НИИИС

МИКРОСХЕМЫ ИНТЕГРАЛЬНЫЕ
1914ВА018
Работа со средой Keil

Содержание

1. Установка среды Keil

Скачать MDK Keil 5 можно с официального сайта. (Процесс установки Keil 4 отличается не значительно)

После запуска исполняемого файла, появится окно приветствия установки. Необходимо нажать кнопку «Next»

Рисунок 1.1

Содержание следующего окна является лицензионным соглашением.

Необходимо установить галочку, которая подтверждает согласие с содержимым лицензионного соглашения, и нажать кнопку «Next»

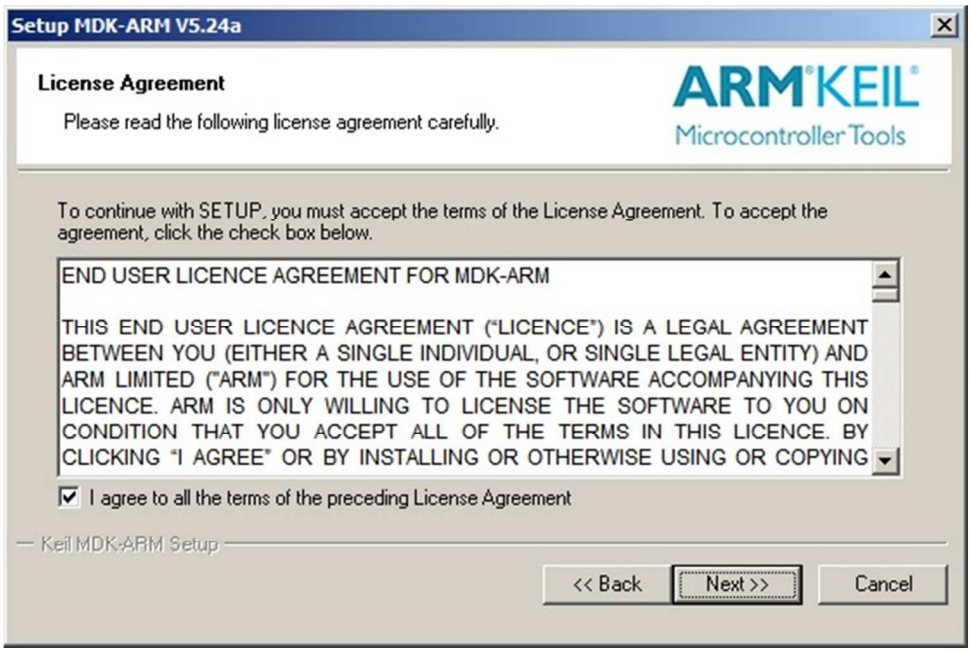


Рисунок 1.2

В следующем окне необходимо произвести выбор папки для установки среды Keil и расположение папки PACK. (PACK используется в среде Keil версии 5 и выше. Более подробно о данном инструменте будет далее.) И нажать кнопку «Next»

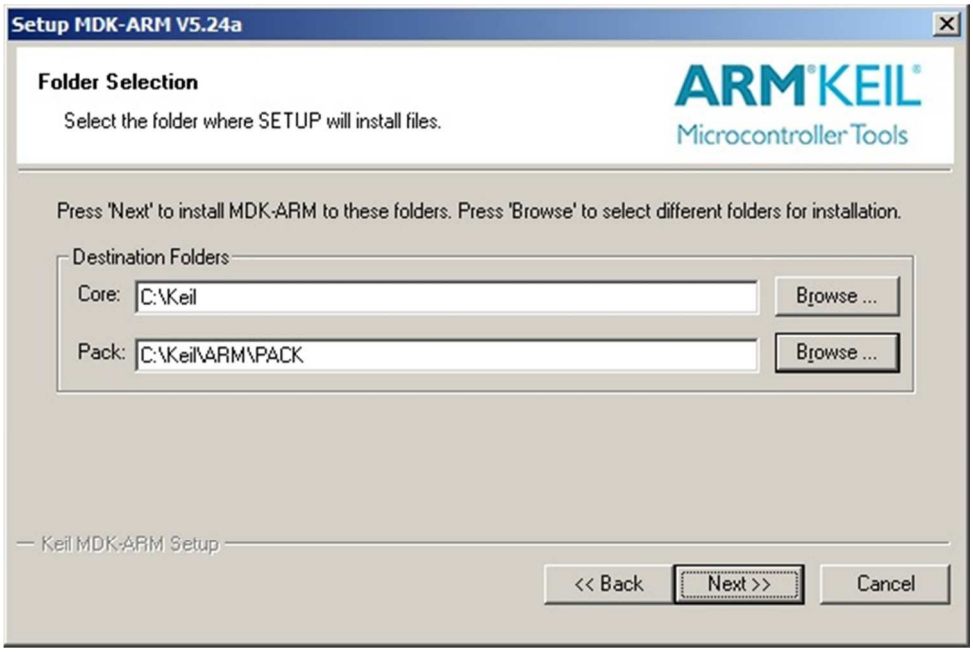


Рисунок 1.3

В данном окне необходимо указать личные данные и нажать кнопку «Next»

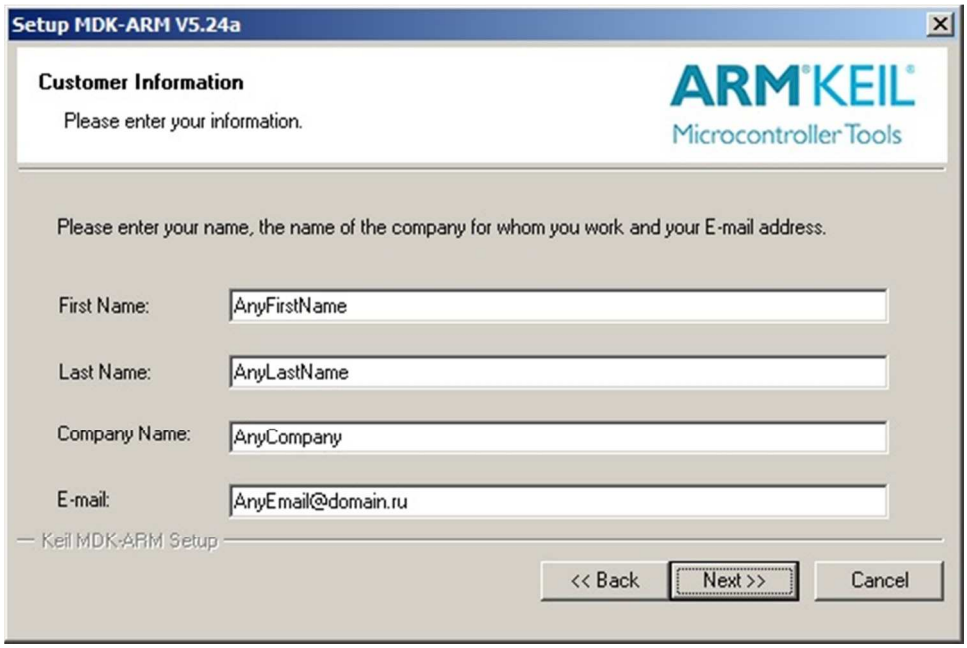


Рисунок 1.4

Начнется установка среды Keil.

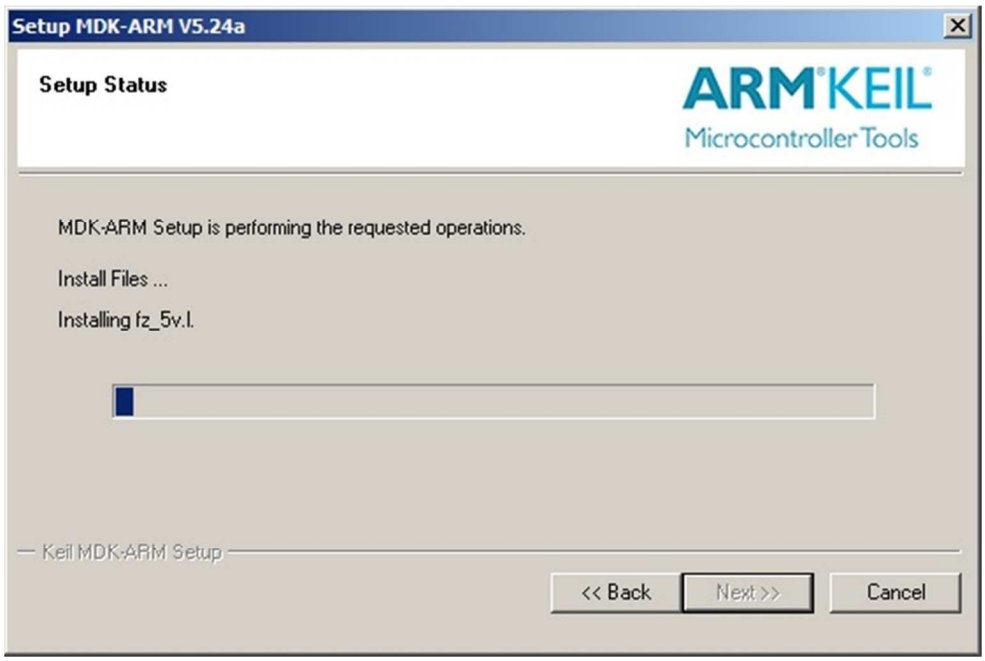


Рисунок 1.5

Когда прогресс статус-бара дойдет до конца, установка завершится.

Появится следующее окно:

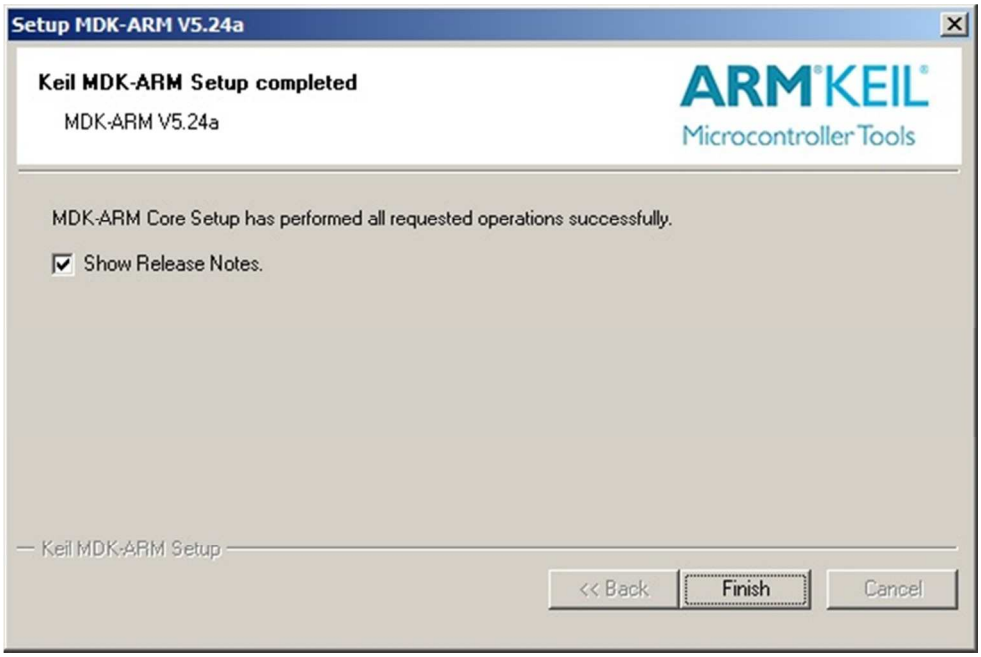


Рисунок 1.6

Наличие галочки в пункте «Show Release Notes» означает, что после нажатия кнопки «Finish» откроется текстовый файл с описанием изменений в релизах среды Keil.

Установка завершена.

2. Интеграция NIIIS в среду Keil (только для Keil 5)

Необходимо добавить поддержку семейства МК в среду Keil, для этого откроем файл ..\UV4\PACK.xsd в текстовом редакторе

В блоке <!-- Registered Device Vendors -->

Добавить строчку

<xs:enumeration value=”NIIIS:200”/>

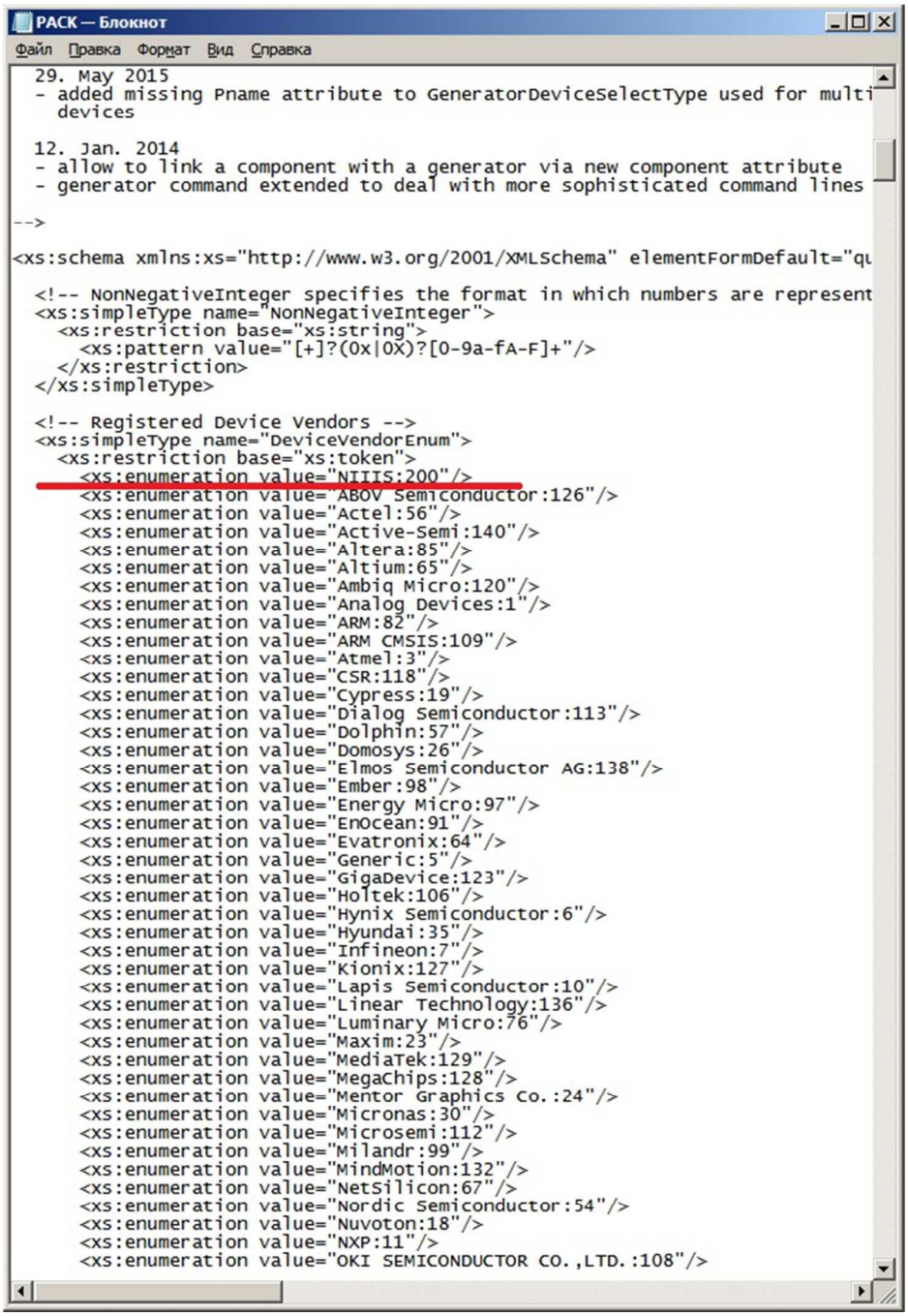


Рисунок 2.1

Необходимо сохранить редактируемый файл, и запустить среду Keil

Рисунок 2.2

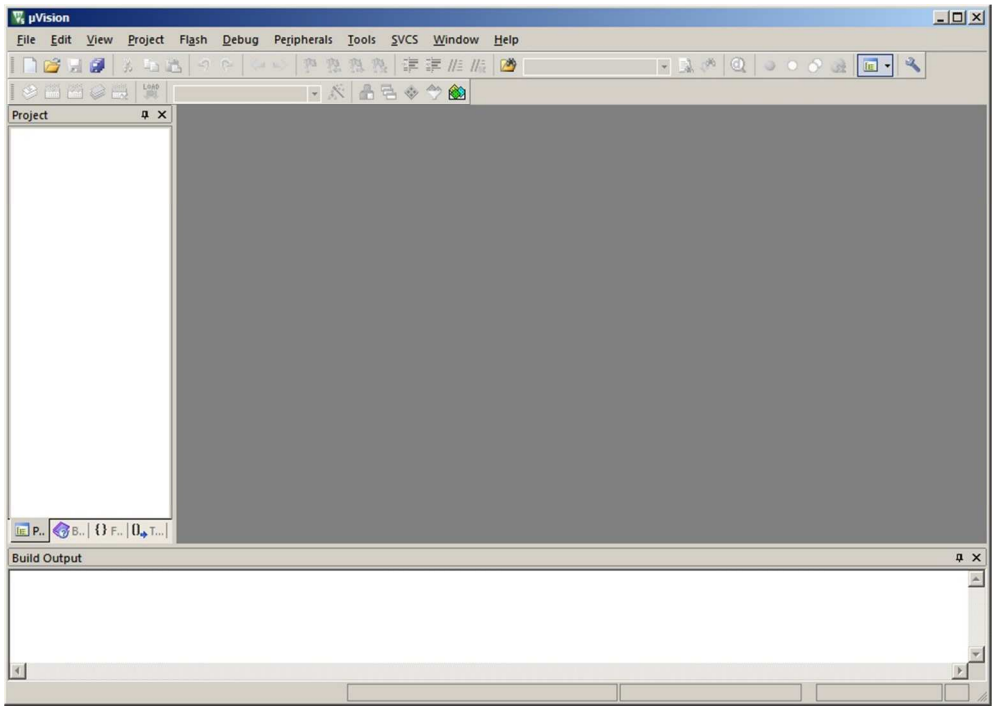


Рисунок 2.3

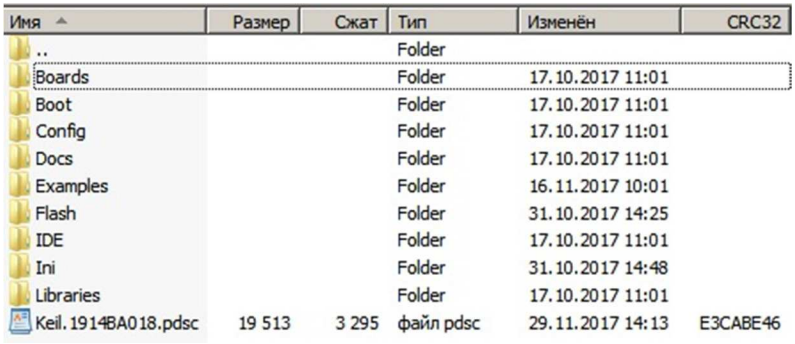
3. Установка Pack

3.1 Keil 4

Для удобства мы предлагаем поместить папку с файлами МК по адресу ../Keil/ARM/Device/

Далее необходимо создать папку 1914BA018

Необходимо извлечь из PACK-файла любым удобным архиватором содержимое в папку 1914BA018.



Имя	Размер	Сжат	Тип	Изменён	CRC32
..			Folder		
Boards			Folder	17.10.2017 11:01	
Boot			Folder	17.10.2017 11:01	
Config			Folder	17.10.2017 11:01	
Docs			Folder	17.10.2017 11:01	
Examples			Folder	16.11.2017 10:01	
Flash			Folder	31.10.2017 14:25	
IDE			Folder	17.10.2017 11:01	
Ini			Folder	31.10.2017 14:48	
Libraries			Folder	17.10.2017 11:01	
Keil.1914BA018.pdsc	19 513	3 295	файл pdsc	29.11.2017 14:13	E3CABE46

Рисунок 3.1.1

Для использования алгоритмов загрузки во внешнюю память, необходимо из папки ../Keil/ARM/Device/1914BA018/Flash скопировать *.flm файлы в папку ../Keil/ARM/Flash

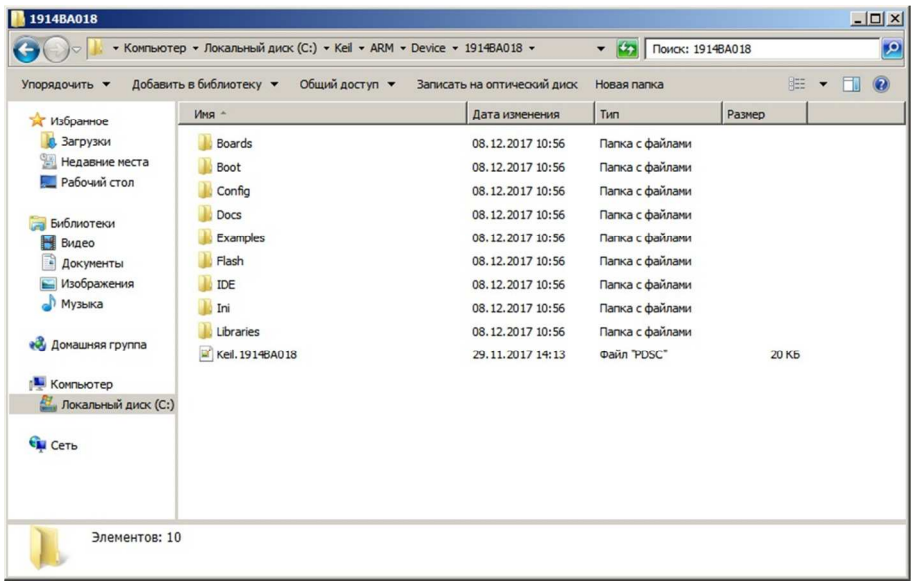


Рисунок 3.1.2

3.2 Keil 5

Необходимо открыть среду Keil

Нажать на иконку «Pack installer»

