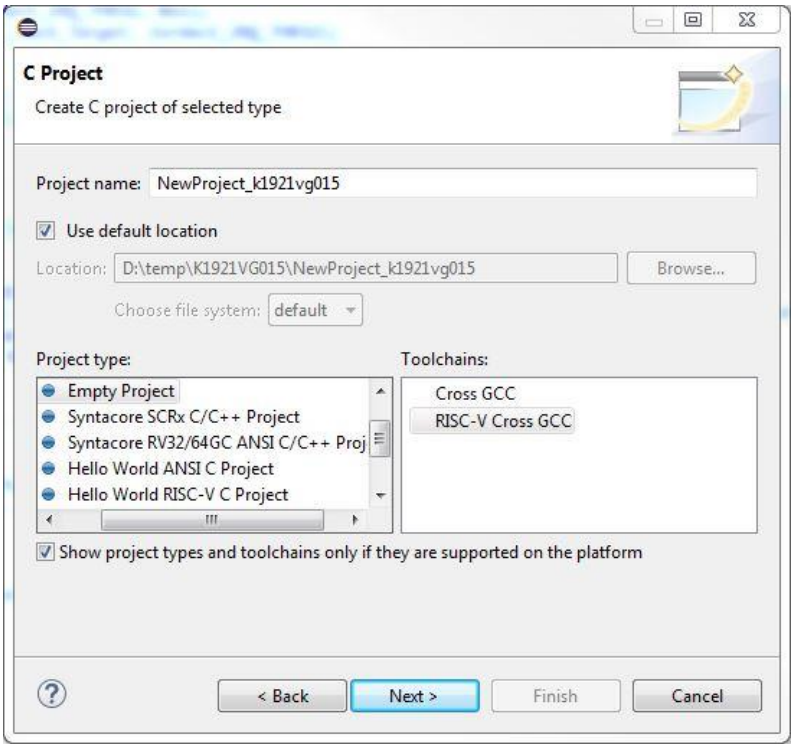

Рисунок 3.20 – Выбор Toolchains для нового проекта

2. Создать каталог под названием «platform» в каталоге с названием нового проекта в рабочей среде workspace (см. рисунок 3.21). Скопировать внутрь каталога «platform» каталоги «Include», «ldscripts», и «Source» (находятся в niiet_riscv_sdk\platform\Device\K1921VG015).

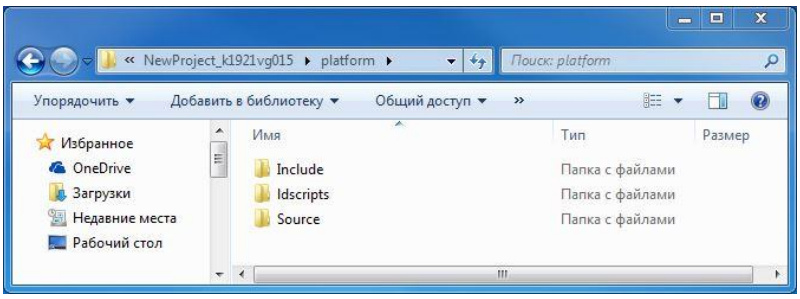

Рисунок 3.21 – Содержание каталога platform

3. Установить флаги –march и –mabi в настройках созданного проекта (см. рисунок 3.22).

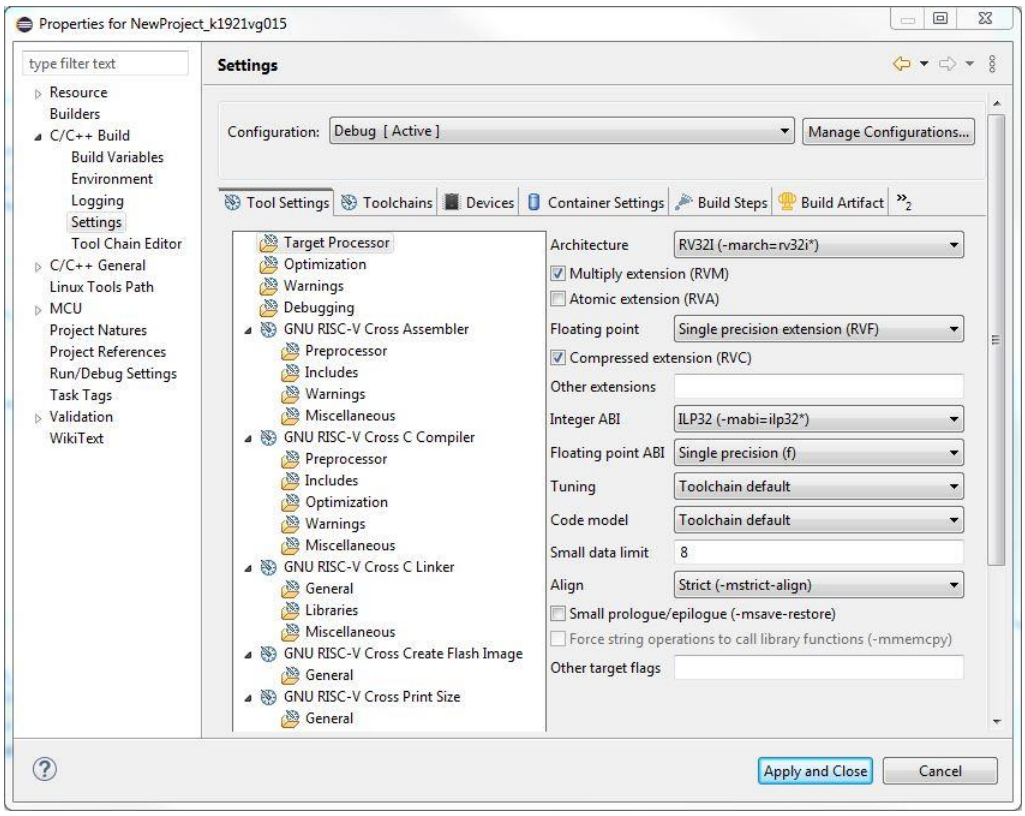

Рисунок 3.22 – Установка флагов–march и –mabi

4. Прописать пути к заголовочным файлам и скриптам в разделе GNU RISC-V Cross Assembler (см. рисунок 3.23) и в разделе GNU RISC-V Cross C Compiler (см. рисунок 3.24).

