

NuEclipse User Manual.

1 Introduction.

The NuEclipse is designed for cross-platform embedded ARM development. It includes a series of Eclipse plug-ins and tools. The plug-ins allow the user to create, build, and debug ARM-based projects within the Eclipse framework. Its features are listed below:

- Creating projects by the New Project Wizard: The New Project Wizard provides several templates for different target chips.
- Building projects by the GNU ARM Toolchain: The toolchain contains the ARM Embedded GCC compiler. The user can use it to build projects without restriction.
- Debugging projects by GDB: The user can halt, step, run, and monitor target chips. Accessing memory and flash is allowed. Setting hardware breakpoints and watchpoints is supported. In addition, the user can erase target chips and program the user configuration. Through the NuEclipse, the user can develop projects of the NuMicro® Family within the Eclipse framework.

1. Введение. NuEclipse предназначен для кроссплатформенной разработки встроенных ARM. Он включает серию плагинов и инструментов Eclipse. Плагины позволяют пользователю создавать, строить и отлаживать на базе ARM проекты в рамках Eclipse. Его особенности перечислены ниже:

- Создание проектов с помощью мастера нового проекта: мастер создания проекта предоставляет несколько шаблоны для разных целевых фишек.
- Сборка проектов с помощью GNU ARM Toolchain: Toolchain содержит ARM Embedded Компилятор GCC. Пользователь может использовать его для создания проектов без ограничений по объёму памяти МК.
- Отладка проектов с помощью GDB: пользователь может останавливать, переходить, запускать и контролировать целевые микросхемы. Доступ память и флэш разрешены. Поддерживается установка аппаратных точек останова и наблюдения. С помощью NuEclipse пользователь может разрабатывать проекты семейства NuMicro® в Eclipse.

2 Системные требования и руководство по установке.

2.1 Системные требования.

В следующей таблице перечислены системные требования для пользователя, запускающего NuEclipse, их можно посмотреть в оригинальном документе если так интересно....

2.2 Поддерживаемые чипы.

Чтобы увидеть список поддерживаемых чипов, перейдите на сайт производителя.

2.3 Установка.

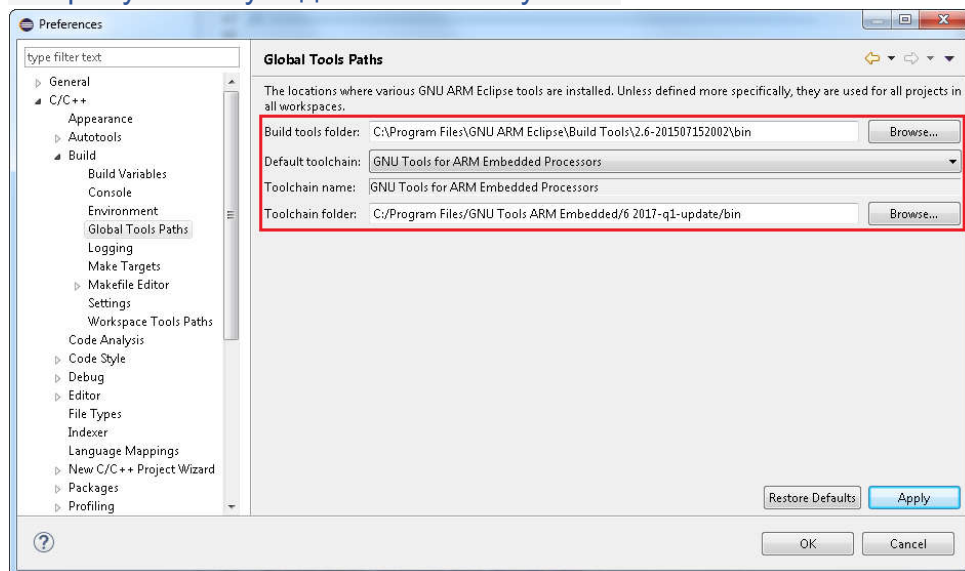
Далее идет описание процесса подготовки к установке NuEclipse и сама установка в ОС Microsoft Windows.

2.3.3 Verifying the Eclipse Preferences

After the installation, the Eclipse preferences are automatically written on Windows. To verify them, click **Window > Preferences**, the Preferences wizard appears. Go to **C/C++ > Build > Global Tools Paths** and make sure the Build tools and Toolchain folder be correctly configured to what the installer has installed in the previous step. Click the **Apply** button to take effect. On GNU/Linux, Build tools folder path is not required. The path should be empty.

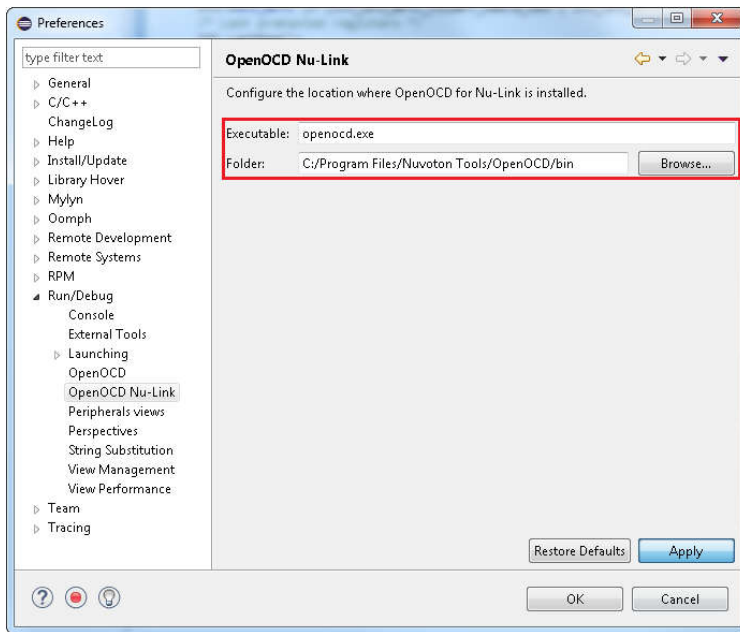
2.3.3 Проверка настроек Eclipse.

После установки настройки Eclipse автоматически записываются в Windows. Проверять их, щелкните «Окно»> «Настройки», откроется мастер настроек. Перейдите в C / C ++> Сборка> Global Tools Paths и убедитесь, что инструменты сборки и папка Toolchain правильно настроены. то, что установщик установил на предыдущем шаге. Нажмите кнопку «Применить», чтобы применить изменения. На GNU / Linux, путь к папке с инструментами сборки не требуется. Путь должен быть пустым.



Subsequently, go to **Run/Debug > OpenOCD Nu-Link** and make sure the OpenOCD folder be configured to where the installer has put the OpenOCD executable. On Microsoft Windows, for example, the path of OpenOCD folder could be C:/Program Files/Nuvoton Tools/OpenOCD. Similarly, on GNU/Linux it could be /usr/local/OpenOCD. The OpenOCD executable provided by Nuvoton is customized for Nu-Link. If the user tries to use other OpenOCD executable, OpenOCD and Nu-Link may not cooperate well. Click the **Apply** button to take effect.

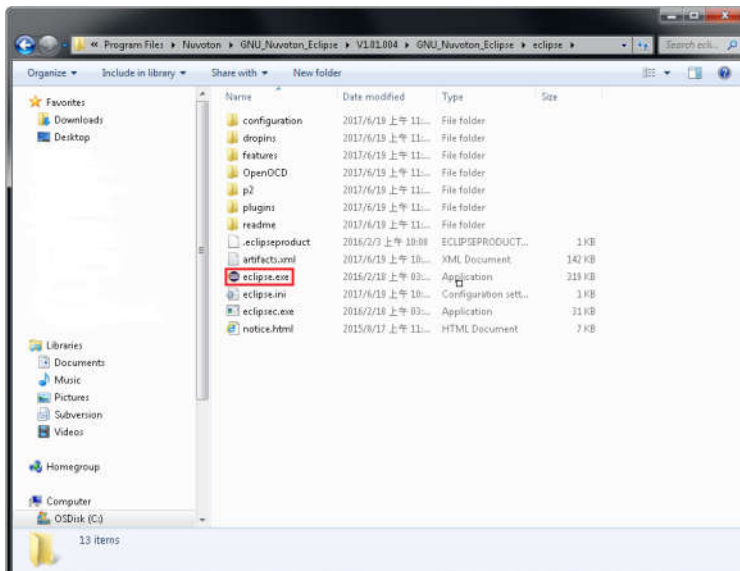
Затем перейдите в меню «Выполнить / Отладка»> «OpenOCD Nu-Link» и убедитесь, что папка OpenOCD настроена таким образом, чтобы установщик поместил исполняемый файл OpenOCD. В Microsoft Windows, например, путь к папке OpenOCD может быть C: / Program Files / Nuvoton Tools / OpenOCD. Точно так же в GNU / Linux это может быть / usr / local / OpenOCD. Исполняемый файл OpenOCD, предоставляемый Nuvoton, адаптирован для Nu-Link. Если пользователь пытается использовать другой исполняемый файл OpenOCD, OpenOCD и Nu-Link могут плохо взаимодействовать. Нажмите кнопку «Применить», чтобы применить изменения.



2.4 Running Eclipse

To run **NuEclipse**, double-click the **Eclipse.exe**. Note that the .exe file and the related folders, such as the OpenOCD folder, should stay in the same directory; otherwise, the application will not work properly.

2.4 Запуск Eclipse Чтобы запустить NuEclipse, дважды щелкните файл Eclipse.exe. Обратите внимание, что файл .exe и связанные с ним папки, такие как папка OpenOCD, должны оставаться в одном каталоге; в противном случае приложение не будет работать должным образом.



3 Development Tutorial

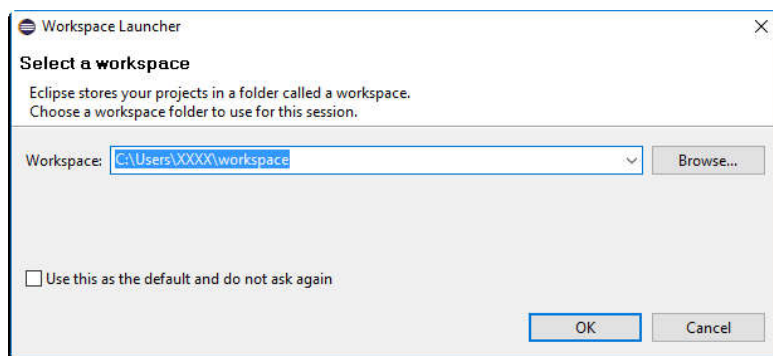
3.1 Select Workspace

When Eclipse launches, we have to select a workspace which groups a set of related projects together that usually make up an application. In addition, some configuration settings for Eclipse and projects are stored here, too. For different computers, the configuration settings may change. We should create our own workspace rather than copying another user's workspace. Only one workspace can be active at one time. The current workspace for Eclipse can be switched by clicking **File->Switch Workspace**

3 Development Tutorial

3.1 Выберите рабочее пространство.