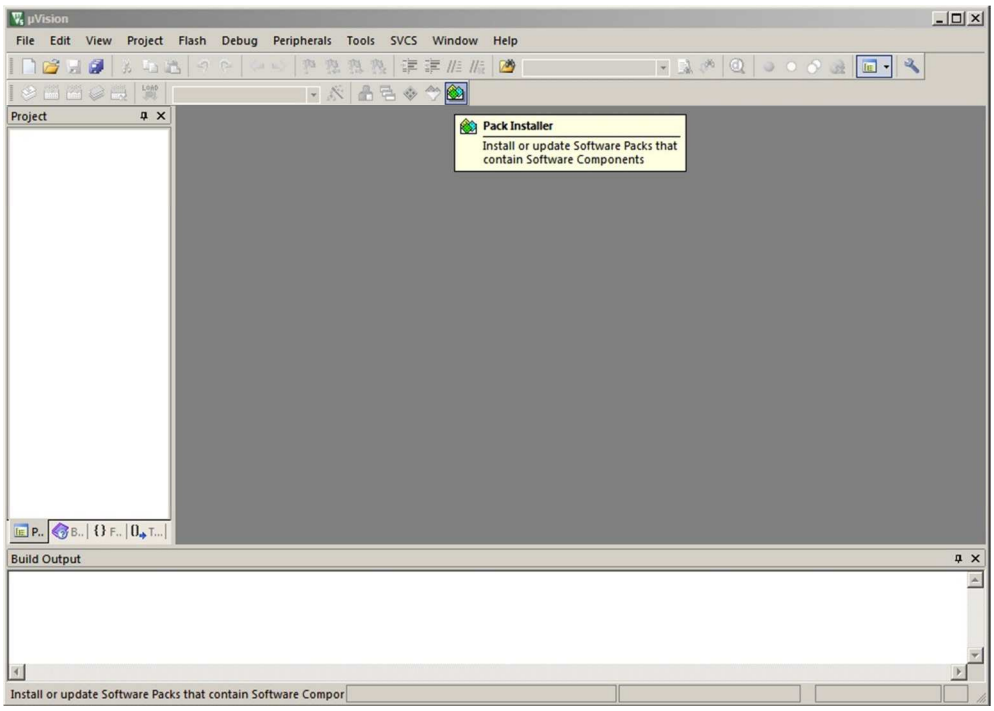


Рисунок 3.2.1

Появится окно приветствия Keil Pack Installer

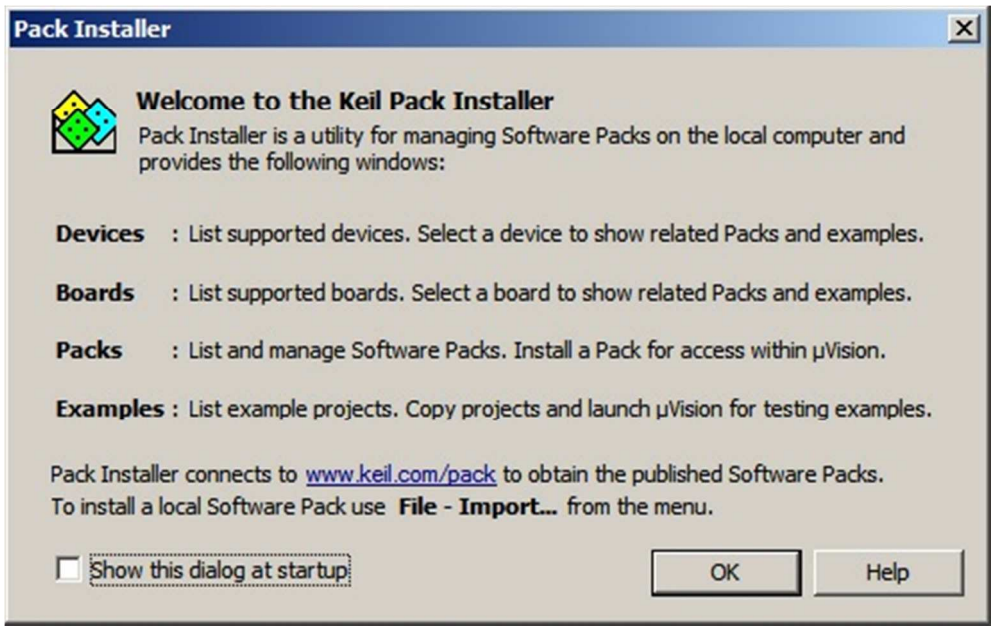


Рисунок 3.2.2

Можно убрать галочку «Show this dialog at startup»

Необходимо нажать кнопку «ОК»

Выбрать File->Import…

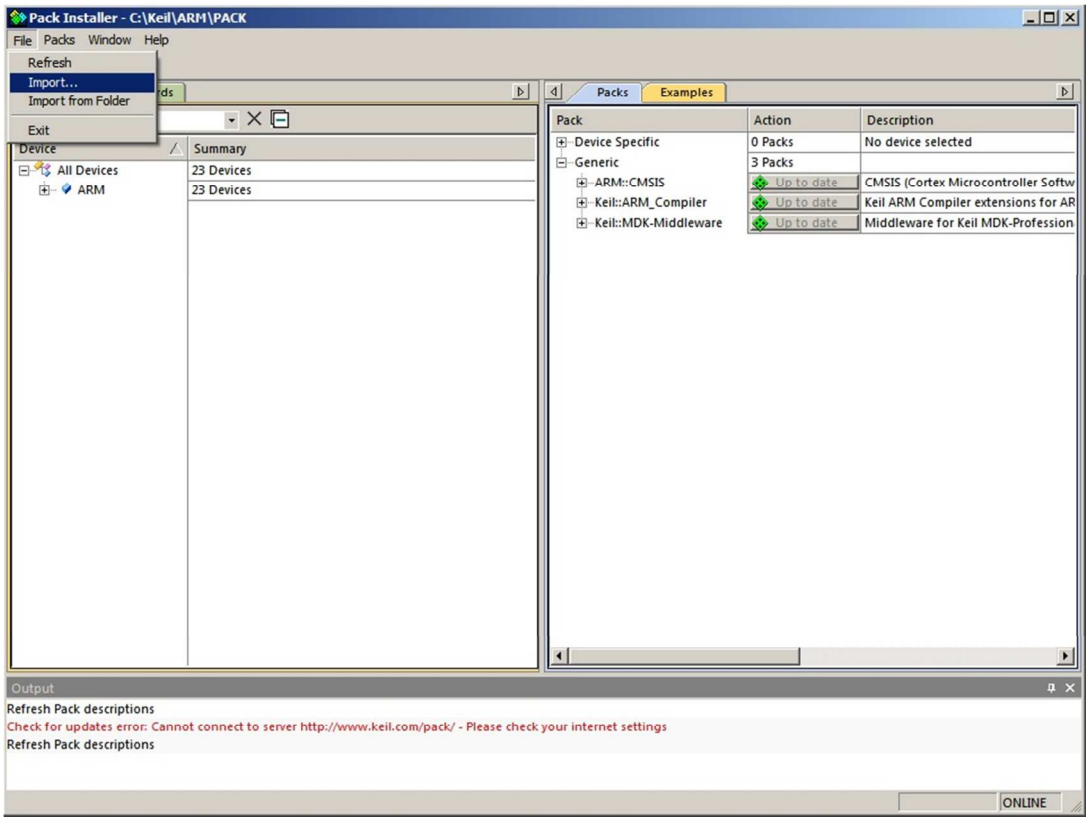


Рисунок 3.2.3

Выбрать файл Keil.1914BA018.pack и нажать кнопку «открыть»

В списке устройств появится МК 1914BA018 в разделе NIIIS

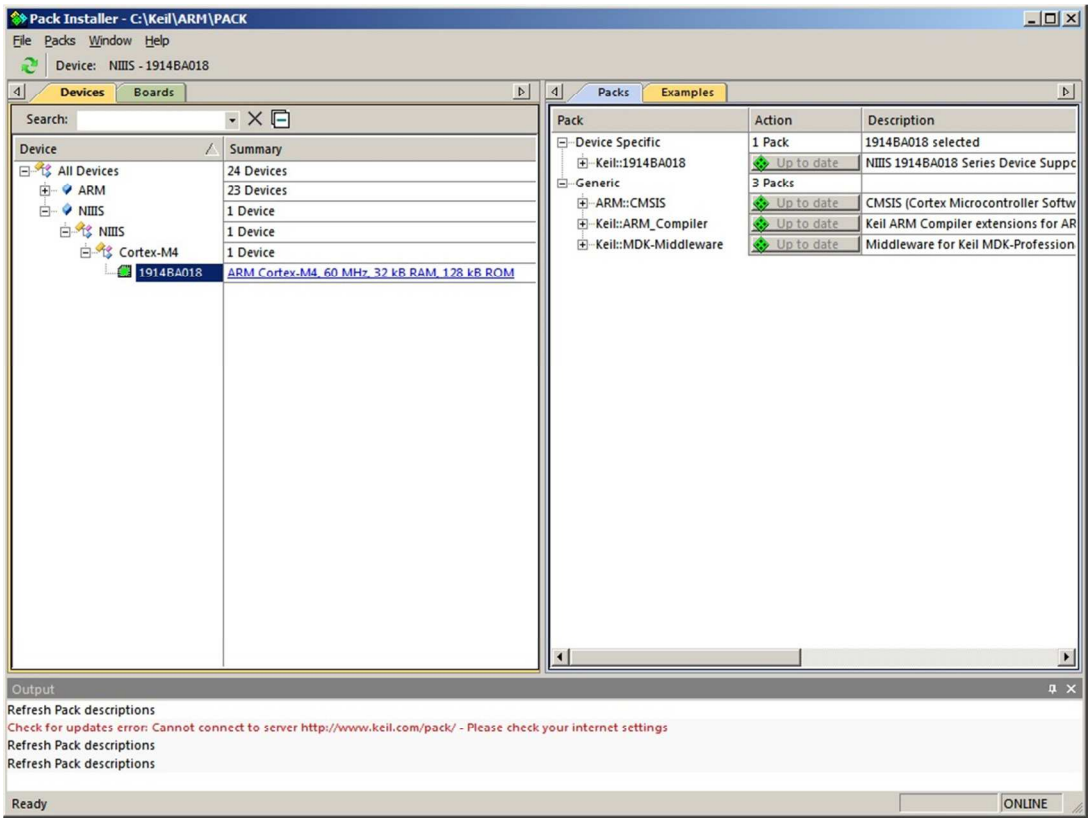


Рисунок 3.2.4

4. Основные средства Keil

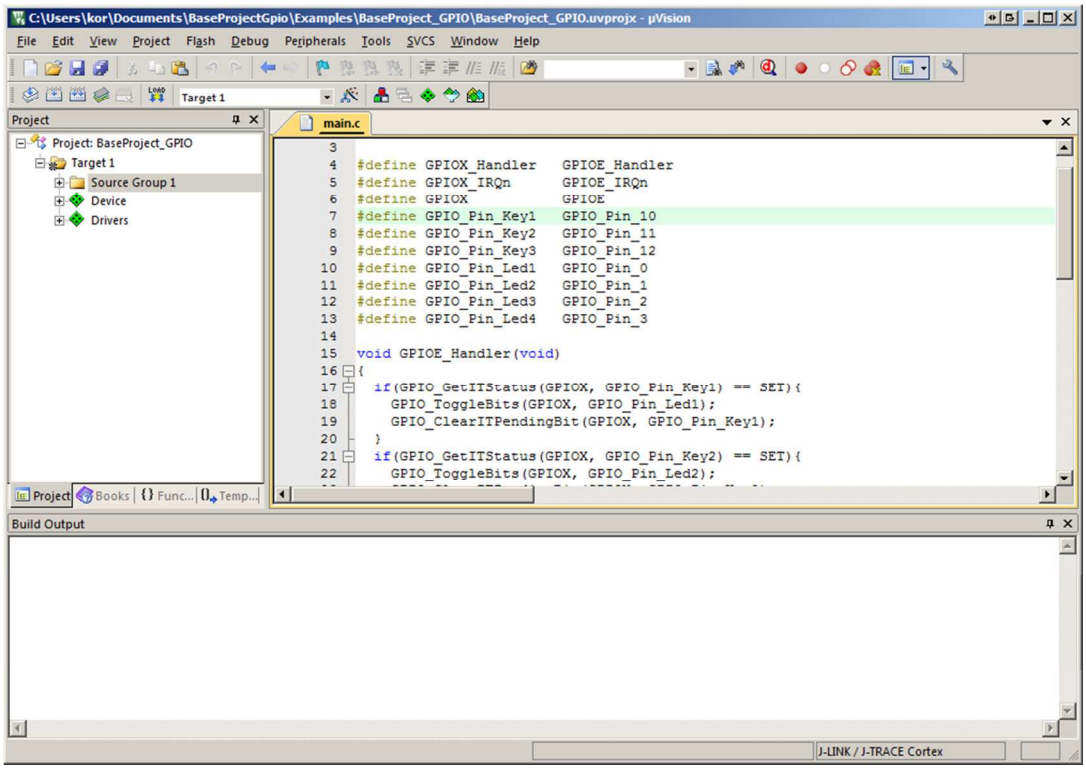


Рисунок 4.1

Главное окно проекта в среде Keil содержит несколько внутренних окон:

Сверху содержатся иконки быстрого доступа (они дублируют пункты меню)

Слева расположено многофункциональное окно с вкладками: Project (файлы проекта), Book (документация и описания), Functions (функции, которые содержатся в проекте) и Templates (средства программирования).

Снизу вывод информационных сообщений линковщика и компилятора.

Основное окно – окно редактора (в данном примере файл main.c)

5. Настройка среды Keil. Создание первой программы.

5.1 Keil 4.

Необходимо создать новый проект.

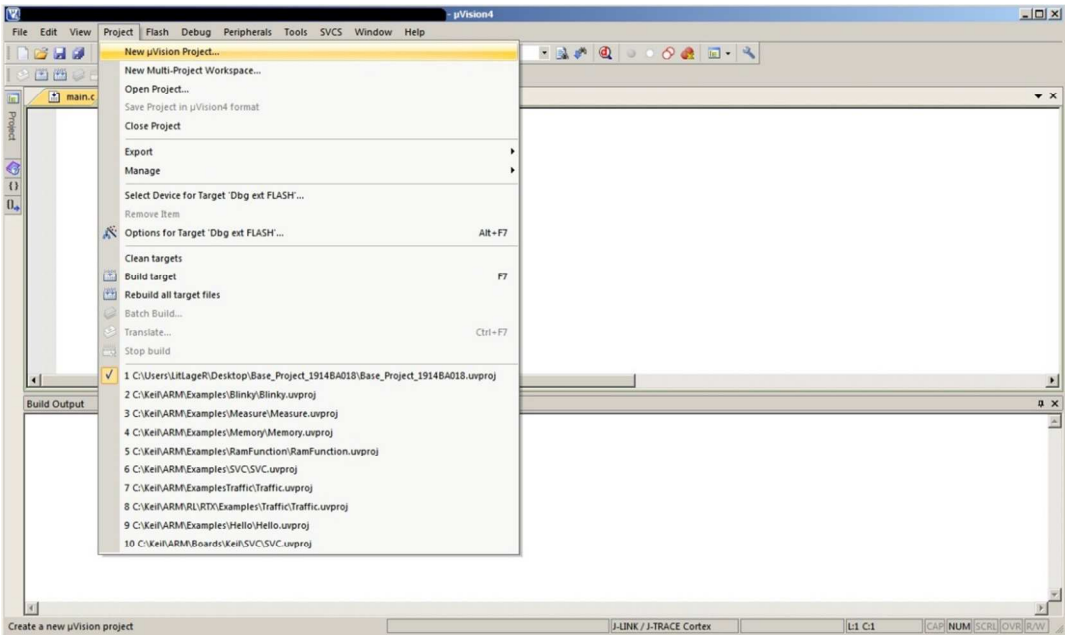


Рисунок 5.1.1

Выбрать папку для сохранения проекта

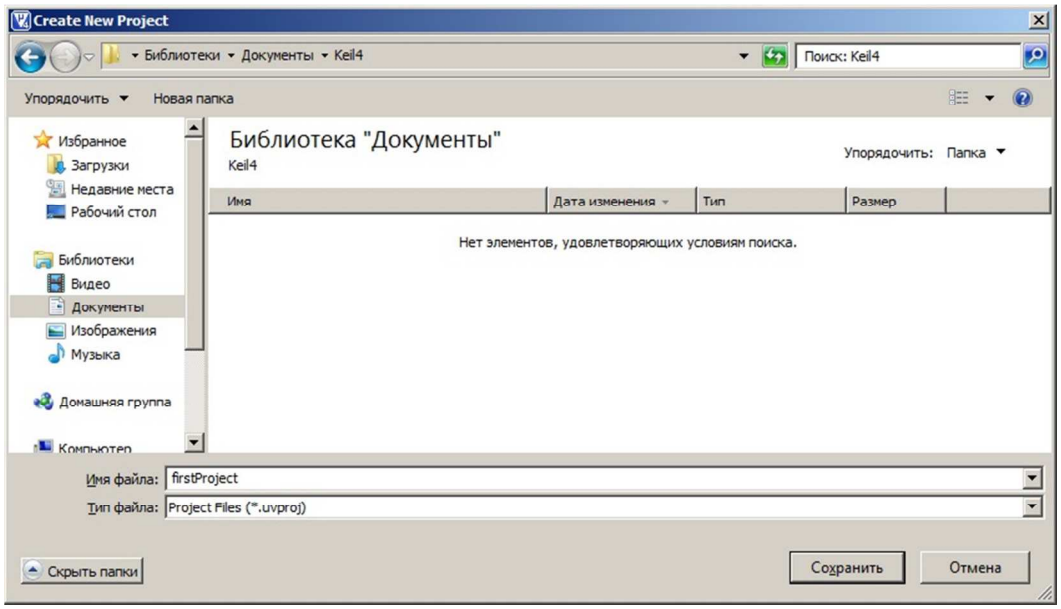


Рисунок 5.1.2

Выбрать ядро микроконтроллера: ARM Cortex-M4 FPU

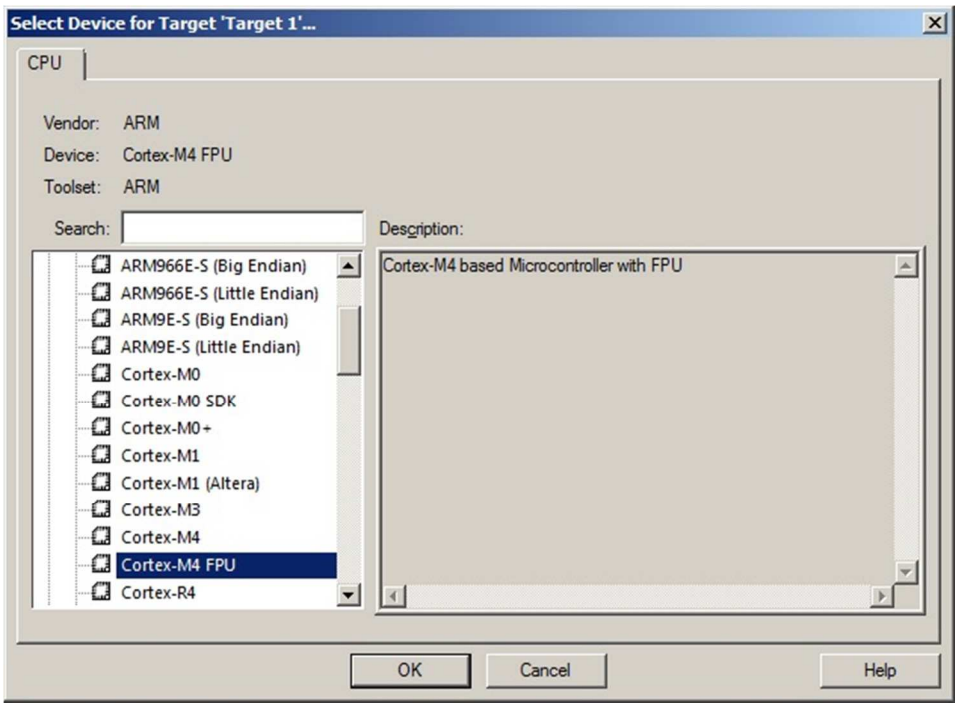


Рисунок 5.1.3

Откроется окно проекта. Слева отображаются файлы проекта. Необходимо добавить две папки: «Device» и «Drivers».

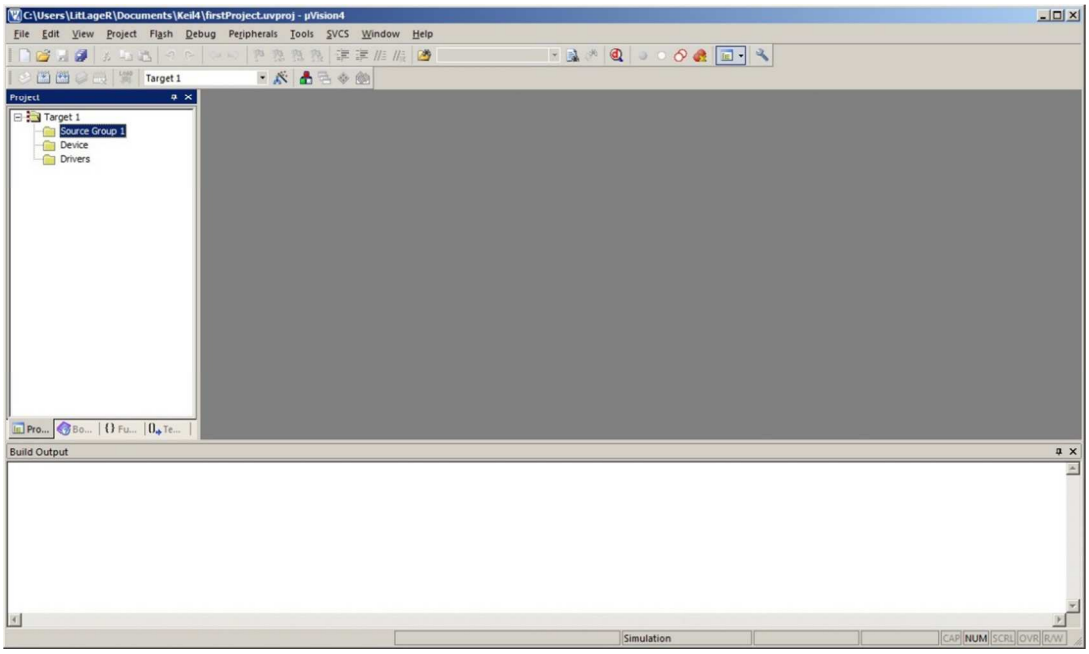


Рисунок 5.1.4

Далее необходимо открыть PACK-файл любым архиватором. Из него понадобятся папки «Config», «IDE», «Ini» и «Libraries». Их необходимо скопировать в папку с проектом.