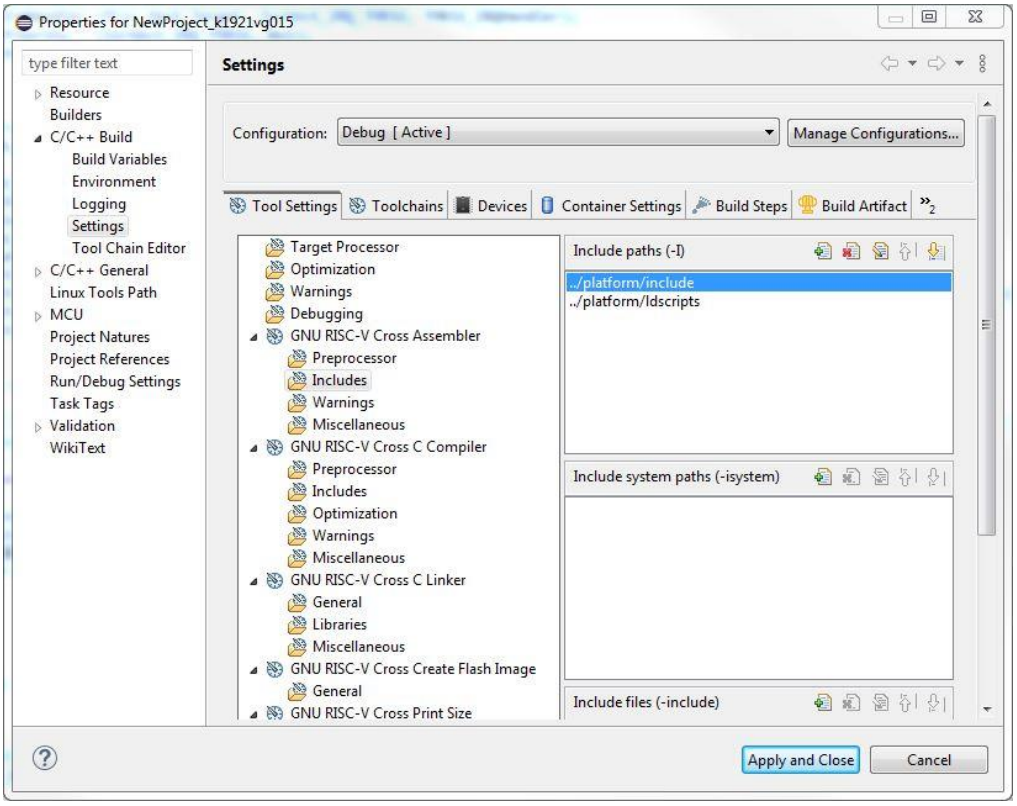

Рисунок 3.23 – Указание путей для транслятора ассемблера

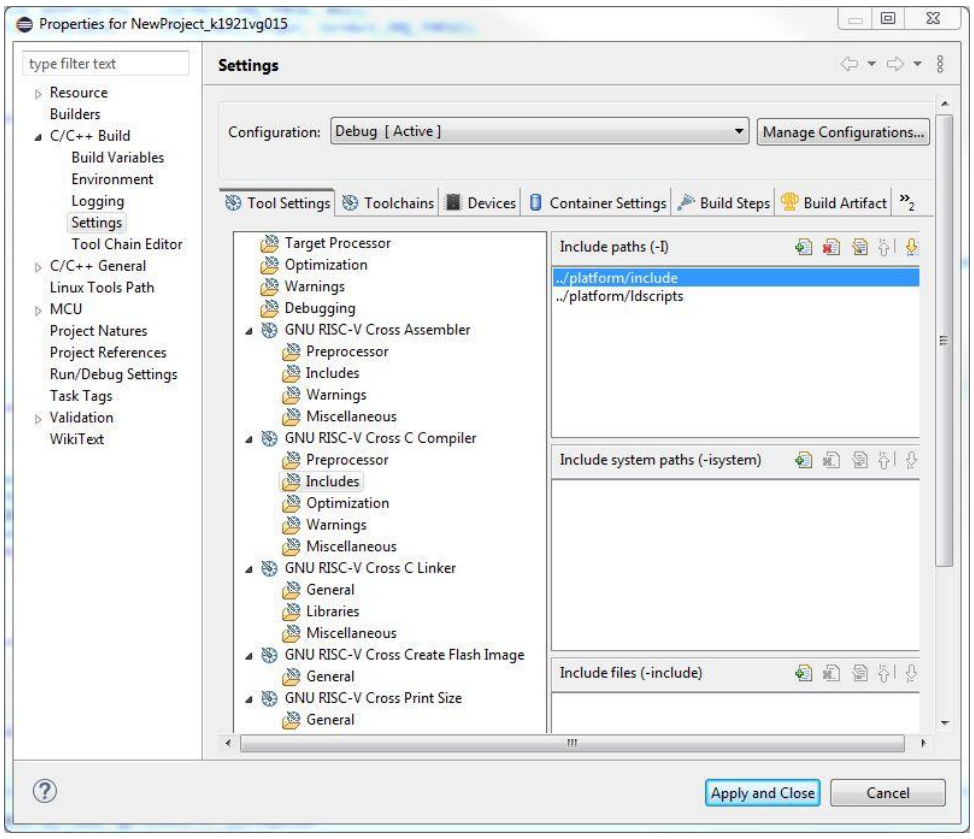

Рисунок 3.24 – Указание путей для компилятора Си

5. Указать директивы препроцессора (см. рисунок 3.25).

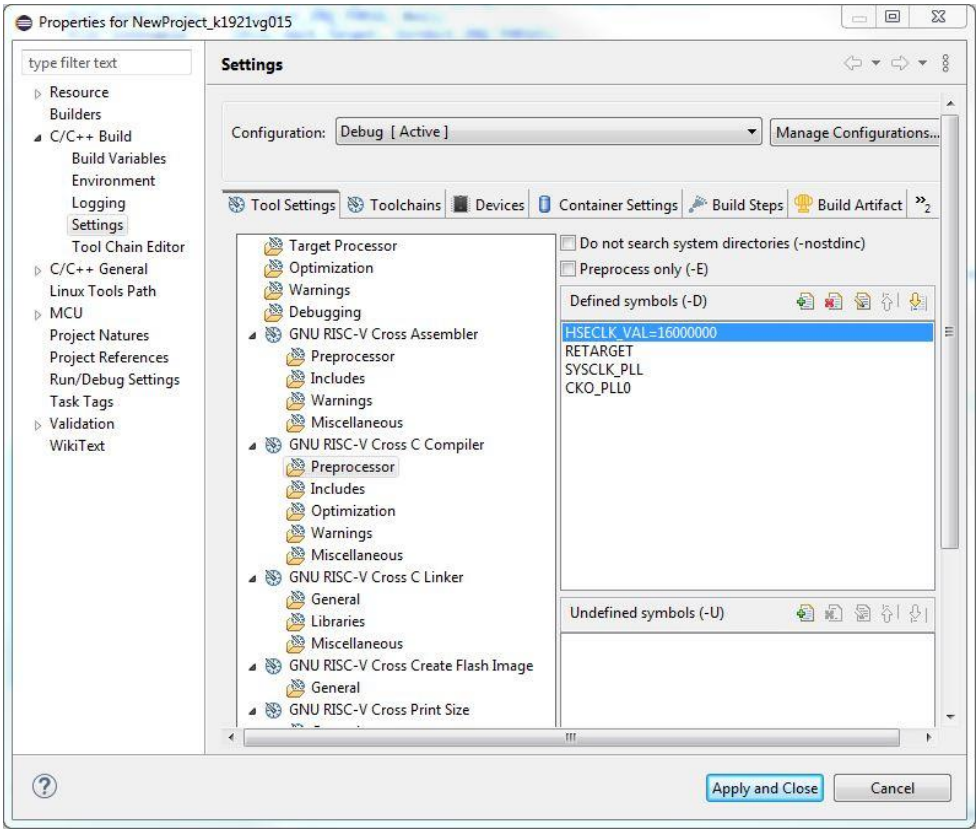

Рисунок 3.25 – Указание директив препроцессора

6. Указать путь к скрипту линкера, а также выбрать настройки линкера (см. рисунок 3.26).

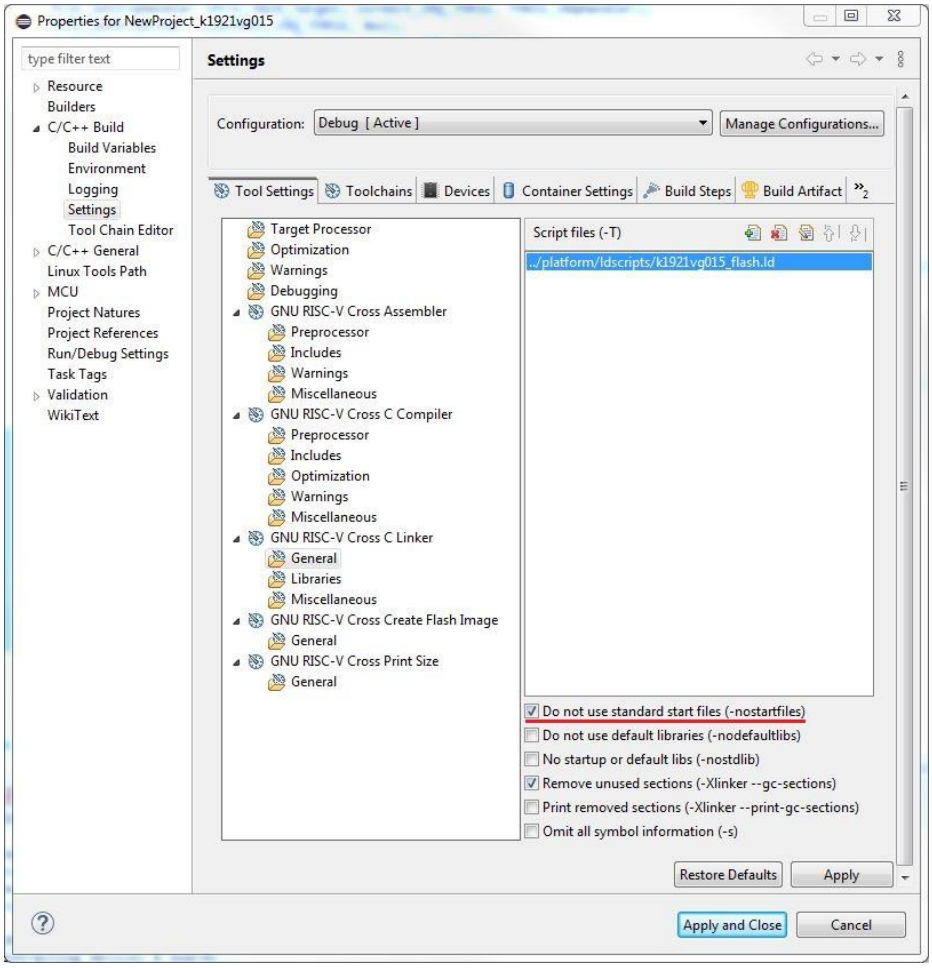

Рисунок 3.26 – Указание пути к скрипту линкера

7. Указать библиотеки линкера и путь к ним (см. рисунок 3.27).