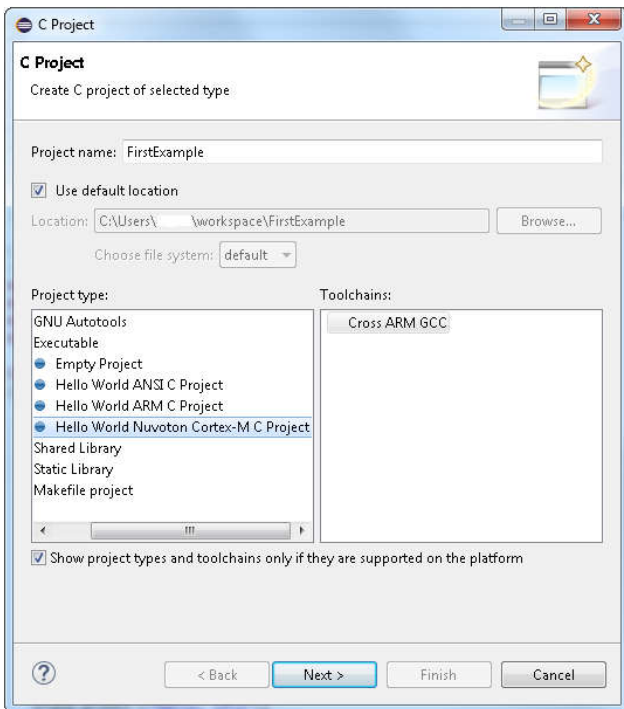
Когда Eclipse запускается, мы должны выбрать рабочую область, которая группирует вместе набор связанных проектов, которые обычно составляют приложение. Кроме того, здесь хранятся некоторые параметры конфигурации для Eclipse и проектов. Для разных компьютеров параметры конфигурации могут изменяться. Мы должны создать собственное рабочее пространство, а не копировать рабочее пространство другого пользователя. Одновременно может быть активна только одна рабочая область. Текущее рабочее пространство для Eclipse можно переключить, щелкнув File-> Switch Workspace



3.2 New Project Wizard

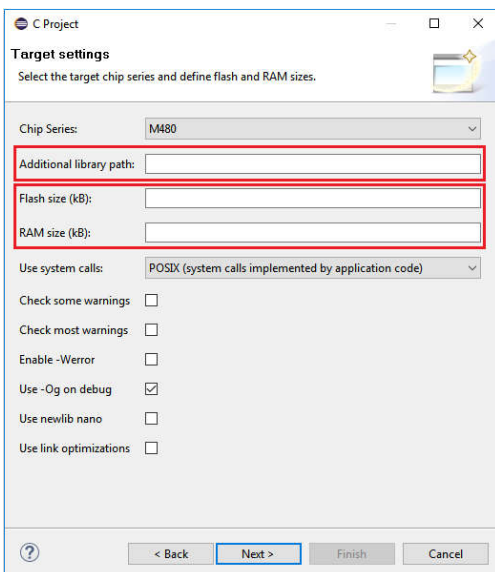
As a beginner, the fastest way to create a C/C++ project is using the New Project Wizard. For instance, to create a C project, click **File > New > C Project**. The New Project Wizard appears. Here we choose **Hello World Nuvoton Cortex-M C Project** for Project type. Input the project name and click the **Next >** button to continue.

3.2 Мастер создания нового проекта Для новичка самый быстрый способ создать проект C / C ++ - использовать мастер создания нового проекта. Например, чтобы создать проект C, щелкните File> New> C Project. Появится мастер создания нового проекта. Здесь мы выбираем Hello World Nuvoton Cortex-M C Project в качестве типа проекта. Введите имя проекта и нажмите кнопку Далее>, чтобы продолжить.



Based on the actual target chip, we select the corresponding chip series. For some chip series, e.g., M2351_NonSecure, we need to input the additional library path. If not, the build process may fail. In addition, input the real values to Flash and RAM size. If not, the default values will be used. When all the settings are done, click the **Next >** buttons until clicking the **Finish** button.

Выбираем соответствующую серию чипа. Для некоторых серий микросхем, например M2351_NonSecure, нам нужно ввести дополнительный путь к библиотеке. В противном случае процесс сборки может завершиться ошибкой. Кроме того, введите реальные значения для размера Flash и RAM. Если нет, будут использоваться значения по умолчанию. Когда все настройки будут выполнены, нажимайте кнопки Далее>, пока не нажмете кнопку Готово.



3.3 Import Existing Projects

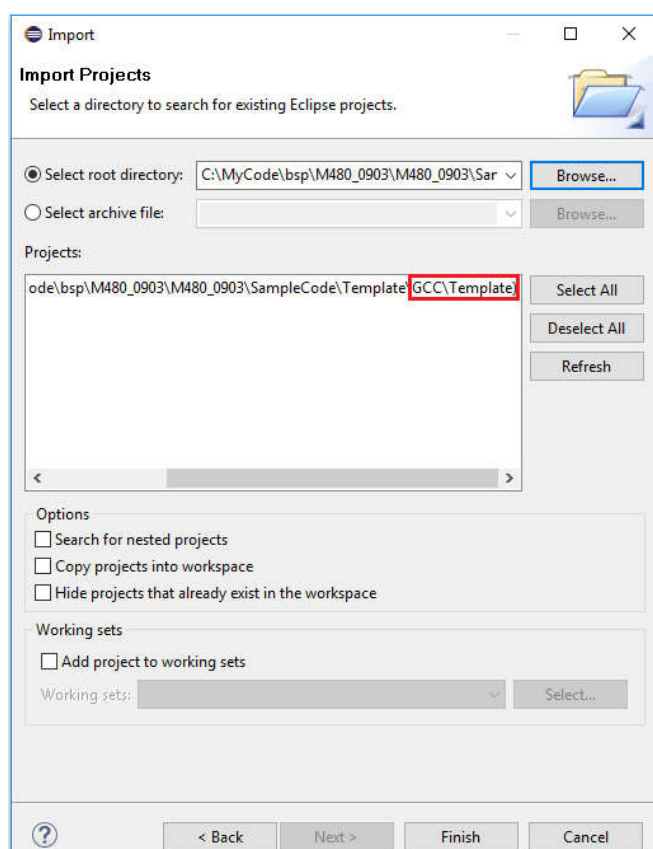
When BSP projects are available, we can import them into the workspace using the following steps:

1. From the main menu bar, select **File > Import**. The Import wizard shows up.
2. Select **General > Existing Project into Workspace** and click **Next**.
3. Choose either **Select root directory** or **Select archive file** and click the associated **Browse** to locate the directory or file containing the projects. In the Nuvoton BSP, the Eclipse projects are stored in the GCC folders (refer to the following picture).
4. Under **Projects** select the project or projects which you would like to import and click **Finish**.

3.3 Импорт существующих проектов.

Когда проекты BSP доступны, мы можем импортировать их в рабочую область, выполнив следующие действия:

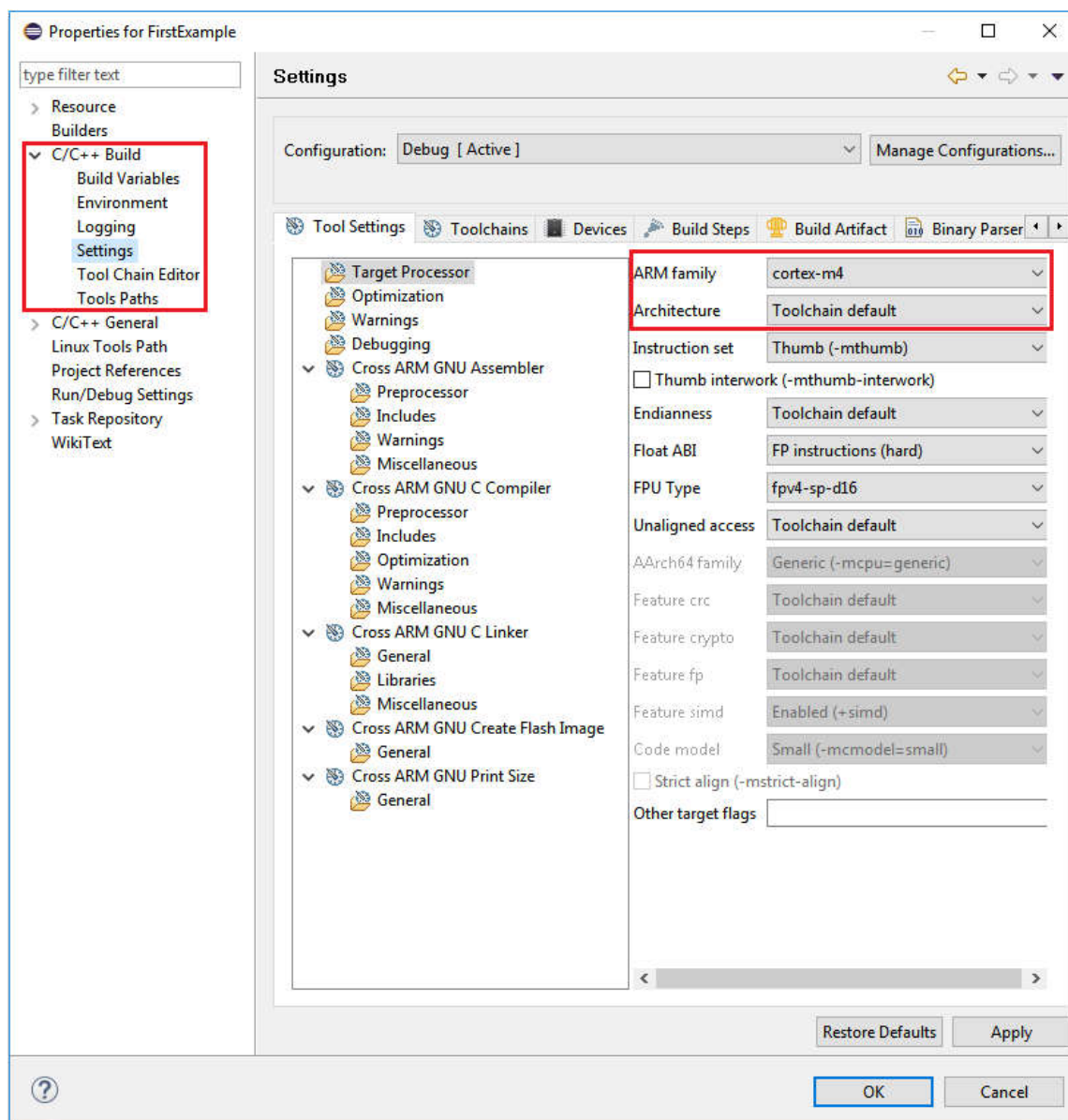
1. В строке главного меню выберите Файл> Импорт. Появится мастер импорта.
2. Выберите «Общие»> «Существующий проект в рабочей области» и нажмите «Далее».
3. Выберите либо «Выбрать корневой каталог», либо «Выбрать архивный файл» и нажмите соответствующий «Обзор», чтобы найти каталог или файл, содержащий проекты. В Nuvoton BSP проекты Eclipse хранятся в папках GCC (см. Следующий рисунок). 4. В разделе «Проекты» выберите проект или проекты, которые вы хотите импортировать, и нажмите «Готово».



3.4 Build Settings

After projects have been created, we still have a chance to alter the build settings by clicking **Project > Properties**. The Properties wizard shows up. Then go to **C/C++ Build > Settings**. From there, we can alter the build settings according to the actual target chip. Then click the **Apply** button to take effect. After applying build settings, we should be able to build projects successfully

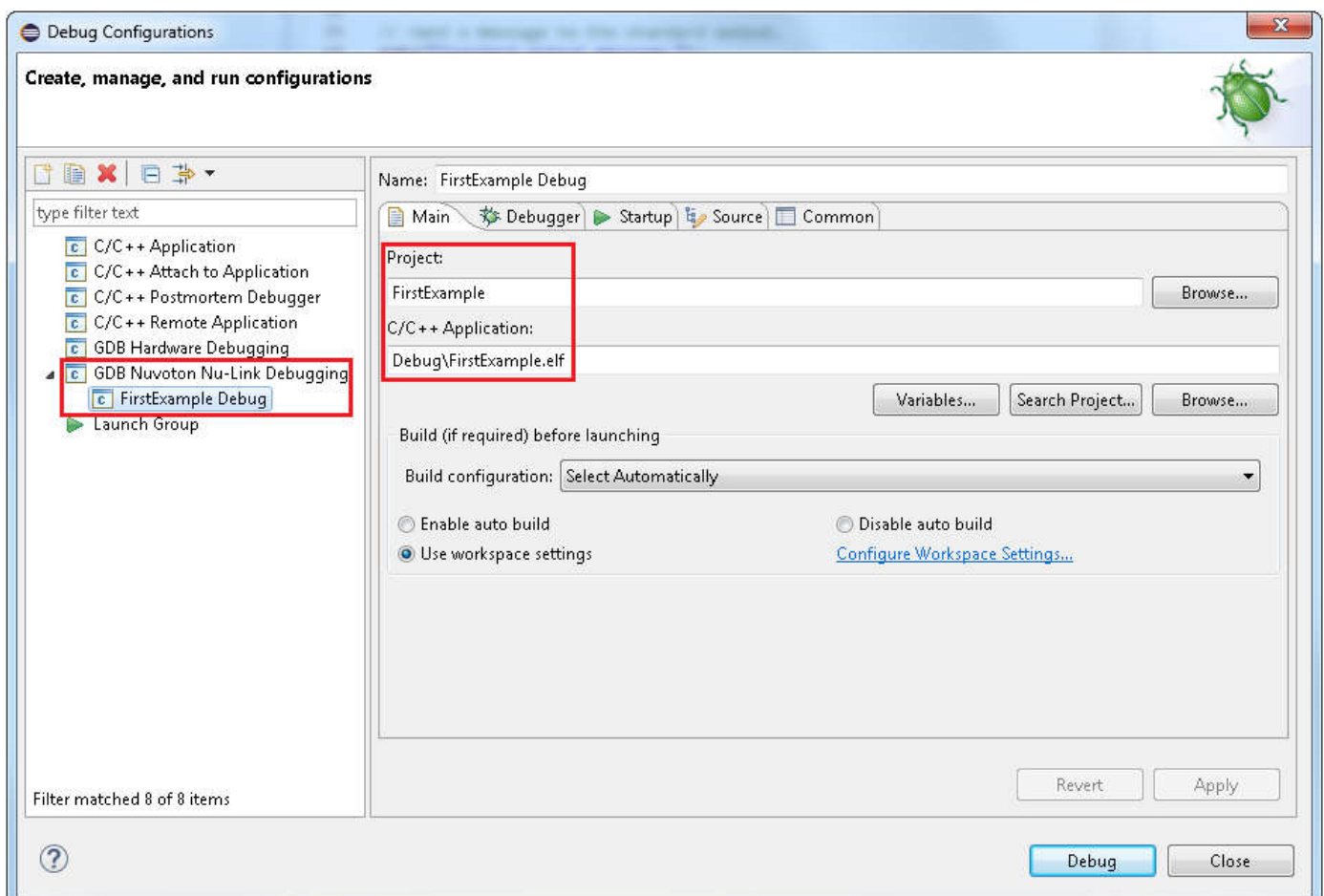
3.4 Настройки сборки После создания проектов у нас все еще есть возможность изменить настройки сборки, щелкнув «Проект»> «Свойства». Появится мастер свойств. Затем перейдите в С / С ++ Build> Settings. Оттуда мы можем изменить настройки сборки в соответствии с фактическим целевым чипом. Затем нажмите кнопку «Применить», чтобы применить изменения. После применения настроек сборки мы сможем успешно создавать проекты.