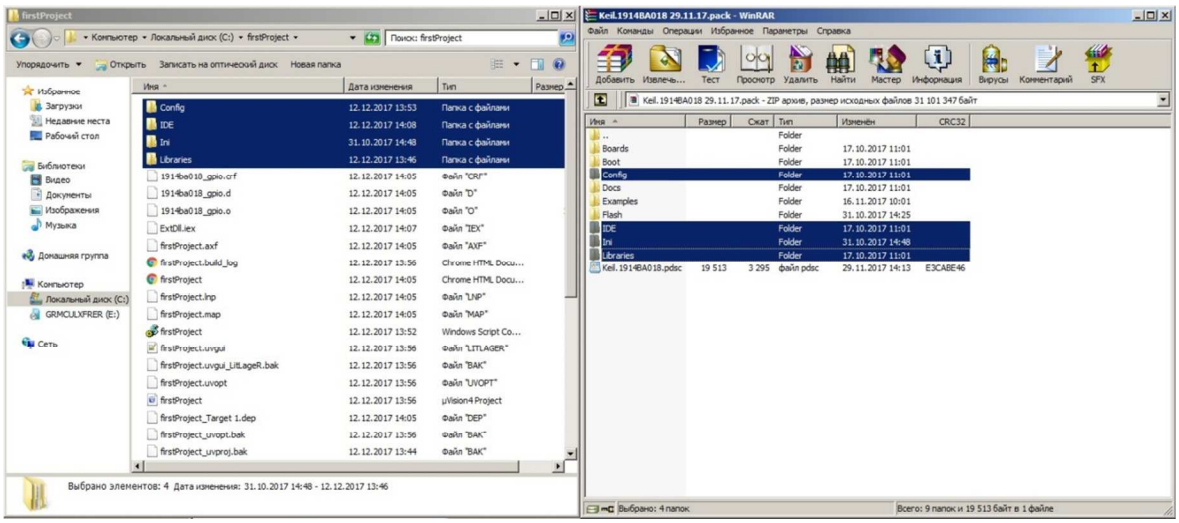


Рисунок 5.1.5

Далее в проект необходимо добавить файлы из скопированных папок, а так-же создать файл main.c

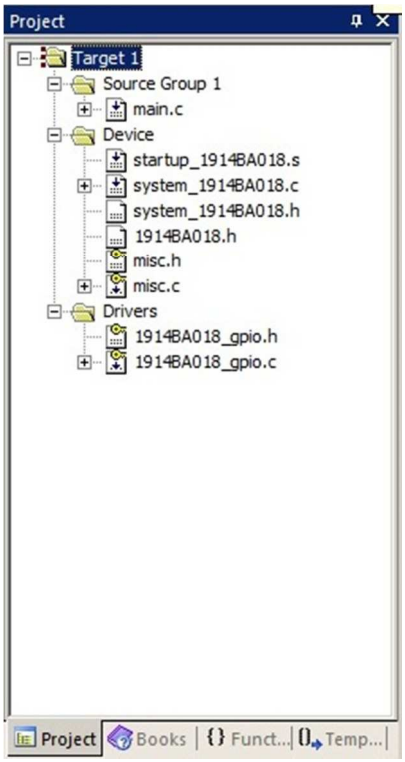


Рисунок 5.1.6

Папка: «Source group 1»:

«main.c»: Необходимо создать новый файл со следующим содержимым:

```
#include "1914BA018.h"
#include "1914BA018_gpio.h"

int main(void){
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_Init(GPIOE, &GPIO_InitStructure);

  while(1){
```

```
    GPIO_SetBits(GPIOE, GPIO_Pin_1);
    for(int i=0;i<100000;i++){};
    GPIO_ResetBits(GPIOE, GPIO_Pin_1);
    for(int i=0;i<100000;i++){};
  }
}
```

Папка «Device»:

«startup_1914BA018.s»:\Libraries\CMSIS\Cm4\DeviceSupport\1914BA018\startup\arm\startup_1914BA018.s

«system_1914BA018.c»:\Libraries\CMSIS\Cm4\DeviceSupport\1914BA018\startup\arm\system_1914BA018.c

«system_1914BA018.h»:\Libraries\CMSIS\Cm4\DeviceSupport\1914BA018\startup\arm\system_1914BA018.h

«1914BA018.h»: .\Libraries\CMSIS\Cm4\DeviceSupport\1914BA018\inc\1914BA018.h

«misc.h»: .\Libraries\1914BA018_StdPeriph_Driver\inc\misc.h

«misc.c»: .\Libraries\1914BA018_StdPeriph_Driver\src\misc.c

Папка «Drivers»:

«1914BA018_gpio.h»: .\Libraries\1914BA018_StdPeriph_Driver\inc\1914BA018_gpio.h

«1914BA018_gpio.c»: .\Libraries\1914BA018_StdPeriph_Driver\src\1914BA018_gpio.c

Далее необходимо войти в настройки проекта (Options for Target…) или Alt+F7. Возможно появление следующего окна

Рисунок 5.1.7

Необходимо выбрать вручную Cortex M4 (см. рисунок 5.1.8)

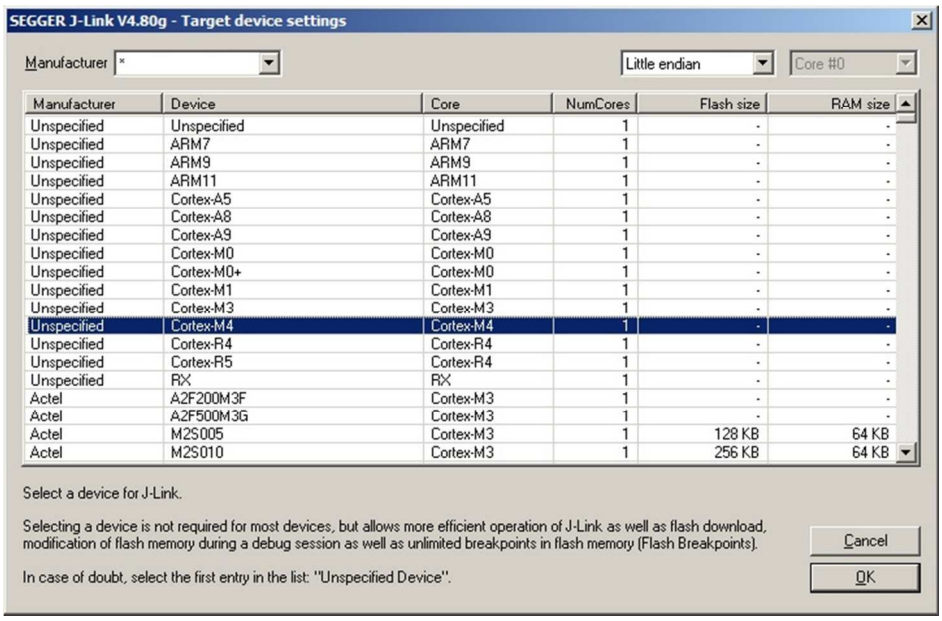


Рисунок 5.1.8

Настройки проекта представлены на следующих рисунках:

Вкладка «Device»

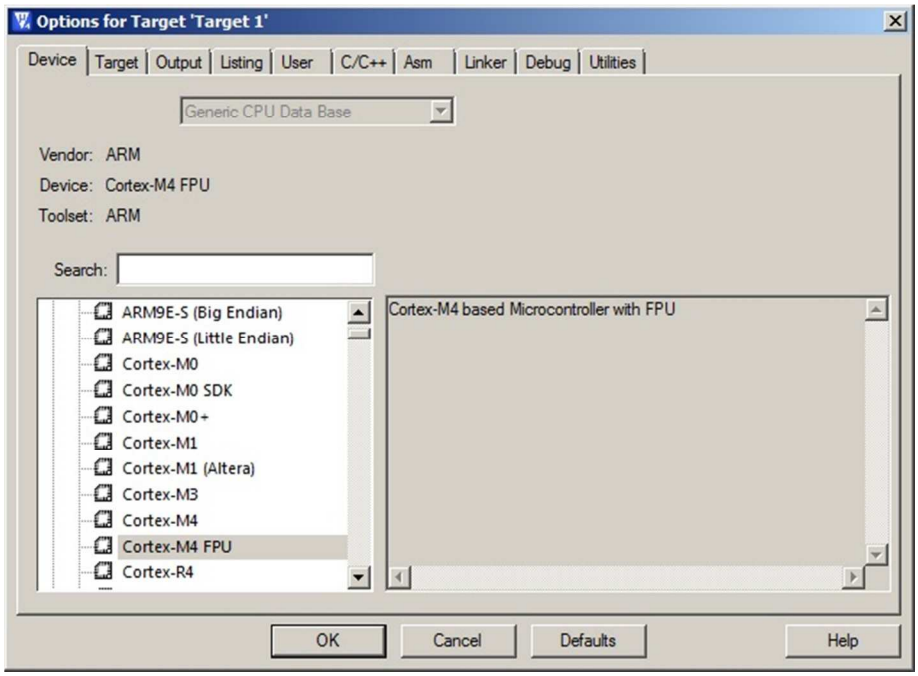


Рисунок 5.1.9

Вкладка «Target». В ней необходимо указать ссылку на System-Viewer File: .\IDE\keil\SVD\1914BA018.SFR, начальный адрес IROM 0x8000000, размер 0x20000, начальный адрес IRAM 0x20000000, размер 0xFC00, а так-же частоту кварцевого резонатора 16мгц.

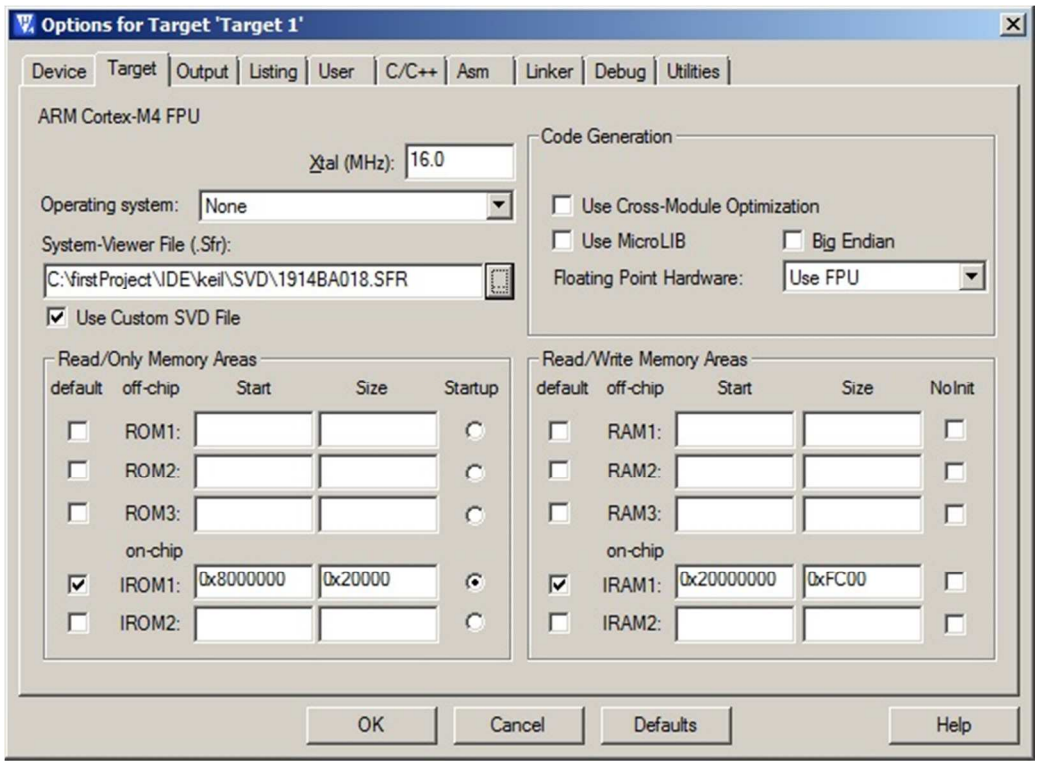


Рисунок 5.1.10

Вкладка «Output»:

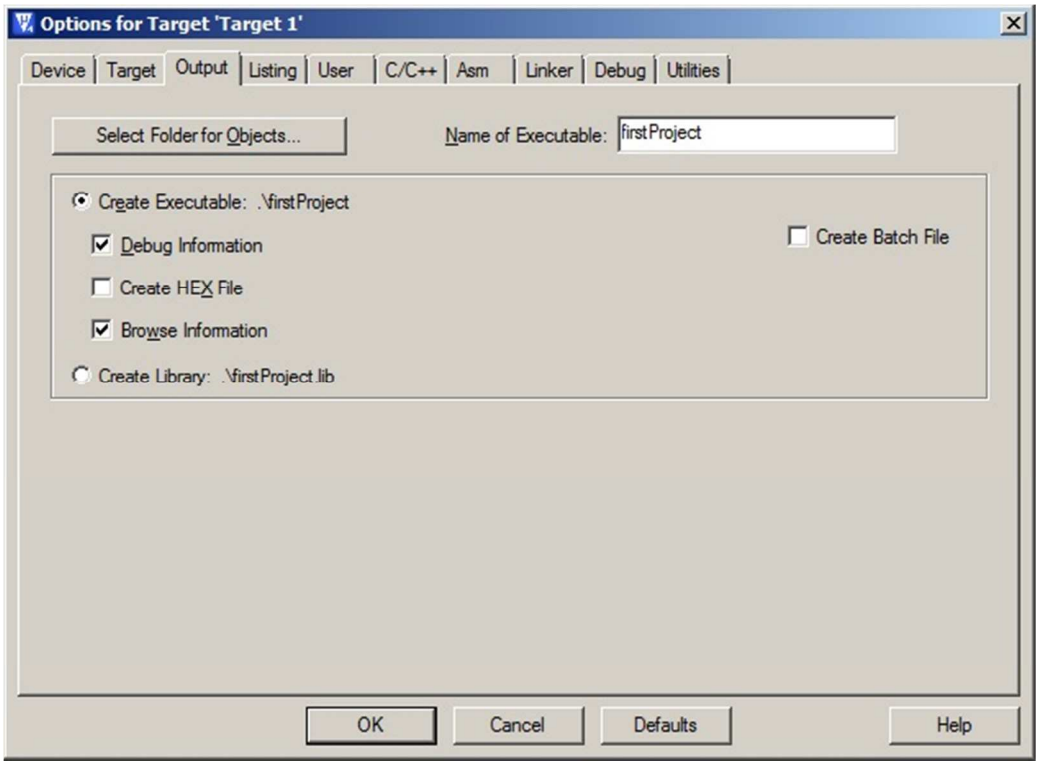


Рисунок 5.1.11

Вкладка «Listing»:

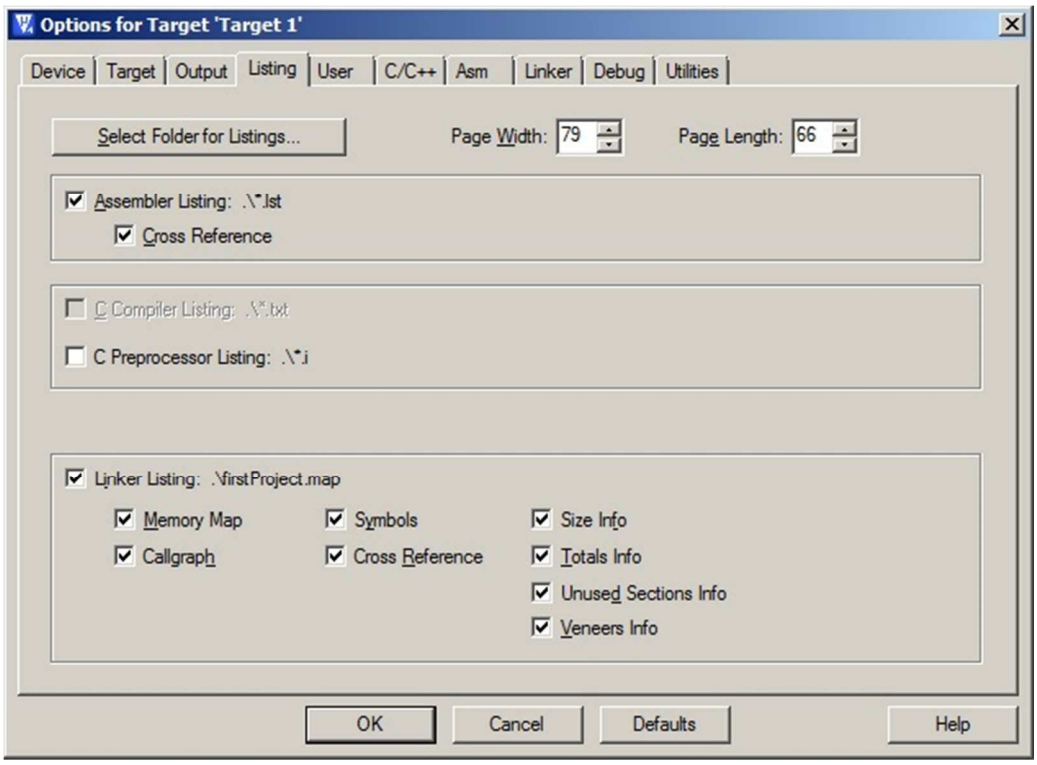


Рисунок 5.1.12

Вкладка «User»:

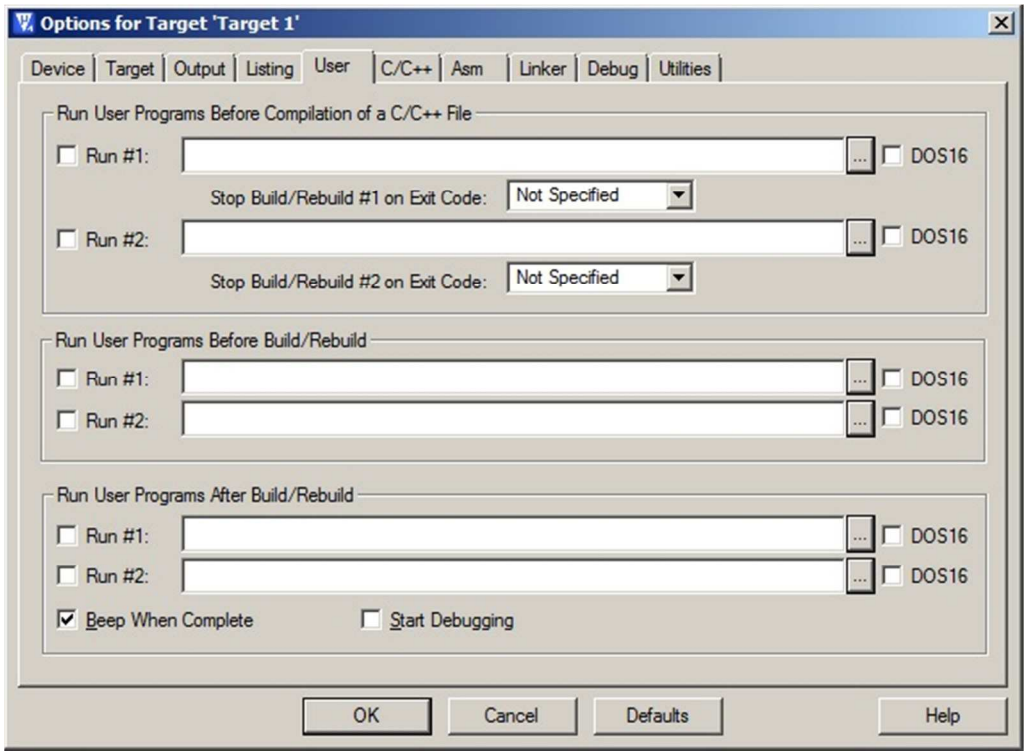


Рисунок 5.1.13

Вкладка «C/C++»:

Установить галочку «C99 Mode», а в строку Include Paths ввести следующую строку

```
./Libraries/1914BA018_StdPeriph_Driver/inc;./Libraries/1914BA018_StdPeriph_Driver/src;./Li
```

braries/CMSIS/Cm4/CoreSupport;./Libraries/CMSIS/Cm4/DeviceSupport/1914BA018/inc;./Libraries/CMSIS/Cm4/DeviceSupport/1914BA018/startup/arm;./Config;

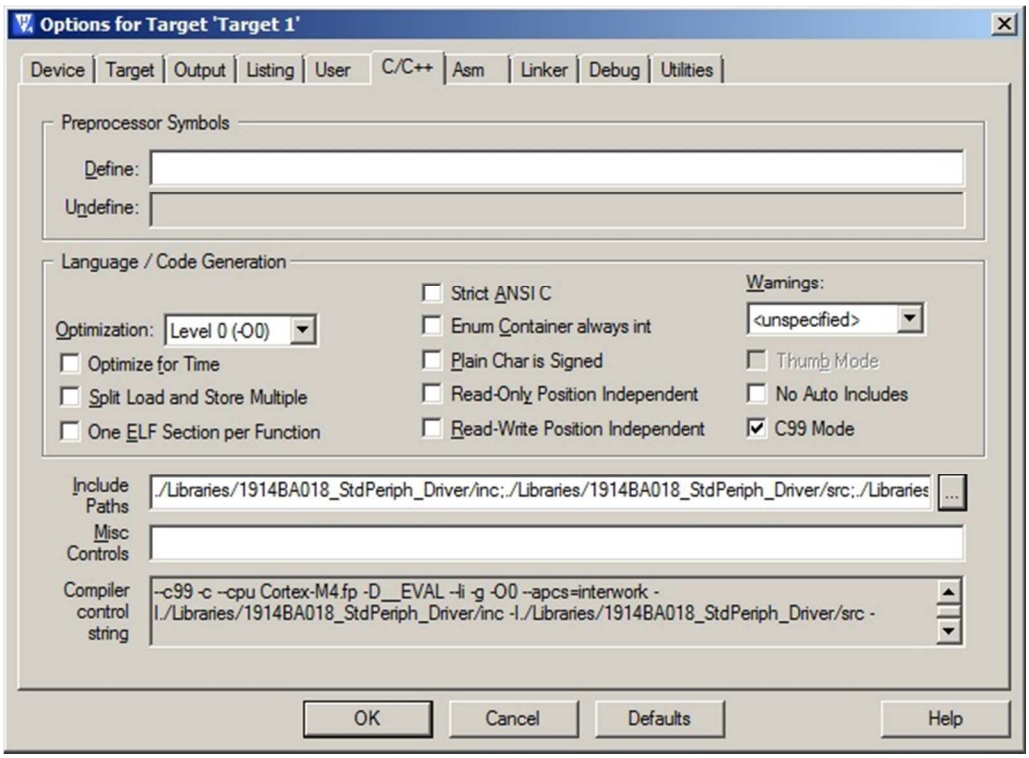


Рисунок 5.1.14

Вкладка «Asm»:

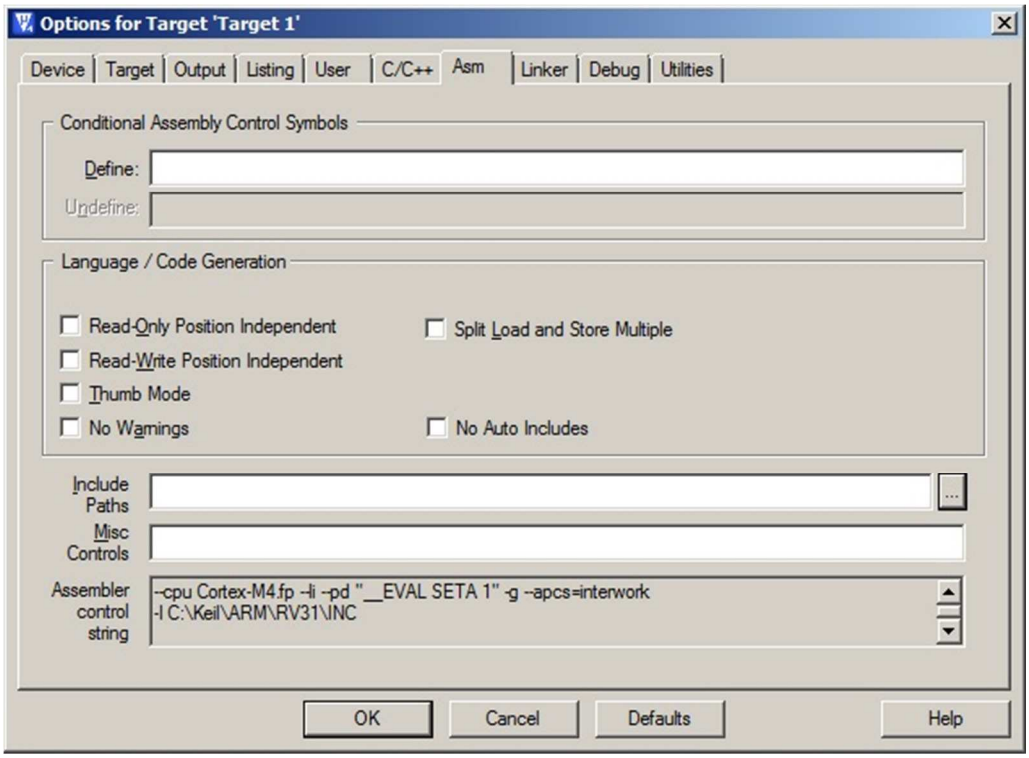


Рисунок 5.1.15

Вкладка «Linker»:

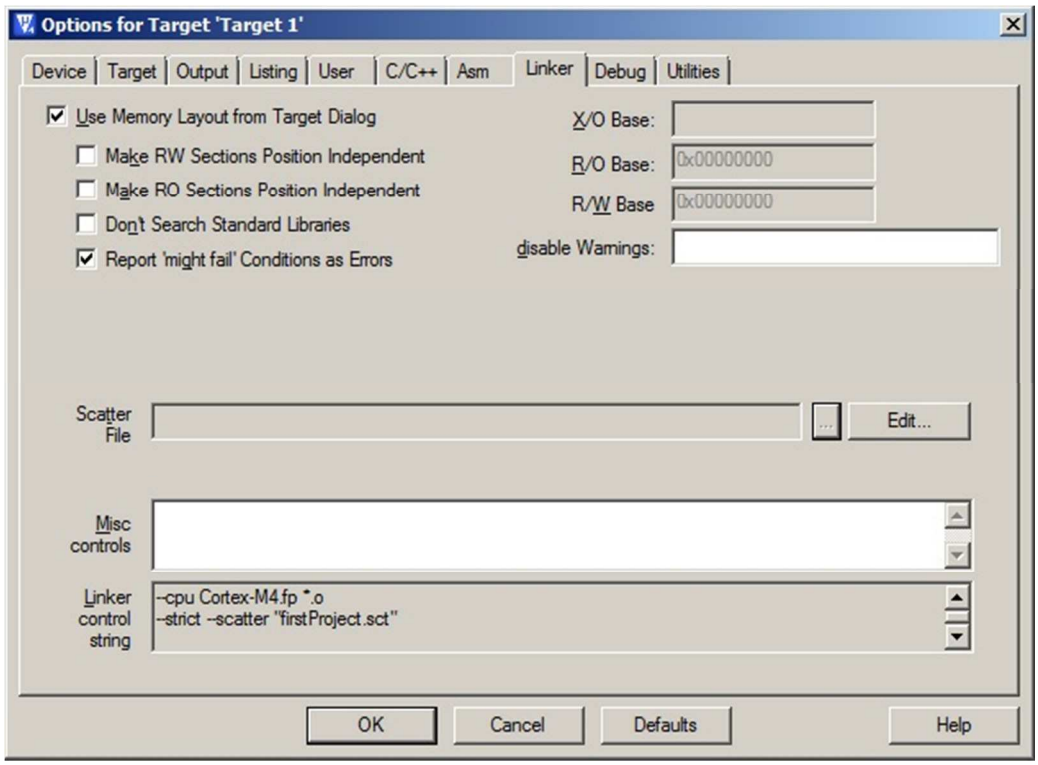


Рисунок 5.1.16

Вкладка «Debug»:

Необходимо добавить ссылку на «Initialization File» .\Ini\ram.ini, выбрать J-LINK/J-TRACE Cortex в качестве устройства для отладки.

