В. И. Бугаев, М. П. Мусиенко, Я. М. Крайнык

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по изучению микроконтроллеров STM8
на базе отладочного модуля STM8S Value Line Discovery

Москва-Николаев – 2013

Лабораторный практикум по изучению микроконтроллеров STM8 на базе отладочного модуля STM8S Value Line Discovery / Бугаев В.И., Мусиенко М.П., Крайнык Я.М. – Москва-Николаев: МФТИ-ЧГУ, 2013. – 24 с.

Бугаев Виктор Иванович, Московский физико-технический институт, г. Москва, Россия

Мусиенко Максим Павлович, д.т.н., профессор, Черноморский государственный университет им. П. Могилы, г. Николаев, Украина

Крайнык Ярослав Михайлович, аспирант, Черноморский государственный университет им. П. Могилы, г. Николаев, Украина

Лабораторный практикум содержит материалы для начала работы с 8-битными микроконтроллерами фирмы ST Microelectronics. Приведено семь лабораторных работ для работы с основной периферией.

Лабораторный практикум будет полезен студентам компьютерных специальностей, изучающим микроконтроллеры, а также аспирантам, преподавателям, которым интересно данное направление.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Традиционно лидерами на рынке 8-битных микроконтроллеров являются компании Atmel и Microchip, производители популярных микроконтроллеров AVR и PIC. Их популярность можно объяснить долгим присутствием на данном сегменте рынка, а также ориентацией на развитие именно этого сектора. Тем не менее, в последнее время в данном сегменте появляются новые производители, одним из которых является ST Microrelectronics. 8-битные устройства данной компании имеют обозначение STM8. По оснащению они не уступают, а чем-то даже опережают аналогичные устройства от лидеров отрасли. При этом цена устройств обычно меньше, чем у конкурентов.

Данный лабораторный практикум предназначен для обучения студентов работе с микроконтроллерами серии STM8 на базе отладочного модуля STM8S Value Line Discovery. Представлены примеры программ на языке С для основных модулей микроконтроллера. Работу с практикумом следует совмещать с изучением документации для данного микроконтроллера от производителя.

Ознакомление с отладочным модулем

STM8S Value Line Discovery – наиболее простой по оснащению среди 8-битных модулей фирмы STM. На модуле размещены:

– микроконтроллер STM8S003K3, который имеет 8 кб Flash-памяти, 1 кб памяти RAM, 128 байт EEPROM-памяти;

– разъем USB для подключения питания и обмена данными (программирование и отладка при помощи ST-Link);

– одна кнопка для использования в программах;

– один светодиод;

– контакты для всех портов ввода/вывода;

– площадка для размещения другой микросхемы или подключения других устройств;

– интерфейс SWD для проведения отладки.

Внешний вид модуля показан на рисунке 1.

Рис. 1. Внешний вид модуля

Особенностью модуля является то, что он может быть физически разделен и в результате получаем два модуля: первый – модуль с отладочным интерфейсом ST-Link, второй – модуль с микроконтроллером.

Установка и настройка средств разработки. Создание базового проекта

Для работы с 8-битными микроконтроллерами STM есть несколько средств разработки: EWSTM8 от IAR, IDEA от Cosmic и др. STM предлагает разработчикам также собственную среду – ST Visual Develop (STVD).

Данная среда разработки поддерживает несколько инструментов от различных фирм. Компилятор среды устанавливается отдельно. Основными

производителями компиляторов выступают Cosmic и Raisonance. Следует отметить, что данные компиляторы могут использоваться бесплатно, но с ограничениями на размер выходного файла и термин использования с повторной регистрацией. Среду разработки можно загрузить с сайта производителя, а средства построения загружаются отдельно с выполнением настроек в STVD.

Рассмотрим последовательность действий, которые необходимо выполнить, чтобы создать проект на языке С в STVD.

Работа с проектами ведется на основе рабочего пространства (workspace), которое может содержать несколько проектов. В рабочем пространстве можно создавать новые проекты или добавлять уже существующие. Для начала работы выполним команду меню *File\New Workspace*. Откроется диалоговое окно (рис. 2) с вариантами создания рабочего пространства.