3.5 Debug Configuration

Before launching an application into the debug mode, we have to prepare a debug configuration, which contains all the necessary information about the debug mode. Click **Run > Debug Configuration...** to open the debug configuration dialog. Double click on the **GDB Nuvoton NuLink Debugging** group. The Nuvoton Nu-Link debug configuration appears on the right-hand side. In the Main tab, the name of Project should coincide with the project name. The C/C++ Application should point to the .elf application generated by the build process. If the project name or C/C++ Application is incorrect, please select the expected project first in the project view, build the project to generate the executable, and expand the tree to make sure the existence of the generated executable. Then repeat the former operations again

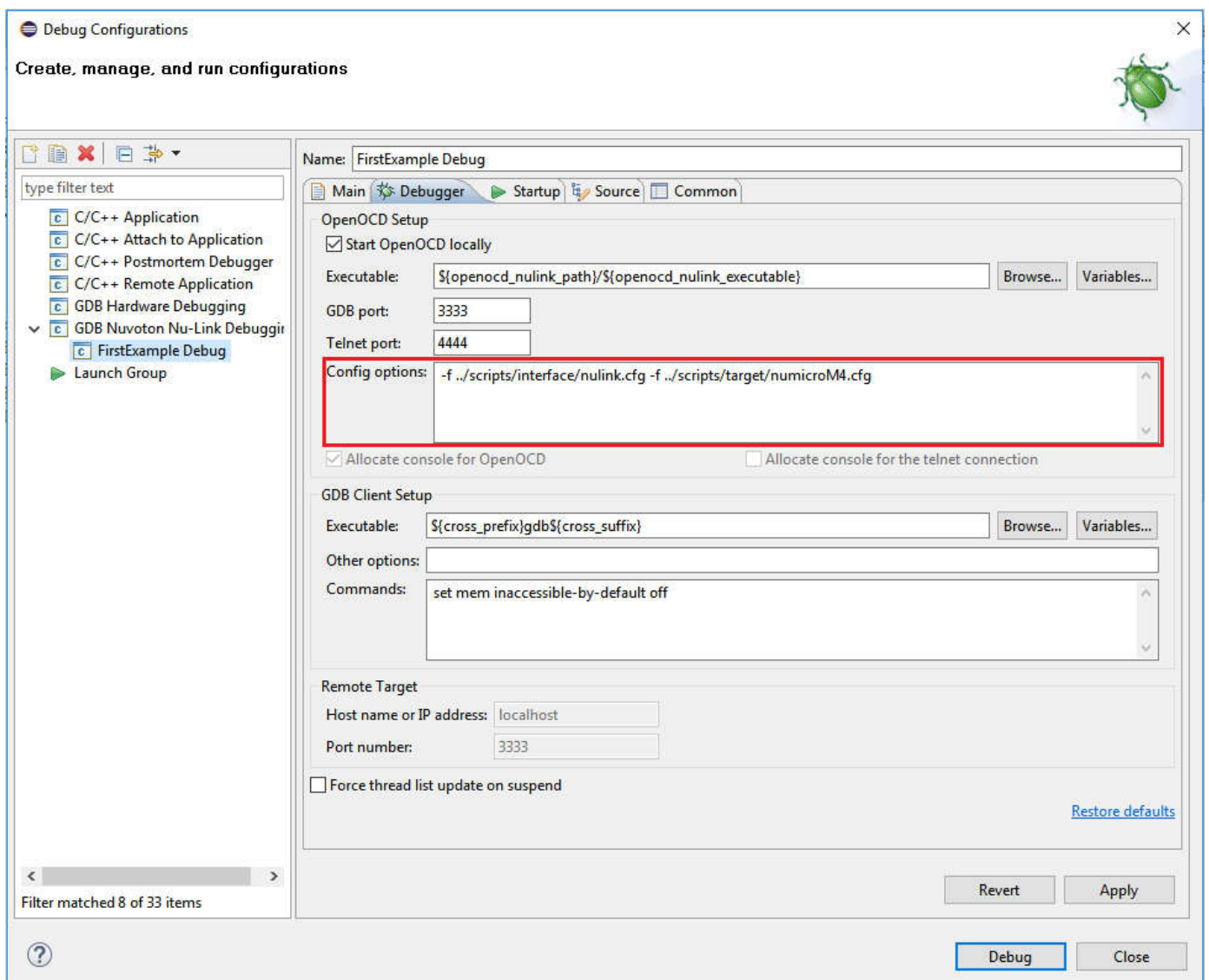
3.5 Конфигурация отладки Перед запуском приложения в режиме отладки мы должны подготовить конфигурацию отладки, которая содержит всю необходимую информацию о режиме отладки. Щелкните Выполнить> Конфигурация отладки..., чтобы открыть диалоговое окно конфигурации отладки. Дважды щелкните группу GDB Nuvoton NuLink Debugging. Отладка Nuvoton Nu-Link На правой стороне появляется конфигурация. Во вкладке Main название проекта должно совпадать с названием проекта. Приложение C / C ++ должно указывать на приложение .elf, созданное в процессе сборки. Если имя проекта или приложение C / C ++ неверно, сначала выберите ожидаемый проект в представлении проекта, постройте проект для создания исполняемого файла и разверните дерево, чтобы убедиться в существовании сгенерированного исполняемого файла. Затем повторите предыдущие операции еще раз.



3.5.1 Debugger Tab

The Debugger tab is used to provide the OpenOCD and GDB Client setup. OpenOCD requires correct configuration files to know how to work with adapters and target chips. The configuration files are specified in the **Config options** field. Nuvoton's adapter is Nu-Link, which uses the interface configuration file named **nulink.cfg**. In addition, Nuvoton has three different ARM families, such as M0, M4, and M23. The corresponding target configuration files are **numicroM0.cfg**, **numicroM4.cfg**, and **numicroM23.cfg**. For M23 2nd development, the target configuration file would be **numicroM23_NS.cfg**

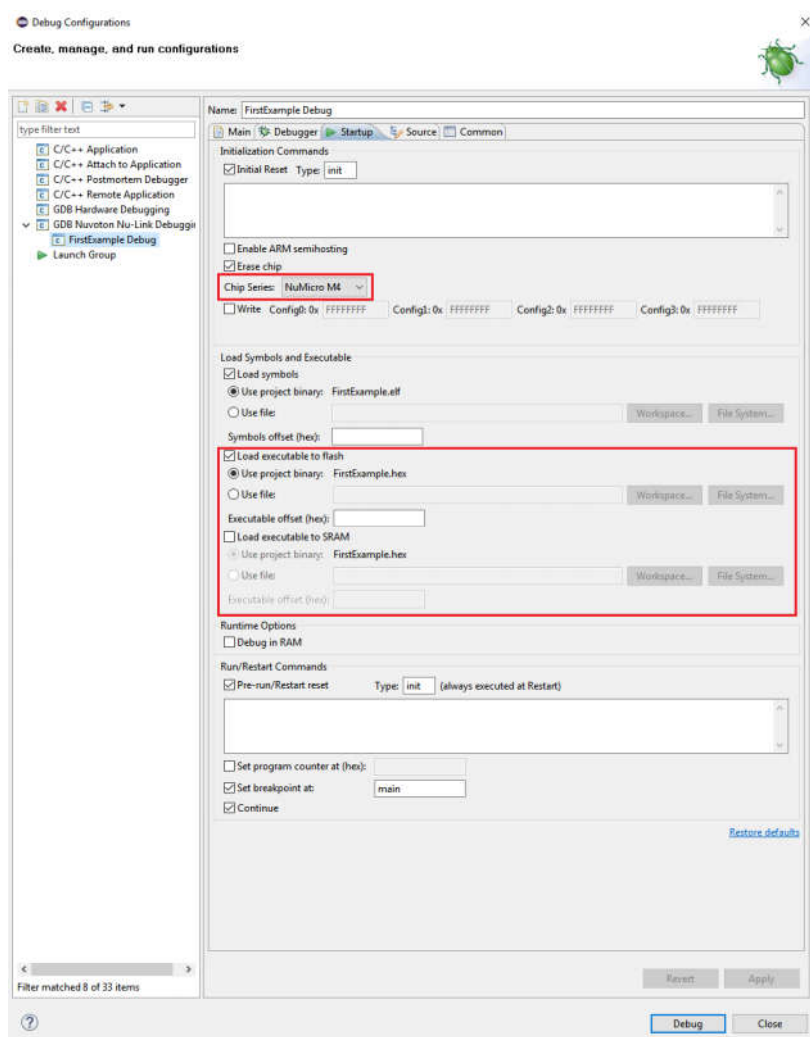
3.5.1 Вкладка отладчик Вкладка Debugger используется для настройки OpenOCD и клиента GDB. OpenOCD требует правильных файлов конфигурации, чтобы знать, как работать с адаптерами и целевыми микросхемами. Файлы конфигурации указываются в поле Параметры конфигурации. Адаптер Nuvoton - это Nu-Link, который использует интерфейсный файл конфигурации с именем nulink.cfg. Кроме того, Nuvoton имеет три разных семейства ARM, таких как M0, M4 и M23. Соответствующие файлы целевой конфигурации - umicroM0.cfg, numicroM4.cfg и numicroM23.cfg. Для второй разработки M23 целевой файл конфигурации будет umicroM23_NS.cfg



3.5.2 Startup Tab

As the first step, we should **choose the right Chip Series** in the Startup tab. When done, the corresponding target configuration file will be automatically written in the **Config options** field of the Debugger tab. To load executable to flash, we need to select the **Load executable to flash** checkbox. To load executable to RAM, we need to select the **Load executable to SRAM** checkbox. When all the settings are done, click the **Apply** button to take effect. To launch the application into the debug mode, click the **Debug** button.

3.5.2 Вкладка Startup В качестве первого шага мы должны выбрать правильную серию чипов на вкладке «Запуск». Когда это будет сделано, соответствующий целевой файл конфигурации будет автоматически записан в поле «Параметры конфигурации» на вкладке «Отладчик». Чтобы загрузить исполняемый файл во флэш-память, нам нужно установить флажок Загрузить исполняемый файл во флэш-память. Чтобы загрузить исполняемый файл в ОЗУ, нам нужно установить флажок Загрузить исполняемый файл в SRAM. Когда все настройки выполнены, нажмите кнопку «Применить», чтобы они вступили в силу. Чтобы запустить приложение в режиме отладки, нажмите кнопку «Отладка».



