МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

РАЗРАБОТКА И ОТЛАДКА ПРОГРАММ ДЛЯ МИКРОКОНТРОЛЛЕРОВ С ЯДРОМ CORTEX-M3 В СРЕДЕ μVision

Рекомендовано редакционно-издательским советом федерального государственного автономного образовательного учреждения высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» в качестве методических указаний для студентов Самарского университета, обучающихся по основным образовательным программам высшего образования 12.03.04 Биотехнические системы и технологии, 12.03.05 Лазерная техника и лазерные технологии, 03.03.01 Прикладные математика и физика

Составители: *И.А. Кудрявцев,*
Д.В. Корнилин,
О.О. Мякинин

С А М А Р А
Издательство Самарского университета
2020

УДК 004(075)
ББК 32.973я7

Составители: ***И.А. Кудрявцев, Д.В. Корнилин, О.О. Мякинин***

Рецензент канд. техн. наук М. П. К а л а е в

Разработка и отладка программ для микроконтроллеров с ядром Cortex-M3 в среде µVision: методические указания / составители: *И.А. Кудрявцев, Д.В. Корнилин, О.О. Мякинин*. – Самара: Издательство Самарского университета, 2020. – 24 с.

В методических указаниях рассмотрены вопросы разработки программ для микроконтроллеров с ядром Cortex-M3. Продемонстрированы разработка и основные приемы отладки программ для данного типа микропроцессоров в среде программирования Keil µVision 5. Приведены порядок выполнения лабораторных работ и требования к отчету.

Методические указания предназначены для студентов, обучающихся по направлениям 12.03.04 Биотехнические системы и технологии, 12.03.05 Лазерная техника и лазерные технологии, 03.03.01 Прикладные математика и физика, выполняющих лабораторные работы по дисциплинам «Цифровые устройства и микропроцессоры», «Основы микропроцессорных систем и программирование микроконтроллеров». Подготовлены на кафедре лазерных и биотехнических систем.

УДК 004(075)
ББК 32.973я7

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время микроконтроллеры с ядром CORTEX-M3 являются одними из наиболее распространенных в данном сегменте. Большинство ведущих производителей микроконтроллеров имеют в своей продуктовой линейке микросхемы на базе ядра CORTEX-M3. В методических указаниях рассматривается микроконтроллер К1986ВЕ92QI производства российской компании «Миландр».

Микроконтроллеры К1986ВЕ92QI, помимо ядра CORTEX-M3, способного работать на частоте до 80 МГц, обладают значительным объемом FLASH-памяти на кристалле и большим набором периферийных устройств, включая АЦП, ЦАП, порты ввода-вывода, таймеры и модули различных интерфейсов. Микроконтроллеры снабжены системой внутрисхемного программирования на базе интерфейса JTAG и специальным загрузчиком, позволяющим загружать код программы через асинхронный приемопередатчик. Для разработки ПО используется широко известная среда Keil μVision, которая рассматривается в данных методических указаниях.

Среда разработки Keil μVision позволяет работать в режиме симулятора и с поддержкой аппаратной отладки. В методические указания включены упражнения с использованием обеих возможностей. Большая часть работы посвящена изучению особенностей микроконтроллера посредством отладочной платы, предоставленной компанией «Миландр». Установленные модули (LCD дисплей, светодиоды и др.) позволяют изучить основные варианты применения микроконтроллера в системах управления.

Методические указания позволяют студентам изучить основы применения языка С для разработки программ микроконтроллеров с ядром CORTEX-M3. Указания не претендуют на полноту описания особенностей ядра, равно как и микроконтроллеров К1986ВЕ92QI или среды разработки. Приводятся лишь краткие пояснения, необходимые для понимания приведенных фрагментов кода. Методические указания содержат также список дополнительных заданий для самостоятельных экспериментов с микроконтроллером и вопросы для самоконтроля.

1 СОЗДАНИЕ ПРОЕКТА В СРЕДЕ MVISION ФИРМЫ KEIL

Первый проект предназначен для усвоения базовых навыков работы со средой µVision и микроконтроллером К1986ВЕ92.

Запустите среду разработки с помощью ярлыка  , после чего в рабочем окне выберите пункт меню **Project/New uVision Project** и далее укажите путь к **папке проекта** и **имя проекта**. Настоятельно рекомендуется **не** использовать русскоязычные символы и пробелы в пути и названии проектов и файлов.

В открывшемся далее окне (рис. 1) выберите микроконтроллер **MDR1986BE92** и нажмите **ОК**.

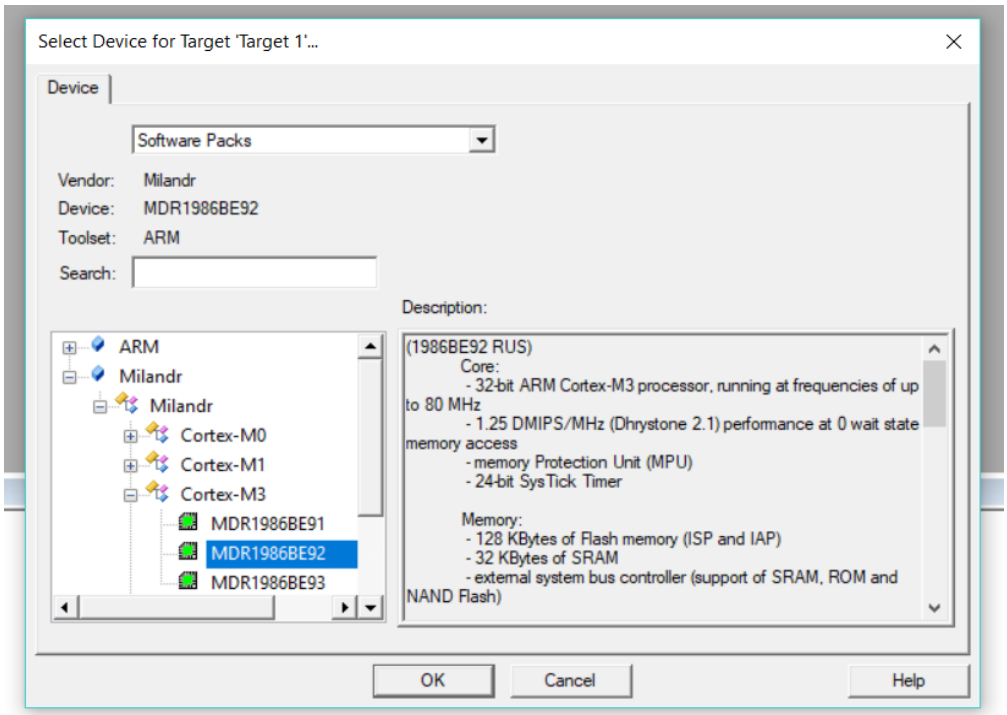

Рис. 1. Выбор микроконтроллера

В следующем окне отметьте галочками модуль **Startup** для выбранного микроконтроллера и **драйверы** системы тактирования и управления портами, как показано на рис. 2. В дальнейшем, изменяя выбор модулей, можно изменять конфигурацию проекта, подключая и отключая отдельные модули. Окно конфигурации можно вызвать с помощью ярлыка  .