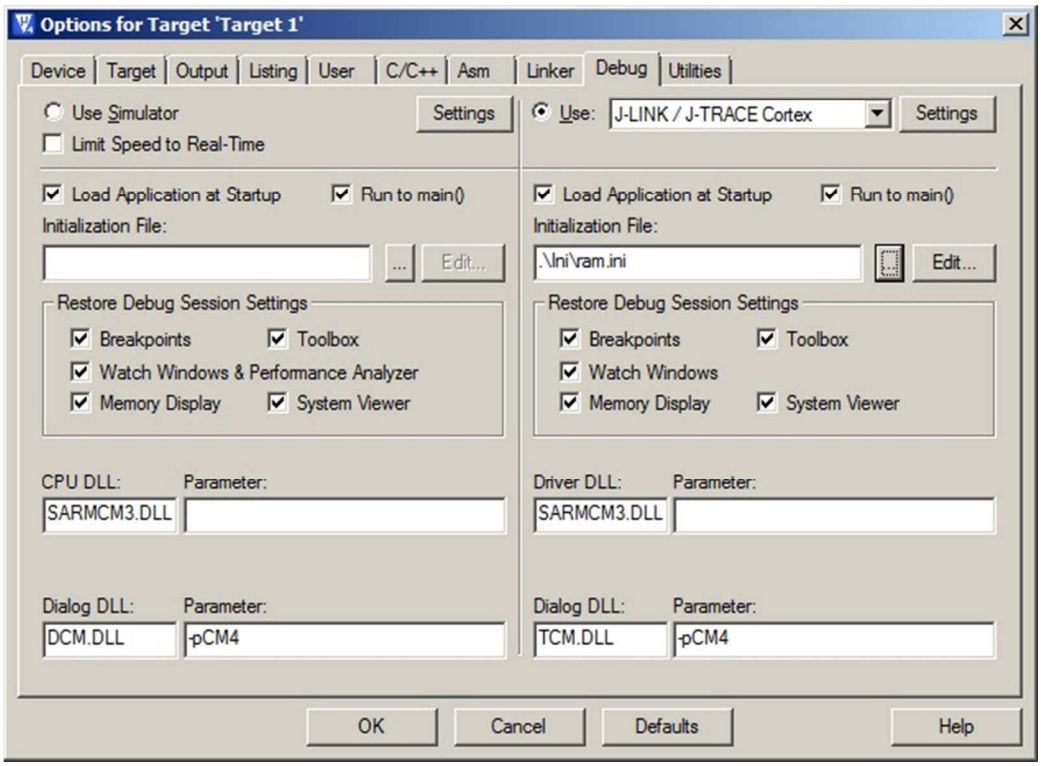


Рисунок 5.1.17

Затем перейти в его настройки

Вкладка «Debug»:

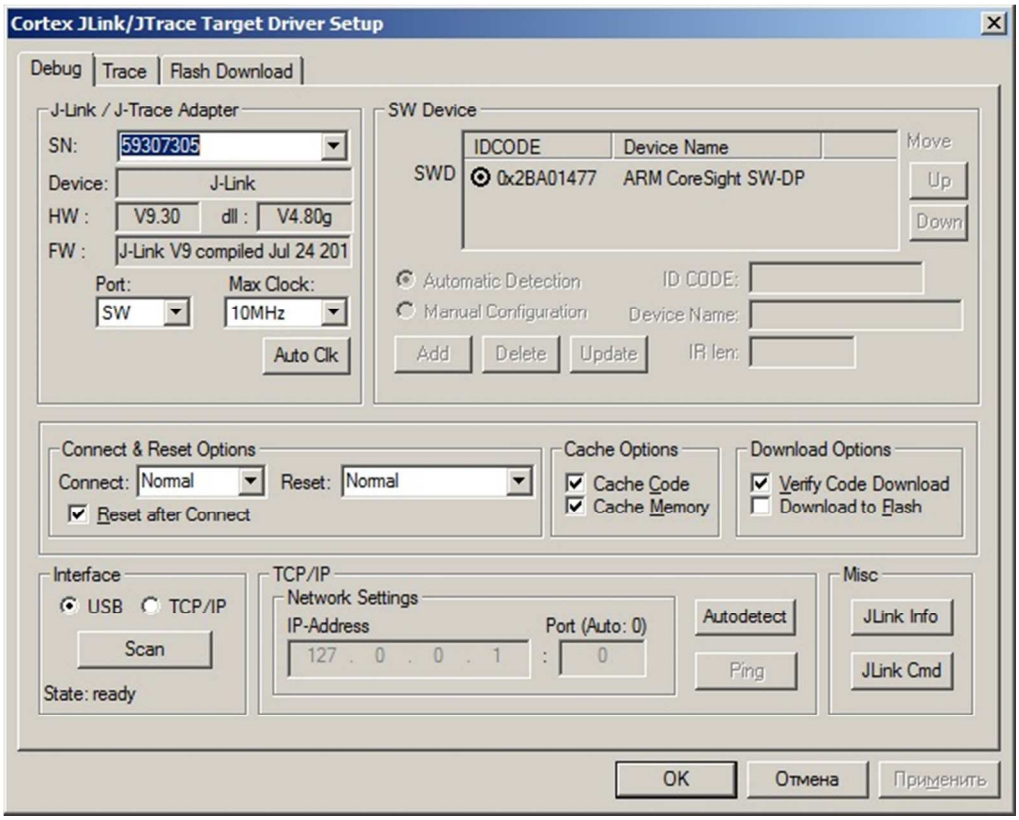


Рисунок 5.1.18

Вкладка «Trace»:

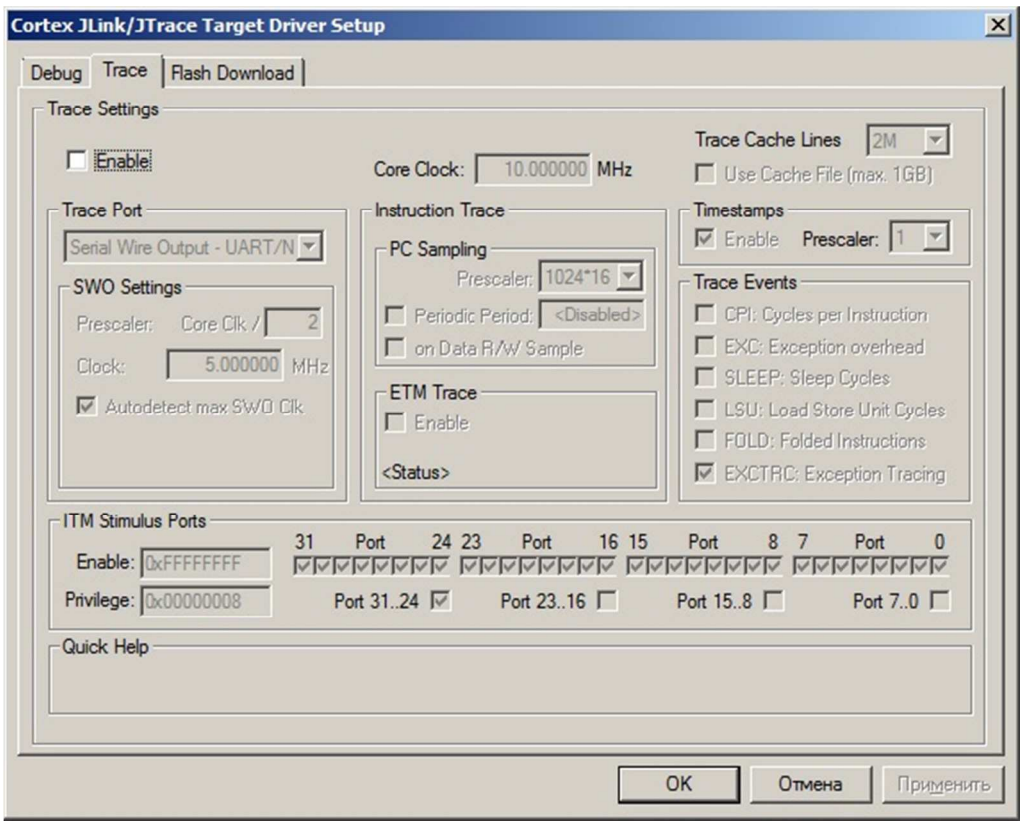


Рисунок 5.1.19

Вкладка «Flash Download»:

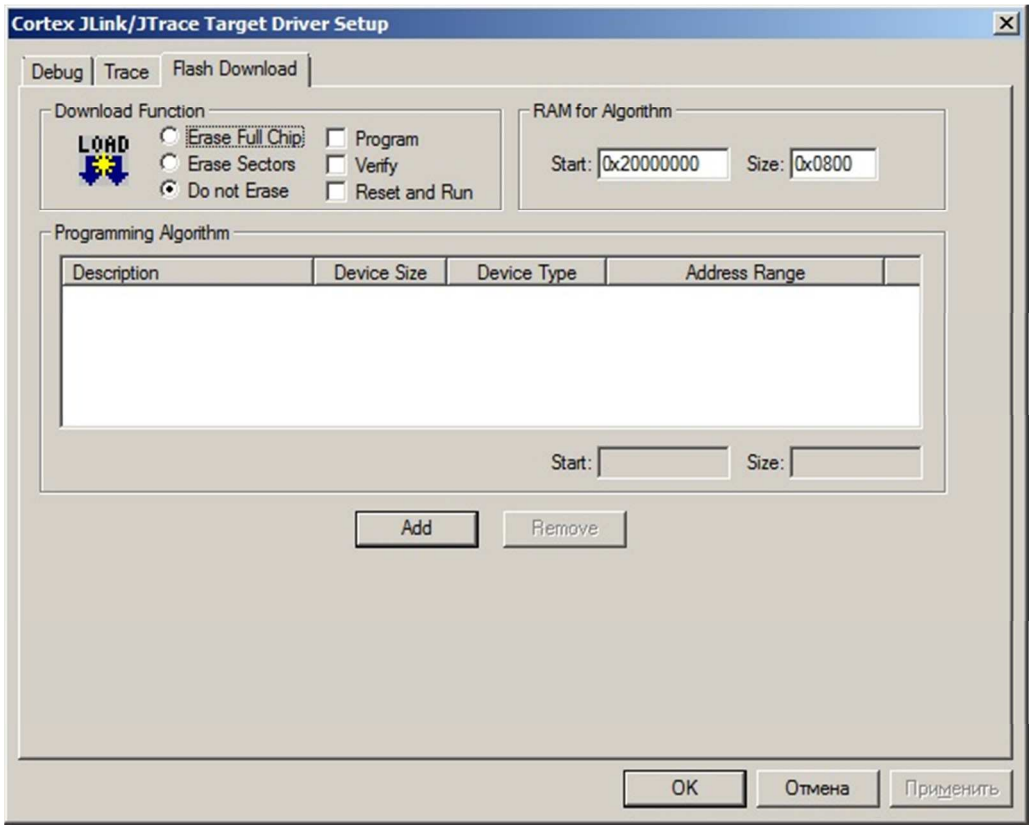


Рисунок 5.1.20

Вкладка «Utilities» настроек проекта:

Необходимо вновь добавить ссылку на .\Ini\ram.ini файл

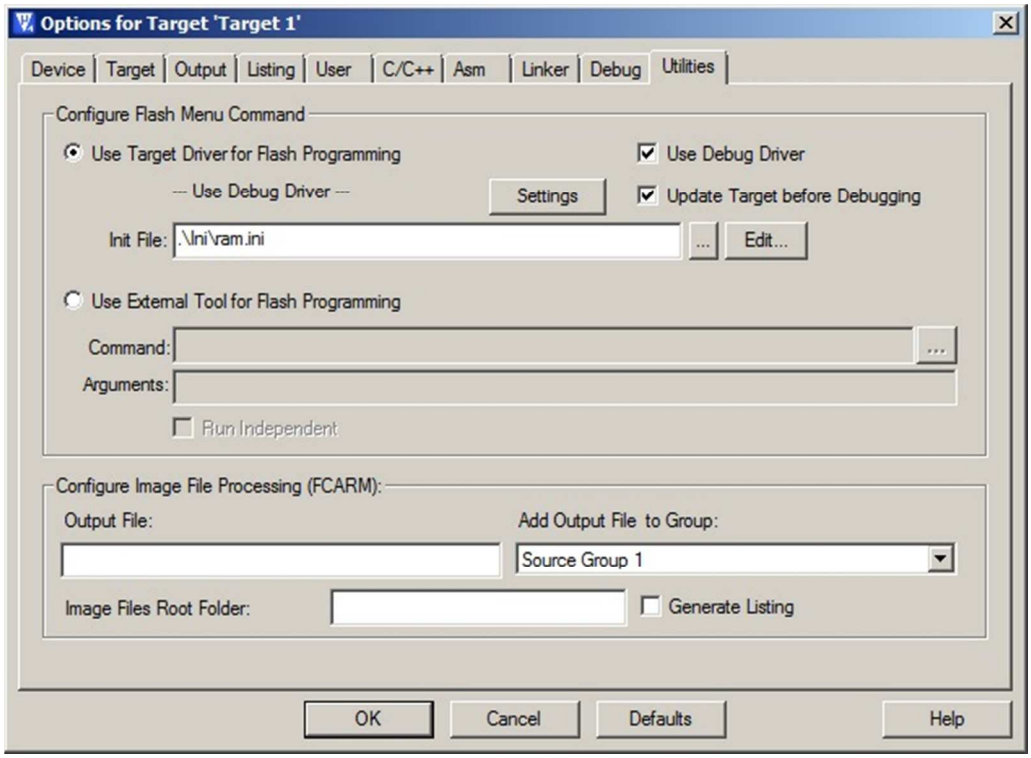


Рисунок 5.1.21

Настройка проекта завершена, теперь можно скомпилировать проект «Project->Rebuild all target files»

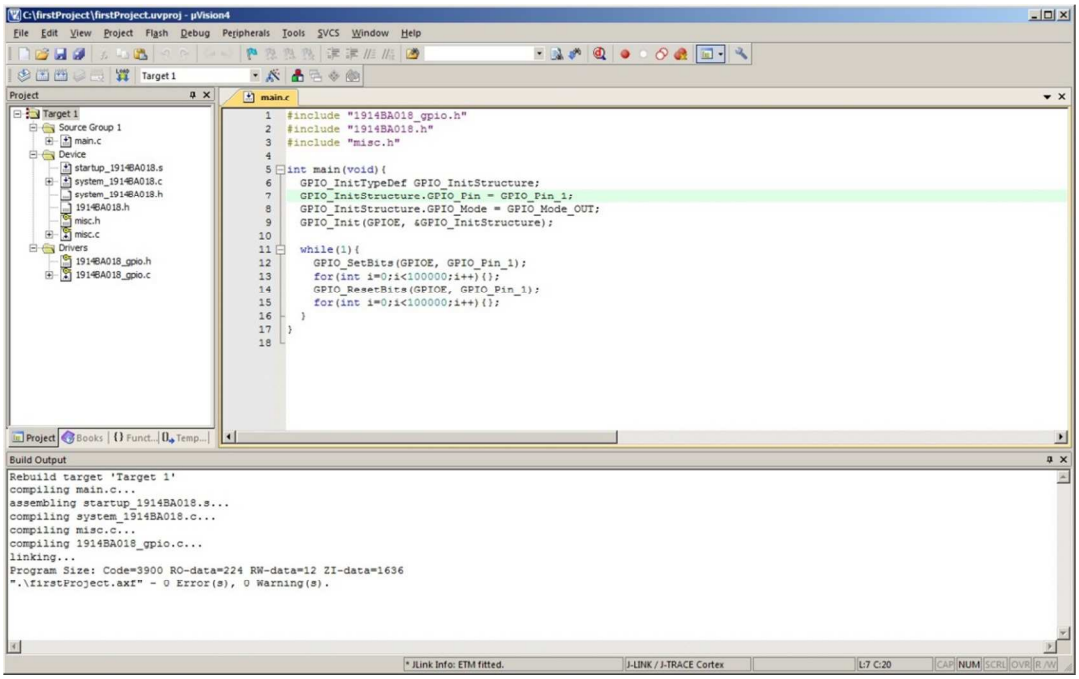


Рисунок 5.1.22

В окне «Build Output» будет информация об успешной компиляции проекта. Теперь можно переходить к отладке внутри микроконтроллера. Для этого необходимо перейти «Debug->Start/Stop Debug Sessions» или Ctrl+F5. Может появиться информация об отсутствии алгоритмов прошивки. Необходимо нажать кнопку «ОК».

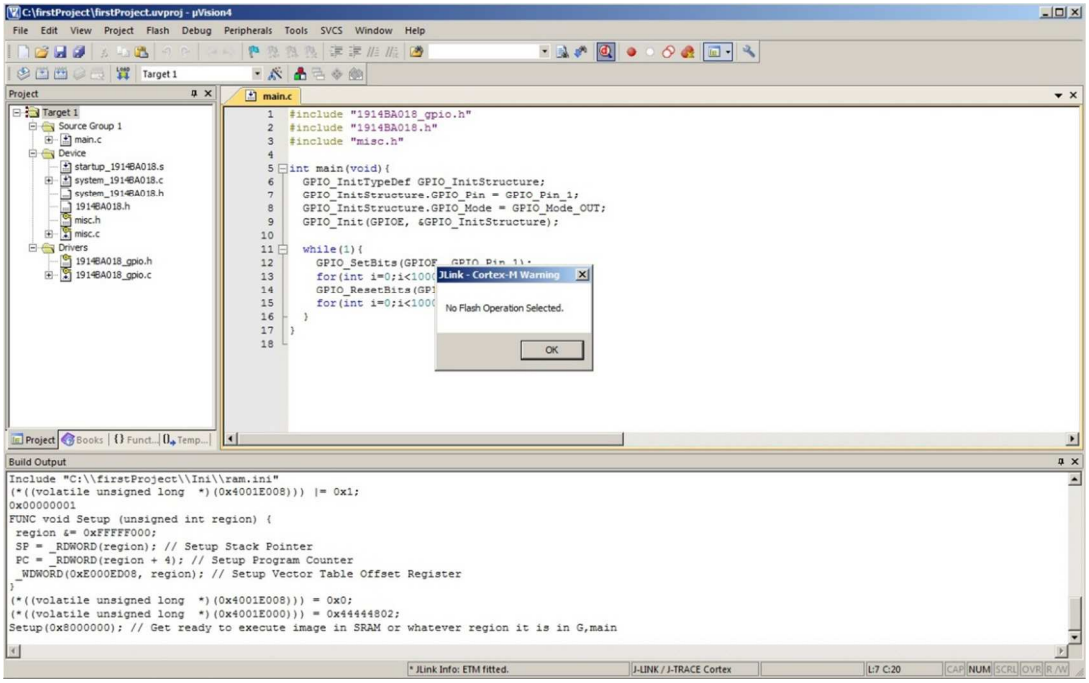


Рисунок 5.1.23

Открылось окно отладки, необходимо запустить исполнение программы. Для этого необходимо нажать кнопку «Debug->Run» или нажать клавишу F5

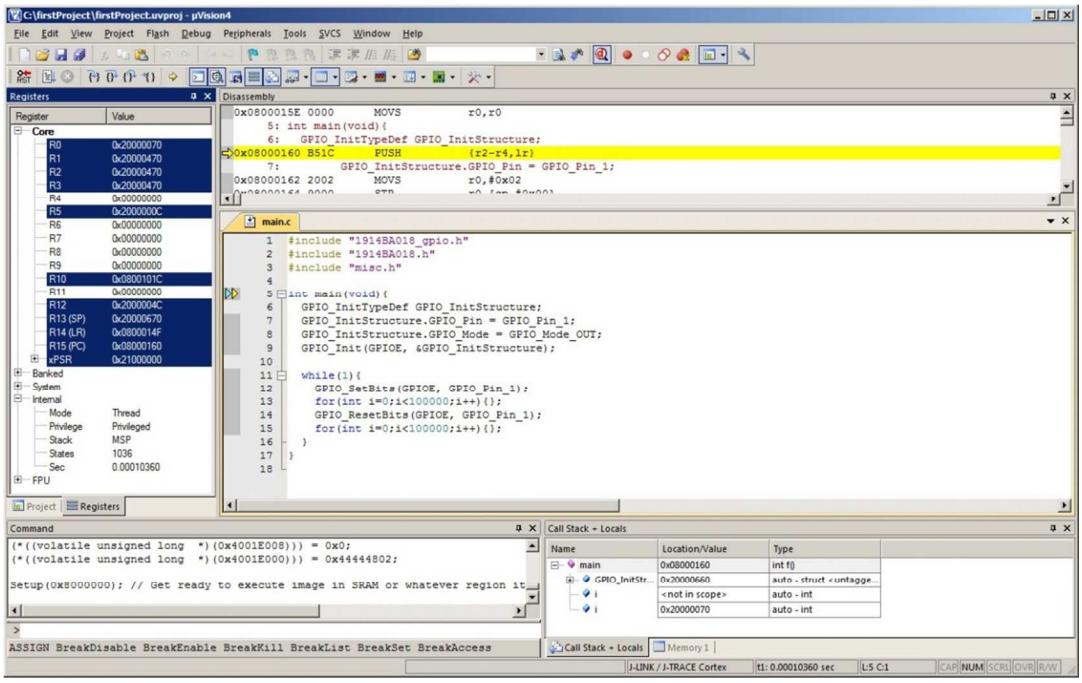


Рисунок 5.1.24

5.2 Keil 5.

Необходимо создать новый проект.