3.6 Debug Views

Eclipse provides many debug views. Each of them contains specific information for debugging.

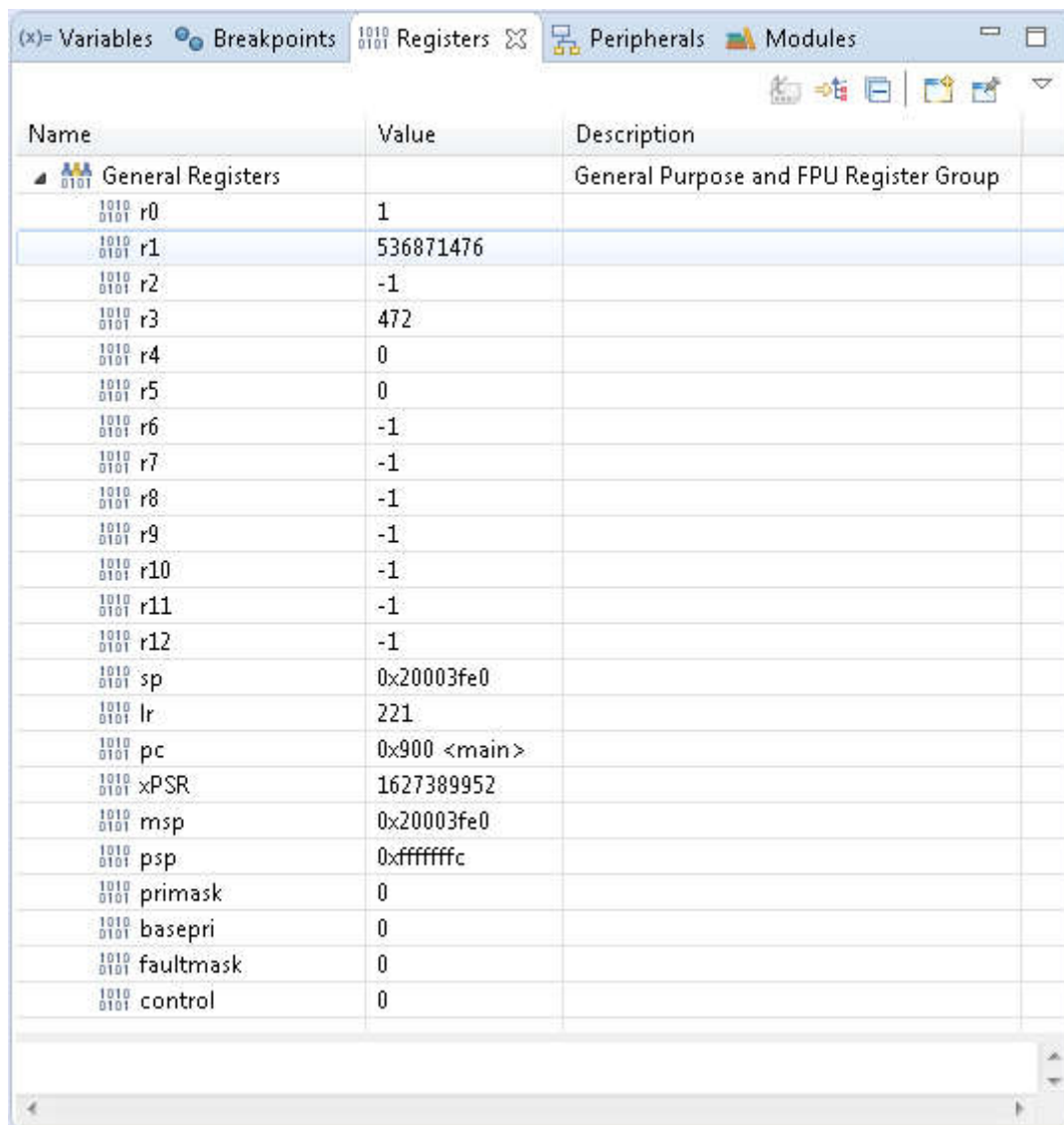
3.6.1 Registers View

When entering the debug mode, we can open the **Registers view** in the upper-right corner of Debug perspective. The Registers view lists information about the registers in a selected stack frame.

3.6 Просмотры отладки.

Eclipse предоставляет множество видов отладки. Каждый из них содержит определенную информацию для отладки.

3.6.1 Представление регистров. При входе в режим отладки мы можем открыть представление регистров в правом верхнем углу перспективы отладки. В представлении «Регистры» отображается информация о регистрах в выбранном кадре стека.

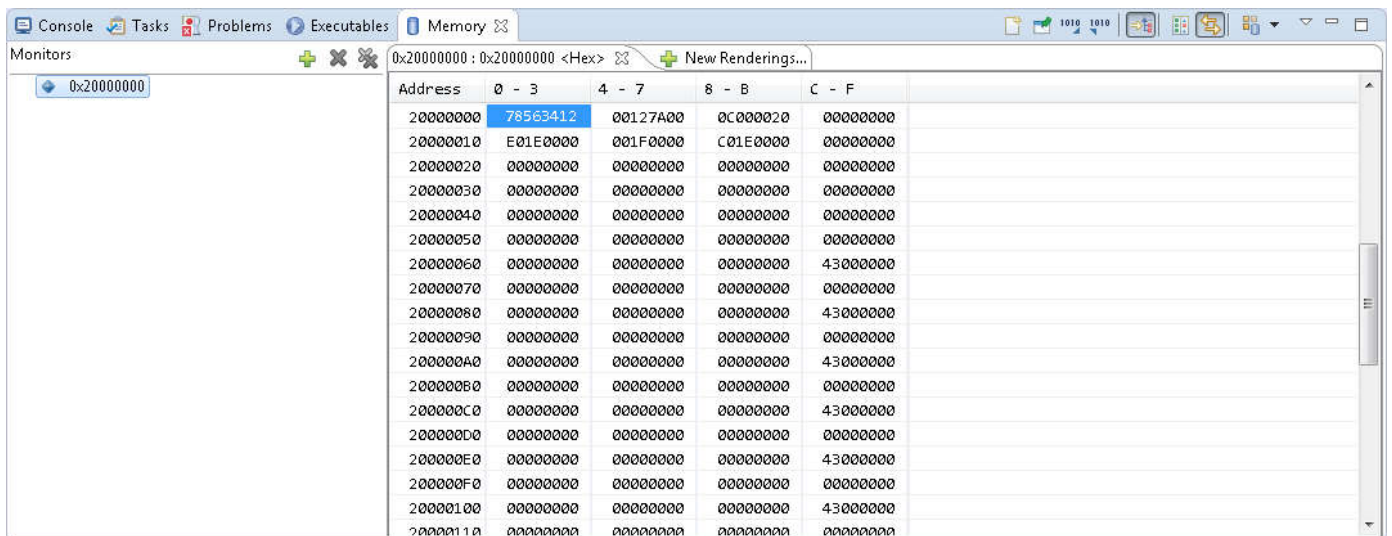


Name	Value	Description
General Registers		General Purpose and FPU Register Group
r0	1	
r1	536871476	
r2	-1	
r3	472	
r4	0	
r5	0	
r6	-1	
r7	-1	
r8	-1	
r9	-1	
r10	-1	
r11	-1	
r12	-1	
sp	0x20003fe0	
lr	221	
pc	0x900 <main>	
xPSR	1627389952	
msp	0x20003fe0	
psp	0xffffffffc	
primask	0	
basepri	0	
faultmask	0	
control	0	

3.6.2 Memory View

The **Memory view** of the Debug perspective is used to monitor and modify the process memory. The process memory is presented as a list of so called **memory monitors**. Each monitor represents a section of memory specified by its location called **base address**. To open it, click the Memory tab on the lower side of Debug perspective.

3.6.2 Просмотр памяти Представление «Память» перспективы «Отладка» используется для отслеживания и изменения памяти процесса. Память процесса представлена в виде списка так называемых мониторов памяти. Каждый монитор представляет собой раздел памяти, определяемый его местоположением, называемый базовым адресом. Чтобы открыть его, щелкните вкладку «Память» в нижней части перспективы «Отладка».



3.6.3 Disassembly View

The **Disassembly view** shows the loaded program as assembler instructions mixed with source code for comparison. We can do the following tasks in the Disassembly view:

1. Setting breakpoints at the start of any assembler instruction.
2. Enabling and disabling breakpoints.
3. Stepping through the disassembly instructions of the program.
4. Jumping to specific instructions in the program.

To open it, we need to click the **Instruction Stepping Mode** button on the upper toolbar, as follows:

3.6.3 Disassembly View.

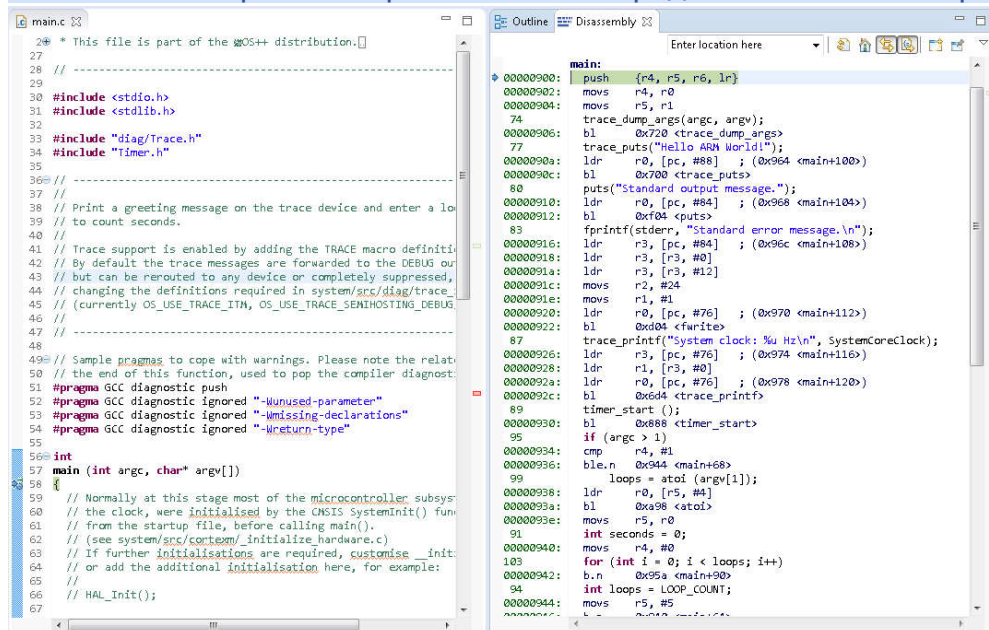
Представление Disassembly показывает загруженную программу как инструкции ассемблера, смешанные с исходным кодом. код для сравнения. В представлении «Разборка» мы можем выполнять следующие задачи:

1. Установка точек останова в начале любой инструкции ассемблера.
2. Включение и отключение точек останова.
3. Выполнение инструкций по разборке программы.
4. Переход к конкретным инструкциям в программе. Чтобы открыть его, нам нужно нажать кнопку «Пошаговый режим инструкций» на верхней панели инструментов, так как следует:



Then the Disassembly view will appear on the right-hand side.

После этого с правой стороны появится представление «Разборка».