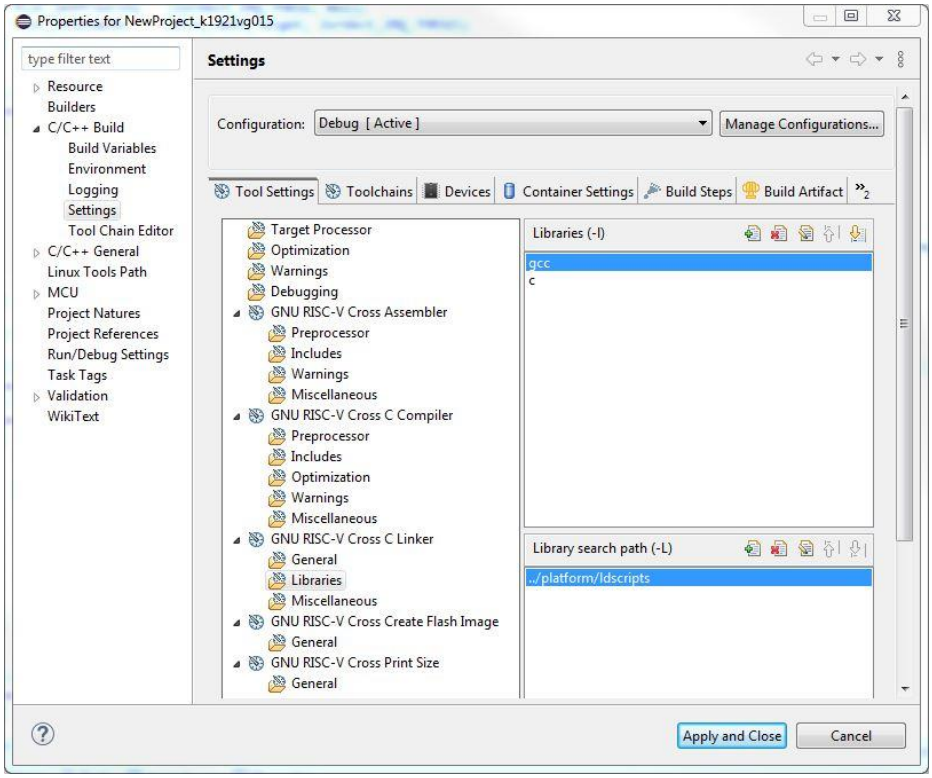

Рисунок 3.27 – Указание библиотек линкера и пути к ним

8. При необходимости формирования файла прошивки нужно разрешить его формирование в подразделе Toolchains (см. рисунок 3.28), а затем сконфигурировать настройки выходного файла прошивки (см. рисунок 3.29).

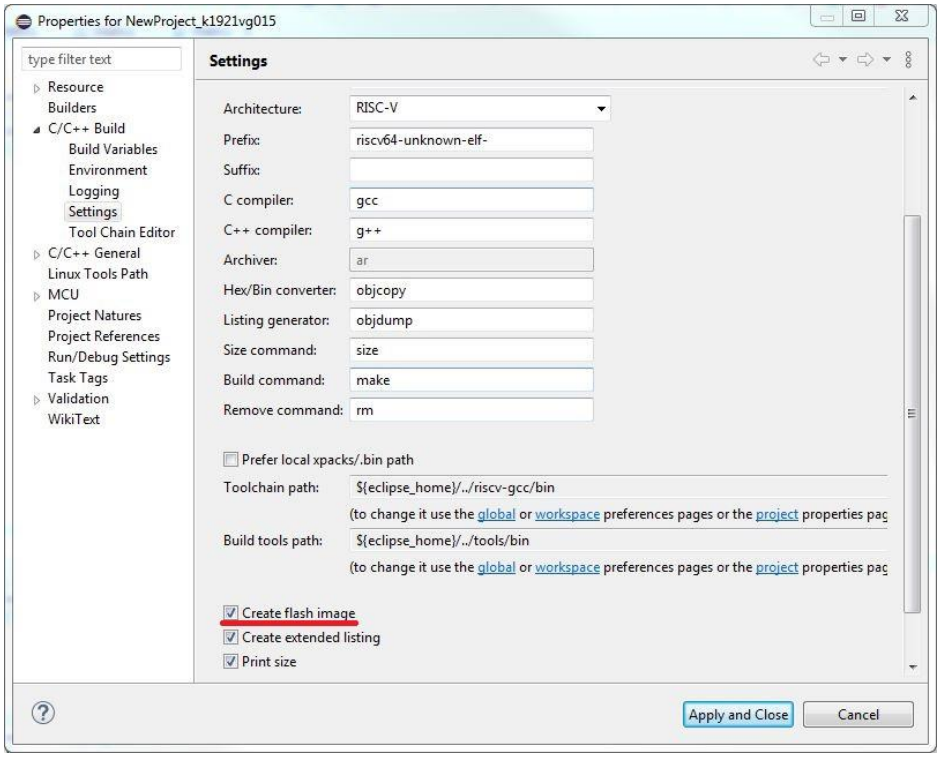

Рисунок 3.28 – Разрешение формирования выходного файла прошивки

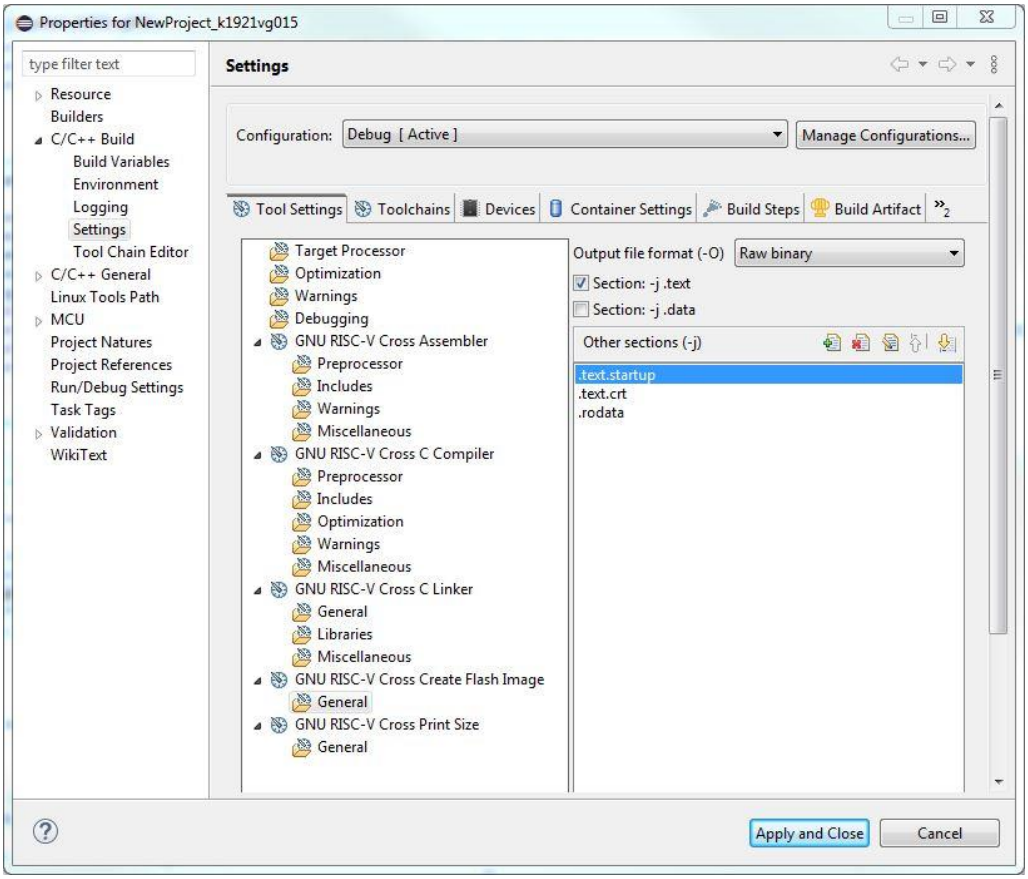

Рисунок 3.29 – Настройки формирования выходного файла прошивки

Теперь вновь созданный проект готов к сборке (см. подраздел «Шаг 5 – Сборка проекта») и запуску отладки (см. подраздел «Шаг 7 – Настройка и запуск отладочной сессии»).

3.2 Настройка среды Syntacore IDE и проекта для работы с компилятором, содержащим программный обход п.5 Errata для К1921ВГ015

Загрузка компилятора, содержащего программный обход (добавление команды «NOP») для К1921ВГ015.

Переходим на страницу сайта https://tools.cloudbear.ru и загружаем пакеты GCC версии 14.1 «riscv_gnu_toolchain_elf», содержащие программный обход (добавление команды «NOP»), для ОС Linux или ОС Windows.

Из загруженного архива распаковываем каталог «riscv-gnu-toolchain-elf» (Рисунок 3.30) в каталог с установленной sc-dt, рядом с каталогом «riscv-gcc» (Рисунок 3.31).