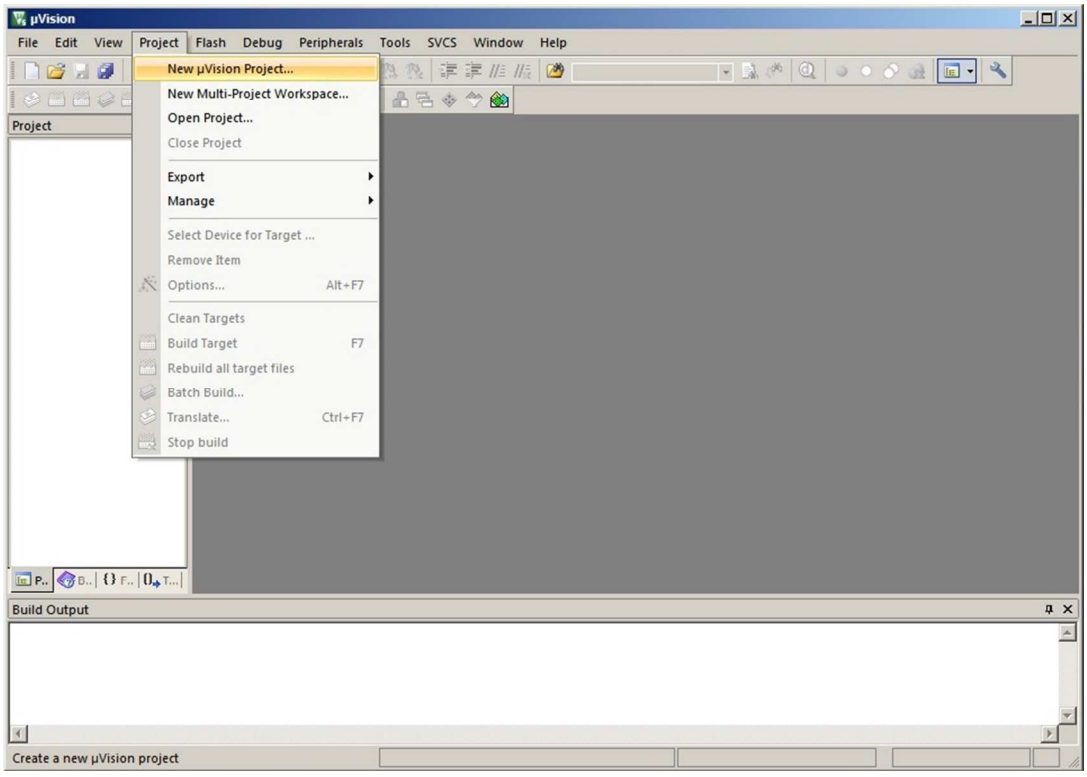


Рисунок 5.2.1

Выбрать папку для сохранения проекта

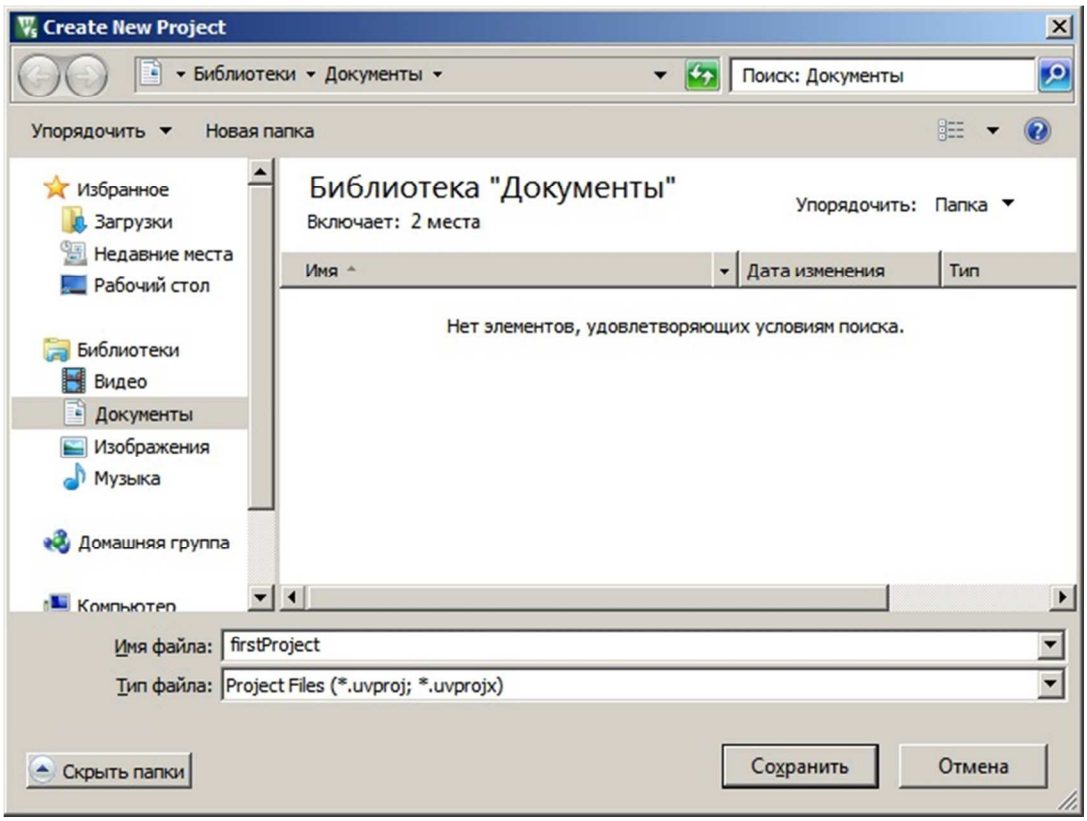


Рисунок 5.2.2

Выбрать микроконтроллер 1914BA018

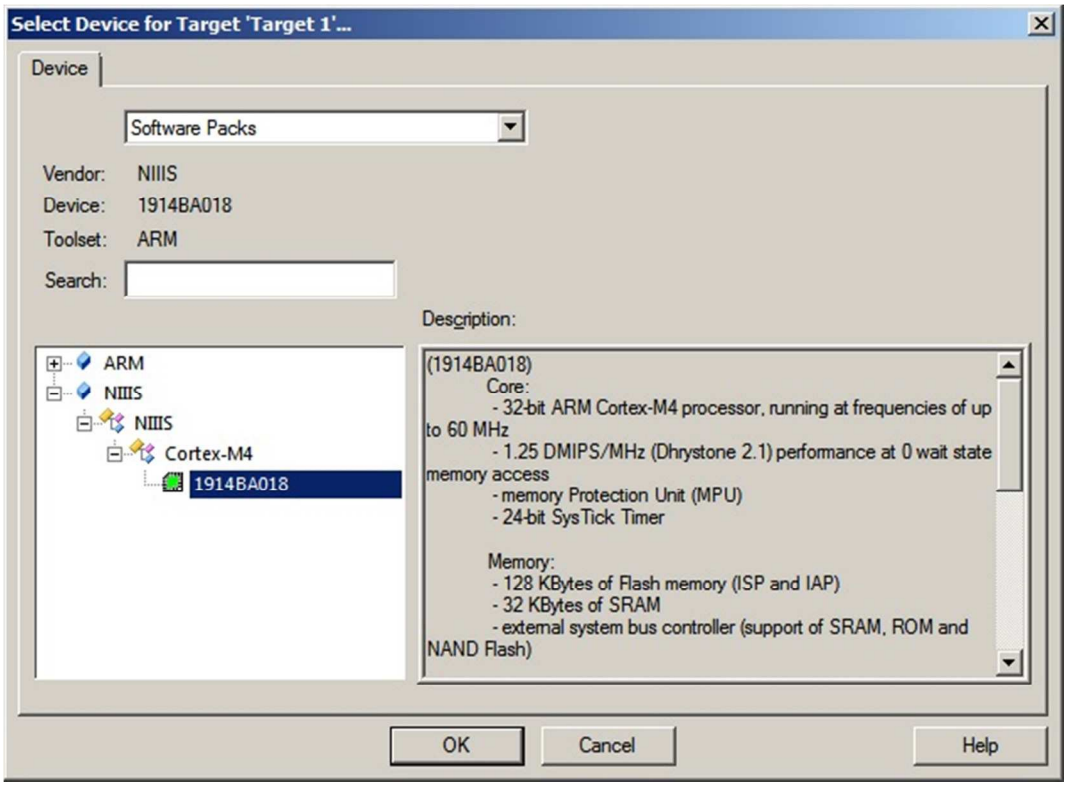


Рисунок 5.2.3

Необходимо выбрать, какие библиотеки подключить. Startup подключается для любого проекта. В данном примере мы будем использовать библиотеку GPIO.

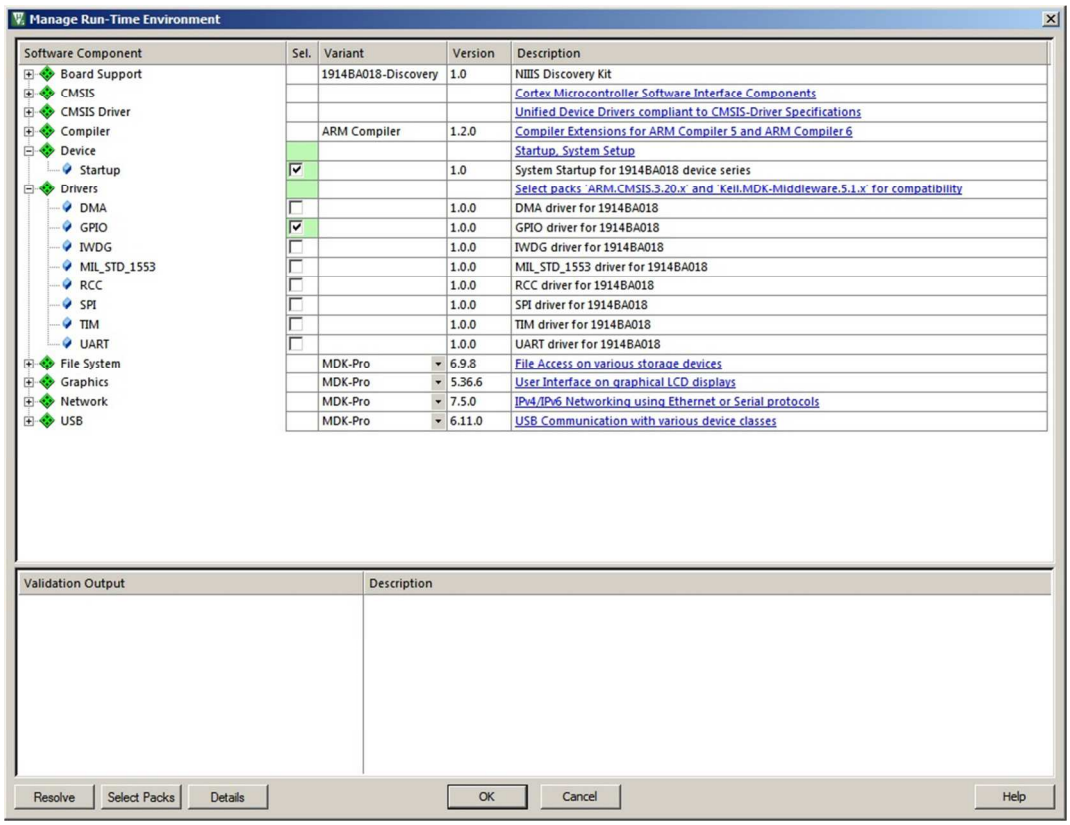


Рисунок 5.2.4

После нажатия кнопки «ОК» появится окно проекта

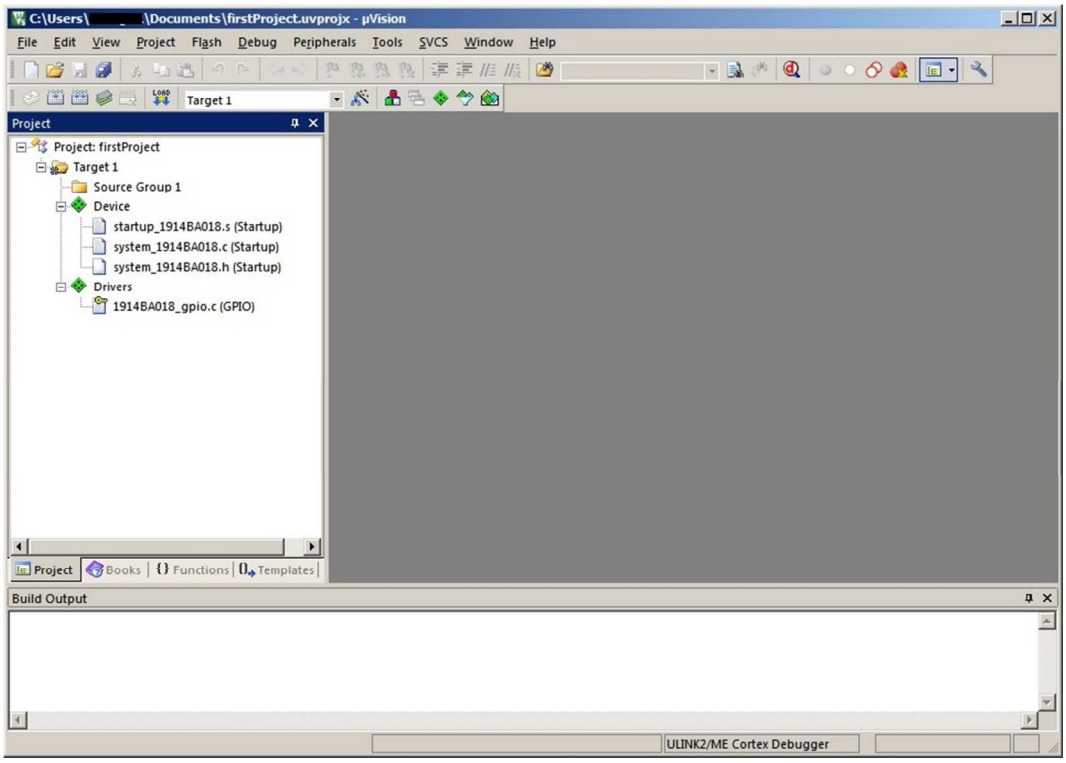


Рисунок 5.2.5

Необходимо добавить файл main.c, для этого нажать на Source Group 1 правой клавишей мыши и выбрать Add New Item. В появившемся окне выбрать C File (.c) и ввести название «main»

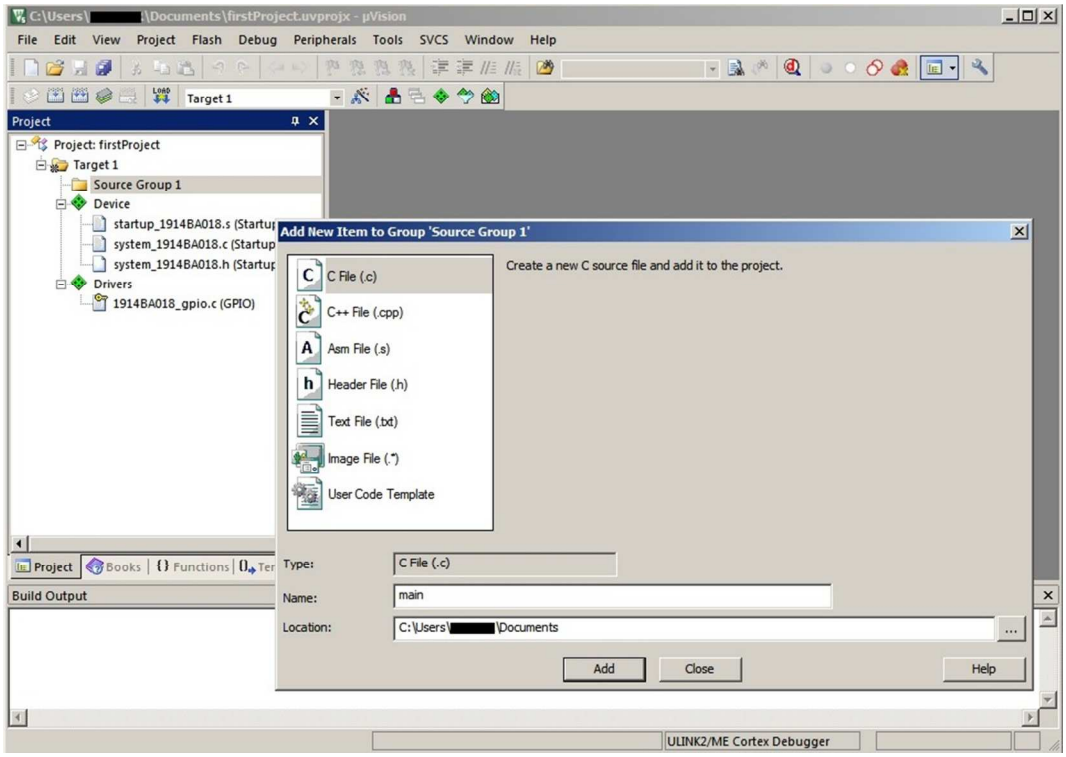


Рисунок 5.2.6

В папке Source Group 1 появился файл main.c

Необходимо кликнуть на него дважды, справа откроется окно редактора.

Необходимо подключить Device header и драйвер GPIO. Для этого правой клавишей мыши кликнуть в пустом месте открытого файла main.c, перейти в раздел “Insert #include file” и выбрать сначала 1914BA018.h, а затем 1914BA018_gpio.h

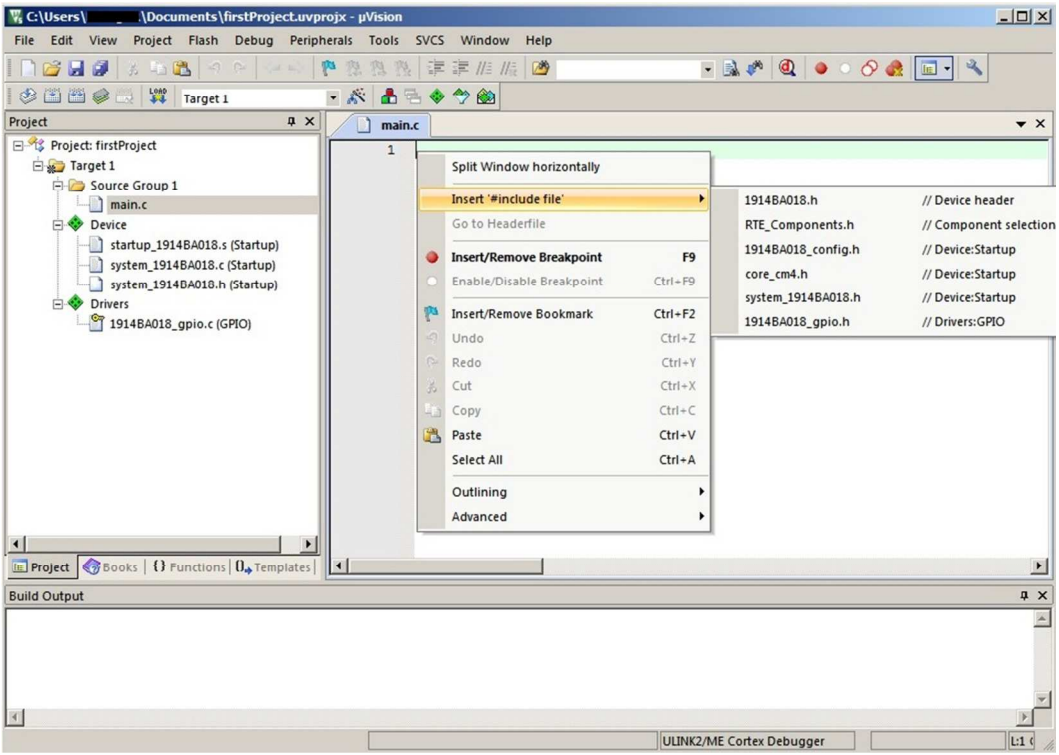


Рисунок 5.2.7

Основное тело программы

```
int main(void){

}
```

В него добавим настройку порта GPIOE и управление первым пином в бесконечном цикле

```
while(1){

}
```

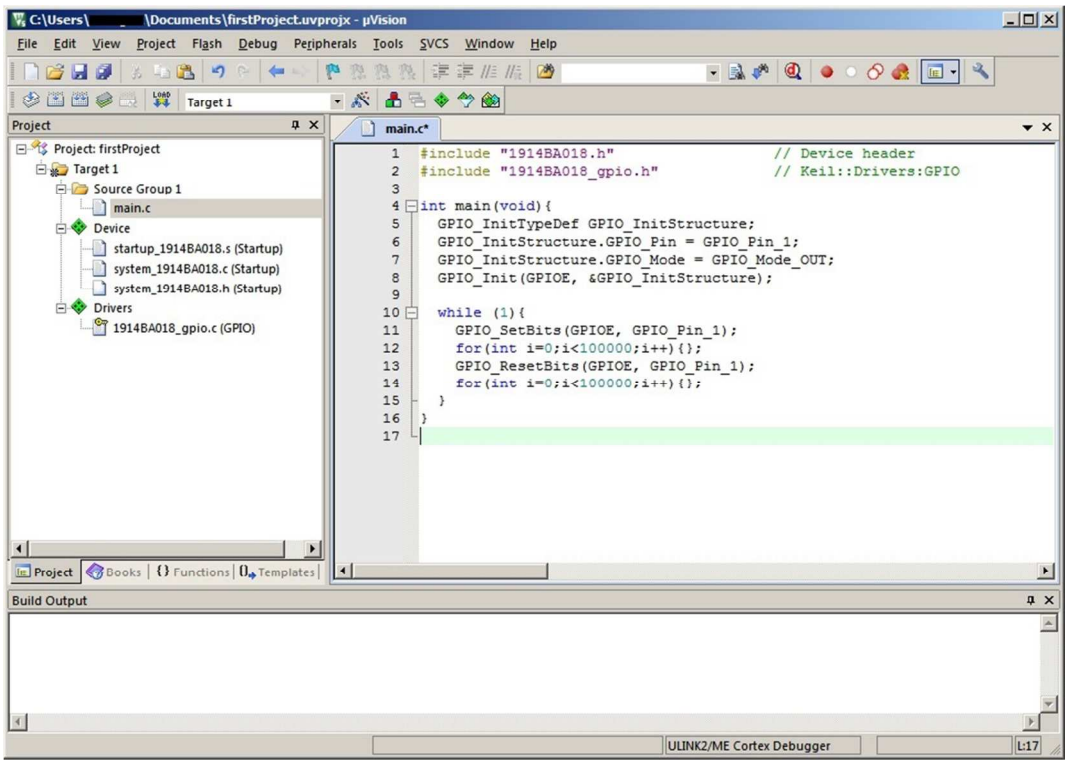


Рисунок 5.2.8

Текст программы на рисунке 5.8:

```
#include "1914BA018.h"
#include "1914BA018_gpio.h"

int main(void){
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_Init(GPIOE, &GPIO_InitStructure);

  while(1){
    GPIO_SetBits(GPIOE, GPIO_Pin_1);
    for(int i=0;i<100000;i++){};
    GPIO_ResetBits(GPIOE, GPIO_Pin_1);
    for(int i=0;i<100000;i++){};
  }
}
```

Далее необходимо скомпилировать проект (иконка «Rebuild»)

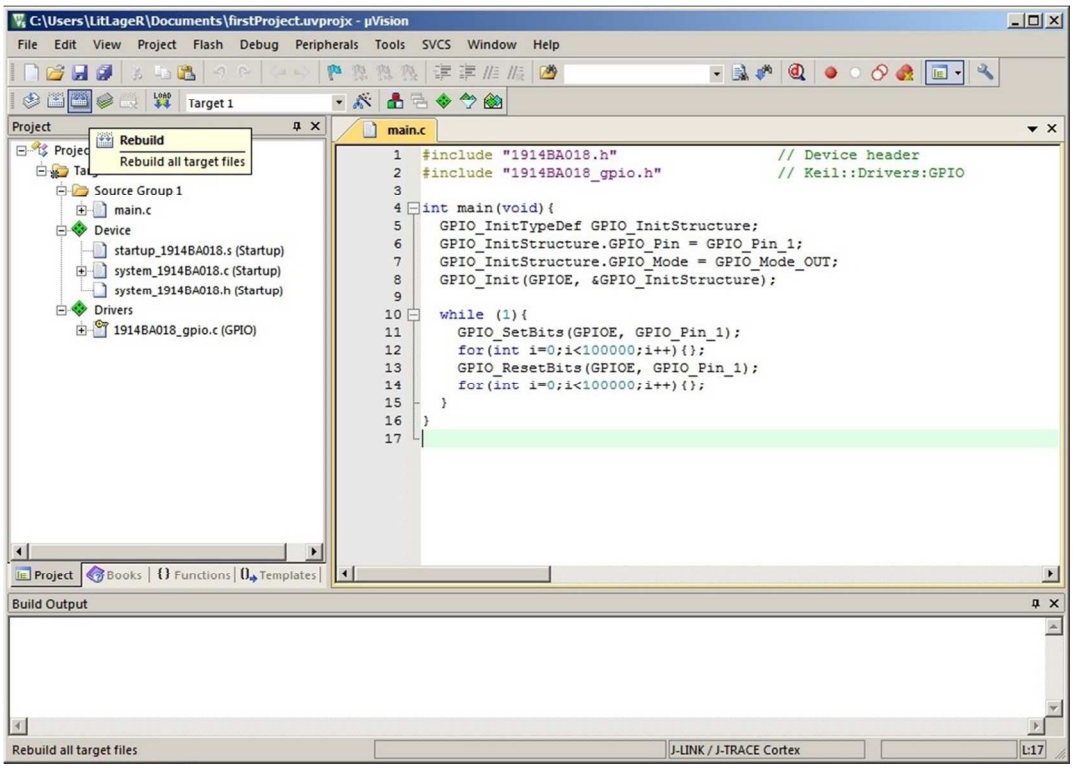


Рисунок 5.2.9

Последняя строчка в программе должна быть пустой. Если это не так, keil выведет сообщение “warning” в окне “Build Output”

6. Загрузка программы в ОЗУ

В загрузчике микроконтроллера не предусмотрен запуск программы из ОЗУ после сброса. Для реализации отладки в ОЗУ необходимо добавить инициализацию в Debug режим. (Открыть настройки Target. Нажать сочетание клавиш Alt+F7 или Project->Options for Target)

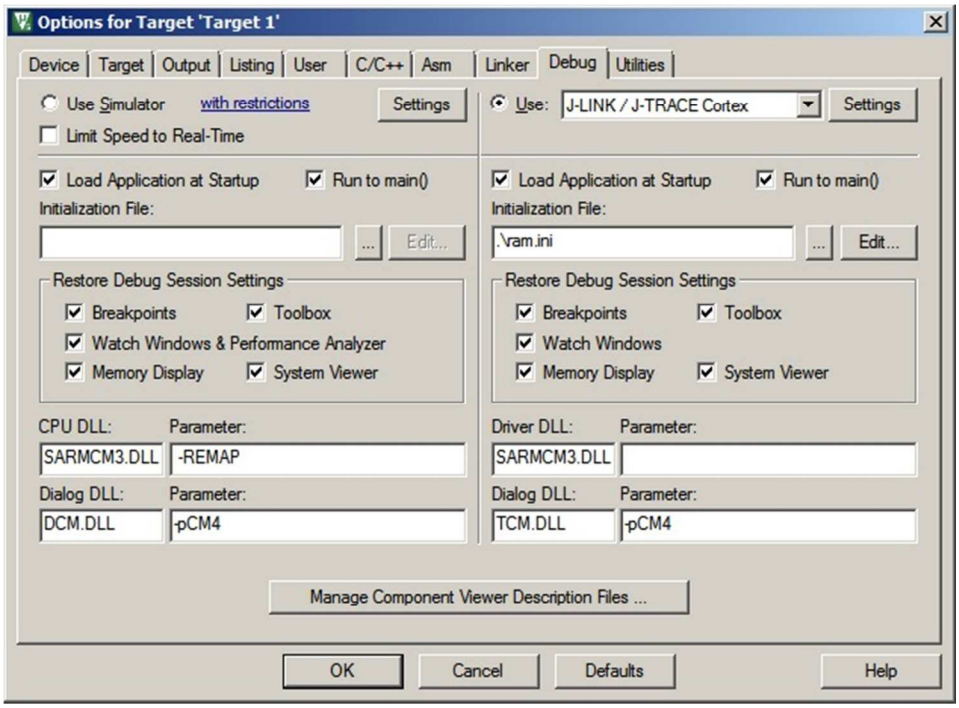


Рисунок 6.1

Это делается путем создания ram.ini файла следующего содержания:

```
(*((volatile unsigned long  *)(0x4001E008))) |= 0x1;
FUNC void Setup (unsigned int region) {
       region &= 0xFFFFF000;
       SP = _RDWORD(region); // Setup Stack Pointer
       PC = _RDWORD(region + 4); // Setup Program Counter
       _WDWORD(0xE000ED08, region); // Setup Vector Table Offset Register
}
Setup(0x8000000); // Get ready to execute image in SRAM or whatever region it is in G,main
```

Этот файл будет автоматически исполняться микроконтроллером перед запуском средства отладки.