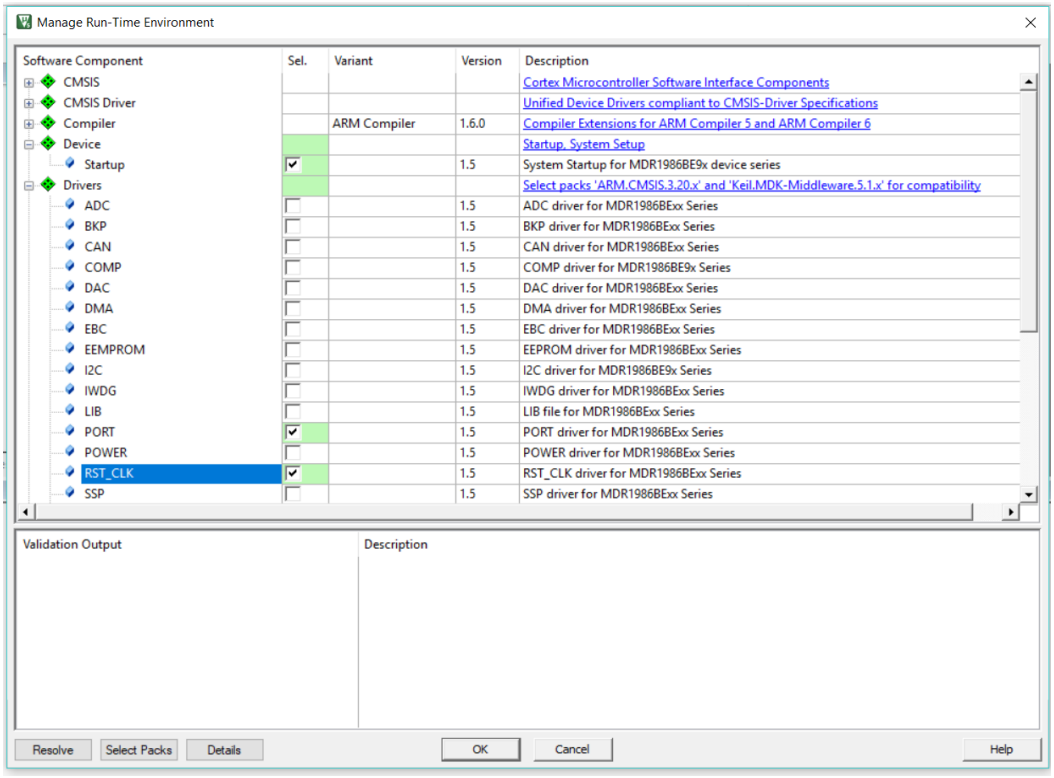

Рис. 2. Окно конфигурации проекта

В рабочем окне программы откройте окно конфигурирования свойств проекта с помощью **пиктограммы** на панели инструментов, как показано на рис. 3.

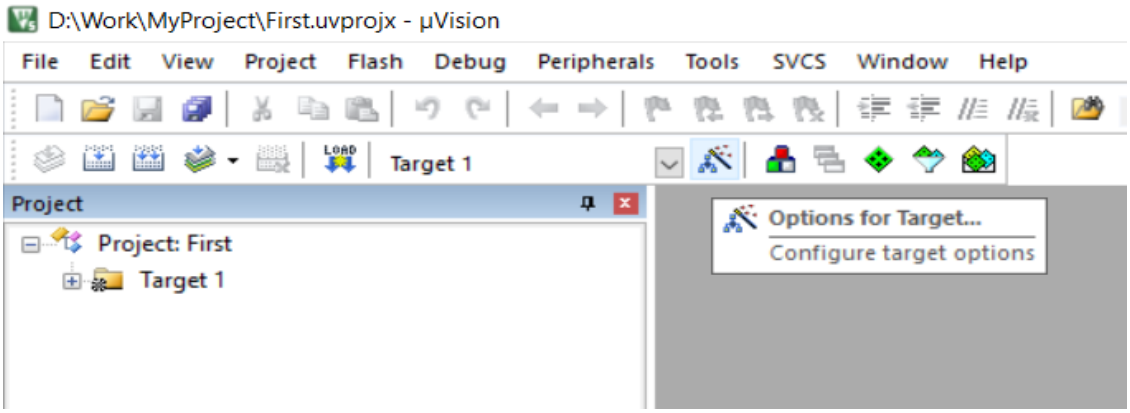

Рис. 3. Вызов окна настройки свойств проекта

В открывшемся диалоговом окне во вкладке **Target** установите частоту тактового генератора 8 МГц, как показано на рис. 4. Эта величина соответствует частоте встроенного тактового генератора HSI, который включается по умолчанию после сброса микроконтроллера [1].

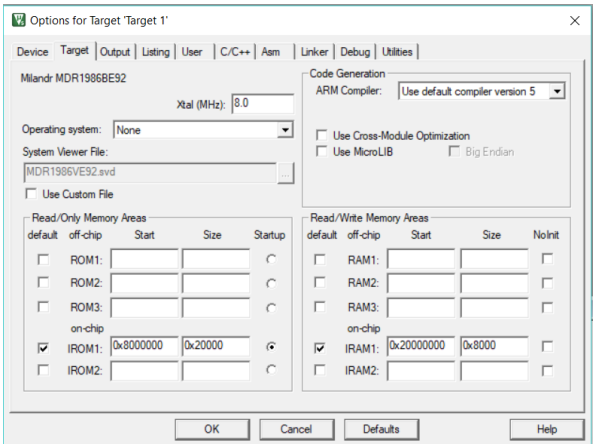

Рис. 4. Окно конфигурации частоты тактового генератора и областей памяти

Как видно на рисунке, по умолчанию система памяти конфигурируется в соответствии с физически реализованными ЗУ микроконтроллера, т.е. область 0x08000000 – 0x0801FFFF отводится для размещения кода программы, а область 0x20000000 – 0x2007FFFF – для размещения данных [1].

На вкладке **Output** отметьте галочкой пункт **Create HEX File**, как показано на рис. 5.

