Лабораторная работа №1
«Порты ввода-вывода микроконтроллера семейства STM32. Средства отладки и симуляции»

1 Цель работы

Ознакомиться со средой разработки Keil uVision: встроенными средствами разработки, отладки и симуляции; освоить программные средства работы с портами ввода-вывода микроконтроллера STM32.

2 Теоретические сведения

2.1 Среда разработки Keil uVision

Keil uVision позволяет работать с проектами любой степени сложности, начиная с введения и правки исходных текстов и заканчивая внутрисхемной отладкой кода и программированием ПЗУ микроконтроллера.

uVision Debugger позволяет вести отладку исходных текстов программ, написанных на C и ассемблере или в смешанном формате, сохраняет историю трассировки и позволяет выбирать между симулятором, монитором и внутрисхемным эмулятором. uVision Simulator - программный продукт, который осуществляет отладку в исходных кодах, симуляцию на уровне символов и отладку непосредственно на рабочей плате - *target debugging*. Моделируется вся система команд и все периферийные устройства, с некоторыми ограничениями. Симулятор полностью поддерживает периферийные устройства микроконтроллера посредством специальных драйверов - для STM32F103RB драйвера содержатся в библиотеке DARMSTM.DLL.

В распоряжение пользователя предоставляется ряд окон, отображающих состояния таймеров, портов, прерываний, сторожевого таймера, последовательного порта, аналогово-цифрового преобразователя и т.д. Параметры этих устройств могут быть установлены и изменены в соответствии с контекстом приложения через вкладку *Peripherals*.

uVision Simulator позволяет проводить пошаговую отладку программы, просматривая ее в окне *Debug*. Трассировщик запоминает команды и позволяет их просматривать в окне *Trace*. Изменение заранее заданных переменных отслеживает окно *Watch*. Последовательность вызова процедур отображается в окне *Call-Stack*.

2.2 Микроконтроллер STM32F103RB

Базовое семейство STM32F1 имеет сбалансированные характеристики и компромиссные значения производительности, потребления и цены. STM32F1 – исторически первая линейка 32-битных микроконтроллеров от ST, построено на ядре Cortex-M3 с рабочей частотой до 72 МГц.

STM32F103RB - 32-разрядный микроконтроллер средней производительности на базе Cortex-M3 с 64 или 128 КБ флэш-памяти, USB, CAN, 7 таймерами, 2 аналого-цифровыми преобразователями, 9 последовательными интерфейсами.

2.3 Типы данных

Все действия в программе производятся над переменными. Переменные должны быть объявлены с указанием типа данных до использования в программе. Тип данных задает параметры переменных. Некоторые базовые типы данных стандартов языка программирования Си зависят от используемого микроконтроллера. В значительной мере от разрядности данных, с которыми он оперирует.

Имена целочисленных типов данных образуются из символов:

– *int* –целое знаковое;

– *uint* –целое беззнаковое (к int добавили u);

– число разрядов;

– *_t* – тип.

Формат таких типов данных не зависит от разрядности микроконтроллера:

– *int8_t* – целое знаковое 8 разрядов;

– *uint8_t* – целое беззнаковое 8 разрядов;

– *int16_t* – целое знаковое 16 разрядов;

– *uint16_t* – целое беззнаковое 16 разрядов;

– *int32_t* – целое знаковое 32 разрядов;

– *uint32_t* – целое беззнаковое 32 разрядов;

– *int64_t* – целое знаковое 64 разрядов;

– *uint64_t* – целое беззнаковое 64 разрядов.

Для хранения кодов символов допускается использование типа *char*.

Типы с плавающей запятой:

– *float* - формат с плавающей запятой одинарной точности;

– *double* - формат с плавающей запятой двойной точности;

– *long double* - формат с плавающей запятой повышенной точности.

2.4 Библиотеки CMSIS и SPL

CMSIS (Cortex Microcontroller Software Interface Standard) - это стандартная библиотека содержит обозначение регистров микроконтроллера, областей памяти, битов, векторов, прерываний. В файле stm32f10x.h регистры микроконтроллера объявлены как указатели на структуры, поэтому обращаться к регистрам надо как к элементам структуры через указатели. Для битов регистров в файле stm32f10x.h содержатся имена и битовые маски.

Пример установки битового значения в регистр:

GPIOB -> BSRR = GPIO_BSRR_BS13;

В результате работы с регистрами прямым обращением через имена библиотеки CMSIS, получаются самые компактные, быстрые программы.