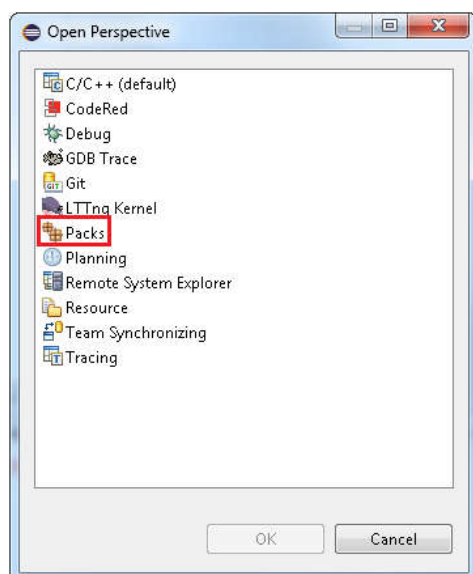


3.6.4 Peripheral Registers View

To display the Peripheral Registers view, we need to utilize **Packs** mechanism. Packs can help the user download SFR (special function register) files from the Keil repository. Firstly, we open the Packs perspective by choosing it in the **Open Perspective** dialog

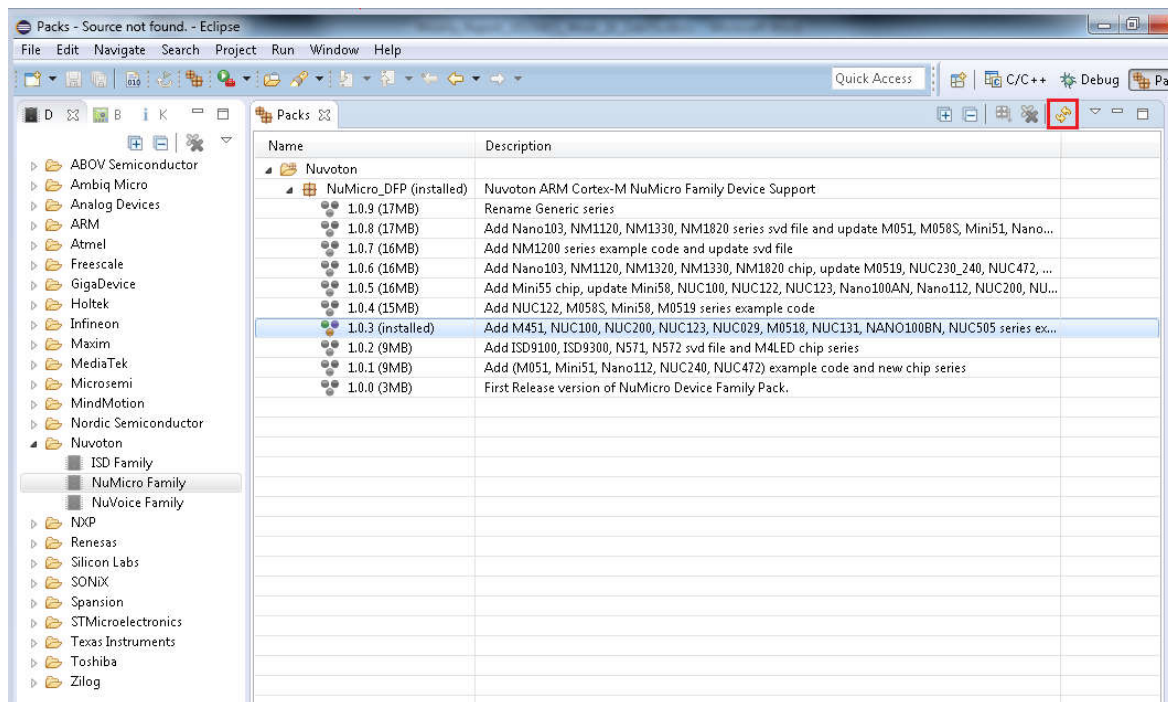
3.6.4 Просмотр периферийных регистров.

Чтобы отобразить представление Peripheral Registers, нам нужно использовать механизм Packs. Пакеты могут помочь пользователю загрузить файлы SFR (регистр специальных функций) из репозитория Keil. Во-первых, мы открываем перспективу Packs, выбирая ее в диалоговом окне Open Perspective.



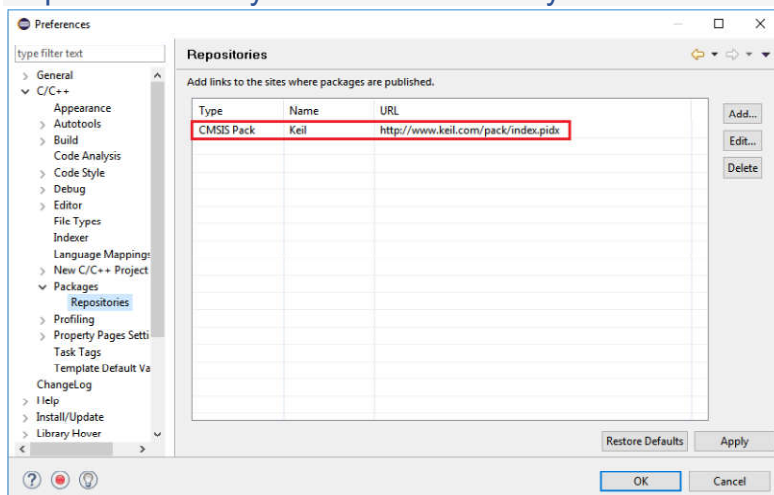
For the first time we see the Packs perspective, its content is provided by the NuEclipse installer. If the default content is missing, please switch to a new workspace and try again. To get the latest version, click the **Update the packages definitions from all repositories** button at the upperright corner. After clicking, Eclipse begins downloading all the SFR files from a repository.

Впервые мы видим перспективу Packs, ее содержимое предоставляется установщиком NuEclipse. Если содержимое по умолчанию отсутствует, переключитесь на новую рабочую область и повторите попытку. Чтобы получить самую последнюю версию, нажмите кнопку «Обновить определения пакетов из всех репозиторий» в правом верхнем углу. После щелчка Eclipse начинает загрузку всех файлов SFR из репозитория.



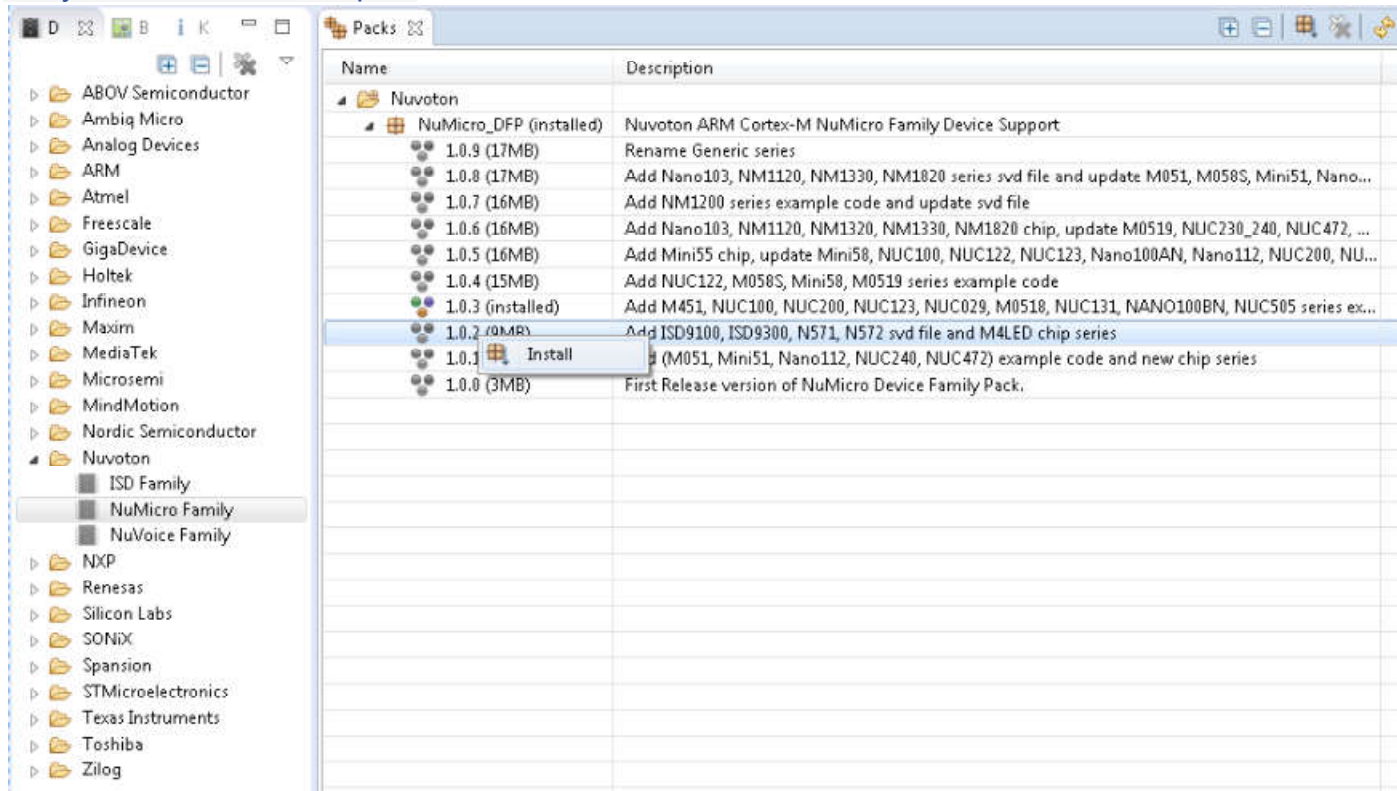
The locations of repositories are specified in the **Window > Preferences > C/C++ > Packages > Repositories**. The default is from Keil's CMSIS pack.

Расположение репозитория указывается в Window > Preferences > C / C ++ > Packages > Repositories. По умолчанию используется пакет CMSIS Кейла.



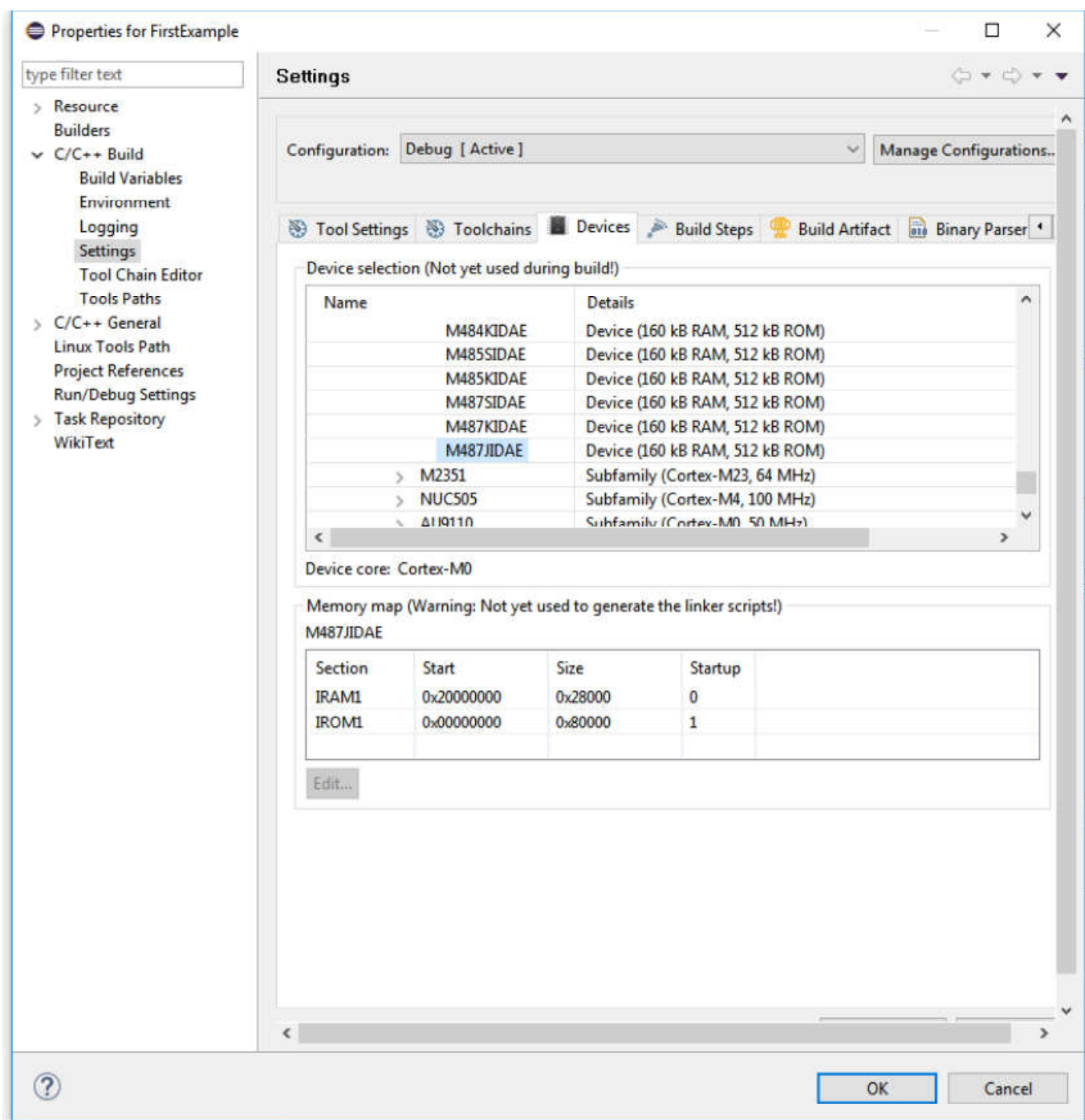
When the download is completed, we can find the Nuvoton SFR files and install them on Eclipse if needed.