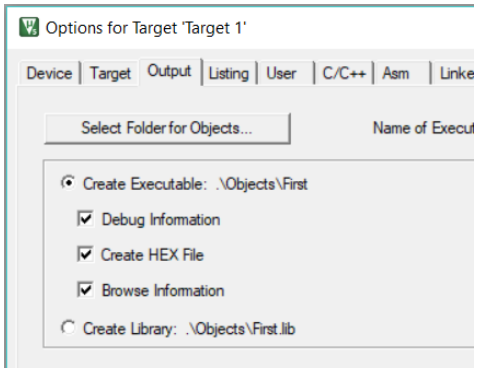

Рис. 5. Создание HEX файла

Во вкладке **Linker** отметьте блок **Use Memory Layout from Target Dialog**, как показано на рис. 6.

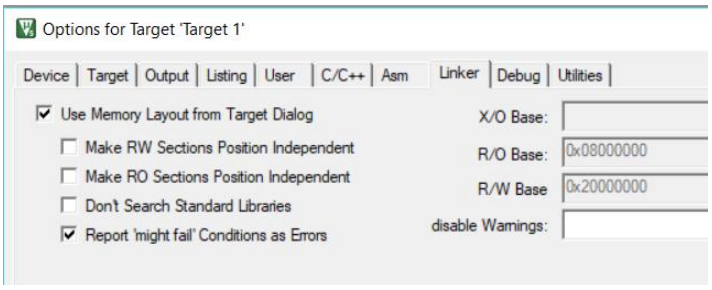

Рис. 6. Вкладка конфигурации компоновщика

Во вкладке **Debug** выберите симулятор (**Use Simulator**) для программной отладки проекта, как показано на рис. 7. Следует отметить, что программная симуляция имеет существенные ограничения и финальная отладка проекта чаще всего проводится с использованием аппаратного отладчика, однако программная симуляция имеет то очевидное преимущество, что не требует внешних подключений. Нажмите **ОК** для сохранения сделанных настроек.

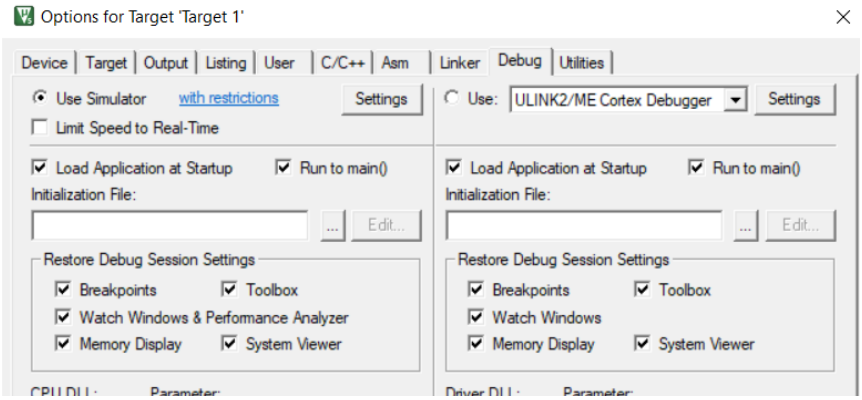

Рис. 7. Конфигурация программного отладчика

В качестве первой отлаживаемой программы воспользуемся рекомендациями, приведенными в [2]. Текст основной программы приведен ниже. Программа представляет собой использование порта ввода-вывода для управления светодиодом. Вывод **PORT_Pin_0** устанавливается и сбрасывается в бесконечном цикле **while (1)**. Длительность фаз включения и выключения светодиода определяется функцией **DELAY**. Такой способ задания длительности, конечно, имеет свои недостатки, однако в данном случае он позволяет избежать применения таймеров и сосредоточиться на основах разработки программного обеспечения с использованием среды **µVision**.

```
#include <MDR32F9Qx_port.h>
#include <MDR32F9Qx_rst_clk.h>
#define DELAY(T) for (i = T; i > 0; i--)
int i;

int main()
{
       RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC, ENABLE);
    PORT_InitTypeDef Nastroyka;
```

```
    Nastroyka.PORT_Pin = PORT_Pin_0;
    Nastroyka.PORT_OE = PORT_OE_OUT;
    Nastroyka.PORT_FUNC = PORT_FUNC_PORT;
    Nastroyka.PORT_MODE = PORT_MODE_DIGITAL;
    Nastroyka.PORT_SPEED = PORT_SPEED_SLOW;
    PORT_Init (MDR_PORTC, &Nastroyka);

    while (1)
    {
        PORT_SetBits(MDR_PORTC, PORT_Pin_0);
        DELAY(100000);
        PORT_ResetBits(MDR_PORTC, PORT_Pin_0);
        DELAY(100000);
    }
}
```

Управление портами ввода-вывода и их настройка подробно описаны в [1], здесь отметим, что функции **RST_CLK_PCLKcmd**, **PORT_SetBits**, **PORT_ResetBits** реализованы как элементы библиотеки **Standard Peripherals Library**, разработанной компанией «Миландр» и доступной для скачивания [3]. В архиве, кроме исходных кодов, имеется файл помощи, содержащий подробное описание всех используемых функций, структур и типов. Коды функций и соответствующие заголовочные файлы автоматически подключаются в среде **µVision**, если подключен пакет (**Pack**) для соответствующего семейства микроконтроллеров. Подключение необходимых пакетов может быть сделано на любом этапе с использованием соответствующего диалога, вызываемого так, как показано на рис. 8. В нашем случае необходимый пакет от производителя (компания «Миландр») уже установлен, так что дополнительные действия не требуются. Для удобства разработчиков существует удобная утилита **Milandr_PLL v2.8**, краткое описание которой приведено в Приложении А. При ее использовании автоматически генерируется код для вставки в разрабатываемую программу.

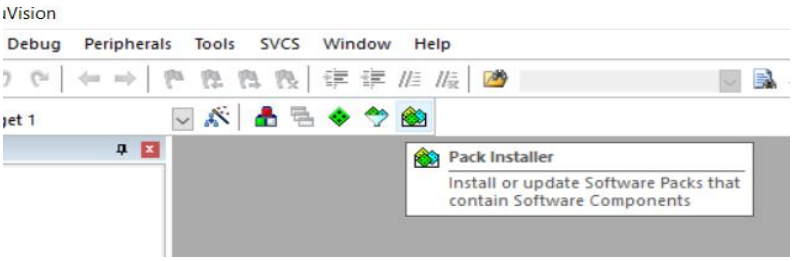

Рис. 8. Вызов установщика программных пакетов

Для создания файла с программой раскройте папку **Target1** и выберите пункт **Add New Item to Group ‘Source Group 1’**, как показано на рис. 9. Далее выберите тип файла **.c** и создайте новый файл, как показано на рис. 10.