Библиотека SPL (Standard Peripheral Library) - базовая библиотека для работы с периферийными устройствами. Файлы библиотеки содержат функции, структуры и определения, которые могут потребоваться при обращении к периферийным устройствам.

За тактирование в SPL отвечают файлы stm32f10x_rcc.c и stm32f10x_rcc.h, за работу с портами ввода-вывода - stm32f10x_gpio.h и stm32f10x_gpio.c.

2.5 Управление тактированием

Для работы периферийных устройств микроконтроллера необходимо разрешить тактирование устройства через регистр разрешения тактирования периферийных устройств соответствующей шины APB - регистры APB1 peripheral clock enable register (APB1ENR) и APB2 peripheral clock enable register (APB2ENR). Информацию о том, к какой шине подключено устройство, можно получить из технической спецификацией микроконтроллера либо из библиотечного файле stm32f10x_rcc.h.

Пример включения тактирования порта ввода-вывода B с использованием прямого обращения к регистрам через библиотеку CMSIS:

RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;

Пример включения тактирования порта ввода-вывода A с использованием функции библиотеки SPL:

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

2.6 Управление портами ввода-вывода

В системе STM32 применяется общепринятое название портов ввода-вывода общего назначения, а именно GPIO (General purpose input-output), и идентификатор порта: A, B, C или D. Состояние выводов устанавливается или считывается программно.

Для каждого порта (16 выводов) есть два 32х разрядных регистра конфигурации. Они образуют 64 битный регистр конфигурации порта.

Младший регистр конфигурации портов GPIOx_CRL (Port configuration register low) (x - идентификатор порта A, B, C или D) показан на рис.1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис.1

Старший регистр конфигурации портов GPIOx_CRH (Port configuration register high) показан на рис.2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис.2

На каждый вывод отводится 4 бита, которые делятся на 2-битные поля режима (MODE) и конфигурации (CNF). Поле режима определяет направление работы вывода – вход или выход. В случае выхода оно задает максимальную частоту передачи выходного сигнала, что влияет на энергопотребление микроконтроллера.

Состояние регистров конфигурации можно защитить от непредусмотренных изменений. Для этого существует регистр блокировки конфигурации GPIOx_LCKR. Каждому выводу порта соответствует бит блокировки LCK0– LCK15. При установке бита в 1 запрещается изменение соответствующих битов режима и конфигурации. После задания всех нужных битов регистра блокировки необходимо активизировать защиту. Для этого надо в 16 бит регистра блокировки (LCKK) последовательно записать 1, 0, 1. После этого блокировка будет действовать и изменение защищенных битов конфигурации и режима возможно только после сброса микроконтроллера.

Доступ к портам происходит через регистры состояния ввода GPIOx_IDR (Port input data register), показан на рис.3, и состояния вывода данных GPIOx_ODR (Port output data register), показан на рис.4.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Рис.3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис.4

Регистры GPIOx_BSSR (Port bit set/reset register), показан на рис. 5, и GPIOx_BRR (Port bit reset register), показан на рис.6, позволяют менять значения битов в регистре GPIOx_ODR напрямую, без использования битовых масок. Для установки разряда регистра GPIOx_ODR необходимо записать единицу в соответствующий разряд регистра GPIOx_BSSR, чтобы сбросить бит GPIOx_ODR - записать единицу в соответствующий разряд регистра GPIOx_BRR. Старшая часть регистра GPIOx_BSSR может использоваться для сброса бит в регистре GPIOx_ODR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Рис.5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Рис.6

Существуют следующие режимы работы выводов порта:

– работа в качестве выхода;

– работа в качестве входа;

– работа в качестве аналогового вывода;

– использование в качестве вывода выполнения альтернативной функции устройств, которые входят в состав микроконтроллера.

При работе в режиме выхода возможно использование 2 режимов: двухтактного выхода (push-pull) и открытого стока (open-drain). Режим двухтактного выхода используется в большинстве случаев, а работа в режиме открытого стока необходимо, например, при использовании интерфейса передачи данных I^2C. К каждому выводу есть возможность подключения подтягивающего резистора. Есть возможность подтянуть сигнал на выходе к логическому 0 или логической 1. По умолчанию подтягивающие резисторы не используются.

Если необходимо принять данные в виде логических состояний, то необходимо перевести вывод в режим входа. При такой конфигурации появляется возможность считывать данные с выводов.