Когда загрузка будет завершена, мы сможем найти файлы Nuvoton SFR и при необходимости установить их в Eclipse.



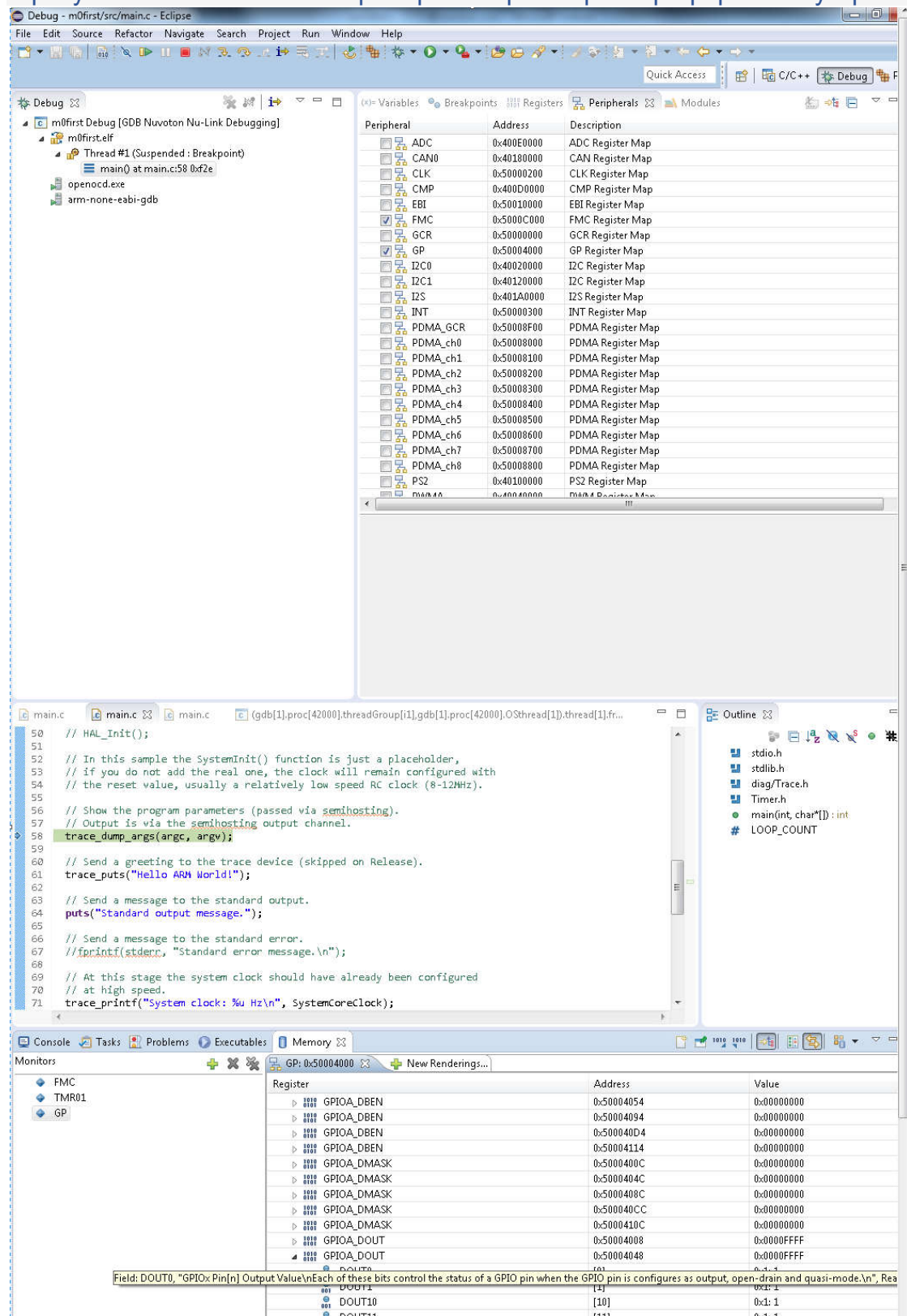
To use the specific SFR file, open the project's properties dialog and go to the **C/C++ Build > Settings**. From there, we should choose the specific device matching the real one. In this case, it is M487JIDAE. Click the **Apply** button to take effect.

Чтобы использовать конкретный файл SFR, откройте диалоговое окно свойств проекта и перейдите в вкладку **C / C ++ Build > Settings**. Оттуда мы должны выбрать конкретное устройство, соответствующее реальному. В данном случае это M487JIDAE. Нажмите кнопку «Применить», чтобы применить изменения.



As a result, we can monitor the peripheral registers when debugging.

В результате мы можем контролировать регистры периферийных устройств при отладке.



The screenshot displays the Eclipse IDE interface for debugging a Nu-Link device. The top toolbar includes standard IDE functions like File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The main workspace is divided into several panels:

- Debug Console:** Shows the current debug session for 'm0first Debug [GDB Nu-Link Debugging]'. It lists the loaded binaries: 'm0first.elf', 'main() at main.c:58 0xf2e', 'openocd.exe', and 'arm-none-eabi-gdb'.
- Peripherals:** A table listing various hardware peripherals and their register maps.

Peripheral	Address	Description
ADC	0x400E0000	ADC Register Map
CAN0	0x40180000	CAN Register Map
CLK	0x50000200	CLK Register Map
CMP	0x400D0000	CMP Register Map
EBI	0x50010000	EBI Register Map
FMC	0x5000C000	FMC Register Map
GCR	0x50000000	GCR Register Map
GP	0x50004000	GP Register Map
I2C0	0x40020000	I2C Register Map
I2C1	0x40120000	I2C Register Map
I2S	0x401A0000	I2S Register Map
INT	0x50000300	INT Register Map
PDMA_GCR	0x50008F00	PDMA Register Map
PDMA_ch0	0x50008000	PDMA Register Map
PDMA_ch1	0x50008100	PDMA Register Map
PDMA_ch2	0x50008200	PDMA Register Map
PDMA_ch3	0x50008300	PDMA Register Map
PDMA_ch4	0x50008400	PDMA Register Map
PDMA_ch5	0x50008500	PDMA Register Map
PDMA_ch6	0x50008600	PDMA Register Map
PDMA_ch7	0x50008700	PDMA Register Map
PDMA_ch8	0x50008800	PDMA Register Map
PS2	0x40100000	PS2 Register Map
- Code Editor:** Displays the source code for 'main.c'. The code includes initialization functions, trace output, and a loop counter.


```

50 // HAL_Init();
51
52 // In this sample the SystemInit() function is just a placeholder,
53 // if you do not add the real one, the clock will remain configured with
54 // the reset value, usually a relatively low speed RC clock (8-12MHz).
55
56 // Show the program parameters (passed via semihosting).
57 // Output is via the semihosting output channel.
58 trace_dump_args(argc, argv);
59
60 // Send a greeting to the trace device (skipped on Release).
61 trace_puts("Hello ARM world!");
62
63 // Send a message to the standard output.
64 puts("Standard output message.");
65
66 // Send a message to the standard error.
67 //fprintf(stderr, "Standard error message.\n");
68
69 // At this stage the system clock should have already been configured
70 // at high speed.
71 trace_printf("System clock: %u Hz\n", SystemCoreClock);
      
```
- Outline:** A list of files and symbols loaded during the debug session, including 'stdio.h', 'stdlib.h', 'diag/Trace.h', 'Timer.h', 'main(int, char*[]): int', and 'LOOP_COUNT'.
- Monitors:** A panel showing the state of various peripherals. The 'GP' (GPIO) peripheral is selected, displaying a list of registers and their values.

Register	Address	Value
GPIOA_DBEN	0x50004054	0x00000000
GPIOA_DBEN	0x50004094	0x00000000
GPIOA_DBEN	0x500040D4	0x00000000
GPIOA_DBEN	0x50004114	0x00000000
GPIOA_DMASK	0x5000400C	0x00000000
GPIOA_DMASK	0x5000404C	0x00000000
GPIOA_DMASK	0x5000408C	0x00000000
GPIOA_DMASK	0x500040CC	0x00000000
GPIOA_DMASK	0x5000410C	0x00000000
GPIOA_DOUT	0x50004008	0x0000FFFF
GPIOA_DOUT	0x50004048	0x0000FFFF
GPIOA_DOUT	0x50004088	0x0000FFFF
GPIOA_DOUT	0x500040C8	0x0000FFFF
GPIOA_DOUT	0x50004108	0x0000FFFF
DOUT10	[10]	0x1: 1
DOUT11	[11]	0x1: 1

3.7 Watchpoints

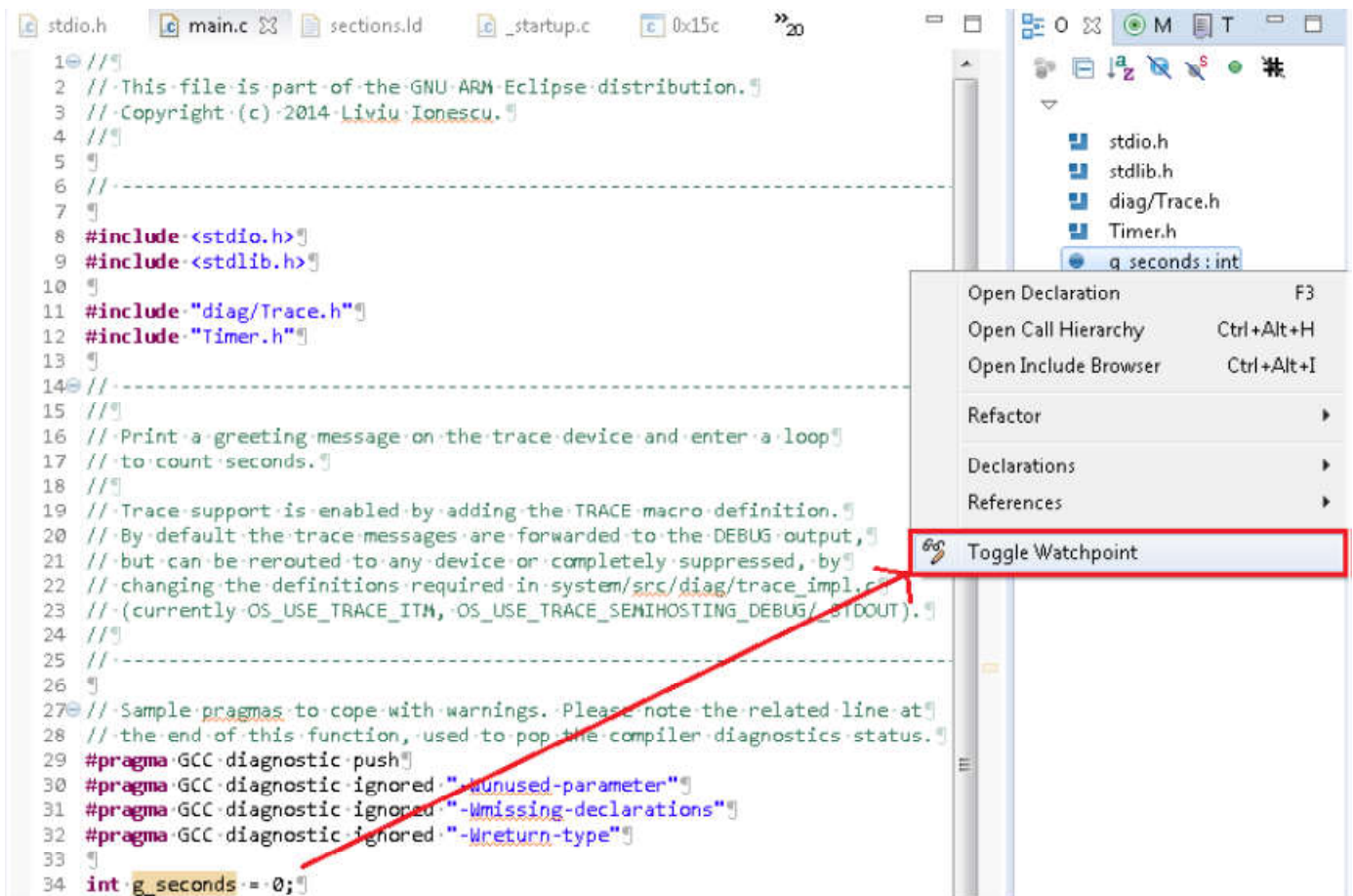
To add watchpoints on Eclipse, we need to do the following steps:

1. Selecting a **globe variable**, i.e. `g_seconds`, in the Outline view.
2. Right-clicking on the global variable and choosing **Toggle Watchpoint**.