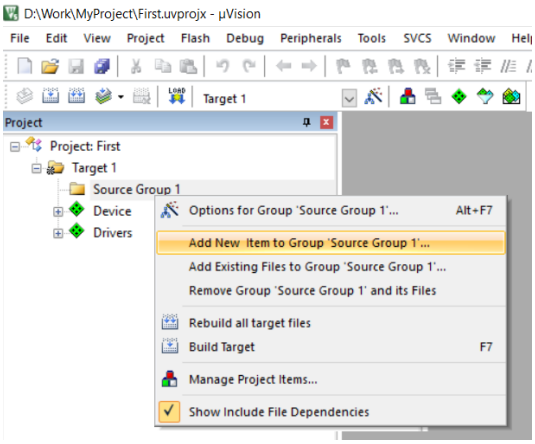

Рис. 9. Добавление файла к проекту

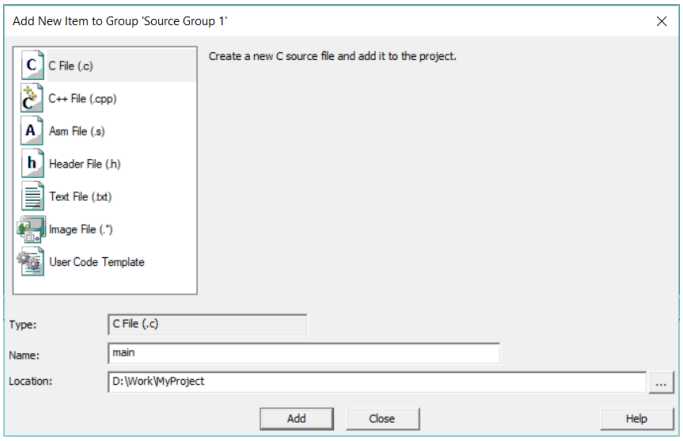

Рис. 10. Создание нового файла

Скопируйте вышеприведенный код программы в созданный файл. Проследите, чтобы после последней фигурной скобки был добавлен перевод строки. Откомпилируйте проект, выбрав опцию **Rebuild all target files**, как показано на рис. 11. Процесс и результаты компиляции будут отображаться в окне Build Output. При возникновении ошибок проверьте правильность исходного кода в соответствии с вышеприведенным образцом.

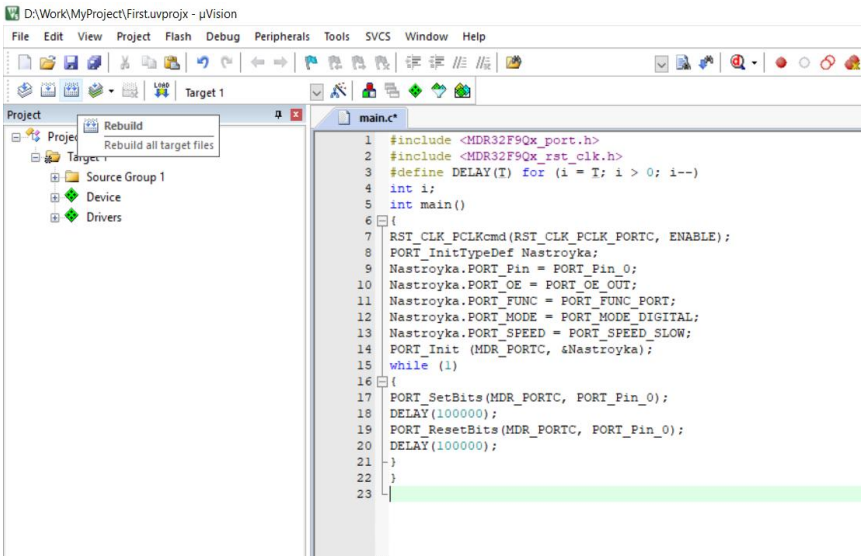

Рис. 11. Компиляция проекта

2 ОТЛАДКА ПРОЕКТА

2.1 Работа с симулятором

Запустите отладочную сессию с помощью меню **Debug – Start/Stop Debug session** или пиктограммы на панели инструментов, как показано на рис. 12.

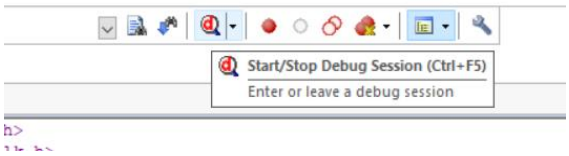

Рис. 12. Запуск сессии отладчика

В открывшемся окне можно видеть одновременно исходный и дизассемблированный код, содержимое памяти, регистров и много другой полезной информации. В окне **Command** отображаются сообщения, в командной строке можно вводить команды и запускать файлы сценариев. Сейчас в командном окне отображается следующий текст:

```
*** error 65: access violation at 0x4002001C : no
'write' permission
```

Причина этого сообщения заключается в том, что стартовый код нашего проекта обратился по адресу, для которого не задано разрешение на запись. Воспользуйтесь меню **Debug/Memory Map** и в открывшемся окне вручную задайте разрешения на чтение и запись, как показано на рис. 13, и нажмите кнопку **Map Range**, после чего закройте окно. С описанием карты памяти **К1986ВЕ92QI** можно подробно ознакомиться в [1].

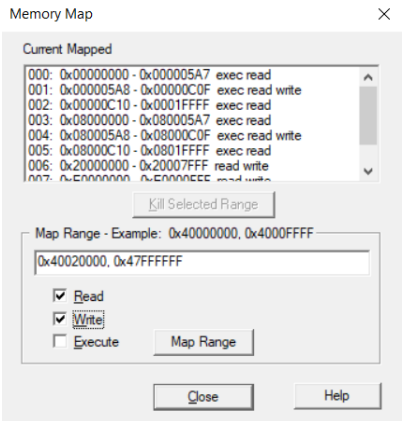

Рис. 13. Окно карты памяти

В правой части экрана отображается окно стека вызываемых функций (рис. 14). Из этого окна можно видеть, что в данный момент процессор остановлен внутри функции **SystemInit()**, которая, в свою очередь, вызвана из обработчика события **Reset**(**Reset_Handler**). В центральном окне и окне дизассемблера можно видеть соответственно строку файла исходного кода и инструкцию процессора, на которой сейчас остановилась отладка.

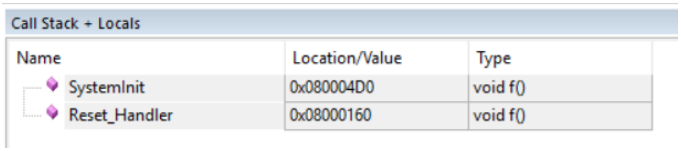

Рис. 14. Стек вызываемых функций

Прежде чем приступить собственно к выполнению программы, полезно выяснить, как процессор перешел к текущей инструкции. **К1986ВЕ92QI** построен на ядре **Cortex-M3** и действует по алгоритму, описанному, например, в [4]. После сброса процессор обращается по фиксированному адресу **0x00000000**, загружает оттуда указатель стека **SP** и из следующего двойного слова получает адрес обработчика события **RESET**. Для проверки выберите вкладку **Memory1** (меню **View / Memory Windows / Memory1**) и укажите начальный адрес, как показано на рис. 15.