На той же вкладке необходимо зайти в настройки адаптера, в данном примере это «J-LINK/J-TRACE Cortex».

При первом входе в данные настройки, может появиться сообщение

Рисунок 6.2

Необходимо нажать кнопку «ОК». Откроется меню выбора устройства и ядра:

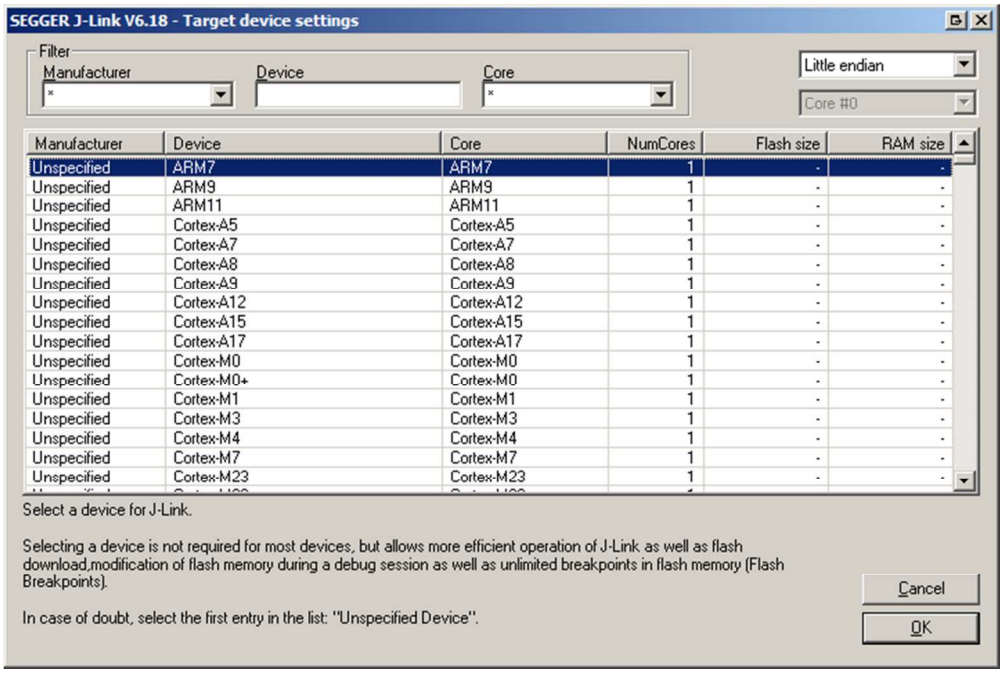


Рисунок 6.3

С помощью фильтра выбрать ядро «Cortex-M4»

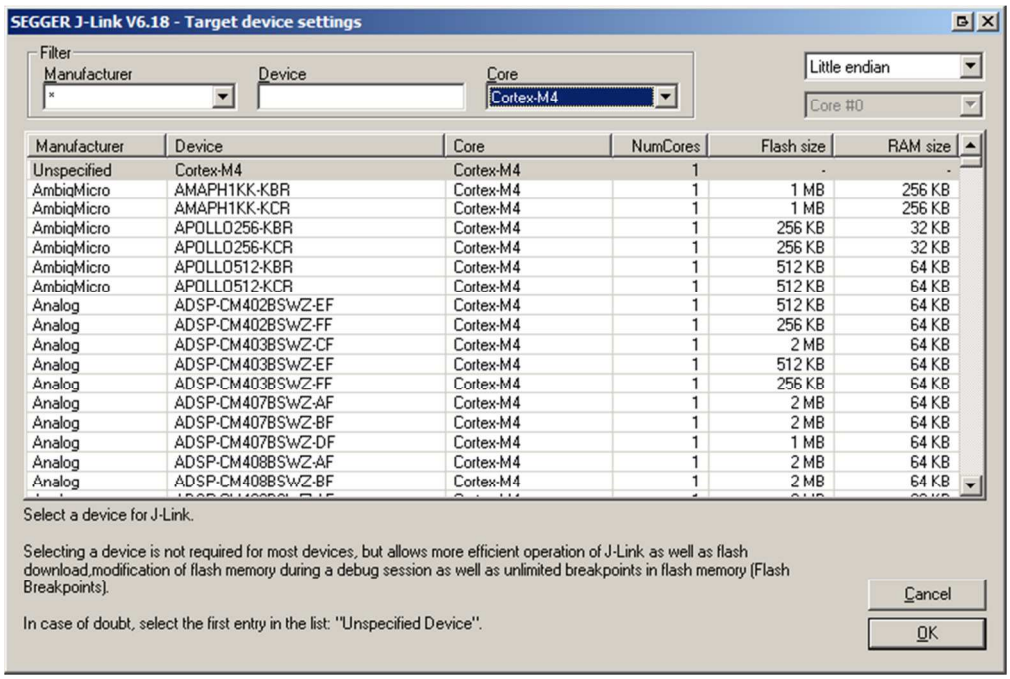


Рисунок 6.4

Необходимо выбрать первую строку «Unspecified / Cortex-M4 / Cortex-M4 / 1 / - / - » и нажать кнопку «ОК»

Далее необходимо выбрать порт загрузки (Поддерживаются режимы JTAG и SW. Переключение между портами производится с помощью перемычек на плате). Во всех примерах далее используется SW.

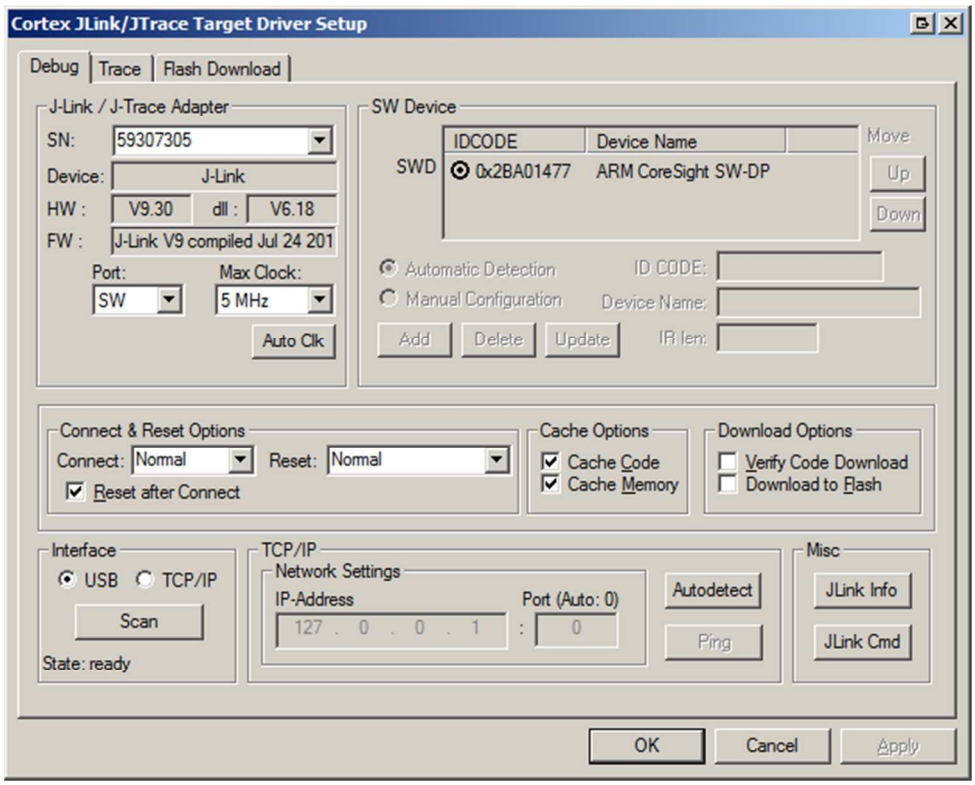


Рисунок 6.5

В данном окне выбрать вкладку Flash Download и выставить настройки согласно рисунку 6.6:

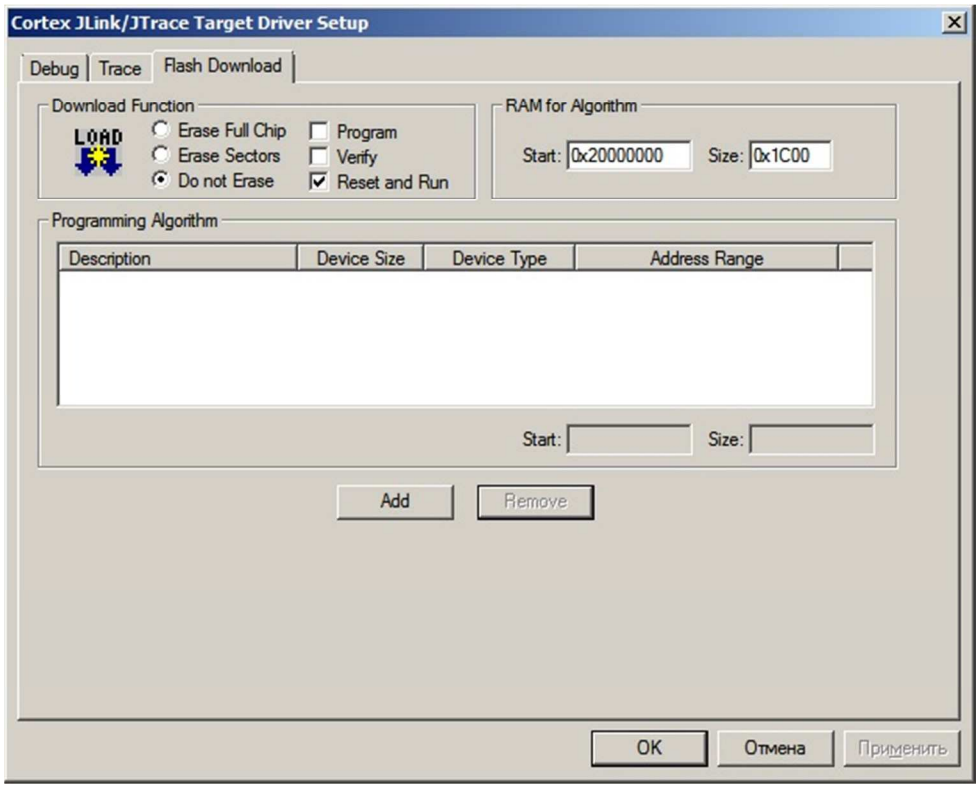


Рисунок 6.6

Далее необходимо настроить адресацию для линковщика.

Согласно п 5.2 технического описания: PROG SRAM расположена по адресу 0x08000000 и имеет размер 128кб, а SRAM по адресу 0x20000000, её размер 64кб.

В настройках проекта необходимо указать адресацию, но регистры ПДП в библиотеках расположены в конце ОЗУ и занимают 1024 байта. Их нужно исключить из обработки линковщика.

IROM 0x8000000, размер 0x4000. IRAM 0x20000000, размер 0x1C00 (0x2000 – 0x400 на ПДП)

А так-же указать частоту работу микроконтроллера.

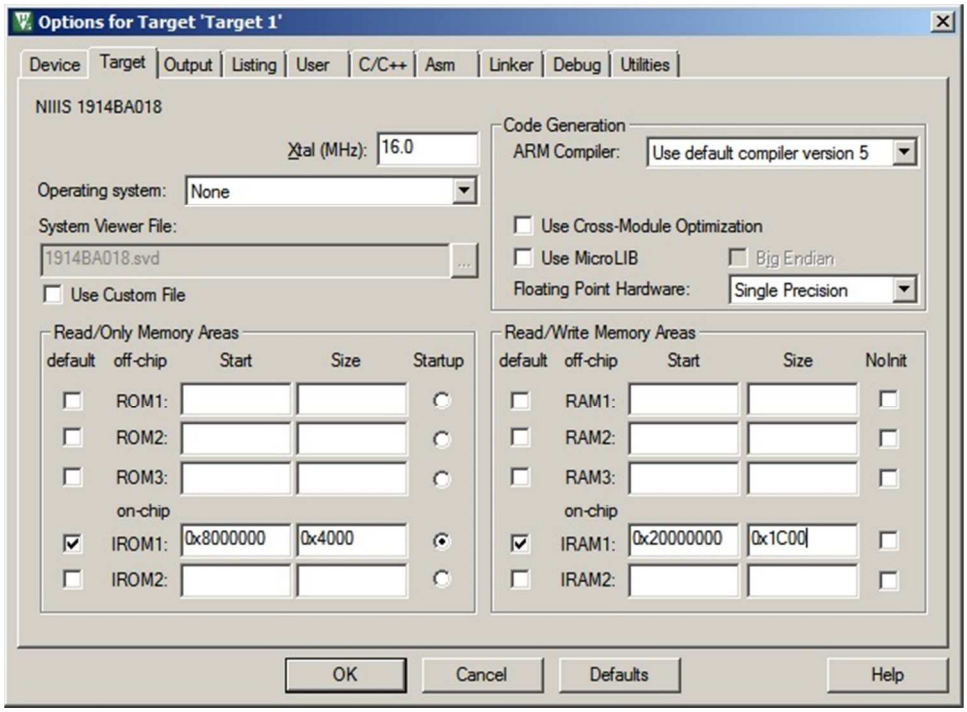


Рисунок 6.7

Необходимо нажать ОК и скомпилировать программу (Rebuild)

В окне Build Output не должно возникать ошибок.

Следующим шагом необходимо запустить Debug->Start или использовать сочетание клавиш «Ctrl + F5»

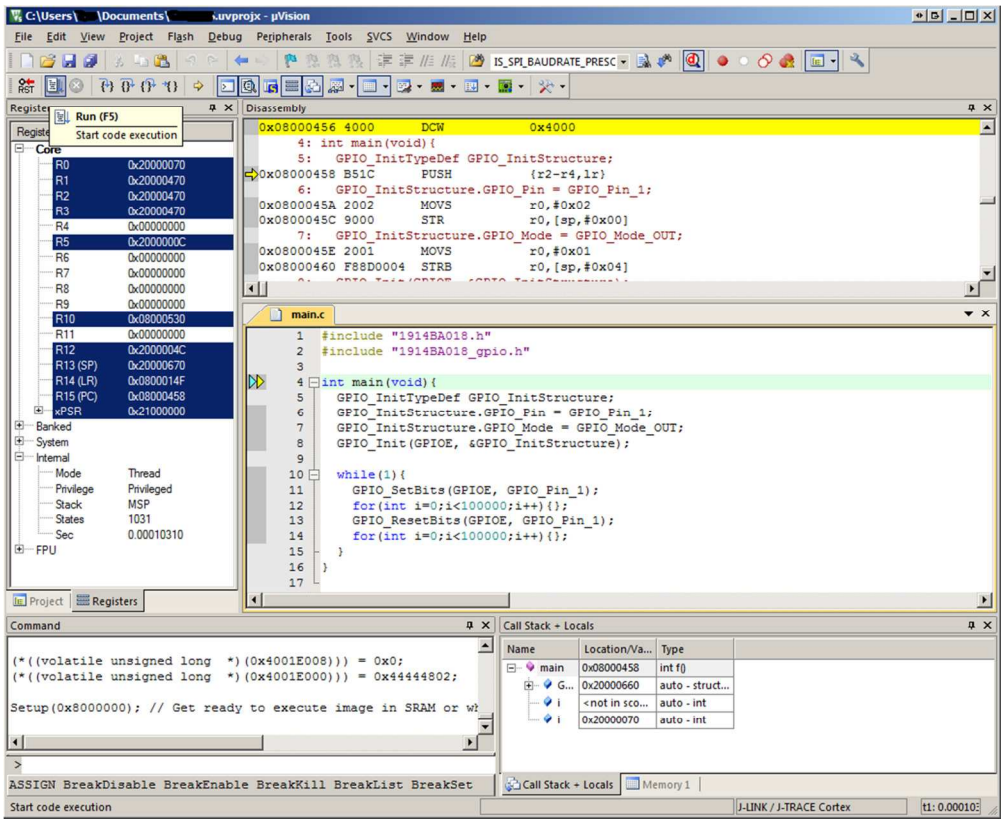


Рисунок 6.8

Теперь необходимо запустить программу (Run), нажав соответствующую иконку или клавишу F5.

7. Загрузка программы во внешнее ПЗУ

7.1 Загрузчик

Загрузчик микроконтроллера имеет несколько режимов работы (см. п.7 технического описания) с памятью на внешней шине данных.

Помимо перечисленных режимов работы есть еще режим REMAP (п. 7.1.2 технического описания).

Для наглядного восприятия, представим режимы работы внутреннего загрузчика (таблица 7.1 технического описания) в следующем виде:

	Шина в 32-битном режиме	Шина в 8-битном режиме
Непосредственное выполнение программы из внешней памяти	**PE[15:13] = 000** Микроконтроллер конфигурирует внешнюю шину на работу в 32-битном режиме со значением WAIT_STATES = 4 и начинает выполнять программу из внешней памяти.	**PE[15:13] = 100** Микроконтроллер конфигурирует внешнюю шина на работу в 8-битном режиме со значением WAIT_STATES = 4 и начинает выполнять программу из внешней памяти.
Загрузка памяти из внешней шины	**PE[15:13] = 010** Микроконтроллер конфигурирует внешнюю шину на работу в 32-битном режиме со значением WAIT_STATES = 4, копирует пользовательскую программу из внешней памяти во внутреннее ОЗУ и передаёт ей управление.	**PE[15:13] = 001** Микроконтроллер конфигурирует внешнюю шину на работу в 8-битном режиме со значением WAIT_STATES = 4, копирует пользовательскую программу из внешней памяти во внутреннее ОЗУ и передаёт ей управление.

Таблица 7.1.1

Среда Keil позволяет работать с памятью на внешней системной шине, используя ОЗУ подключенного микроконтроллера (см Рисунок 6.6, RAM for Algorithm).

В данную область памяти загружается программа, содержащая алгоритм. В примерах использовалась память AM29LV400B в 8-битном режиме, две микросхемы памяти AM29LV160D в 16 битных режимах для организации 32 битной памяти, а также СОЗУ 1658ру2t.

Алгоритмом для работы с памятью является файл *.FLM, который находится в папке ../FLASH

7.2 Режим работы шины.

Встроенный загрузчик позволяет работать с внешней системной шиной только в 2-х режимах: 8-битном и 32-битном. Перед началом загрузки выбранным алгоритмом, необходимо переконфигурировать внешнюю системную шину в нужный режим. Для

этого в настройках Target (Открыть настройки Target. Нажать сочетание клавиш Alt+F7 или Project -> Options for Target) выбрать вкладку Utilities.

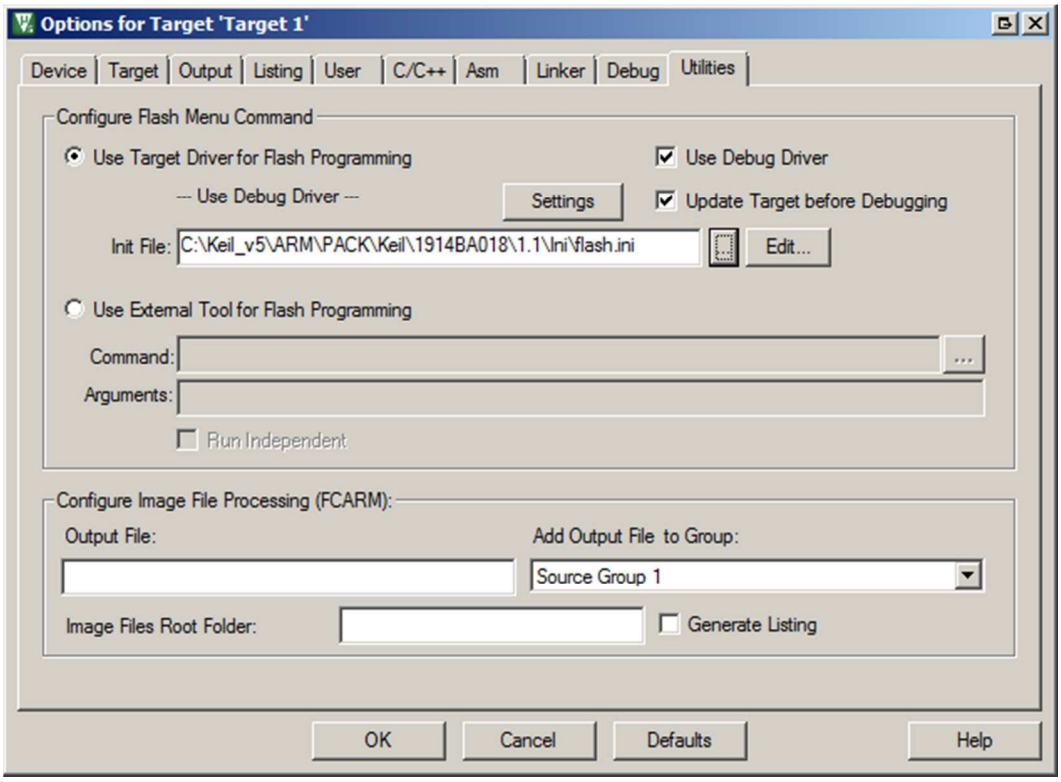


Рисунок 7.2.1

Настроим конфигурацию внешней системной шины перед исполнением алгоритма загрузки в микросхему памяти путем создания flash.ini файла следующего содержания:

```
(*((volatile unsigned long  *)(0x4001E008))) = 0x0;
(*((volatile unsigned long  *)(0x4001E000))) = 0xFFFFFF02;
```

Для конфигурации в 32-битном режиме или

```
(*((volatile unsigned long  *)(0x4001E008))) = 0x1;
(*((volatile unsigned long  *)(0x4001E000))) = 0xFFFFFF02;
```

Для конфигурации в 8-битном режиме.