3.7 Точки наблюдения.

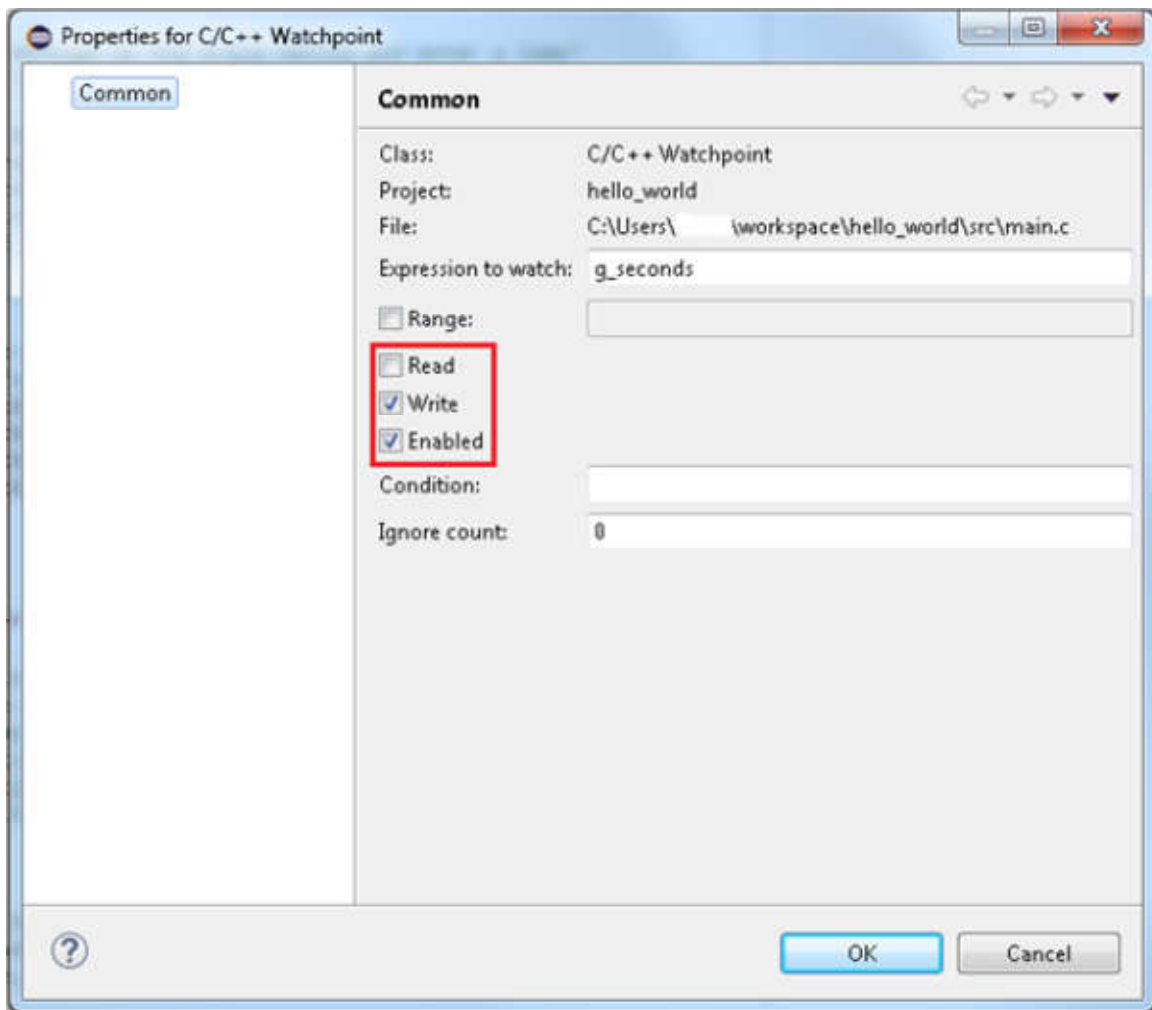
Чтобы добавить точки наблюдения в Eclipse, нам нужно сделать следующие шаги:

1. Выбрать глобальную переменную, например, `g_seconds`, в режиме Outline.
2. Щелкните глобальную переменную правой кнопкой мыши и выберите Toggle Watchpoint.



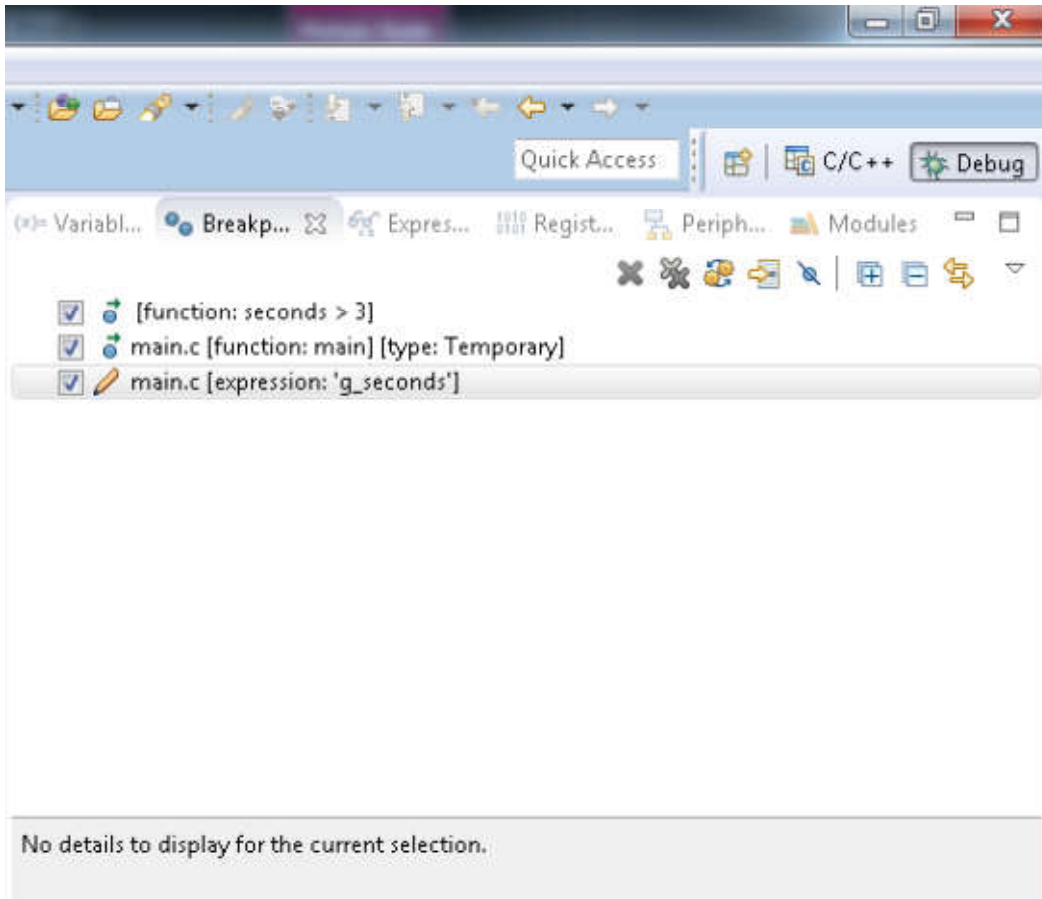
3. Configuring the settings for watchpoints. To stop execution when the watch expression is read, select the **Read** checkbox. To stop execution when the watch expression is written to, select the **Write** checkbox.

3. Настройка параметров точек наблюдения. Чтобы остановить выполнение при чтении контрольного выражения, установите флажок Читать. Чтобы остановить выполнение при записи контрольного выражения, установите флажок Запись.



When the watchpoint is added, it appears in the **Breakpoints view**.

Когда точка наблюдения добавлена, она появляется в представлении «Точки останова».



3.8 Debug in RAM

To debug in RAM, there are several steps to follow:

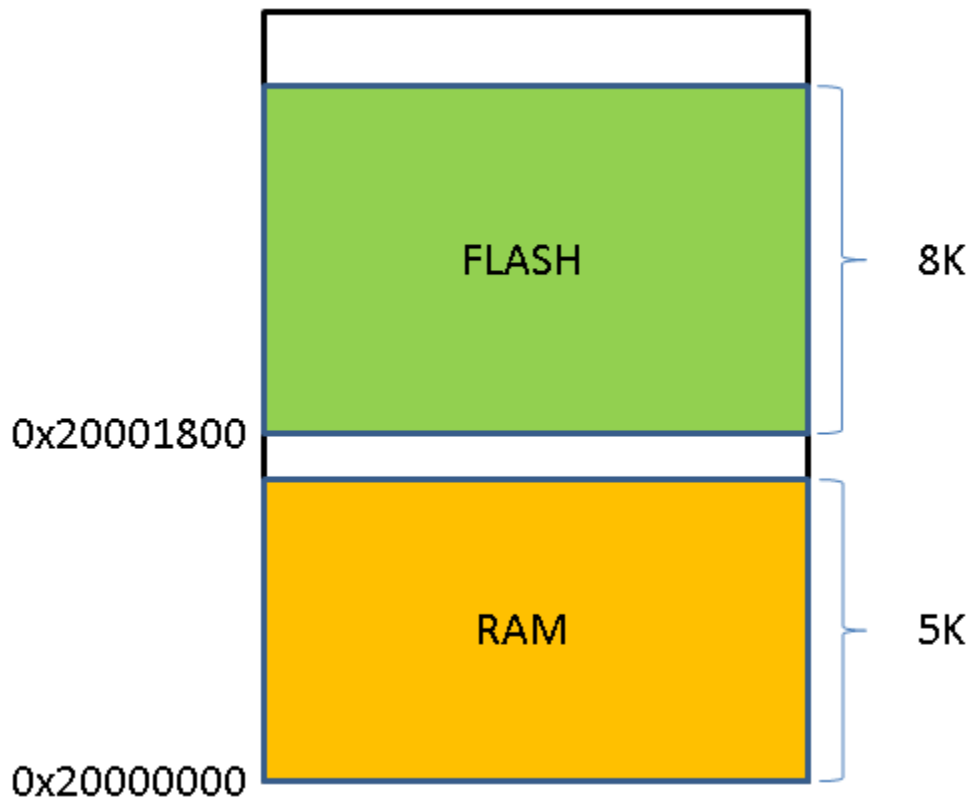
1. Modifying the ld script.
2. Assigning PC to the specific RAM address.
3. Assigning SP to the specific RAM address.
4. Downloading the binary file to RAM.

The ld script is responsible for telling the linker the layout of the compiled executable. For example, the memory layout looks like:

3.8 Отладка в ОЗУ.

Чтобы выполнить отладку в ОЗУ, необходимо выполнить несколько шагов:

1. Модификация ld скрипта.
2. Назначение ПК конкретному адресу RAM.
3. Назначение SP конкретному адресу RAM.
4. Загрузка бинарного файла в оперативную память. Скрипт ld отвечает за сообщение компоновщику макета скомпилированного исполняемого файла. Например, схема памяти выглядит так:



The modified mem.ld script should meet the memory layout design.