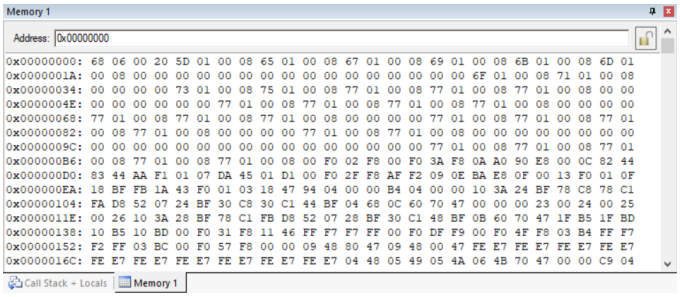

Рис. 15. Дамп памяти

Из рисунка видно, что указатель стека настроен на адрес **0x20000668**, а обработчик события **сброс** расположен по адресу **0x0800015C** (младший бит адреса в таблице векторов должен быть установлен, указывая на команду **Thumb**). Нажмите кнопку **сброса** (RST) и проверьте содержимое регистров **PC** и **SP** в окне **Registers** в левой части рабочего экрана. Итак, наша программа размещена в па-

мяти программ контроллера, начиная с адреса 0x0800015C, а таблица векторов прерываний – начиная с адреса 0x00000000, в соответствии с классическим описанием ядра **Cortex-M3**. Файл **startup_MDR32F9Qx.s** содержит описание необходимых базовых настроек и таблицы векторов прерывания. В качестве упражнения рекомендуется изучить его содержимое.

Используя клавиши **F5**, **F10** и **F11** можно осуществить пошаговую отладку или запуск программы на выполнение. Также можно использовать кнопки-пиктограммы () на панели инструментов. Перейдите в окно исходного кода основной программы (**main.c**) и поставьте **точки останова** щелчком левой кнопки мыши слева от команды **DELAY(100000)**, как показано на рис. 16.

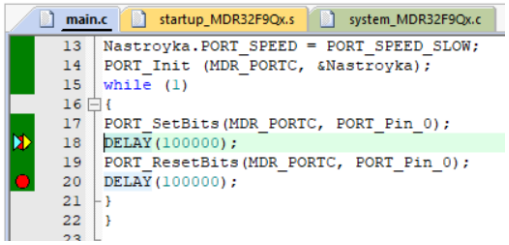

Рис. 16. Активные точки останова

Посмотреть изменение нужного вывода порта можно с использованием модуля **System Viewer**, включение которого показано на рис. 17.

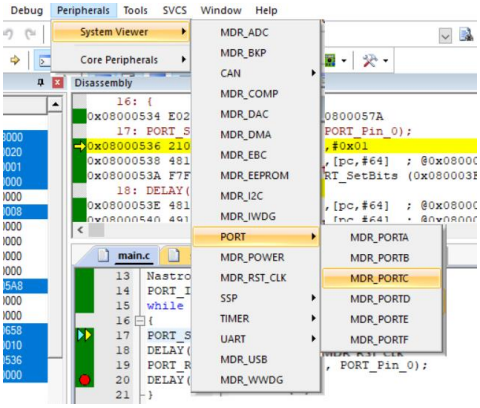

Рис. 17. Вызов System Viewer

В открывшемся окне состояние нужного бита отображается в регистре RXTX. Запустите программу на выполнение (**F5**) и наблюдайте изменения. Определите величину задержки, формируемую функцией **DELAY(1000000)**.

2.2 Работа с аппаратным отладчиком

Для перехода к работе с аппаратным отладчиком необходимо подготовить **отладочную плату**. Описание отладочной платы и ее ресурсов содержится в [2] и [5]. Проверьте **наличие и установку перемычек**, как показано на рис. 18, и подключите соединитель адаптера к разъему **JTAG-B**.

ВНИМАНИЕ!!! Все коммутации на плате должны осуществляться при отключенном питании!

Рис. 18. Отладочная плата микроконтроллера К1986ВЕ92QI

Присоедините кабель USB программатора к компьютеру и включите внешний источник питания.

Если в памяти контроллера имеется какая-то программа, она начнет выполняться.

Для настройки аппаратного отладчика убедитесь в том, что **текущая сессия отладки** завершена, и перейдите к настройке свойств проекта. В окне, показанном на рис. 7, выберите аппаратный отладчик, в выпадающем окне выберите **J-LINK/J-TRACE CORTEX** и нажмите кнопку **Settings**.

В окне настроек аппаратного отладчика установите настройки, как показано на рис. 19, в том числе укажите **Unspecified Cortex-M3** в качестве устройства назначения, если система попросит сделать это. Обратите внимание, что при правильно установленном драйвере адаптера и надежном соединении в окне SN должен высветиться номер версии **ПО адаптера**. Если на этом этапе возникают проблемы, необходимо обратиться к преподавателю.

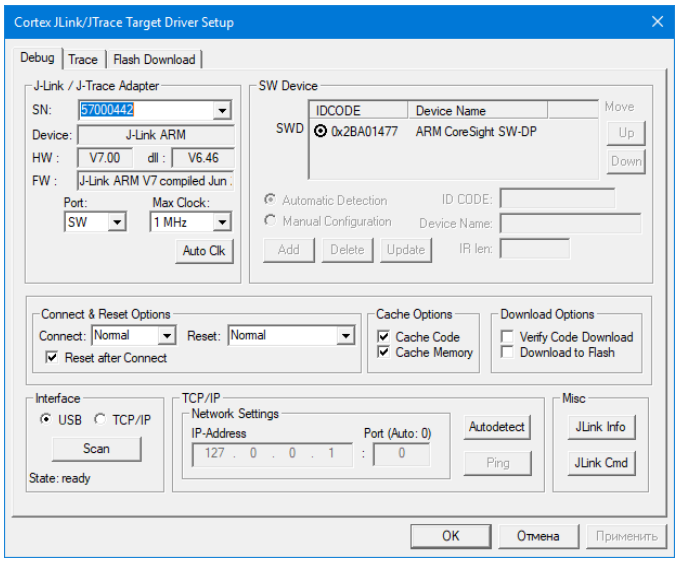

Рис. 19. Окно настроек аппаратного отладчика

Выберите вкладку **Flash Download** и установите настройки программирования FLASH-памяти, как показано на рис. 20.