Для инициализация портов ввода-вывода используется экземпляр структуры типа GPIO_InitTypeDef. Вначале выполняется инициализация структуры, устанавливаются параметры инициализации порта: режим работы, скорость, выводы, использование подтягивающих резисторов. Возможные значения содержатся в файле stm32f10x_gpio.h.

После установки необходимых значений вызывается функция инициализации на основе значений, которые содержит структура – GPIO_Init.

2.7 Файлы сценария и сигнальные функции

Среда uVision Simulator позволяет использовать файлы сценариев, имитирующие работу внешних устройств. Файлы сценария сохраняются с расширением .ini в папке проекта и запускаются через командную строку отладчика с использованием команды *include*. После подключения файла

сценария сигнальная функция вызывается через командную строку отладчика.

Для определения сигнальных функций используется ключевое слово signal. Сигнальные функции разрабатываются в С-подобном языке. В языке доступны все внешние порты MCU, которые могут быть обозначены виртуальными обозначениями. Для получения списка обозначений внешних портов используется команда отладчика *DIR VTREG*.

Для задания временных интервалов используется функция *twatch()*, принимающая в качестве аргумента значение интервала задержки.

Пример сигнальной функции:

```
signal void input(void)
{
    while(1)
    {
        PORTA |= 0x04;
        twatch(500);
        PORTA &= ~0x04;
        twatch(500);
    }
}
```

3 Задание

Разработать программу для микроконтроллера STM32F103RB, реализующую опрос состояние входного порта ввода-вывода: при установленном значении логической единицы выполняется формирование очередного значения бита псевдослучайной последовательности и передачу его в выходной порт ввода-вывода. Передачу значений во входной порт реализовать через файл сценария с сигнальной функцией.

Параметры программы выбираются согласно варианту и приведены в табл. 1.

Таблица 1

Варианты заданий к лабораторной работе №1

Номер варианта	Полином	Входной порт	Выходной порт
1	$x^5 + x^2 + 1$	PA1	PB1
2	$x^{10} + x^3 + 1$	PA13	PB3
3	$x^9 + x^4 + 1$	PB0	PA10
4	$x^6 + x^1 + 1$	PA12	PB2
5	$x^7 + x^3 + 1$	PA11	PB1
6	$x^7 + x^3 + 1$	PA4	PB4
7	$x^{20} + x^3 + 1$	PB5	PA15
8	$x^{17} + x^3 + 1$	PB7	PA5
9	$x^{13} + x^1 + 1$	PA0	PA10
10	$x^{21} + x^2 + 1$	PA2	PB1
11	$x^{23} + x^5 + 1$	PB1	PA11
12	$x^{25} + x^3 + 1$	PA3	PA13
13	$x^{28} + x^9 + 1$	PA15	PA11
14	$x^{14} + x^1 + 1$	PA8	PB6
15	$x^{15} + x^4 + 1$	PA9	PB2
16	$x^{16} + x^7 + 1$	PB7	PA9
17	$x^{19} + x^7 + 1$	PB2	PA4
18	$x^{20} + x^3 + 1$	PB5	PB0
19	$x^{22} + x^1 + 1$	PA10	PA0
20	$x^{31} + x^3 + 1$	PA6	PB2

4 Порядок выполнения работы

4.1 Получить вариант задания у преподавателя.

4.2 Создать проект в среде Keil uVision5, для этого необходимо запустить Keil uVision5, выбрать пункт меню «Project - New uVision Project» (рис.7).