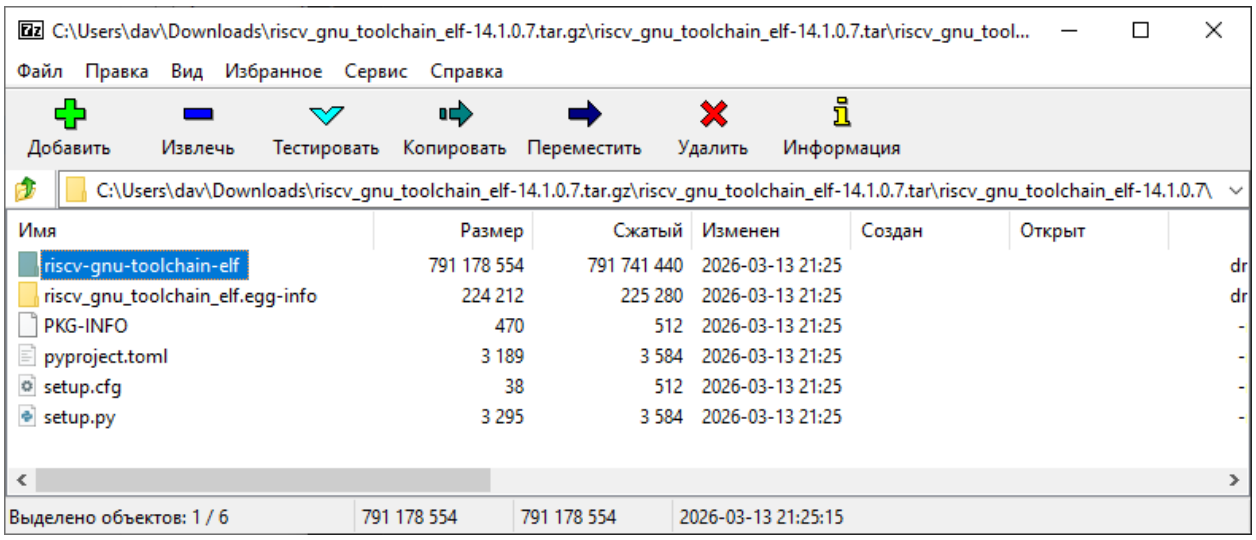

Рисунок 3.30 – Содержимое архива «riscv_gnu_toolchain_elf-14.1.0.7.tar.gz»

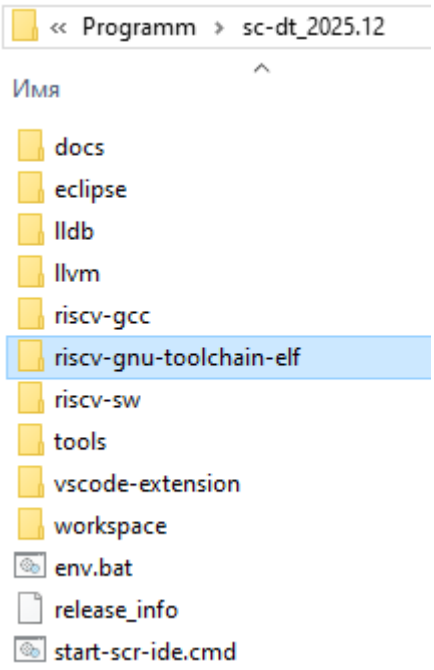

Рисунок 3.31 – Копирование каталога «riscv-gnu-toolchain-elf» в каталог с установленной sc-dt

Настройки проекта для использования компилятора, содержащего программный обход п.5 Errata для К1921ВГ015.

После открытия рабочей области (workspace) с проектами, или после импорта примеров, переходим к настройке проекта. Для этого выбираем проект из списка проектов (Project Explorer) и нажимаем «Alt + Enter» или выбираем пункт «Properties» из контекстного меню, вызываемого нажатием правой клавиши мыши по названию проекта.

Далее в окне настроек выбираем «C/C++ Build -> Settings», после этого в правой части окна выбираем вкладку «Toolchains» (Рисунок 3.32). В этом окне необходимо изменить путь к компилятору «Toolchain path».

По умолчанию используется путь из глобальных настроек (global), если в настройках рабочей области (workspace) или настройках проекта (project) указана пустая строка.

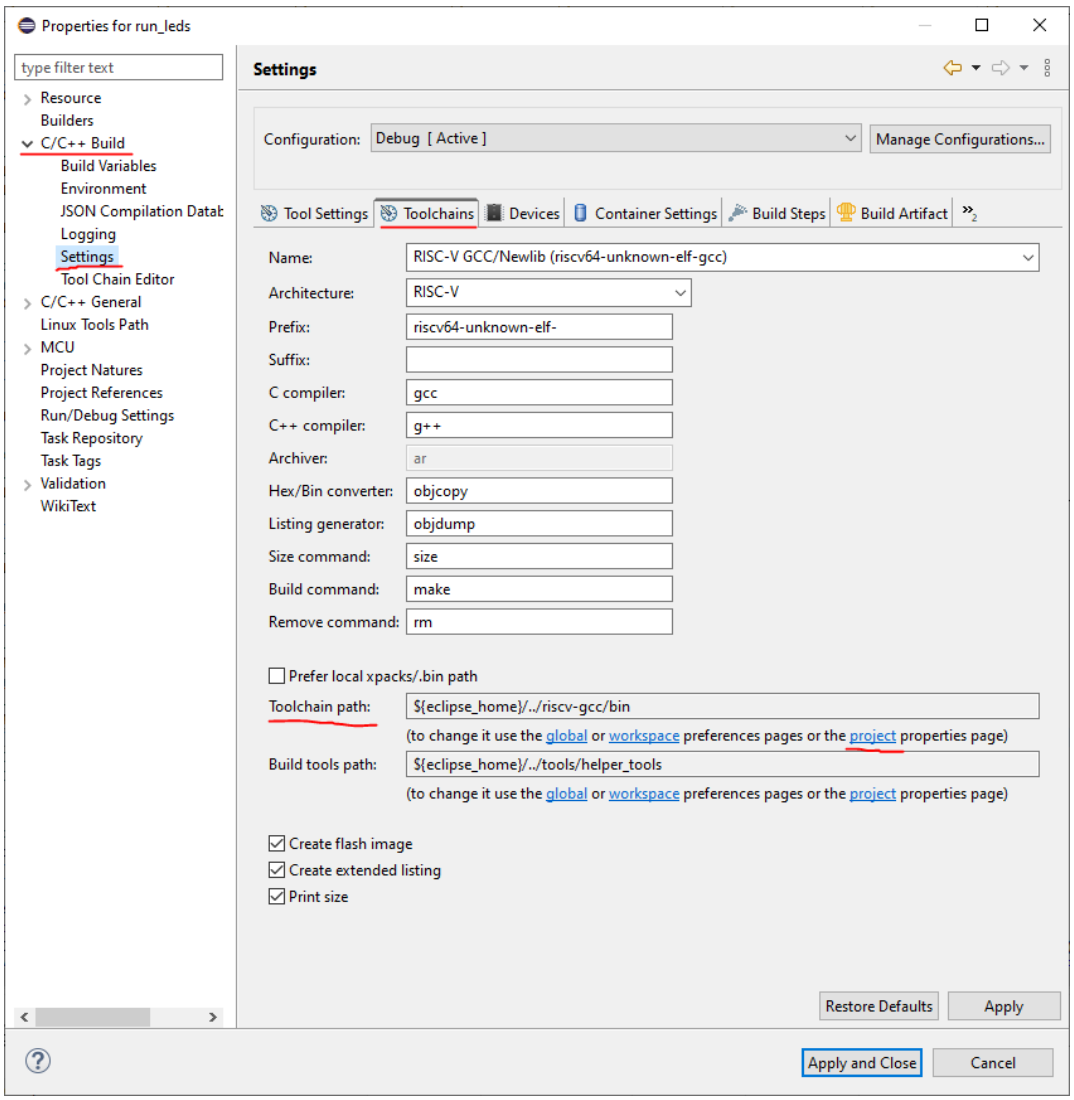

Рисунок 3.32 – Настройки проекта «Toolchains»

Для изменения «Toolchain path» в настройках проекта нажимаем ссылку «project» и в появившемся окне указываем путь к скачанной компилятору *«${eclipse_home}/../riscv-gnu-toolchain-elf/bin»* (Рисунок 3.33) и нажимаем «Apply and Close». После этого в настройках проекта будет указан путь к Toolchain, который мы указали (Рисунок 3.34).

В настройках проекта (Target Processor) при использовании Toolchain от CloudBear необходимо указать параметр "Other extensions" : "zicsr_zifencei_zba_zbb_zbc_zbs" (Рисунок 3.35)

Далее для активации обхода ошибки п.5 Errata необходимо в настройках компилятора добавить ключ «-mfix-cloudbear-0001». Для этого в настройках проекта выбираем вкладку «Tool Settings», пункт настроек «Miskcellaneous» компилятора «GNU RISC-V Cross C Compiler» и в текстовое поле «Other compiler flags» вводим ключ «-mfix-cloudbear-0001» (Рисунок 3.36).

В результате, при сборке данного проекта на языке Си, перед командами div*, rem*, clmul*, fdiv*, fsqrt* будет добавлена ассемблерная команда «nop».