По адресу 0x4001E008 располагается регистр EBC->LOW8 (таблица 16.9 технического описания), с помощью которого происходит конфигурация режимов работы для соответствующих регионов памяти.

По адресу 0x4001E000 располагается регистр EBC->CONTROL (таблица 16.6 технического описания), с помощью которого происходит управление внешней системной шиной. В данном регистре нас интересует длительность фаз обращения для региона памяти (таблица 16.2 технического описания). В примерах используется максимальная длительность фаз обращений.

Примечание. При работе с внешней микросхемой памяти, важно учитывать скорость ее работы и длину соединительных линий.

Например, используемая в примерах память AM29LV400B в 8-битном режиме. По документации работает на частоте 12мгц. Стандартная длительность фаз (после работы

загрузчика) для любого региона выборки 0x4 (расшифровку см. в таблице). При конфигурации PLL выше 48мгц, микроконтроллер начинает считывать данные с ошибками. Корректировка выборки до 0x5 позволяет стабильно работать до 60мгц включительно.

7.3 Способ исполнения программы

7.3.1 Непосредственное исполнение программы из внешней шины. Необходимо настроить линковщик на адрес внешней памяти, в данном случае, это 0x10000000 и указать её размер.

Настройка линковщика для AM29LV400B (4мбит):

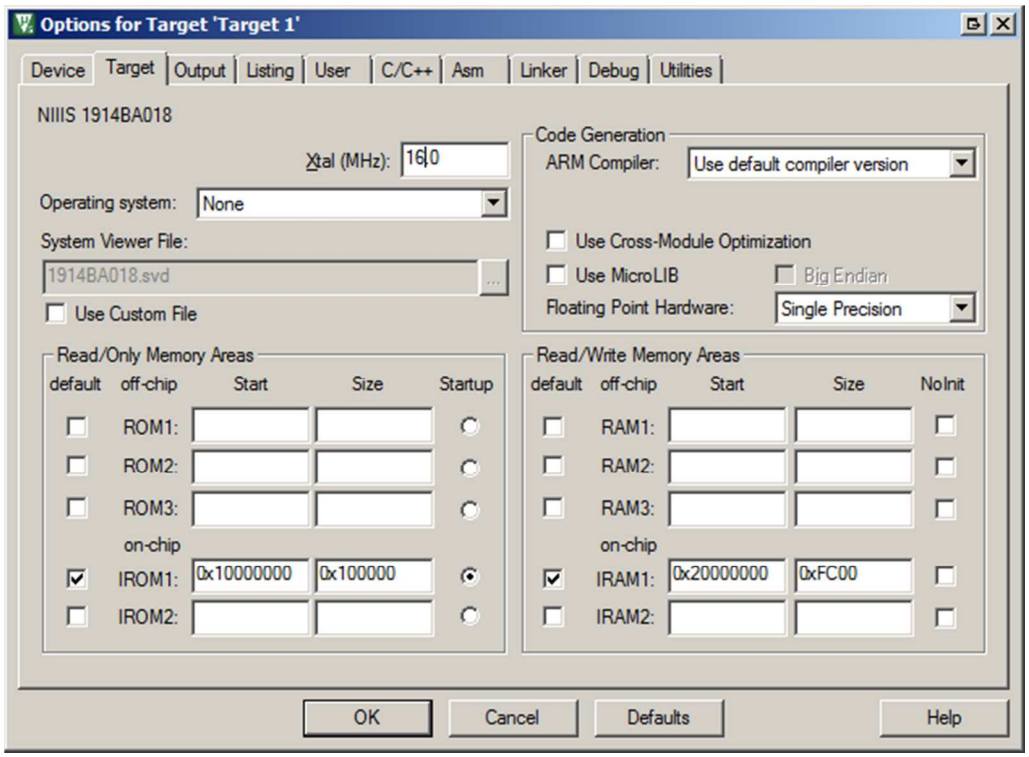


Рисунок 7.3.1.1

настройка линковщика для двух AM29LV160D (2x16мбит)

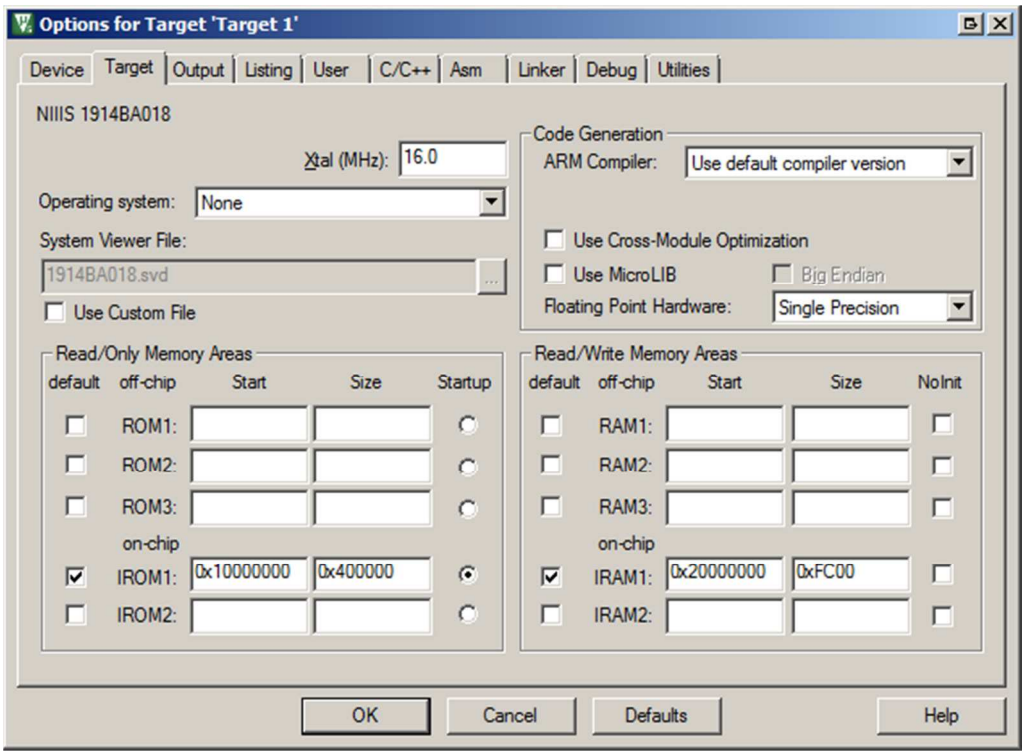


Рисунок 7.3.1.2

Далее во вкладке Debug необходимо настроить адаптер (J-LINK)

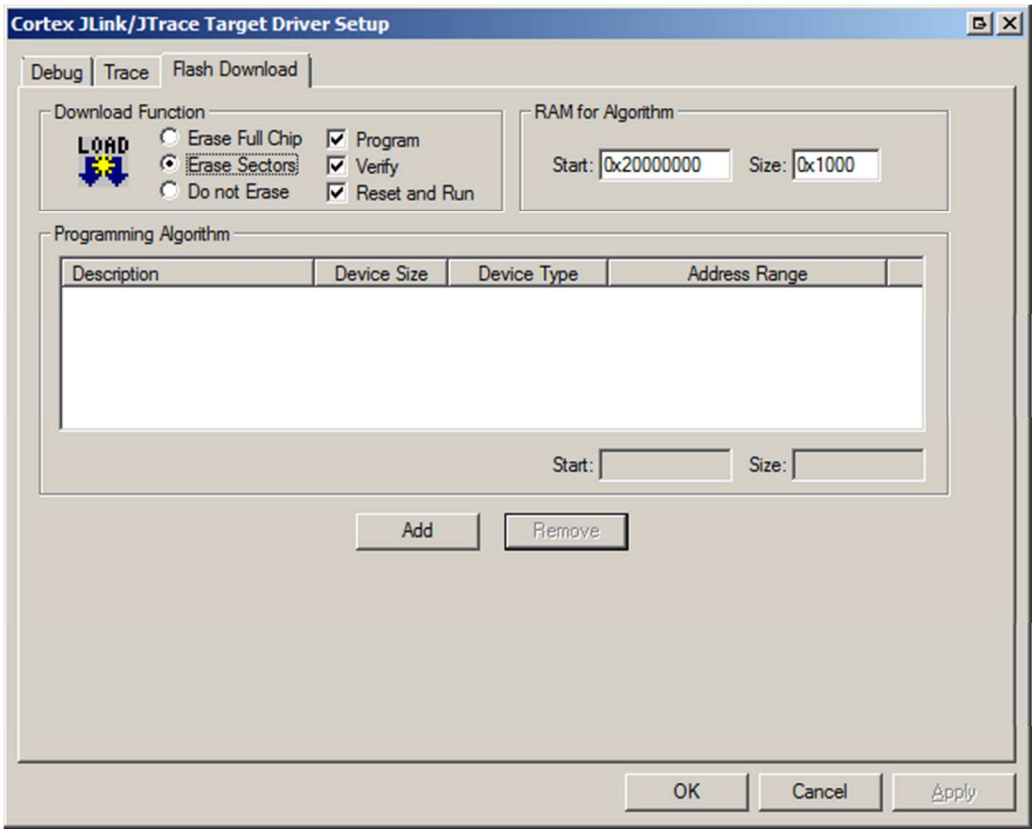


Рисунок 7.3.1.3

В разделе Download Function нами использованы настройки, приведенные на рисунке 7.3.1.3

После нажатия кнопки «Add», появится окно выбора алгоритмов программирования памяти.

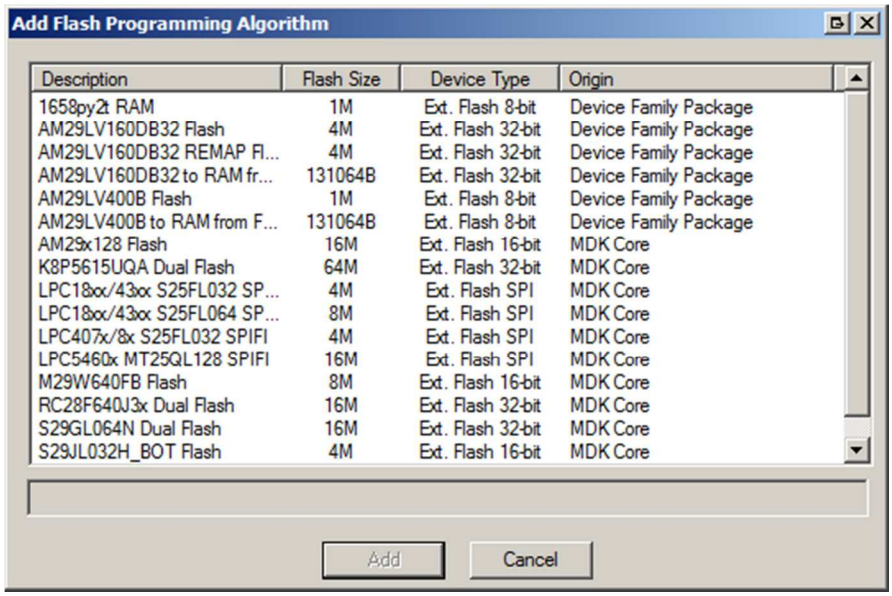


Рисунок 7.3.1.4

В качестве примера выберем AM29LV160DB32 (2 микросхемы памяти), как показано на рисунке 7.3.1.5. Стартовый адрес такой, как и в линковщике, 0x10000000, размер 0x400000

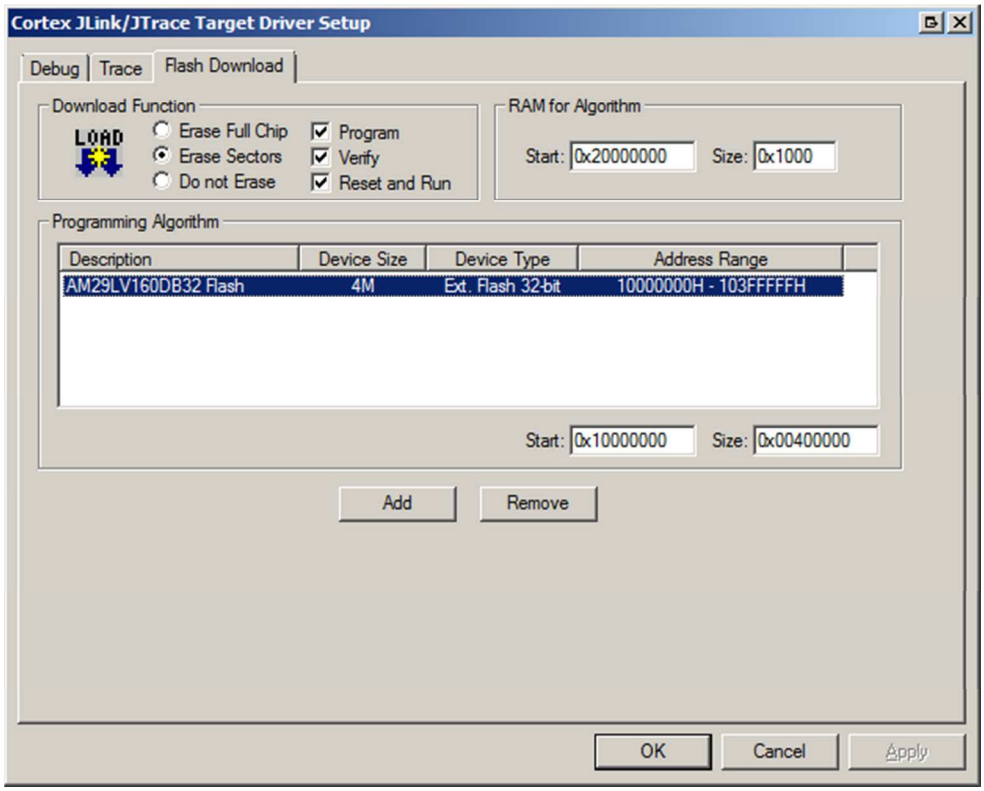


Рисунок 7.3.1.5

Осталось загрузить программу. Для этого можно нажать клавишу F8 или в меню Flash->Download.

7.3.2 Копирование программы из внешней шины в ОЗУ и передача управления.

Данный режим работы подробно описан в п.7.1 технического описания.

Для удобной работы, в PACK содержатся алгоритмы для программирования микросхемы памяти в данный режим.

Программа из внешней памяти перекачивается в ОЗУ по адресу 0x8000000 с передачей управления, значит, линковщик должен быть настроен на данный адрес.

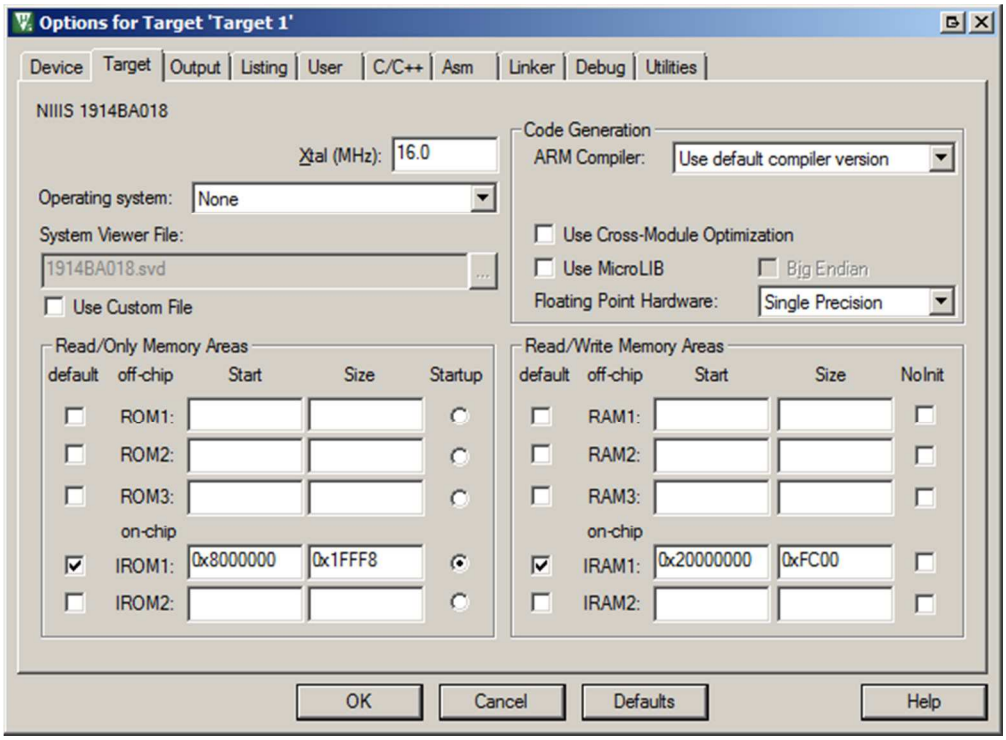


Рисунок 7.3.2.1

Размер программы 128 кб-8 байт (размер программы + адрес, см техническое описание), что составляет 0x1FFF8.

В настройках адаптера во вкладке Flash Download выбираем алгоритм с пометкой «toRAM from Flash»

Стартовый адрес, как в линковщике 0x8000000, адрес 0x1FFF8. См. рисунок 7.3.2.2

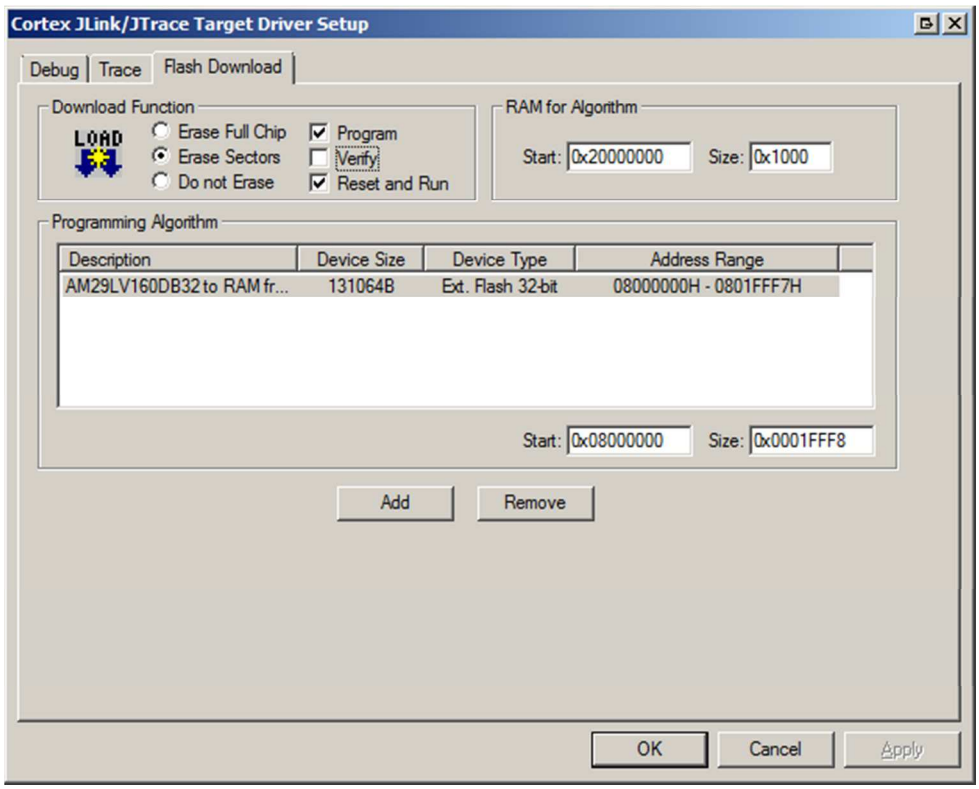


Рисунок 7.3.2.2

Примечание: физически микросхема памяти находится по адресу 0x10000000, но в алгоритме программирования адрес программирования подменяется с 0x8000000 на 0x10000008. В первые 8 байт пишется служебная информация. Потому функция «Verify» работать сразу после прошивки не будет, т.к. она проверяет наличие программы по адресу 0x8000000, а не по 0x10000008. Но после того, как программа была перекачана в ОЗУ, можно проверить целостность программы следующим образом: Настроить алгоритм программирования, как указано на рисунке 7.3.2.3

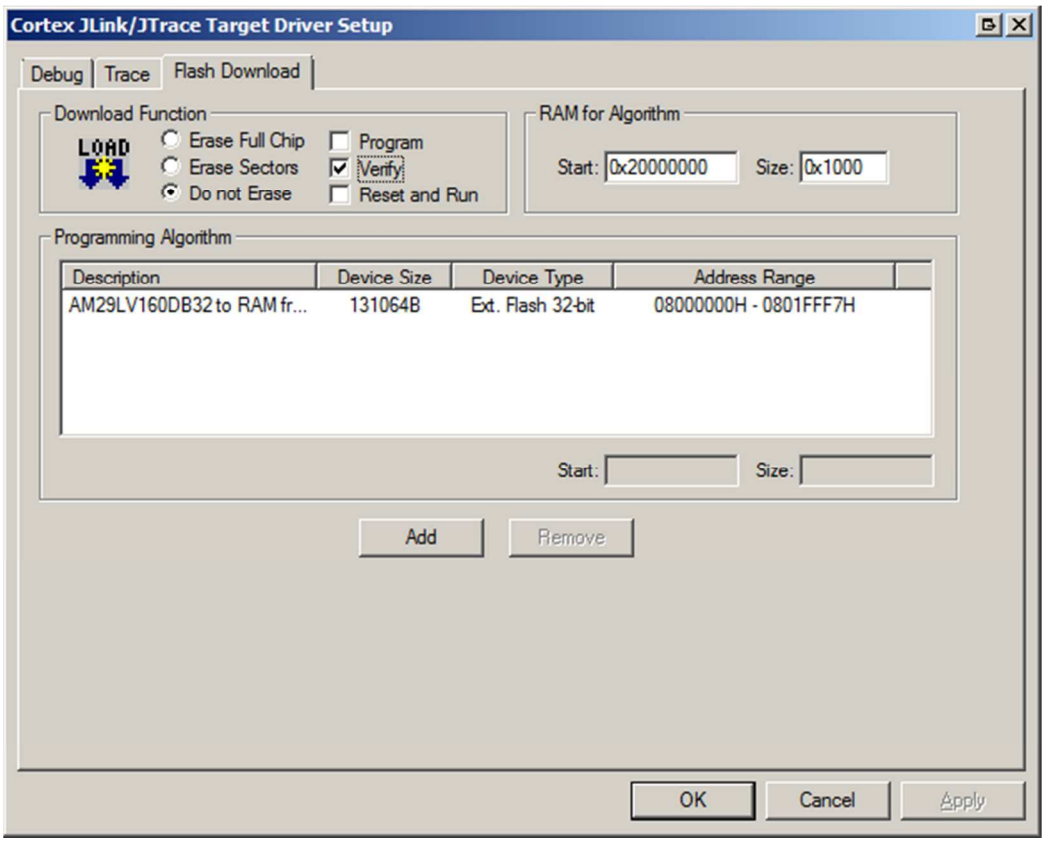


Рисунок 7.3.2.3

И запустить загрузку (нажать клавишу F8 или в меню Flash->Download). Из настроек на вкладке Flash Download видно, что очистка и запись памяти не происходит.

После выполнения загрузчиком перекачки программы в ОЗУ, Keil выполнит сравнение скомпилированной программы и программы в ОЗУ, что подтвердит её целостность.

Remap – использование своего загрузчика. В PACK присутствует алгоритм загрузчика для двух AM29LV160DB32. Данный алгоритм имеет префикс REMAP.

Ограничение исполняемой программы - 4кбайта.

Примечание: исходные данные всех алгоритмов содержатся в папке Flash PACK-файла.