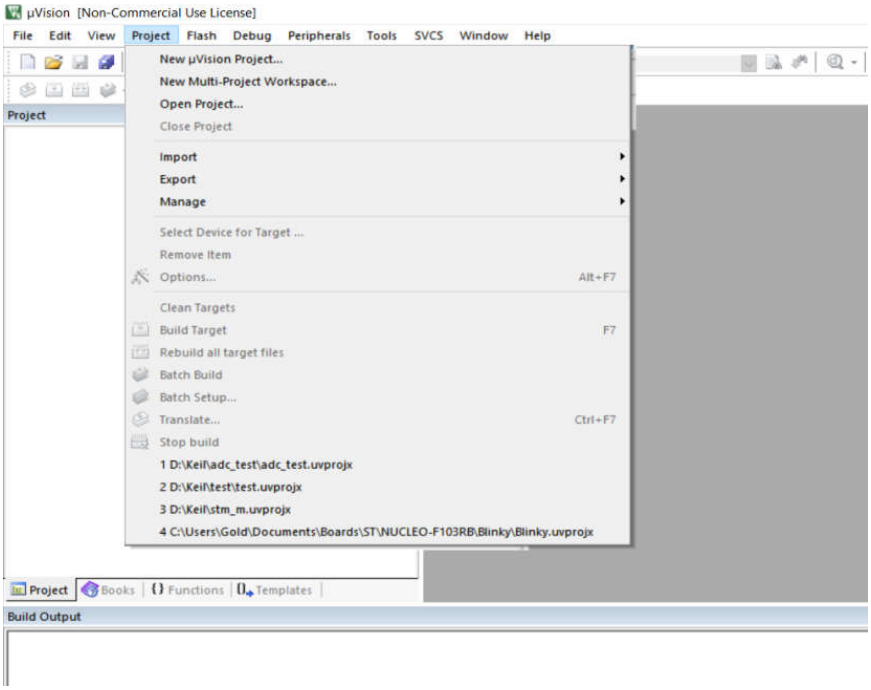

Рис.7

После необходимо создать папку проекта, задать имя проекта и сохранить его в папку. Далее, выбрать модель микроконтроллера для использования в проекте: *STM32F103RB* (рис.8).

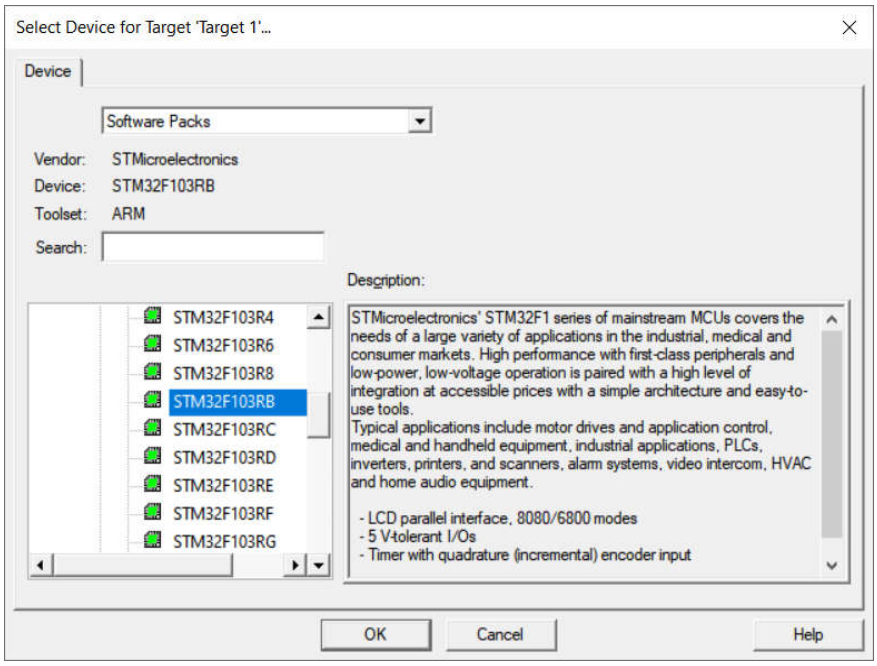

Рис.8

Далее, необходимо выбрать программные компоненты: *CMSIS/Core, Device/Startup, Device/StdPeriph Drivers/Framework, Device/StdPeriph Drivers/GPIO, Device/StdPeriph Drivers/RCC* (рис.9). После чего завершить создание проекта.