Так как мы изменили настройки «Toolchain path» только для данного проекта, при сборке остальных проектов будет использоваться Toolchain по умолчанию из глобальных настроек (global). Если необходимо изменить глобальные настройки Toolchain или настройки для рабочей области (workspace), то следует изменить путь «Toolchain path» в настройках global или workspace, соответственно.

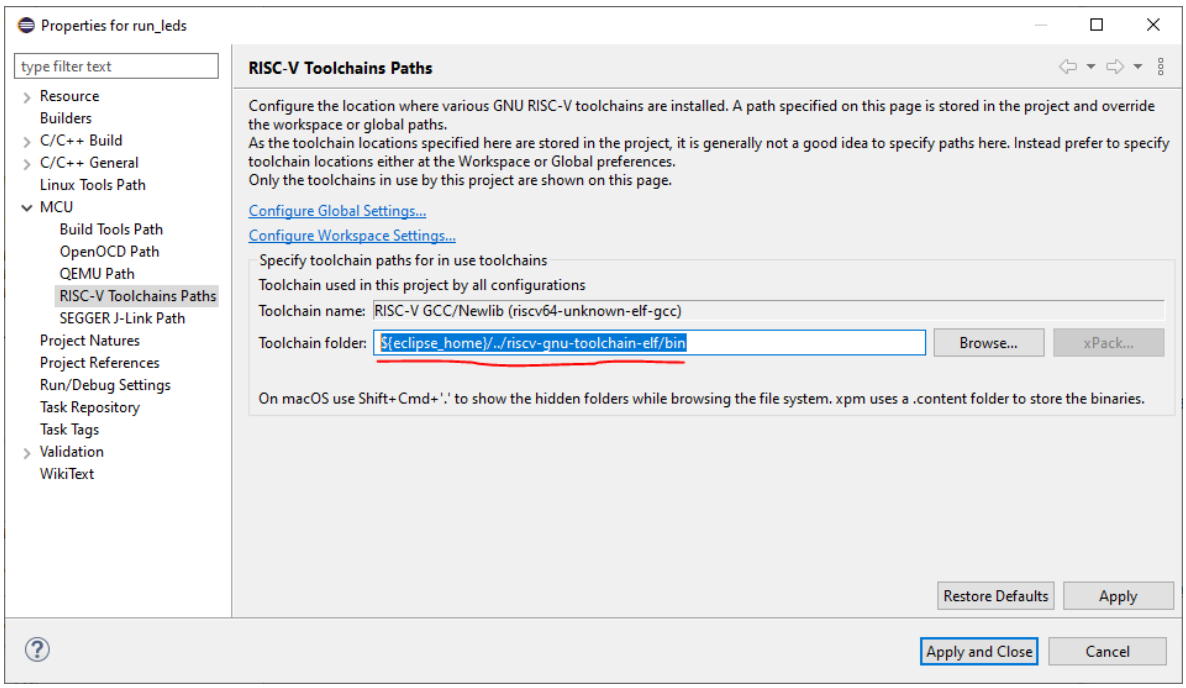

Рисунок 3.33 – Настройки «Toolchains» проекта

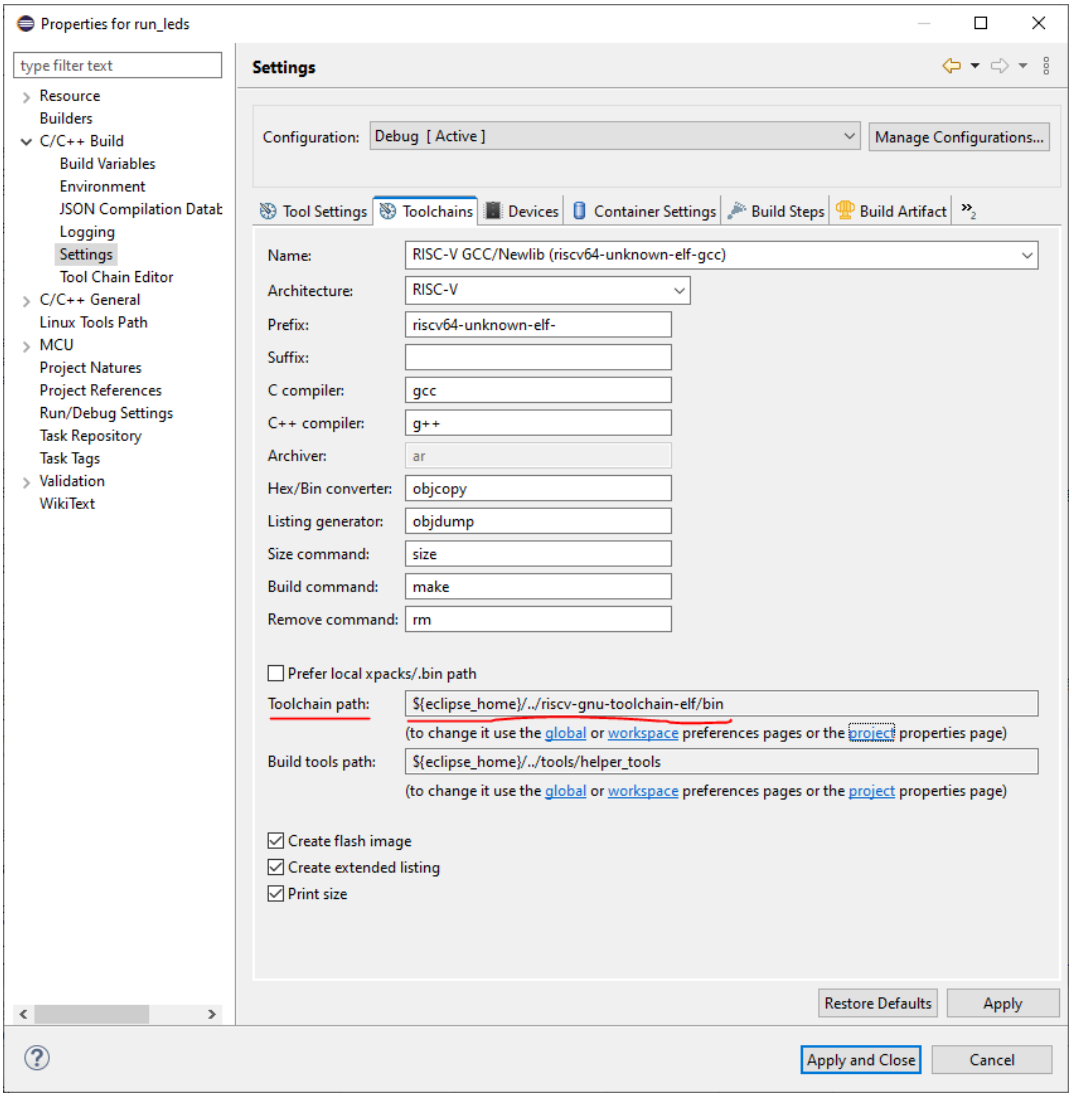

Рисунок 3.34 – Настройки «Toolchains» проекта с измененным каталогом компилятора

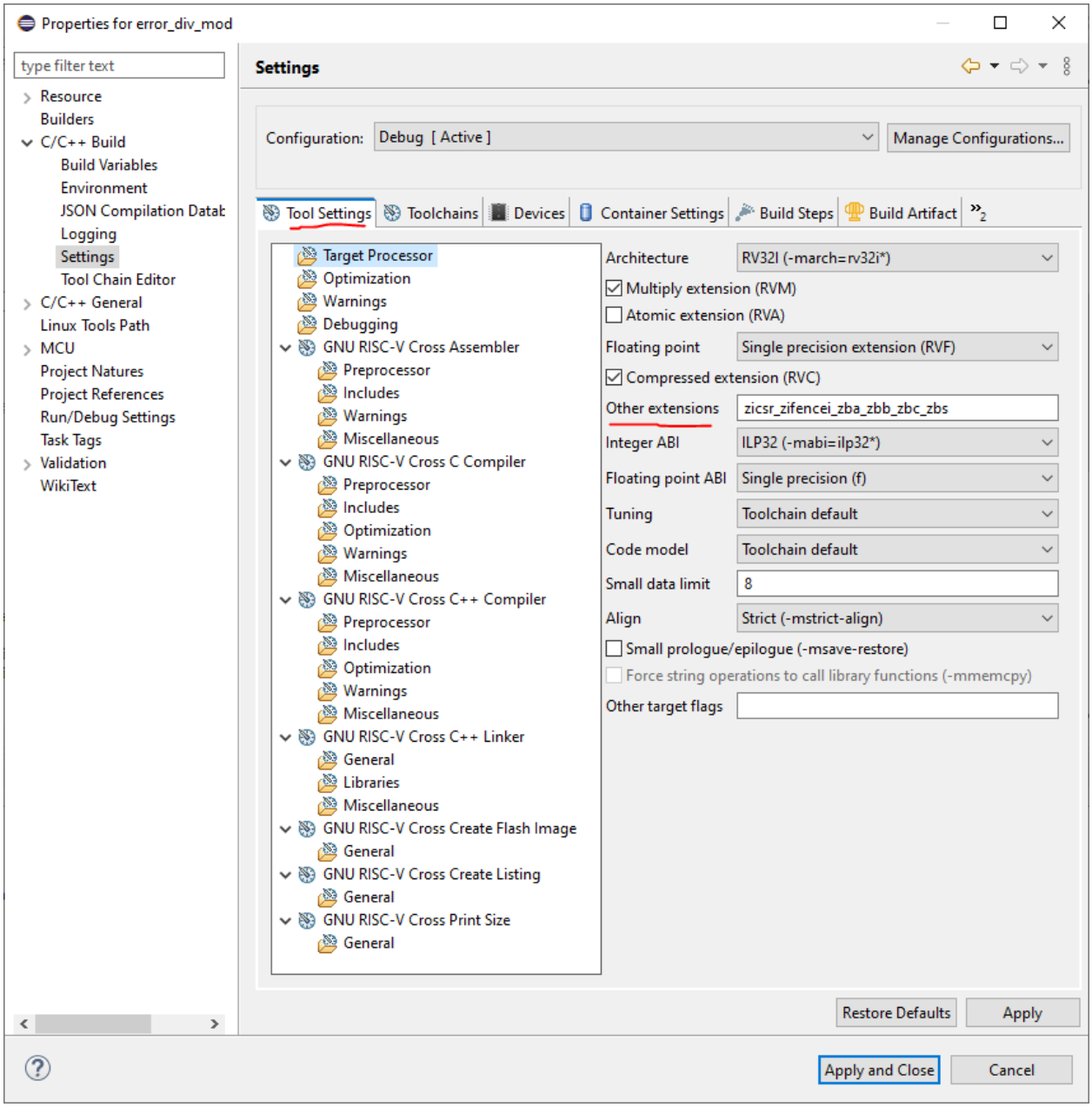

Рисунок 3.35 – Настройки расширения ядра при использовании Toolchain от CloudBear

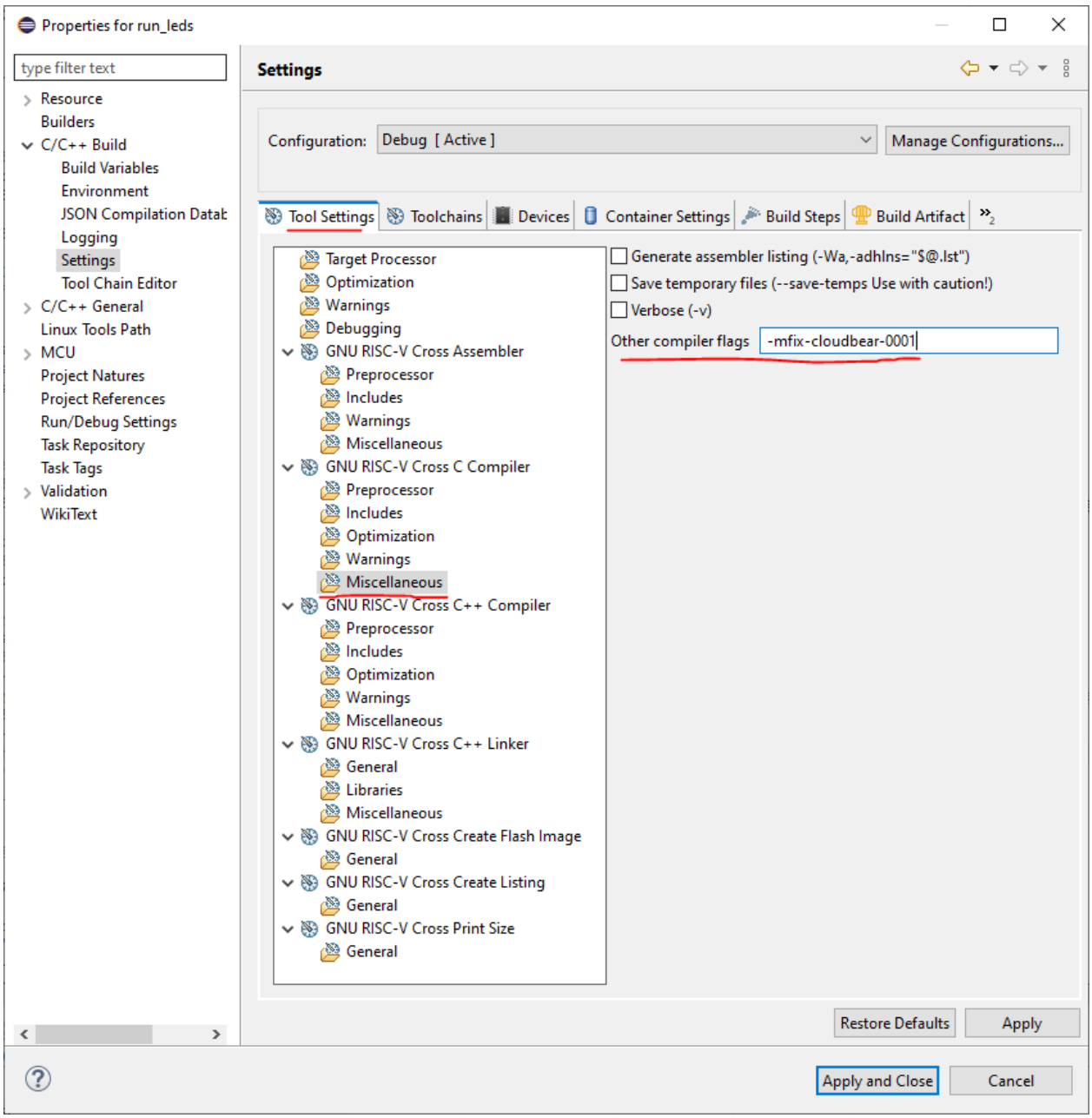

Рисунок 3.36 – Настройки флагов компилятора

4 Работа с примерами проектов

4.1 Структура примеров проектов

Репозиторий NIIET_RISCV SDK (также как и репозитории k1921vg015_sdk, k1921vg5t_sdk, k1921vg7t_sdk) содержит примеры проектов в каталоге **projects**, а также шаблоны для проектов в каталоге **templates**.

Примеры проектов в каталоге **projects** существуют двух типов:
– Проекты с управлением через регистры (например NIIET-DEV-K1921VG015);
– Проекты с использованием библиотек plib (например plib015).

Отличия состоят в том, что каждый из примеров проектов с управлением через регистры содержит весь набор необходимых файлов внутри проекта (каталог platform внутри каждого проекта) и могут быть скопированы в любую рабочую область и запущены локально. Все примеры ссылаются на общий для всех проектов каталог platform расположенный в niiet_riscv_sdk (содержащий все необходимые файлы и библиотеки), поэтому примеры могут быть запущены только из подкаталога SDK. Для получения независимой от путей SDK копии примера для К1921ВГ015 необходимо воспользоваться шаблоном проекта k1921vg015-bare из каталога templates с заменой файла main.c.