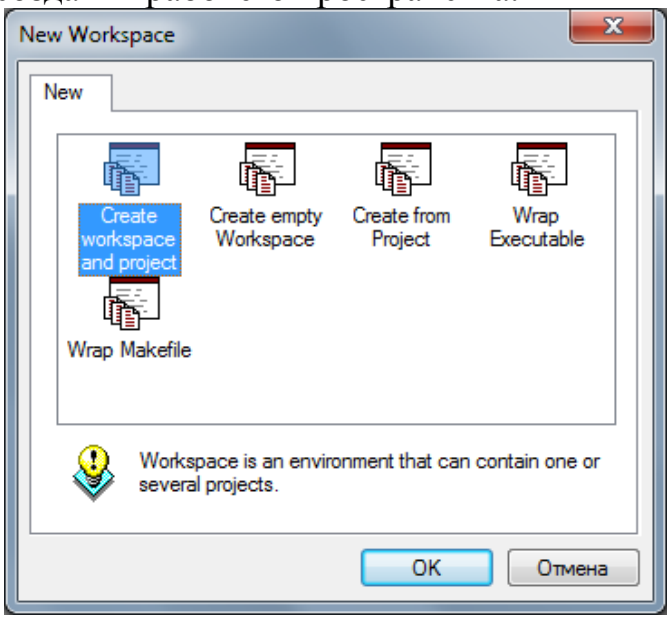

Рис. 2. Диалоговое окно создания нового рабочего пространства

После этого следует указать размещение рабочего пространства (рис. 3).

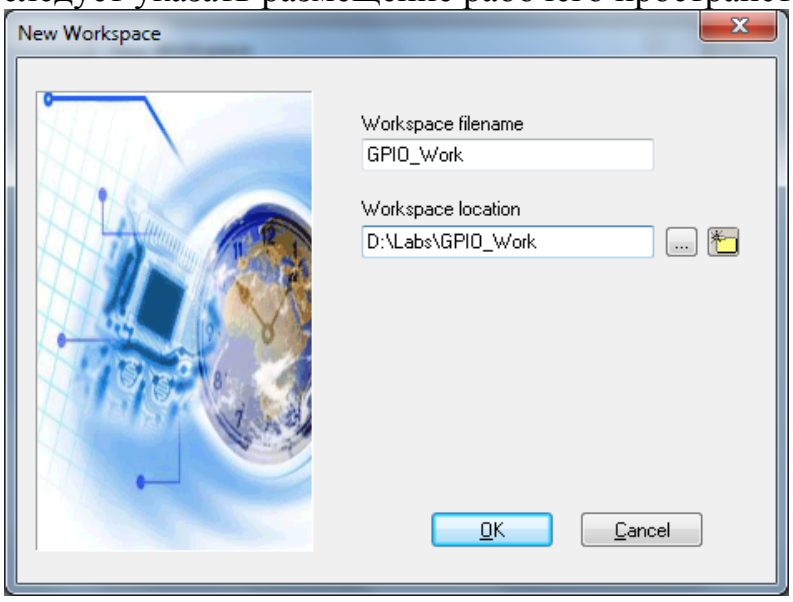

Рис. 3. Окно выбора размещения рабочего пространства

Далее указываем название и размещение проекта, а также инструменты для построения проекта (рис. 4).