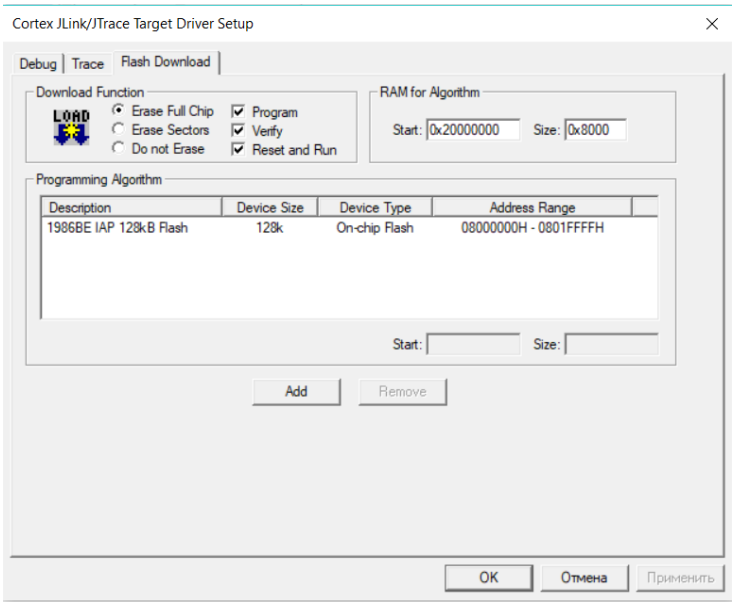

Рис. 20. Настройка алгоритма программирования FLASH-памяти

Завершите настройки и запустите сессию отладчика. Если все сделано правильно, отладчик произведет полное стирание FLASH-памяти микроконтроллера и остановится на точке входа программы. Нажмите кнопку сброса (). Сравните значения указателя стека и счетчика команд с теми, которые наблюдались в сессии симуляции. Откройте окно дампа памяти по адресу **0x00000000** и сравните с предыдущим случаем. Несмотря на то, что мы ничего не меняли в коде программы, на этот раз обработчик события **сброс** находится по адресу **0x000000A6**. Причина этого заключается в том, что область памяти, начиная с адреса **0x00000000** в микроконтроллере **K1986BE92QI**, относится к **BOOTROM**, запрограммированном на заводе-изготовителе. Функция **BOOTROM** заключается в управлении загрузкой при различной аппаратной конфигурации (см. положение переключателей **SA2-SA4** на отладочной плате). Таким образом, содержимое основной таблицы векторов прерывания не может быть изменено. С другой стороны, алгоритм программирования физически ограничен адресом **0x08000000** (см. рис. 20).

Решение этой проблемы заключено в коде загрузчика, который записывает в регистр **VTOR** значение **0x08000000**, смещающее начало таблицы векторов прерывания точно туда, где она располагается после программирования FLASH-памяти, модифицирует указатель стека, вычисляет адрес обработчика события **сброс** в новой таблице векторов прерывания и передает ему управление. После этого все события обрабатываются ядром в соответствии с новой таблицей векторов прерывания, за исключением события **сброс**, которое обрабатывается так, как показано выше.

Запустите программу на выполнение и убедитесь в том, что светодиод **VD3** на отладочной плате переключается в соответствии с отлаживаемой программой.

Согласно [1], код программы рекомендуется размещать в блоке FLASH-памяти, чтобы обеспечить одновременную работу шин адреса и данных. Вместе с тем FLASH-память не обеспечивает высокой скорости доступа, что может являться ограничением при работе на высокой частоте. Эта проблема решена разработчиками **K1986BE92QI** пу-

тем организации считывания нескольких слов памяти в течение одной операции доступа. Тем не менее иногда есть необходимость выполнения кода программы из **ОЗУ**, например при реализации программ-загрузчиков FLASH-памяти. Иногда удобно иметь две копии программы (во FLASH-памяти и ОЗУ) для сравнения и поиска наилучшего решения. Также следует помнить об ограничении количества циклов записи-стирания FLASH-памяти.

Проведем эксперимент, разместив измененную программу в ОЗУ. Для этого завершите сеанс отладчика и измените вызов подпрограммы задержки на **DELAY(10000)**. Код программы вместе с таблицей векторов прерывания разместим в ОЗУ, начиная с адреса **0x20000000**. В окне настроек областей памяти установите значения, как показано на рис. 21.

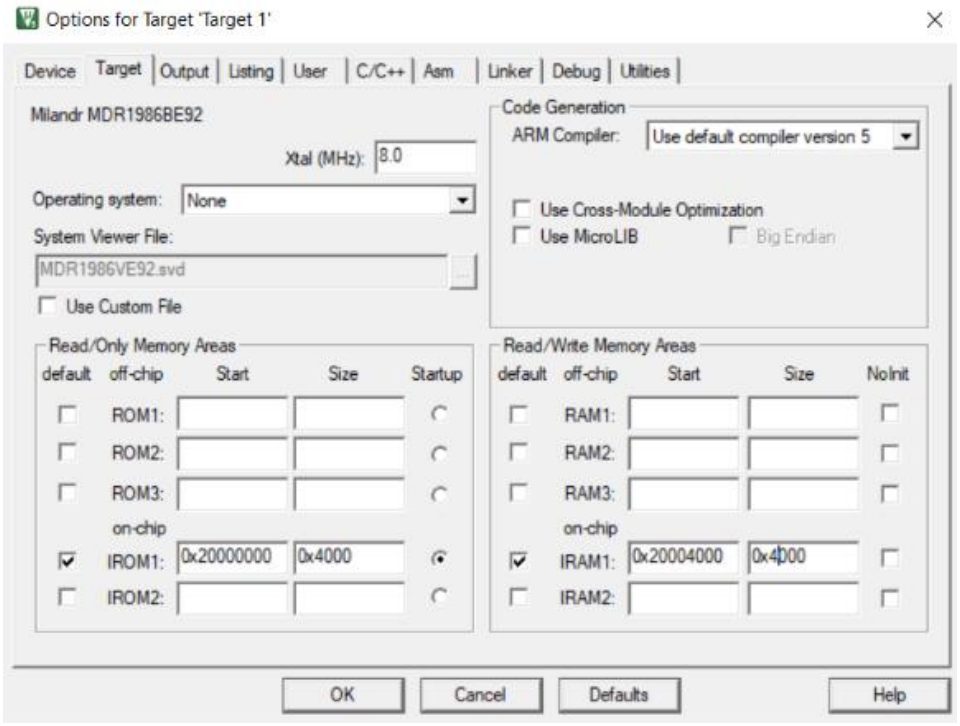

Рис. 21. Настройки проекта для размещения кода в ОЗУ

Для изменения размещения таблицы векторов прерывания откройте окно **настроек** модуля стартового кода, как показано на рис. 22, и в диалоговом окне выберите вкладку **Memory**. В настройках **Code/Const** выберите область **IROM1**, как показано на рис. 23.