Шаблоны проектов в каталоге **templates** также существуют двух типов:
– Проекты с управлением через регистры (например k1921vg015-bare);
– Проекты с использованием библиотек (например k1921vg015-plib015).

Каждый из шаблонов проектов содержит внутри себя все необходимые файлы, а также библиотеки.

Изначально в каталогах sc-dt всех проектов отсутствуют файлы запуска сессии отладки *.launch. Для их генерации в каталоге SDK «tools» расположены скрипты prepare_launch_*.bat. Каждый скрипт в имени файла содержит название отладчика и предназначен для генерации файлов запуска сессии отладки *.launch с настройками соответствующего отладчика. При использовании отладчика для К1921ВГ015 на плате КФДЛ.441461.029 необходимо выполнить скрипт prepare_launch_Onboard_FTDI.bat, при использовании JLink - выполнить скрипт prepare_launch_JLink.bat.

4.2 Использование глобальных определений в примерах проектов

Для удобства работы с примерами можно пользоваться глобальными определениями в настройках каждого проекта (см. рисунок 3.26 «Указание директив препроцессора»). Глобальные определения учитываются в файле system_k1921vg015.c (для К1921VG015) при сборке проекта.

С помощью глобальных определений можно:
– Указать частоту кварцевого резонатора, установленного на плате;
– Разрешить перенаправление функции printf в интерфейс UART (RETARGET);
– Выбрать источник системной частоты SYSCLK;
– Выбрать источник частоты для выходного тактового сигнала CLKOUT.

Типовые частоты для кварцевого резонатора, которые можно задать с помощью глобальных определений (необходимы для подбора коэффициентов делителей в PLL):
– 10 МГц;
– 12 МГц;
– 16 МГц;
– 20 МГц;
– 24 МГц.