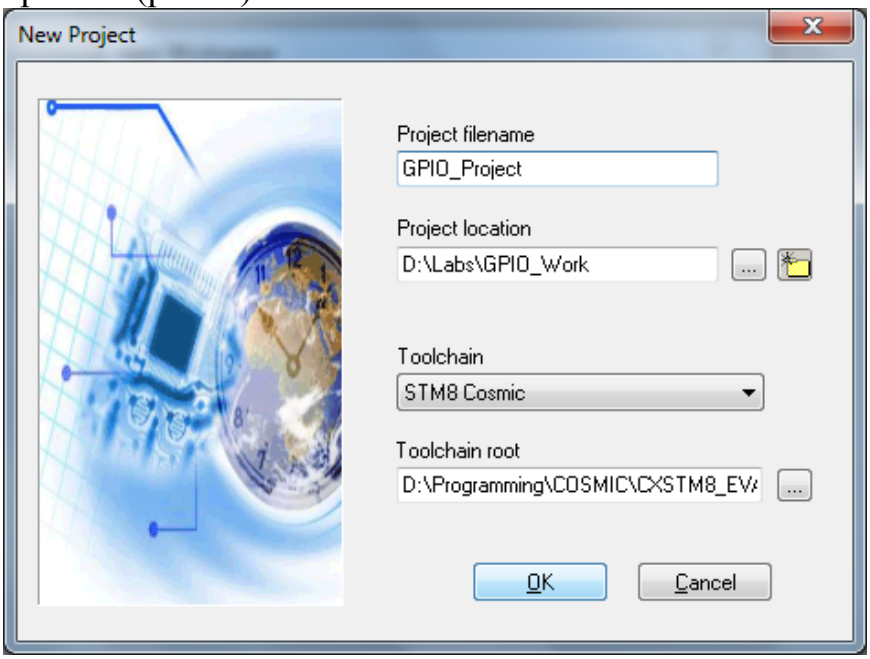

Рис. 4. Окно настройки проекта

В последнем диалоговом окне выбираем микроконтроллер (рис. 5), для которого предназначена программа.

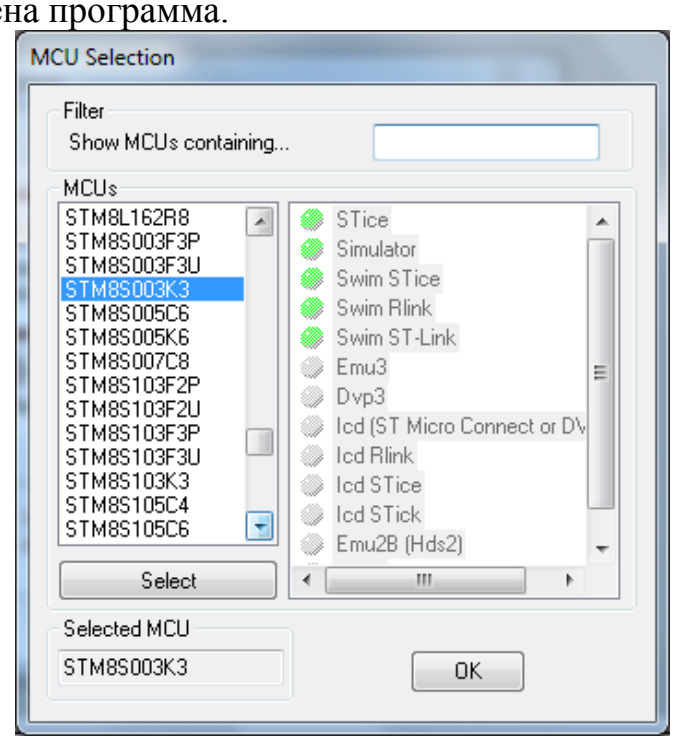

Рис. 5. Окно выбора микроконтроллера

В результате создается каркас проекта с разделением на заголовочные файлы (Include Files) и файлы реализации (Source Files). Для дальнейшей работы нам понадобится стандартная библиотека для работы с периферией, которую также следует загрузить с сайта производителя. Библиотека представляет собой архив с самой библиотекой, примерами работы с

периферией и шаблонами проектов для разных сред разработки. Именно последние удобно использовать для создания проектов и позволяют сэкономить время на первоначальную настройку. Они находятся в папке *STM8S_StdPeriph_Lib_V2.1.0\Project\STM8S_StdPeriph_Template*. Нам понадобятся файлы, которые мы скопируем в папку с проектом:

– stm8s_conf.h;
– stm8s_it.c;
– stm8s_it.h;
– stm8_interrupt_vector.c (находится в папке *STVD\Cosmic*) – данный файл копируется с заменой уже существующего файла.

Скопировав все файлы, добавляем их к проекту (команда контекстного меню *Add Files To Folder…*) в соответствии с расширением.

Также проверьте, чтобы в файле *stm8s_conf.h* была закомментирована директива, определяющая символ USE_FULL_ASSERT. Для начального ознакомления она не нужна.

Если все сделано правильно, то построение проекта (команда меню *Build\Build* или горячая клавиша F7) выполняется без ошибок. В случае присутствия ошибок следует предварительно выполнить команду *Build\Clean*.