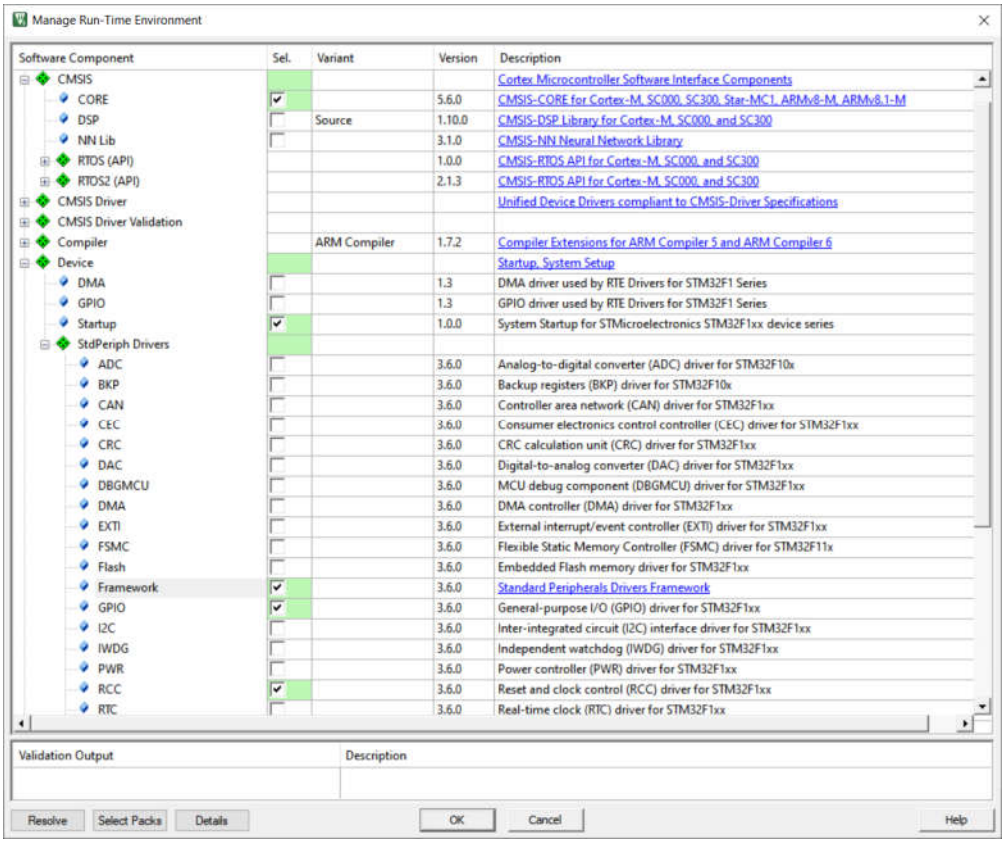

Рис.9

4.3 Выполнить настройку режима отладки для проекта. Необходимо сформировать в папке проекта файл *MAP.ini* (рис.10) со следующим содержанием:

MAP 0x40000000, 0x47FFFFFF READ WRITE;

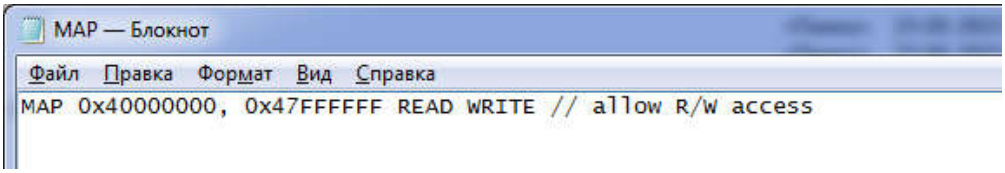

Рис.10

После чего открыть вкладку *«Option for Target…»* (рис.11), переключиться на вкладку *«Debug»*, установить переключатель *«UseSimulator»* для включения симуляции в режиме отладки. В поле *«DialogDLL»* записать *DARMSTM.DLL*, в поле *«Parameter»* *-pSTM32F103RB*. Установить путь к файлу *MAP.ini* в поле *«InitializationFile»*. После чего завершить настройку вкладки *«Debug»* (рис.12).