Источники системной частоты на примере К1921ВГ015:

– SYSCLK_HSI – внутренний высокочастотный тактовый сигнал частотой 1 МГц (тактовый сигнал внутреннего RC-генератора);

– SYSCLK_HSE – внешний тактовый сигнал с выводов XI_OSC и XO_OSC;

– SYSCLK_PLL – тактовый сигнал с выхода 0 PLL;

– SYSCLK_LSI – внутренний низкочастотный тактовый сигнал частотой 32,768 кГц с блока RTC.

Источники частоты для выходного тактового сигнала CLKOUT на примере К1921ВГ015:

– CKO_HSI – внутренний высокочастотный тактовый сигнал частотой 1 МГц (тактовый сигнал внутреннего RC-генератора);

– CKO_HSE – внешний тактовый сигнал с выводов XI_OSC и XO_OSC;

– CKO_PLL0 – тактовый сигнал с выхода 0 PLL;

– CKO_LSI – внутренний низкочастотный тактовый сигнал частотой 32,768 кГц с блока RTC;

– CKO_NONE – запрет выходного тактового сигнала CLKOUT.

4.3 Выбор скрипта линкера

Имеется возможность использования различных скриптов линкера для сборки проекта под ОЗУ (для K1921VG015 – k1921vg015_ram.ld), ПЗУ (для K1921VG015 – k1921vg015_flash.ld) или TCM (для K1921VG3T – k1921vg3t_tcm.ld) при помощи указания файла скрипта в настройках проекта (См. Рисунок 3.25 – «Указание пути к скрипту линкера»). Данные скрипты находятся в каталоге platform→ldscripts.

5 Сборка проекта в makefile

Для сборки проекта можно воспользоваться makefile, расположенном в каталоге platform→Device→…→gcc. Имя проекта задается в строке 6 TARGET_NAME = …, а путь к компилятору в строке 8 COMPILER_PATH =…

6 Использование расширения для Visual Studio Code

6.1 Установка

Для установки плагина вам потребуется скачать архив niiet-aspect-x.x.x.vsix из релиза.

После скачивания зайдите в VSCode, перейдите в раздел `Extensions` в левом меню или при помощи комбинации клавиш `Ctrl+Shift+X`, выбрать меню `Views and More Actions`, где нужно выбрать пункт `Install from VSIX...`. Эта последовательность действий показана на рисунке:

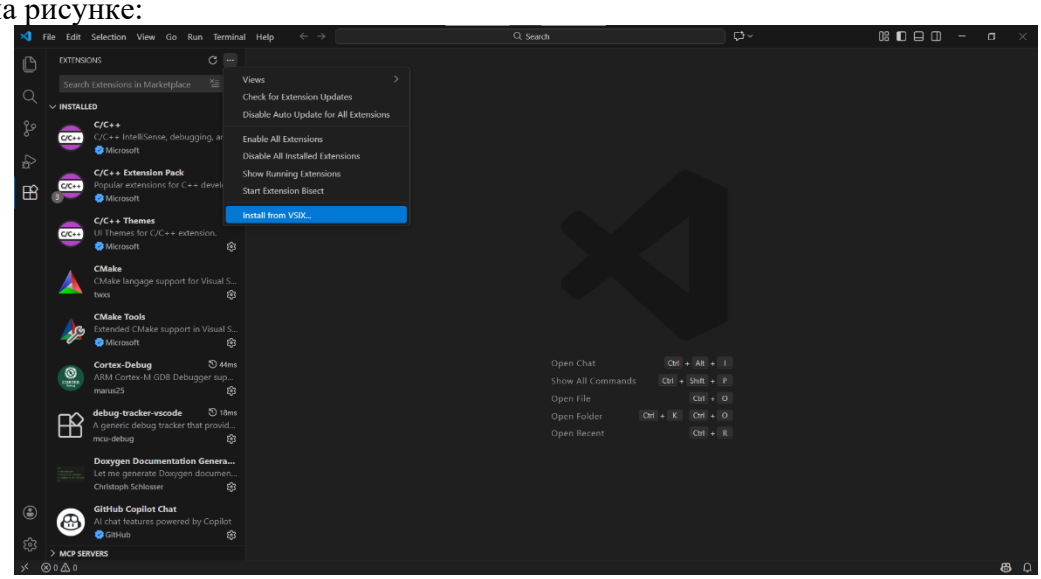

Рисунок 6.1 – Установка расширения из архива .vsix

После этого вы попадете в диалоговое окно с выбором расширения, найдите скачанное расширение и выберете его, затем оно установится в VSCode.

Примечание: в расширении niiet-aspect предусмотрено использование стандартного расширения для языков C/C++ ms-vscode.cpptools, при соединении с интернетом оно установится автоматически. Если этого не произошло установите его отдельно.

6.2 Использование

После установки расширения niiet-aspect вы увидите, что в левом меню появился новый значок с логотипом АО НИИЭТ. При нажатии на него откроется меню расширения.

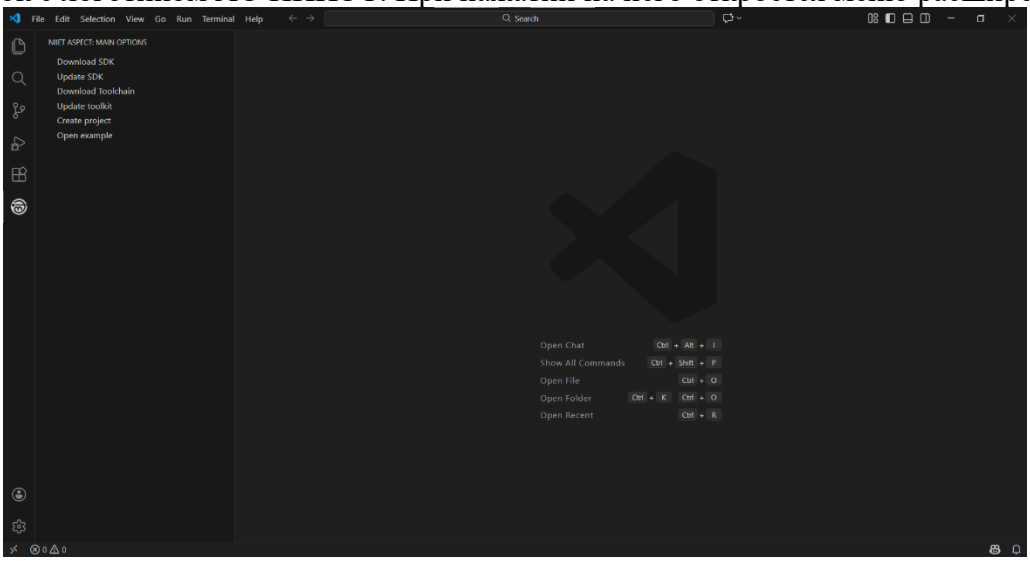

Рисунок 6.2 – Основное меню расширения

Из этого меню можете установить по нажатию кнопок SDK для микроконтроллеров НИИЭТ, компилятор и инструменты для отладки, если они требуются.

Для установки компилятора или инструментов отладки из этого меню необходимо нажать Download toolchain и выбрать то, что вам требуется.

Если нет возможности установить их с этого устройства, то можно перенести эти архивы из репозиториев, предварительно разархивировав, и в настройках указать пути до инструментов. Для того чтобы попасть в эти настройки нужно перейти в «Manage>Settings» или воспользоваться сочетанием клавиш «Ctrl+», перед вами будет меню настроек всех расширений, напишите в поиске niiet-aspect и увидите настройки этого расширения, здесь нужно будет установить пути до установленных вручную расширений.

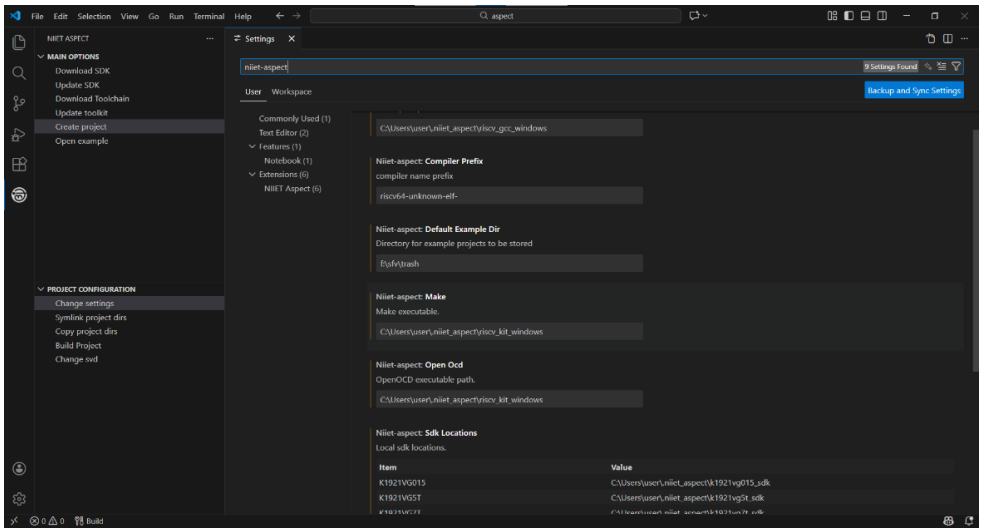

Рисунок 6.3 – Настройка путей расширения

Примечание: пока вы не установите SDK вы не сможете ни создать проект, ни открыть пример!