Файлы, которые отвечают за работу с периферией, находятся в директории *STM8S_StdPeriph_Lib_V2.1.0\Libraries\STM8S_StdPeriph_Driver* и разделены на группы заголовочных файлов (директория inc) и исходных кодов (директория src). Среди них обязательно понадобится файл stm8s.h, а другие файлы – в зависимости от модулей (устройств), которые планируется использовать в программе. Скорее всего, в большинстве проектов будут использоваться функции настройки тактирования устройств (в файлах stm8s_clk.c и stm8s_clk.h) и портов ввода/вывода (в файлах stm8s_gpio.c и stm8s_gpio.h), поэтому копируем их и добавляем к проекту файлы. В результате структура проекта будет выглядеть так, как показано на рисунке 6.

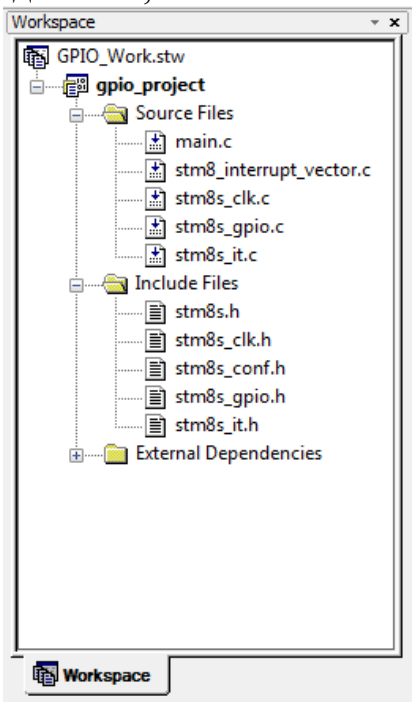

Рис. 6. Базовая структура проекта

Проверьте также, чтобы в настройках компилятора для поиска заголовочных файлов (команда *Tools*/*Options*, вкладка *Directories*), наличие пути к файлам библиотеки или добавьте его (рис. 7).

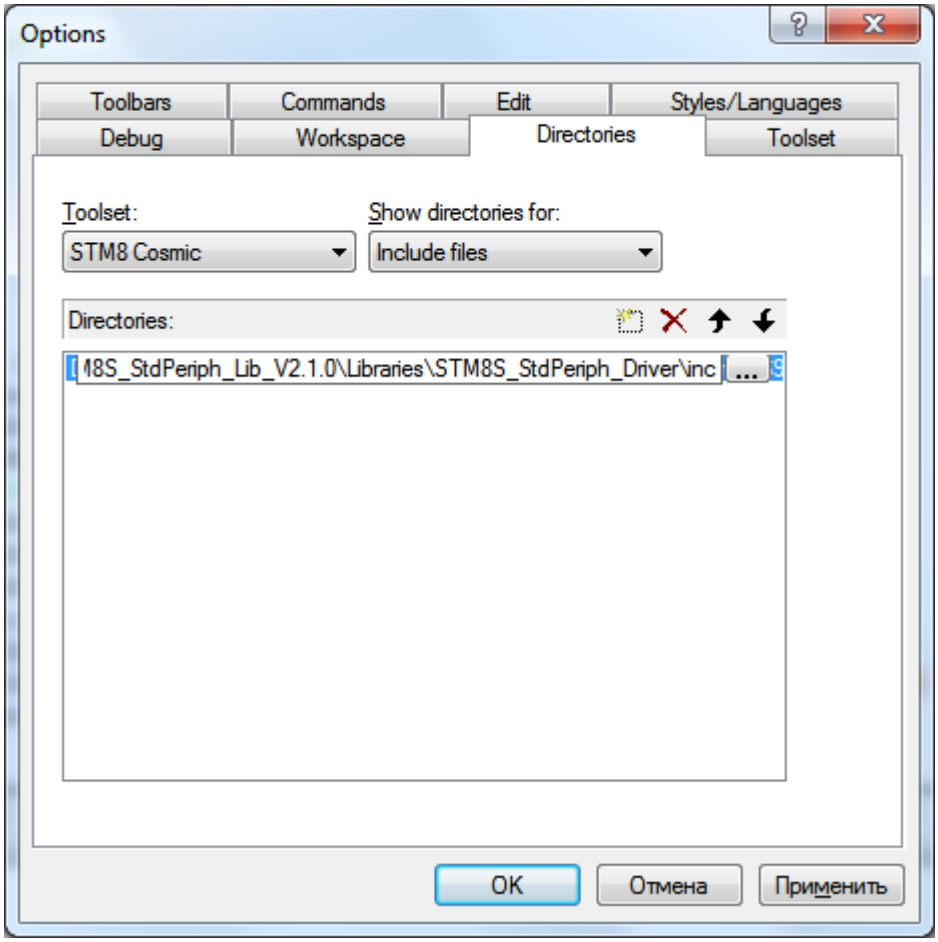

Рис. 7. Настройка директории поиска заголовочных файлов для компилятора

Именно такая последовательность действий с дальнейшим добавлением необходимых файлов выполняется в ходе выполнения заданий.