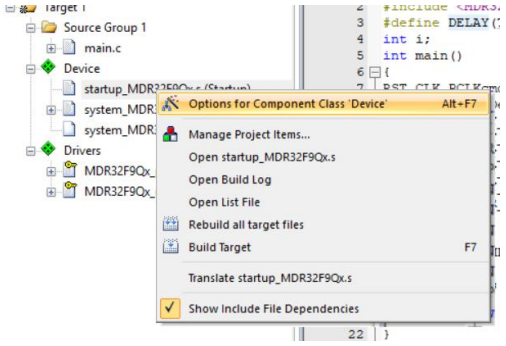

Рис. 22. Настройки загрузчика

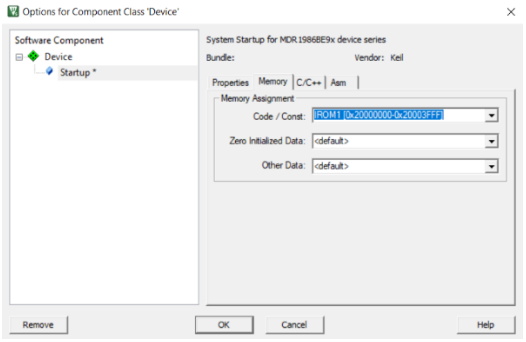

Рис. 23. Настройки области размещения кода

Так как сейчас не требуется программировать FLASH-память, измените настройки аппаратного отладчика, как показано на рис. 24. Поскольку мы не изменяем содержимого FLASH-памяти, желательно остановить выполнение сразу после загрузки. Для этого нужно снять галочку в блоке **Run to main()** в диалоге настройки аппаратного отладчика (окно на рис. 7).

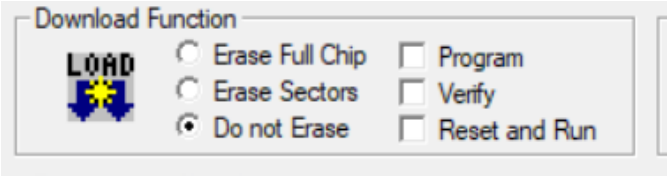

Рис. 24. Отключение программирования FLASH-памяти

Откомпилируйте исправленный код и запустите отладчик. Вы увидите, что отладка остановилась на адресе **0x000000A6**, что соответствует блоку загрузчика. Откройте дамп памяти с адреса **0x20000000** и убедитесь в том, что таблица векторов прерывания скопирована. Вручную в окне **Registers** измените содержимое указателя стека на **0x20004668** (значение из таблицы векторов прерывания). Если мы хотим в дальнейшем использовать прерывания, необходимо также модифицировать регистр **VTOR**, содержащий смещение начального адреса таблицы векторов прерывания. Для этого откройте окно настройки контроллера прерывания (**Peripherals/Core Peripherals/Nested Vectored Interrupt Controller (NVIC)**) и измените адрес **SCB->VTOR** на **0x20000000**. Теперь необходимо изменить значение счетчика команд на **0x2000015C** (адрес обработчика сброса из табли-

цы векторов прерывания). Для этого воспользуемся командной строкой и введем туда **PC=0x2000015C**. Нажмите **ENTER** и убедитесь в наличии сделанных изменений. Сейчас мы проделали это вручную в отладочных целях. Если же необходимо выполнять данные операции в автономном режиме, придется изменить код в файле **startup_MDR32F9Qx.s**. Запустите программу на выполнение (**F5**). Сравните частоту мигания **VD3** на отладочной плате с предыдущим экспериментом. Если сейчас нажать кнопку **сброс**, программа вернется к исходному варианту, зашитому во FLASH-памяти. Проверьте это утверждение.

2.3 Управление системой тактирования микроконтроллера

После сброса микроконтроллер автоматически переходит на тактирование от встроенного тактового генератора HSI, который не обеспечивает высокой стабильности, однако его частоту можно подстраивать [1]. Для многих практических задач необходима более высокая стабильность частоты, кроме того К1986ВЕ92QI способен работать при частоте 80 МГц. Для тактирования таким сигналом необходима активация генератора HSE, синхронизируемого внешним тактовым генератором, и настройка системы ФАПЧ (PLL). Подробно с настройками системы тактирования можно ознакомиться в [1, 5].

Добавьте в файл main.c перед строкой

```
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC, ENABLE);
```

следующий код:

```
    RST_CLK_HSEconfig (RST_CLK_HSE_ON);    //    Включение
HSE
    while (RST_CLK_HSEstatus () != SUCCESS);       //
Ожидание включения
    RST_CLK_CPU_PLLconfig (RST_CLK_CPU_PLLsrcHSEdiv1,
                    RST_CLK_CPU_PLLmul10);
                //Коэффициент умножения - 10
    RST_CLK_CPU_PLLcmd (ENABLE);        //Включение ФАПЧ
    while  (RST_CLK_CPU_PLLstatus  ()  !=  SUCCESS);  //
Ожидание стабилизации ФАПЧ
```

```
    RST_CLK_CPUclkPrescaler (RST_CLK_CPUclkDIV1); //
Деление частоты на 1
    RST_CLK_CPU_PLLuse (ENABLE);// Подключение сигнала с
вых. модуля ФАПЧ
    RST_CLK_CPUclkSelection        (RST_CLK_CPUclkCPU_C3);
   //Подача тактового сигнала на процессор
```

Восстановите значение задержки в вызове функции **DELAY** – **DELAY(100000)**, откомпилируйте программу и запустите сессию отладки, как это было показано выше с размещением программы в ОЗУ. Запустите программу на выполнение и сравните частоту мигания светодиода с изначальной. Посмотрите настройки модуля управления тактированием с помощью меню **Peripheral/System Viewer/MDR_RST_CLK**.

Обратите внимание, что, согласно [1] (табл. 21), для нормального функционирования FLASH-памяти на высоких частотах необходимо устанавливать паузу в регистре **EEPROM_CMD** (биты **DELAY[2:0]**). В нашем случае программа размещалась в ОЗУ, FLASH-память не использовалась, и пауза не требовалась. Утилита **Milandr_PLL v2.8** генерирует соответствующую команду автоматически.

3 ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. Изучите систему цифрового ввода и напишите программу, которая включает и выключает светодиод при нажатии кнопки LEFT.

2. Создайте подпрограмму настройки системы тактирования вида SetClock(int F), где F – частота в мегагерцах.

3. Используя утилиту Milandr_PLL, сформируйте код настройки тактовой частоты ядра и порта C и убедитесь в его эквивалентности ранее протестированному коду.

4. Изучите систему тактирования микроконтроллера и попробуйте настроить тактирование процессора от LSI.

5. С помощью симулятора осуществите пошаговый анализ файла startup_MDR32F9Qx.s.

6. Пользуясь отладчиком, сравните быстродействие процессорного ядра при размещении кода программы во FLASH-памяти и в ОЗУ (используя коды, рассмотренные выше).