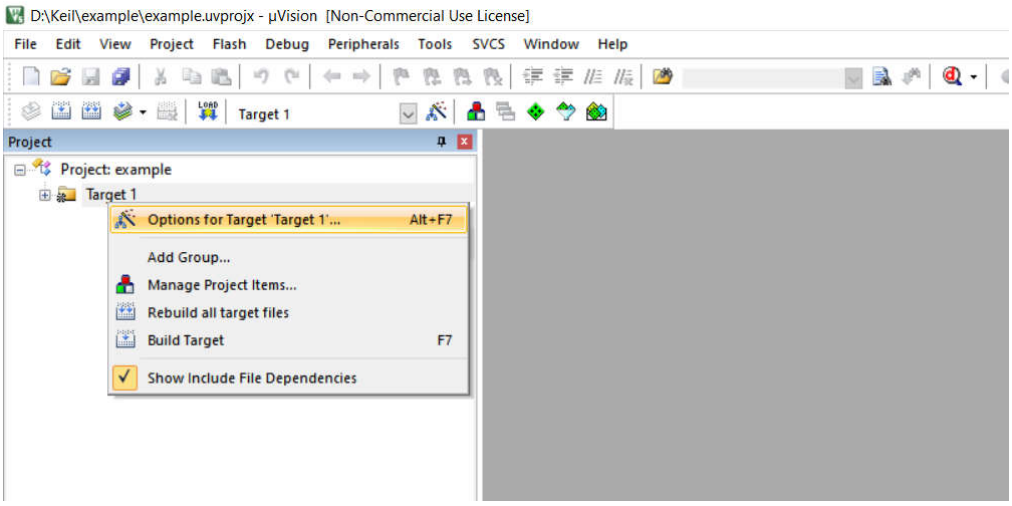

Рис.11

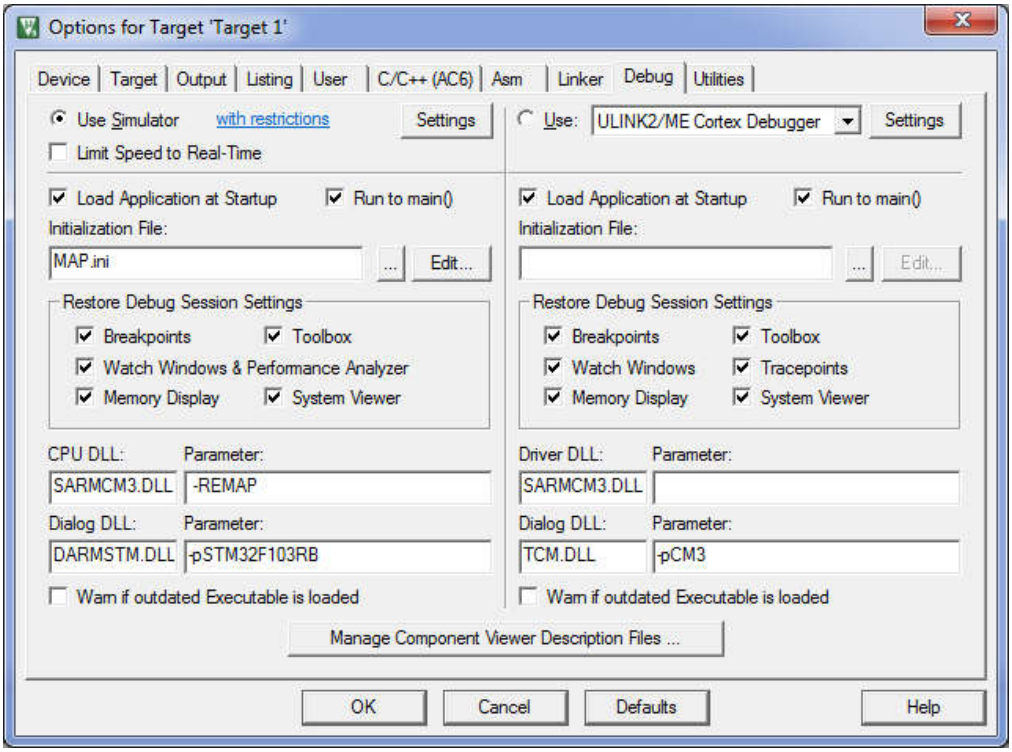

Рис.12

4.4 Создать файлы программы. Для этого необходимо создать группу пользовательских файлов *«Add Group…»* (рис.13), добавить в нее файл *«main.c»* - *«Add New Item to Group…»* (рис.14). После чего добавить в созданный файл директивы включения библиотечных файлов и функцию *int main()*, содержащую бесконечный цикл (рис.15).