Среди средств, которые присутствуют в STVD, присутствует программа, с помощью которой выполняется программирование устройства. Открыть ее окно (рис. 8) можно командой меню *Tools\Programmer*. окно содержит несколько вкладок, которые отвечают за настройки интерфейса подключения, области памяти, дополнительные возможности. Последняя вкладка предназначена для выполнения программирования.

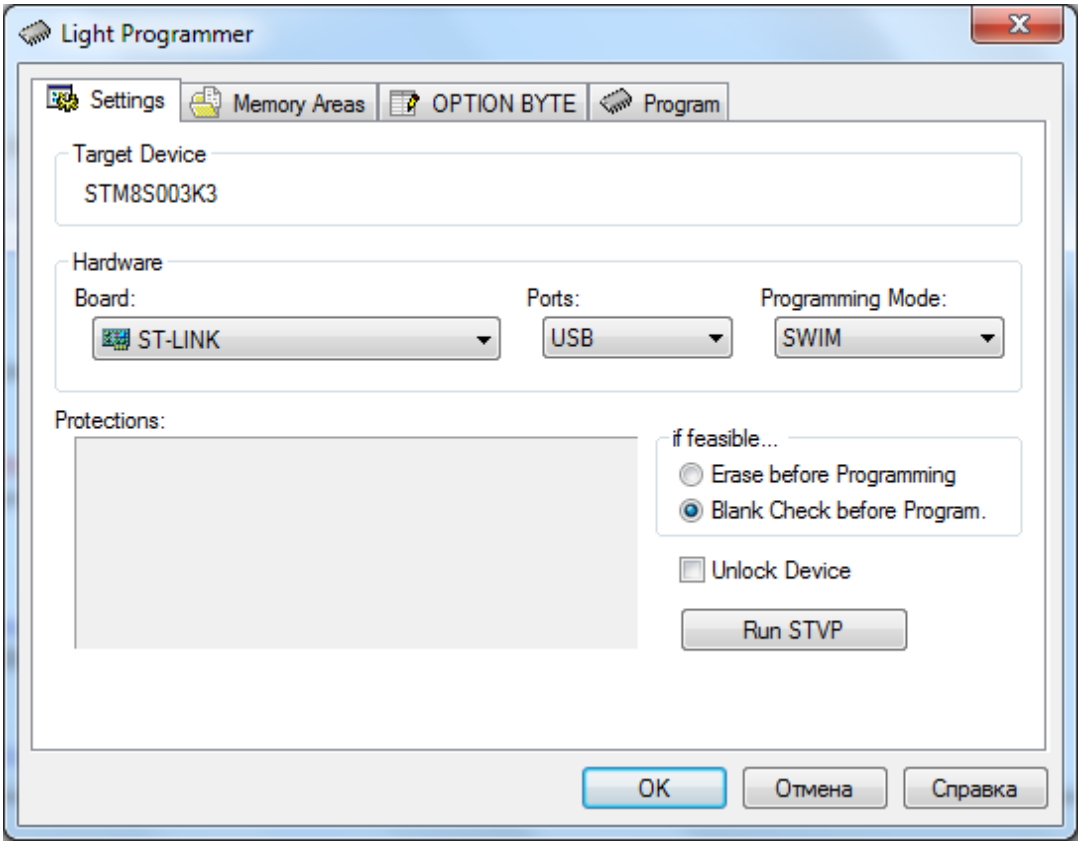

Рис. 8. Вкладка основных настроек

На второй вкладке (рис. 9) можно указать, какие данные записываются в определенную область памяти.