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите последовательность работы микроконтроллера после включения питания.
2. Опишите систему тактирования микроконтроллера.
3. Опишите технологию настройки портов ввода-вывода и назначение отдельных битов.
4. Каково назначение встроенного загрузчика (BOOTROM)?
5. Что такое CMSIS?
6. Для чего используется таблица векторов прерывания? Где она размещается? Какова ее структура?
7. Опишите структуру стартового файла startup_MDR32F9Qx.s. Каково его назначение?
8. Каковы особенности системы команд Cortex-M3?
9. Опишите структуру системы памяти Cortex-M3.
10. Каковы основные особенности микроконтроллеров Cortex-M3?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Спецификация микросхем серии 1986BE9xy, K1986BE9xy, K1986BE9xyK, K1986BE92QI, K1986BE92QC, 1986BE91H4, K1986BE91H4, 1986BE94H4, K1986BE94H4. – Москва: Миландр, 2015. – 521 с.
2. Евдокимов, А.П. Программирование микроконтроллера K1986BE92QI компании «Миландр» / А.П. Евдокимов, Л.Л. Владимиров. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2018. – 76 с.
3. K1986BE92QI // Официальный сайт компании Миландр: [сайт]. – 2019. – URL: https://ic.milandr.ru/products/mikrokontrollery_i_protsessory/k1986ve92qi/ (дата обращения: 15.10.2019).

4. Cortex-M3 Devices Generic User Guide // Официальный сайт ARM: [сайт]. – 2019. – URL: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0552a/index.html (дата обращения: 15.10.2019).

5. Благодаров А.В., Владимиров Л.Л. Программирование микроконтроллеров на основе отечественных микросхем семейства 1986ВЕ9х разработки и производства компании «Миландр» / А.В. Благодаров, Л.Л. Владимиров. – Москва: Миландр, 2016. – 242 с.

Приложение А
Утилита настройки контроллеров семейства 1986

Утилита настройки микроконтроллеров семейства 1986 запускается с помощью ярлыка  на рабочем столе (или из папки, указанной преподавателем). После ее открытия появляется рабочее окно, показанное на рис. А.1.

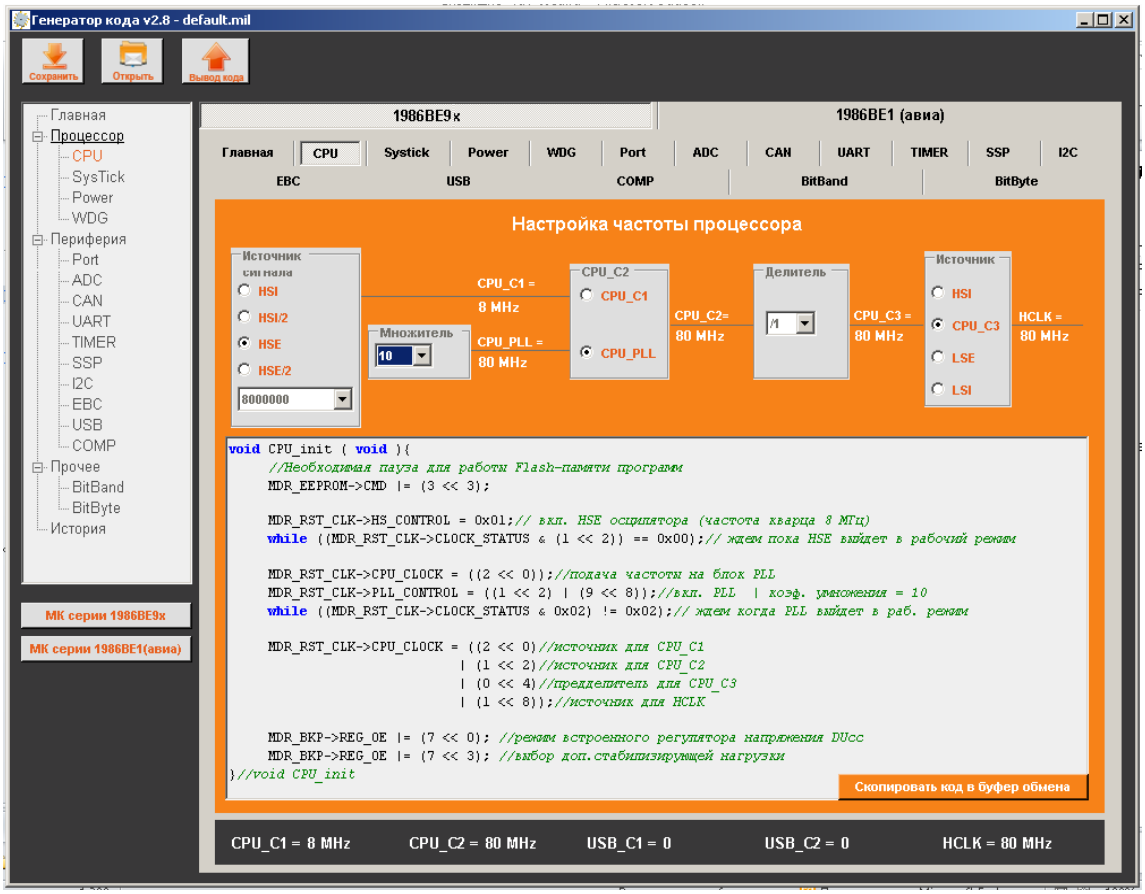

Рис. А.1. Рабочее окно утилиты настройки контроллеров семейства 1986

С помощью выбора необходимого настраиваемого модуля и нужных опций настройки программа генерирует Си-код, который можно вставить в программу либо путем копирования в буфер обмена, либо с помощью формирования отдельного файла – кнопка «Вывод кода». В последнем случае в файле аккумулируются все сделанные настройки для различных модулей и подсистем. Сделанные настройки можно сохранить в специальном формате программы, чтобы впоследствии иметь возможность их редактирования.

Методические материалы

РАЗРАБОТКА И ОТЛАДКА ПРОГРАММ ДЛЯ МИКРОКОНТРОЛЛЕРОВ С ЯДРОМ CORTEX-M3 В СРЕДЕ µVision

Методические указания

Составители: ***Кудрявцев Илья Александрович,***
Корнилин Дмитрий Владимирович
Мякинин Олег Олегович

Редактор *А.С. Никитина*
Компьютерная вёрстка *А.С. Никитиной*

Подписано в печать 10.08.2020. Формат 60x84 1/8.
Бумага офсетная. Печ. л. 3,0.
Тираж 25 экз. Заказ . Арт. – 49(Р1М)/2020.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)
443086, САМАРА, МОСКОВСКОЕ ШОССЕ, 34.

Издательство Самарского университета.
443086, Самара, Московское шоссе, 34.