8. Загрузка программы по UART

Для загрузки по UART из среды Keil используется приложение bootstrap. (оно содержится в папке PACK по адресу ..\boot\bootstrap.exe (для удобства использования, можно скопировать bootstrap.exe в папку с проектом)

Для его выбора, необходимо произвести настройки, представленные на рисунке 8.1:

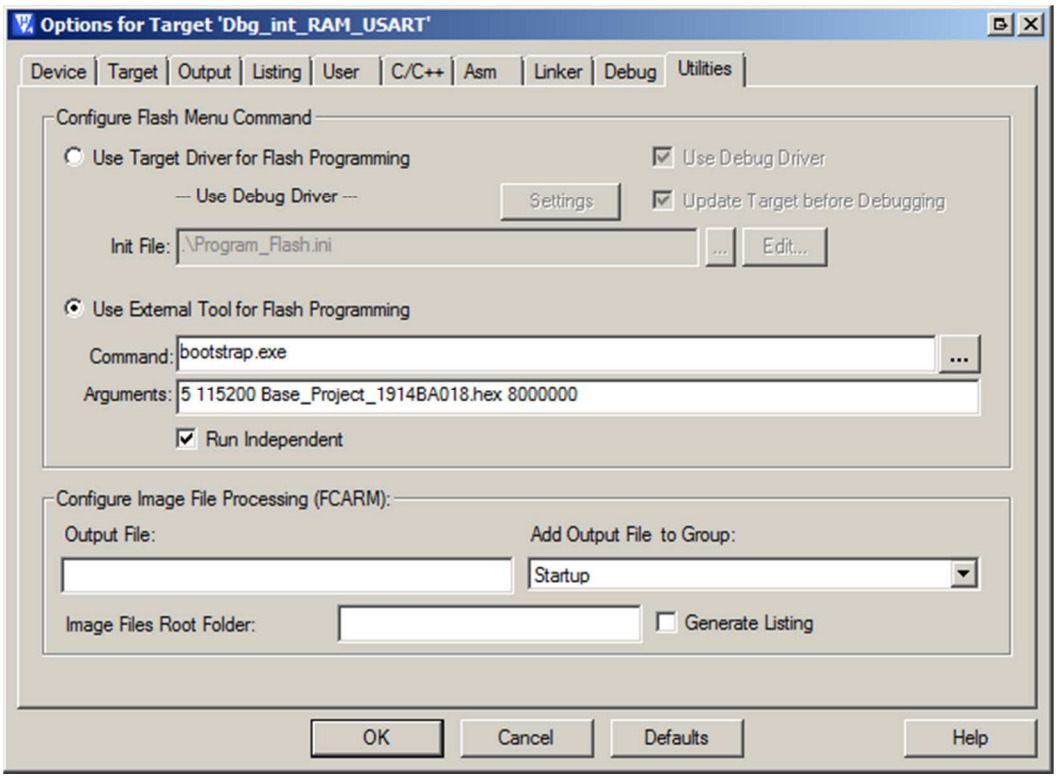


Рисунок 8.1

На вкладке Utilities выбрать «Use External Tool for Flash Programming»

В строке Command указать ссылку на bootstrap.exe (можно использовать относительные пути). В примере, данное приложение лежит в папке с проектом.

В строке «Arguments» необходимо указать 4 аргумента через пробел:

1. Номер порта. В примере это Com5. Первым аргументом идет его номер 5.
2. Скорость загрузки по UART. Микроконтроллер автоматически настраивает скорость, используя внутренний таймер TIM1.
3. Ссылка на hex файл прошивки. В данном случае, он лежит в папке с проектом.
4. Адрес памяти, начиная с которого, будет располагаться программа. Он же является адресом, куда будет передано управление после загрузки (окончания работы bootloader-а). Адрес должен совпадать с адресом, указанным в линковщике.

Для индикации работы bootstrap, необходимо установить галочку «Run Independent»

Для получения hex файла с необходимым форматированием для bootstrap, необходимо добавить инструкцию

«путь»\Keil\ARM\ARMCC\bin\fromelf.exe #L --vhx --8x1 --output

Base_Project_1914BA018.hex

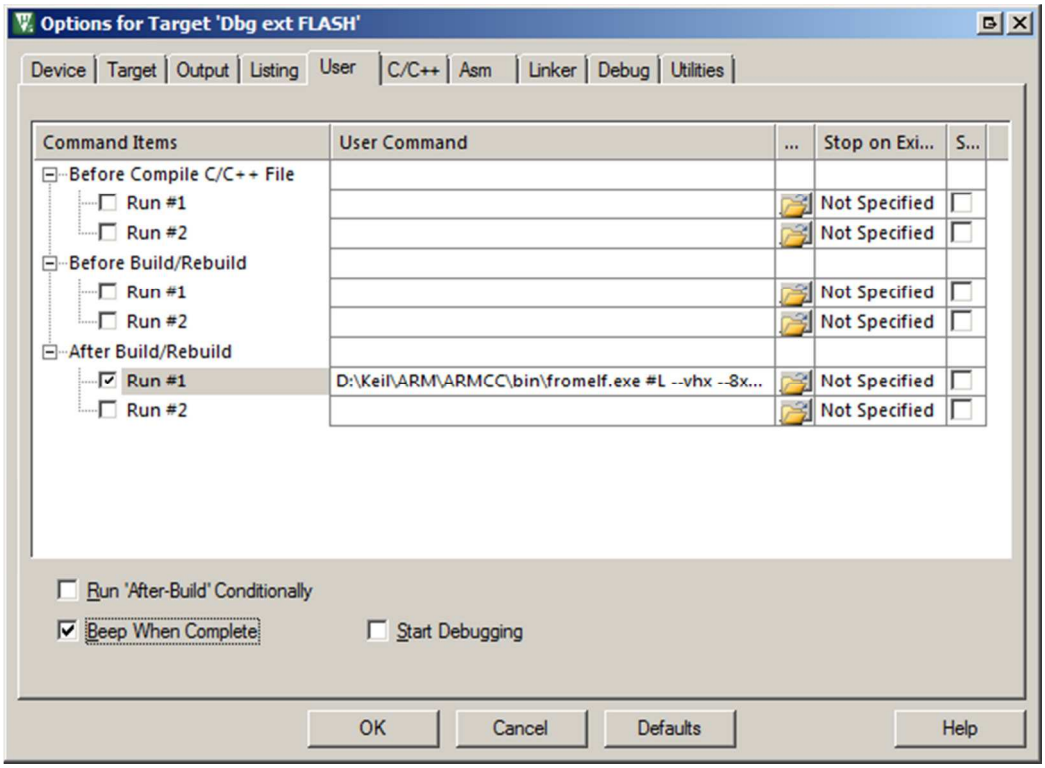


Рисунок 8.2

Линковщик необходимо настроить для исполнения программы в ОЗУ. Начальный адрес программы 0x8000000, размер 0x20000 (128кбайт).

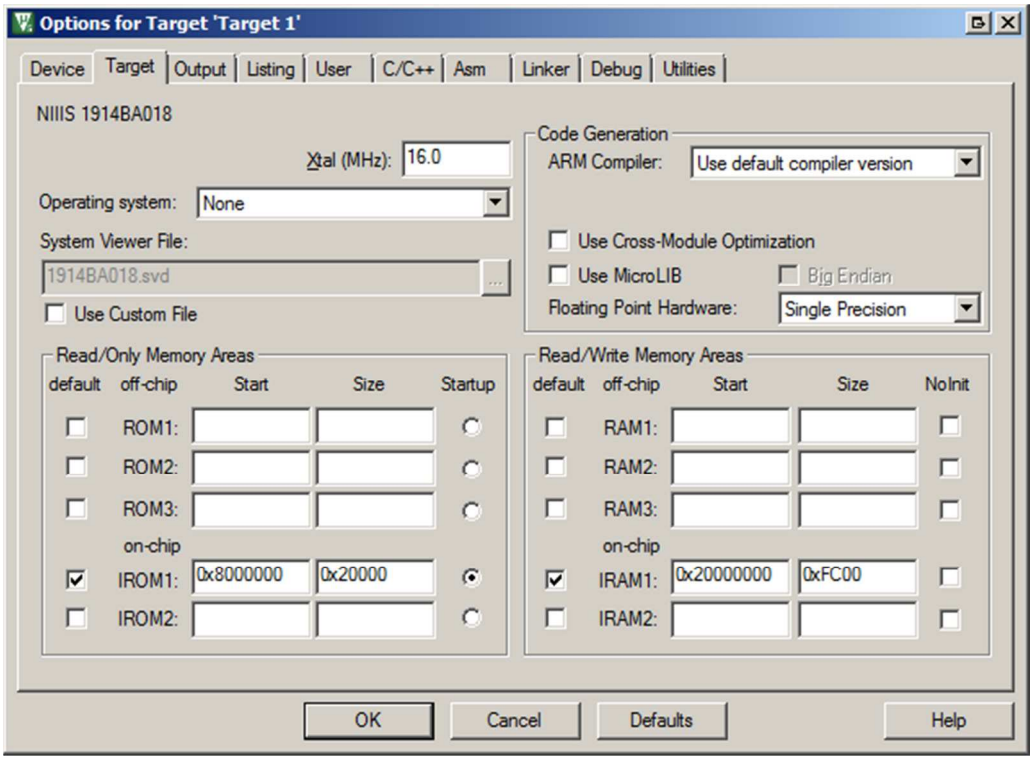


Рисунок 8.3

После сохранения настроек можно начать загрузку программы

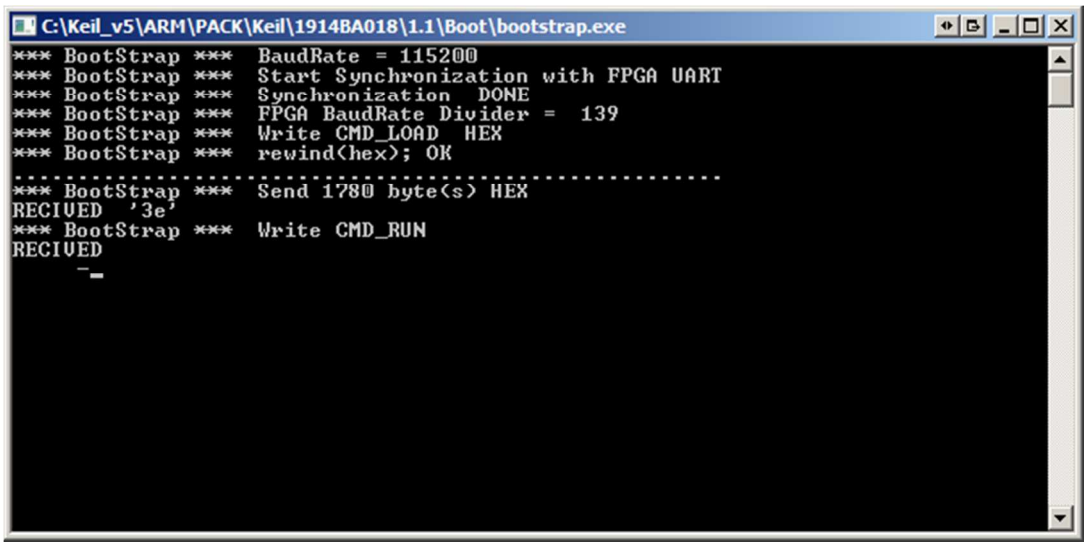


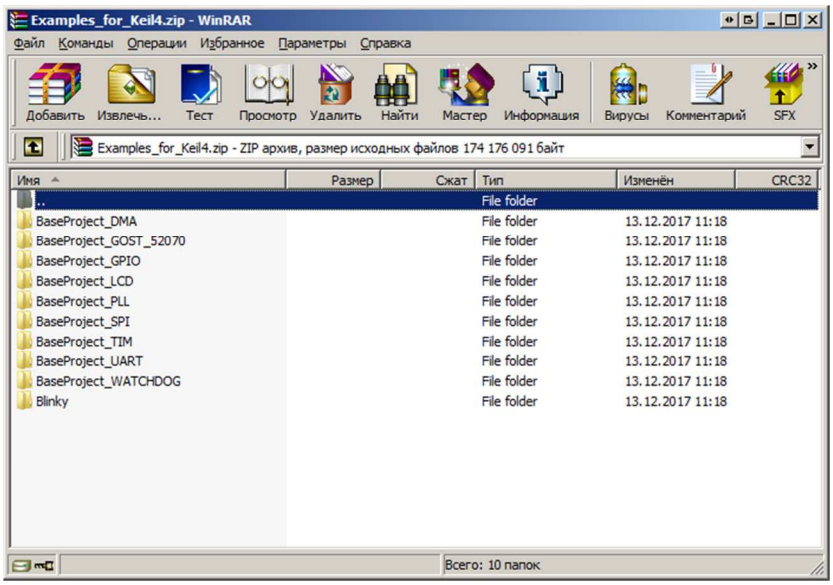
Рисунок 8.4

При использовании USB-TTL преобразователя PL2303 или аналогов, bootstrap перед загрузкой программы, генерирует импульс по линии DTR#, что позволяет ее использовать в качестве сброса микроконтроллера на отладочной плате.

9. Примеры готовых проектов

9.1 Keil 4

В архиве «Examples_for_Keil4» содержатся папки с готовыми и настроенными проектами для Keil4. Для работы с интересующим проектом, необходимо распаковать соответствующую папку с проектом и запустить файл проекта.



Примечание: базовые проекты Blink рассчитаны на 3 режима загрузки (в ОЗУ, по UART, во внешнюю ПЗУ), все остальные - настроены на работу в ОЗУ.

9.2 Keil 5

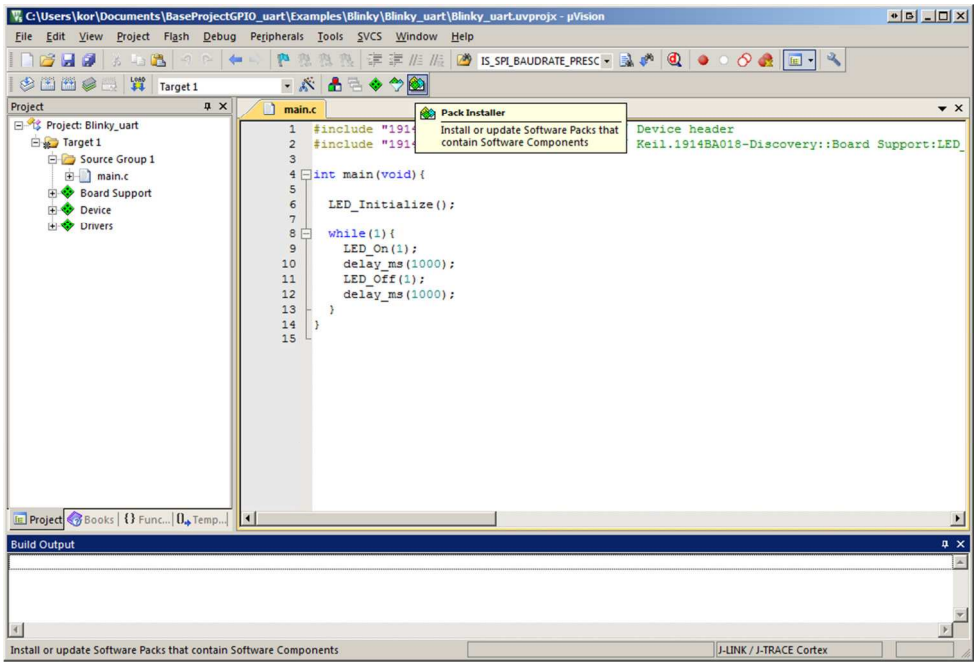


Рисунок 10.2.1

При нажатии на кнопку «Pack Installer», появится окно

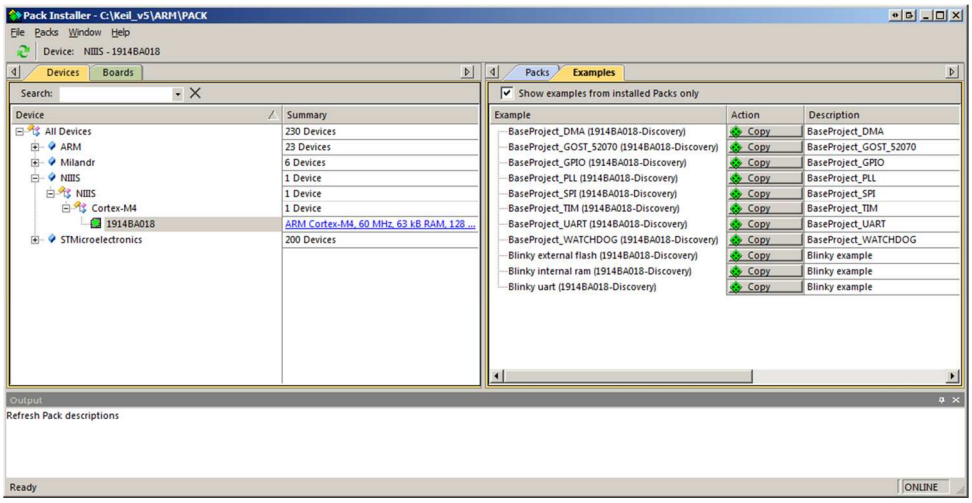


Рисунок 10.2.2

В данном окне, если в левой части, среди «Devices», выбрать интересующий контроллер, а в правой открыть вкладку Examples, то появится список готовых проверенных проектов по каждой библиотеке, а также 3 режима загрузки базовой программы Blink.

При нажатии кнопки «Copy», среда Keil запрашивает указать папку, в которой будет размещен проект.

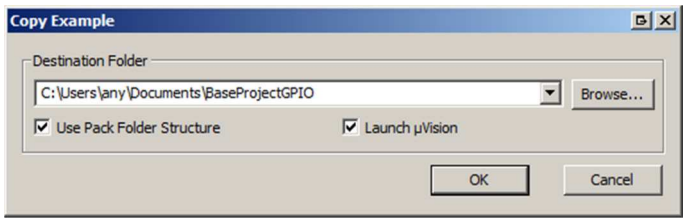


Рисунок 10.2.3

После нажатия кнопки «ОК», готовый базовый проект скопируется в указанную папу и откроется в среде Keil.

Примечание: базовые проекты Blink рассчитаны на 3 режима загрузки (в ОЗУ, по UART, во внешнюю ПЗУ), все остальные - настроены на работу в ОЗУ.

10. Периферия.

В среде Keil имеется возможность работы с периферией. Для среды Keil v5 и выше, все настройки уже произведены при распаковке PACK-файла.

Если вы используете Keil v4, то необходимо в настройках проекта вручную подключить *.SVD файл.

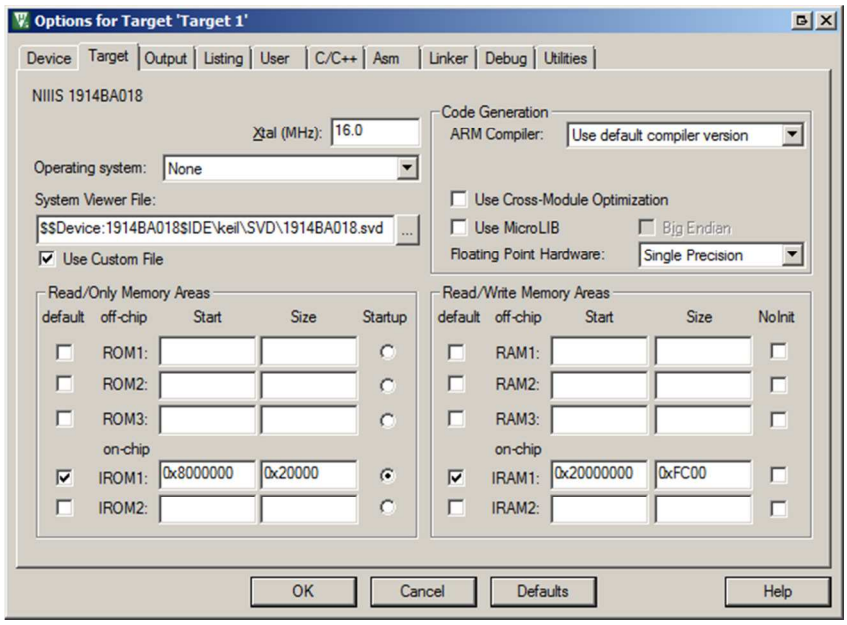


Рисунок 11.1

В режиме отладки можно выбрать Peripherals->System Viewer, затем интересующий интерфейс и номер интерфейса.

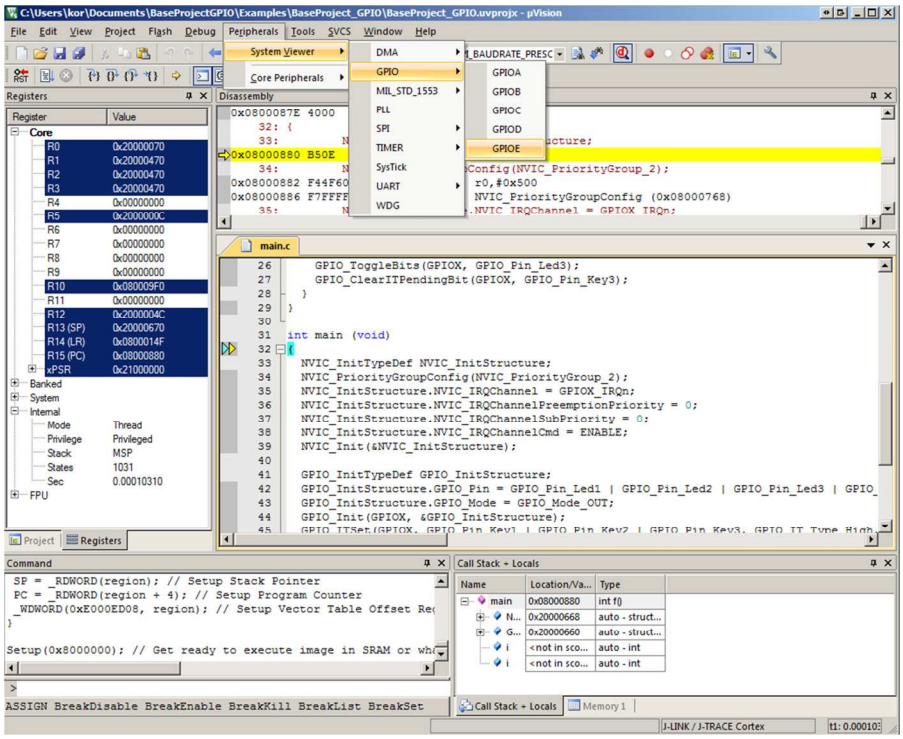


Рисунок 11.2

Справа появится окно с регистрами, которые относятся к данному интерфейсу и их состояние в данный момент паузы программы.

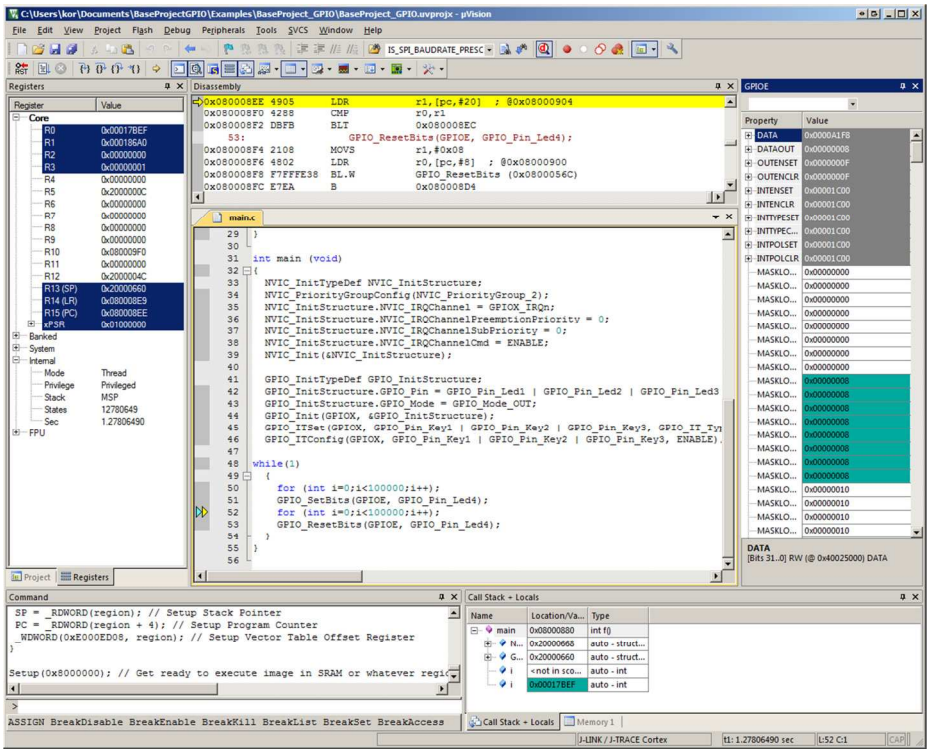


Рисунок 11.3