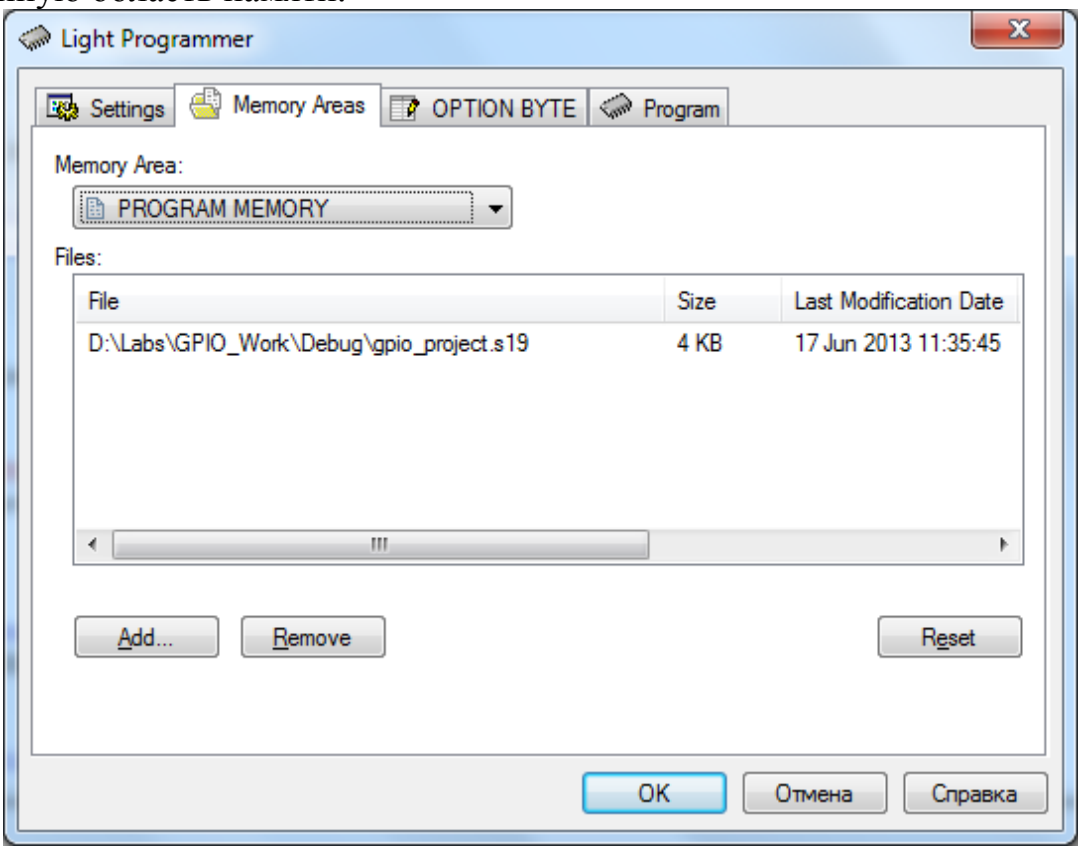

Рис. 9. Выбор файла для программирования памяти

На рис. 10 показана вкладка после проведения процесса программирования.