6.3 Импорт примеров НИИЭТ

После установки SDK вы можете открыть один из разработанных для микроконтроллера примеров, для этого в меню расширения нажмите «Open Example». При нажатии на эту кнопку откроется меню с выбором проекта, выберите то, что вас интересует. Обратите внимание, что пример открывается прямо в репозитории установленного SDK.

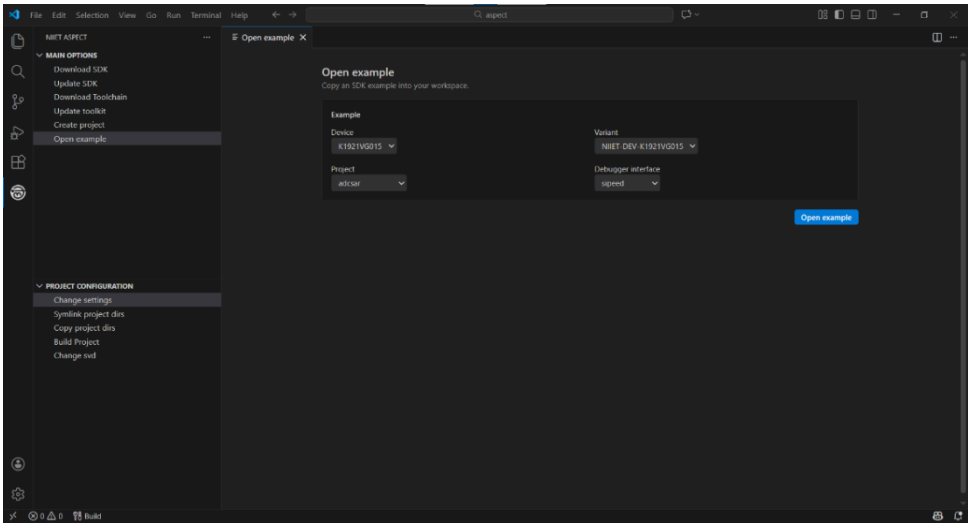

Рисунок 6.4 – Открытие примера

6.4 Создание проекта

При создании проекта вы полностью сами конфигурируете новый проект, вы можете использовать «plib» в проекте, указать собственные зависимости и библиотеки, выбрать готовый скрипт линковщика или установить свой сконфигурировать «march» и «mabi» для компилятора, установить системные для контроллера определения или добавить свои, выбрать уровень оптимизации при компиляции, указать специальные флаги для компилятора самостоятельно.

6.5 Работа с проектом

При создании проекта будет создана директория с проектом и файлами внутри, помимо непосредственно исходных файлов вы можете найти «makefile», «niiet_aspect.json» и «.vscode/launch.json», «.vscode/launch.json», а также импортированные директории, при создании проекта они будут прикреплены к нему как ссылки на те же директории в SDK, или на папки которые вы указали с флагом «-I». «makefile» используется для сборки проекта. «niiet_aspect.json» хранит конфигурацию проекта, «.vscode/launch.json» содержит конфигурацию для отладки проекта. Эти файлы не рекомендуется изменять без должной необходимости.

После того как вы окажитесь в проекте, в меню НИИЭТ станут доступны новые действия.

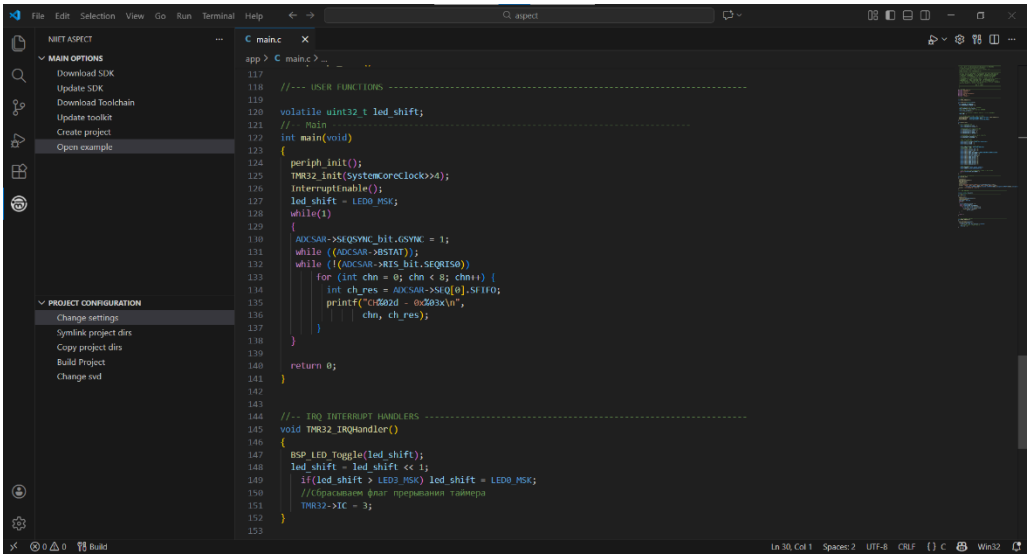

Рисунок 6.5 – Работа внутри проекта

Если у вас верно сконфигурирован компилятор вы можете скомпилировать проект по нажатию на кнопку в этом меню, по кнопке снизу подписанной как «Compile», или если вы находитесь в файле с языком Си, то по кнопке справа сверху, которая имеет иконке отвертки и ключа.

Как уже было упомянуто при создании все папки в проекте прикреплены как ссылки, вы можете это изменить нажатием на кнопку «Copy project dirs» в меню, тогда все ссылки будут убраны и заменены на полные копии указанных директорий. Сделать директории ссылками обратно можно с помощью кнопки «Symlink project dirs».

6.6 Изменение настроек

Если вы хотите посмотреть текущие настройки проекта или изменить их то перейдите в меню «Change settings», в нем представлено несколько отдельных окон, в которых можно изменить настройки проекта, так например в окне «Core» можно выставить настройки расширений ядра для компилятора.

6.7 Отладка

Для начала отладки первым действием будет сборка проекта. После, чтобы начать отладку нужно нажать кнопку «F5» в исходном коде проекта, и выбрать конфигурацию отладки с названием: «NIIET {название_проекта}» после чего будет запущен сервер OpenOCD и IDE перейдет в режим отладки, если GDB сможет подключиться к контроллеру.

Вы вольны использовать иные расширения для отладки, например mcu-debug.memory-view для просмотра памяти или mcu-debug.peripheral-viewer, для использования последнего в конфигурации есть поле «svdPath», что позволит просматривать регистры периферии, поменять его можно в меню проекта при выборе пункта «Change svd».

Примечание: иногда сервер OpenOCD может работать нестабильно, в этом случае рекомендуется перезагружать котроллер и перезапускать сервер OpenOCD и начинать отладку заново.

6.8 Пользовательская настройка расширения

Если вам требуется использовать свой компилятор или вы уже имеете свою сборку OpenOCD или make, то вы можете использовать свои инструменты при разработке. При помощи данного руководства вы также можете сконфигурировать работу расширения с другими ОС.

Настройка SDK: SDK потребуется скачать из репозиториев по контроллерам: [К1921ВГ015, К1921ВГ1Т, К1921ВГ3Т, К1921ВГ5Т, К1921ВГ7Т] и установить путь до него в опциях VS Code niiet-aspect.sdkLocations.

Использование своего компилятора GCC: для этого просто укажите в настройках VS Code опцию niiet-aspect.compiler путь до папки с компилятором, до места где находится папка bin. Так же в опцию niiet-aspect.compilerPrefix укажите префикс компилятора.

Использование своего make: Для использование своего make, или при использовании с другой ОС нужно установить путь до make, таким образом, чтобы он лежал в папке bin, относительно той директории, которую вы укажете в опции пути niiet-aspect.make.

Использование своего OpenOCD: Действия схожи с остальными вам потребуется добавить путь до папки, где находится директория bin, в которой будет находиться OpenOCD, но помимо директории bin в этой директории должен находится файл config.json, как в корне репозитория.

В директориях interface и target должны находится соответсвующие файлы конфигураций. В target конфигурации микроконтроллеров должны называться соответствующе, а сами интерфейсы вы можете называть как пожелаете.

Приложение А
(обязательное)
Перечень изменений

Номер и дата изменения	Описание изменения	Кол-во страниц
№1 17.04.2026	- добавлено приложение А для отражения списка изменений. - сокращено описание микроконтроллеров в разделе 1 - дополнен список отладчиков в п.п. 2.2 - добавлен п.п. 3.2 «Настройка среды Syntacore IDE и проекта для работы с компилятором, содержащим программный обход п.5 Errata для К1921ВГ015»	38