Измененный скрипт mem.ld должен соответствовать дизайну разметки памяти.

```

C/C++ - FirstExample/ldscripts/mem.ld - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

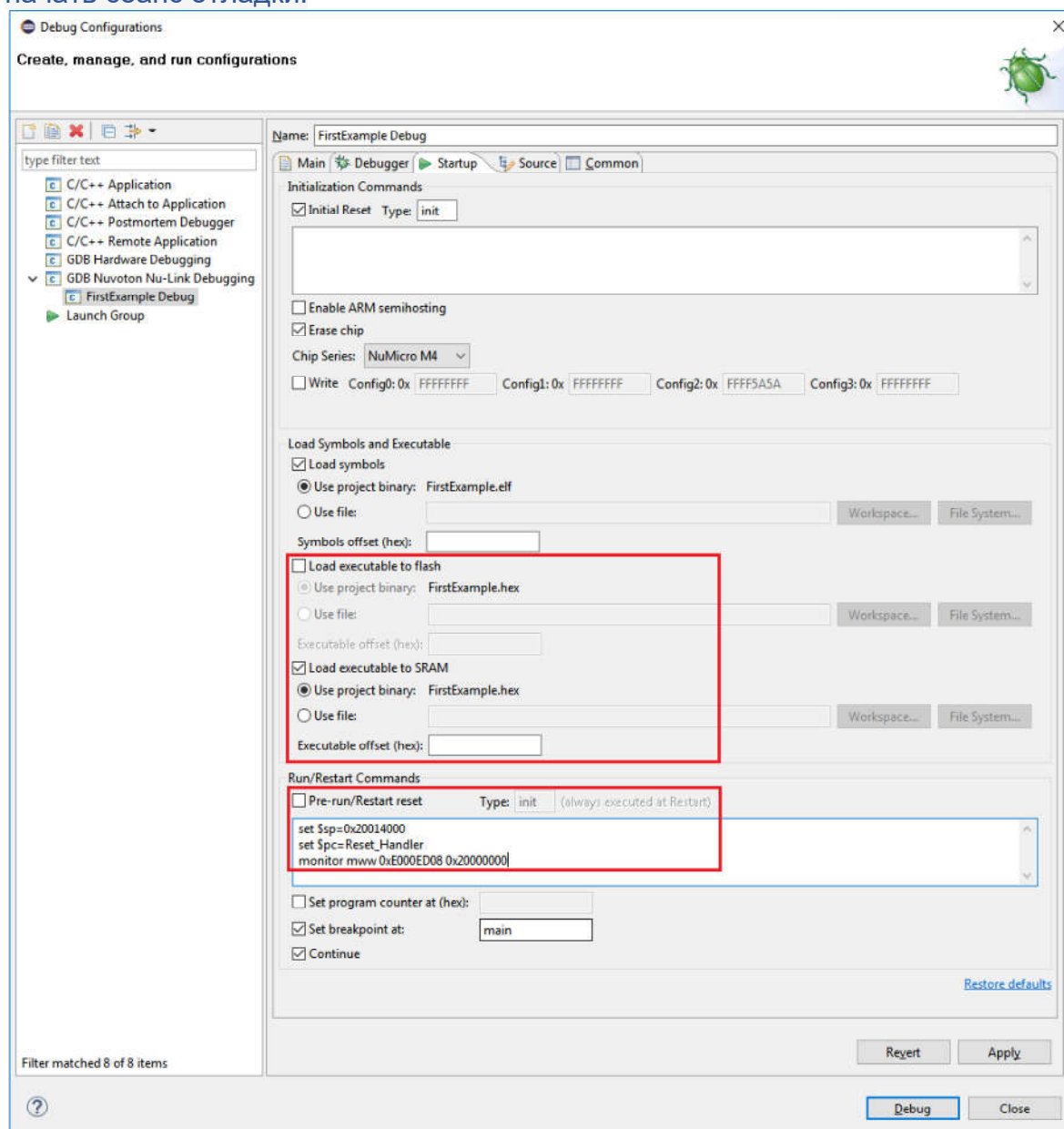
Project Explorer
  FirstExample
    Binaries
    Includes
    src
      _write.c
      main.c
      Timer.c
    system
    Debug
    include
    ldscripts
      libs.ld
      mem.ld
      sections.ld
    t

main.c mem.ld
1 /*
2  * Memory Spaces Definitions.
3  *
4  * Need modifying for a specific board.
5  * FLASH.ORIGIN: starting address of flash
6  * FLASH.LENGTH: length of flash
7  * RAM.ORIGIN: starting address of RAM bank 0
8  * RAM.LENGTH: length of RAM bank 0
9  *
10 * The values below can be addressed in further linker scripts
11 * using functions like 'ORIGIN(RAM)' or 'LENGTH(RAM)'.
12 */
13
14 MEMORY
15 {
16     FLASH (rx) : ORIGIN = 0x20001800, LENGTH = 8K
17     RAM (xrw)  : ORIGIN = 0x20000000, LENGTH = 5K
18 }
19
20 /*
21 * Optional sections; define the origin and length to match
22 * the the specific requirements of your hardware. The zero
23 * length prevents inadvertent allocation.
24 */
25 CCNRAM (xrw) : ORIGIN = 0x10000000, LENGTH = 0
26 FLASHB1 (rx) : ORIGIN = 0x00000000, LENGTH = 0
27 EXTENB0 (rx) : ORIGIN = 0x00000000, LENGTH = 0
28 EXTENB1 (rx) : ORIGIN = 0x00000000, LENGTH = 0
29 EXTENB2 (rx) : ORIGIN = 0x00000000, LENGTH = 0
30 EXTENB3 (rx) : ORIGIN = 0x00000000, LENGTH = 0
31 }
32
33 /*
34 * For external ram use something like:
35 * RAM (xrw) : ORIGIN = 0x64000000, LENGTH = 2048K
36 *
37 * For special RAM areas use something like:
38 * MEMORY_ARRAY (xrw) : ORIGIN = 0x20002000, LENGTH = 32
39 */

```

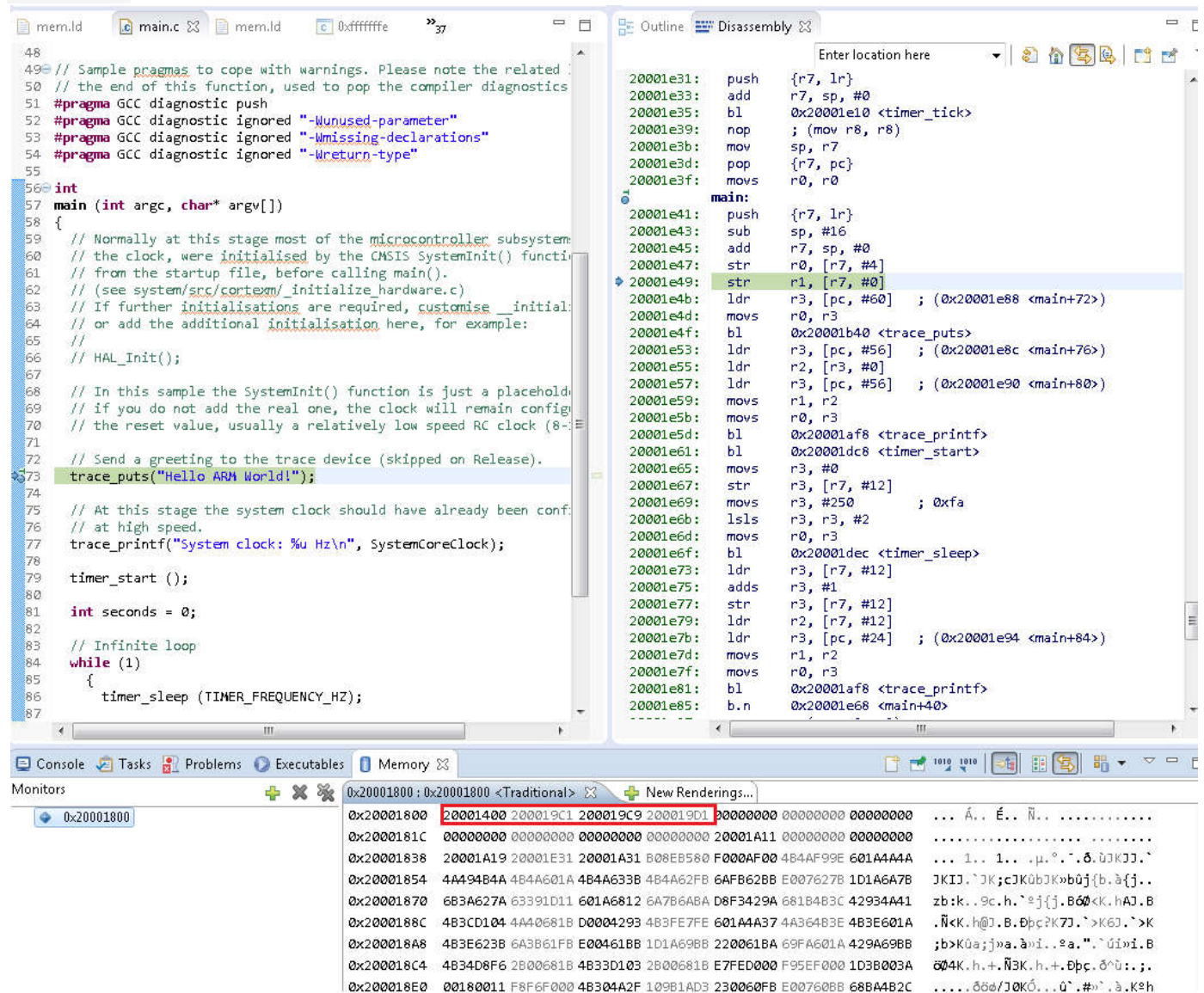
To assign PC and SP to the specific addresses, we need to input them in the debug configuration, as follows. Based on the previous memory layout, the PC and SP addresses should be Reset_Handler and 0x20001400, respectively. In addition, set Vector Table Offset Register (0x0xE000ED08) should be 0x20000000 and unselect the **Pre-run/Restart reset** Button. To download the binary to RAM, we select the **Load executable to SRAM** button and unselect the **Load executable to flash** button. Click the **Debug** button to start a debug session.

Чтобы назначить ПК и SP конкретным адресам, нам необходимо ввести их в конфигурацию отладки, как показано ниже. Основываясь на предыдущей структуре памяти, адреса ПК и SP должны быть Re-set_Handler и 0x20001400 соответственно. Кроме того, установите регистр смещения таблицы векторов (0x0xE000ED08), который должен быть 0x20000000, и отмените выбор кнопки сброса перед запуском / перезапуском. Чтобы загрузить двоичный файл в ОЗУ, мы нажимаем кнопку «Загрузить исполняемый файл в SRAM» и снимаем флажок «Загрузить исполняемый файл для прошивки». Нажмите кнопку «Отладка», чтобы начать сеанс отладки.



When the program stops in the main function, we open the Memory view. From there, we can verify that the binary file is successfully downloaded into RAM. The first word denotes the SP address. The following words denote the addresses of handlers.

Когда программа останавливается в основной функции, мы открываем представление памяти. Оттуда мы можем убедиться, что двоичный файл успешно загружен в оперативную память. Первое слово обозначает адрес SP. Следующие слова обозначают адреса обработчиков.



The screenshot displays an IDE with three main panels:

- Source Code (left):** Shows the `main.c` file. The `main` function is visible, starting with `int main (int argc, char* argv[])`. It includes comments about initialization and a `while (1)` loop containing `timer_sleep (TIMER_FREQUENCY_HZ);`.
- Disassembly (middle):** Shows the assembly code for the `main` function. Key instructions include `push {r7, lr}`, `add r7, sp, #0`, `bl 0x20001e10 <timer_tick>`, `mov sp, r7`, `pop {r7, pc}`, `movs r0, r0`, `push {r7, lr}`, `sub sp, #16`, `add r7, sp, #0`, `str r0, [r7, #4]`, `ldr r3, [pc, #60] ; (0x20001e88 <main+72>)`, `movs r0, r3`, `bl 0x20001b40 <trace_puts>`, `ldr r3, [pc, #56] ; (0x20001e8c <main+76>)`, `ldr r2, [r3, #0]`, `ldr r3, [pc, #56] ; (0x20001e90 <main+80>)`, `movs r1, r2`, `movs r0, r3`, `bl 0x20001af8 <trace_printf>`, `bl 0x20001dc8 <timer_start>`, `movs r3, #0`, `str r3, [r7, #12]`, `movs r3, #250 ; 0xfa`, `lsls r3, r3, #2`, `movs r0, r3`, `bl 0x20001dec <timer_sleep>`, `ldr r3, [r7, #12]`, `adds r3, #1`, `str r3, [r7, #12]`, `ldr r2, [r7, #12]`, `ldr r3, [pc, #24] ; (0x20001e94 <main+84>)`, `movs r1, r2`, `movs r0, r3`, `bl 0x20001af8 <trace_printf>`, `b.n 0x20001e68 <main+40>`.
- Memory View (bottom):** Shows the memory dump starting at address `0x20001800`. The first word is `0x20001400`, which is highlighted in red. The memory dump shows a sequence of words in hexadecimal and their corresponding ASCII representation.