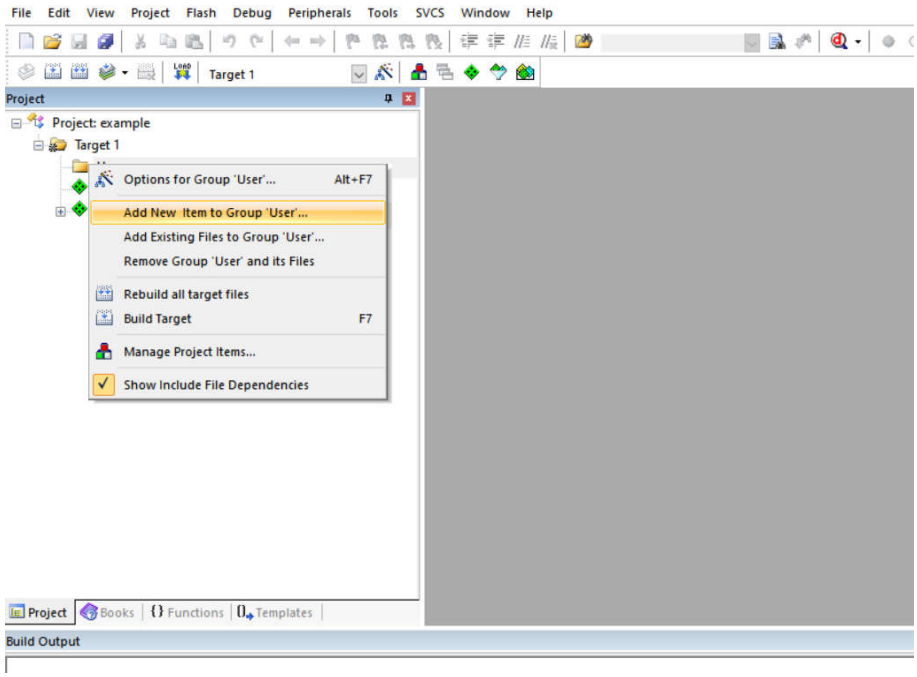

Рис.13

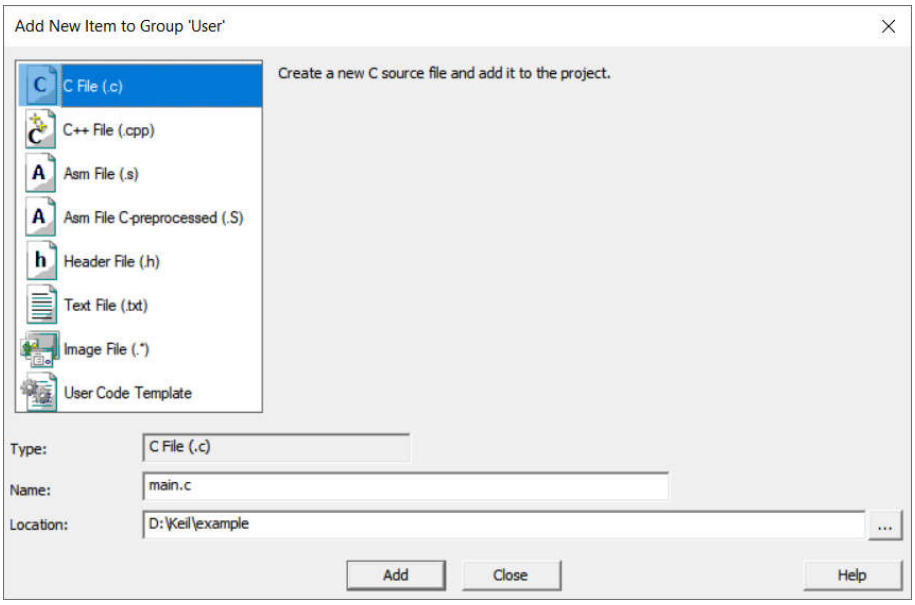

Рис.14

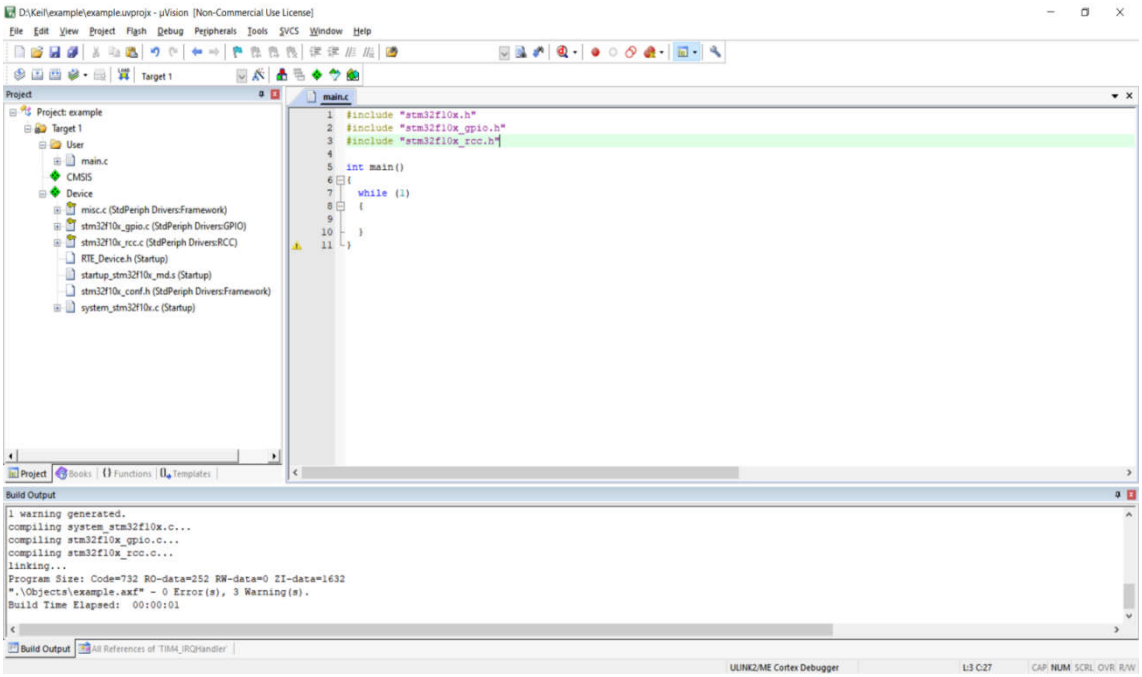

Рис.15

4.5 Разработать файл сценария. Сигнальная функция должна обеспечивать изменение состояния входного порта через заданные интервалы времени. Длительность интервалов выбрать самостоятельно.

4.6 Разработать программу согласно варианту задания. Программа должна быть организована в виде бесконечного цикла. Должны быть реализованы функции инициализации портов ввода-вывода, формирования очередного бита псевдослучайной последовательности, опроса состояния входного порта, установки состояния выходного порта. Опрос состояния входного порта должен выполняться с задержкой. Для реализации задержки использовать задержку, реализуемую через цикл с большим числом итераций. Длительность задержки (количество итераций в цикле) выбрать самостоятельно.

4.7 Выполнить построение проекта - кнопка *«Build»*. При каждом изменении программных функций необходимо выполнять перестроение проекта - кнопка *«Rebuild»*.

4.8 Выполнить симуляцию разработанной. Необходимо запустить отладочный режим - кнопка *«Start/Stop Debug Session»* или сочетание клавиш

«Ctrl + F5». После чего добавить отображение логического анализатора *«View-Analysis Windows-Logic Analyzator»* и окна просмотра данных *«View-Watch Windows-Watch1»*. Необходимо добавить в окно анализатора используемые порты. Для того, чтобы добавить вывод в окно анализатора необходимо ввести в командную строку команду *la* с указанием имени порта (*porta*, *portb*, *portc*...) и маски, соответствующей выводу, например для первого вывода порта B:

la portb&0x01

Пример симуляции порта ввода-вывода в анализаторе показан на рис.16.

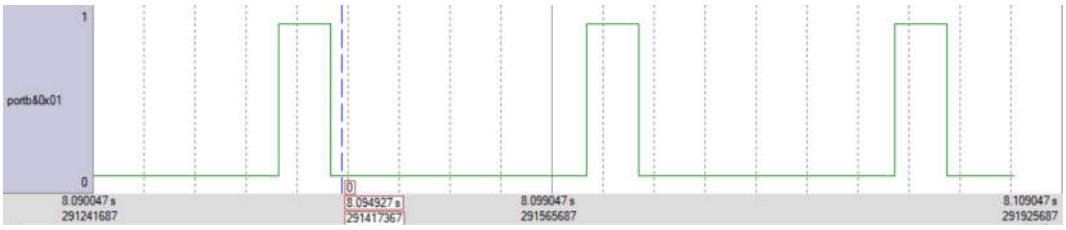

Рис.16

В окно просмотра данных добавить основные переменные программы. Вывести состояние регистров задействованных периферийных устройств в окно просмотра.

Для подключения файла сценария необходимо ввести в командную строку команду *include имя_файла.* Для запуска формирования входного сигнала необходимо ввести в командную строку имя сигнальной функции. Пример:

include gpio.ini

input()

4.9 Сделать выводы по проделанной работе и оформить отчет.

5 Содержание отчета

В отчете по результатам лабораторной работы должны быть приведены:

– используемые периферийные модули микроконтроллера и их регистры управления;

– описание алгоритма работы разработанной программы;

– текст разработанной программы, реализующей управление портами ввода-вывода;

– текст файла сценария имитации входного сигнала;

– результаты симуляции работы разработанной программы.

6 Контрольные вопросы

1. Какой периферийный модуль микроконтроллера STM32F103RB обеспечивает управление тактированием?

2. Какое количество портов ввода-вывода доступно в микроконтроллере STM32F103RB?

3. Через какой регистр осуществляется доступ к входному состоянию порта ввода-вывода?

4. Через какой регистр осуществляется установка выходного состояния порта ввода-вывода?

5. Какие режимы работы могут быть заданы для порта ввода-вывода?