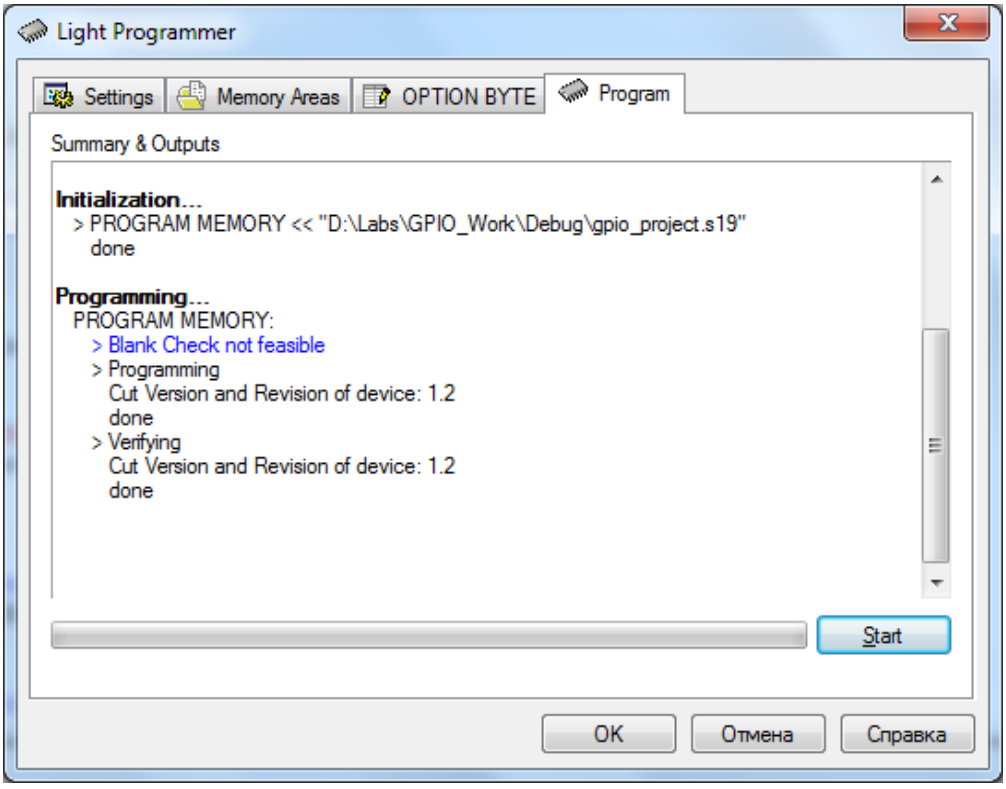

Рис. 10. Вкладка запуска программирования

Таким образом, средства программирования уже встроены в среду разработки. Однако, как более быстрый вариант проведения программирования, предлагается после построения проекта запускать отладку при подключенной к компьютеру плате – в таком случае не расходуется время на открытие другой программы.

Лабораторная работа №1. Работа с портами ввода/вывода

Цель работы: изучить особенности организации проектов в среде STVD, изучить возможности настройки и использования портов ввода/вывода, ознакомится с библиотекой управления периферией микроконтроллера, организацией программ для внешнего устройства.

Оборудование и программное обеспечение: отладочная плата STM8S Value line discovery, среда разработки STVD.

Вместе со средствами программирования доступны также средства отладки. Запуск отладки выполняется командой *Debug/Start Debugging*. Для корректной отладки необходимо проверить настройки командой меню *Debug instrument/Target Settings*. В окне, которое появилось следует указать параметры, которые показаны на рисунке 11.

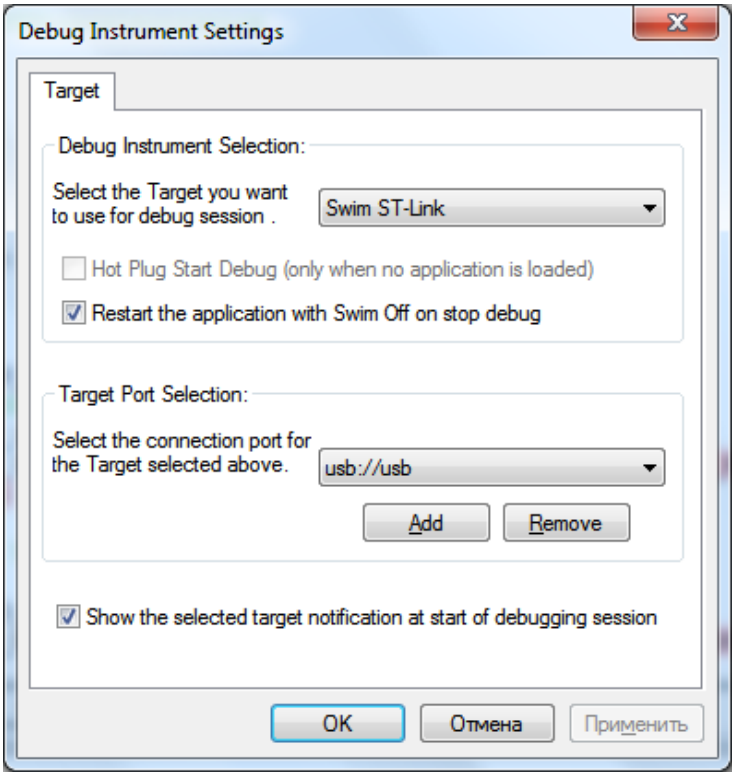

Рис. 11. Настройка устройства отладки

В ходе выполнения данной работы вместе с написанием программы и программированием платы желательно попробовать просмотреть выполнение операций с помощью режима отладки, установив точки прерываний (рис. 12).

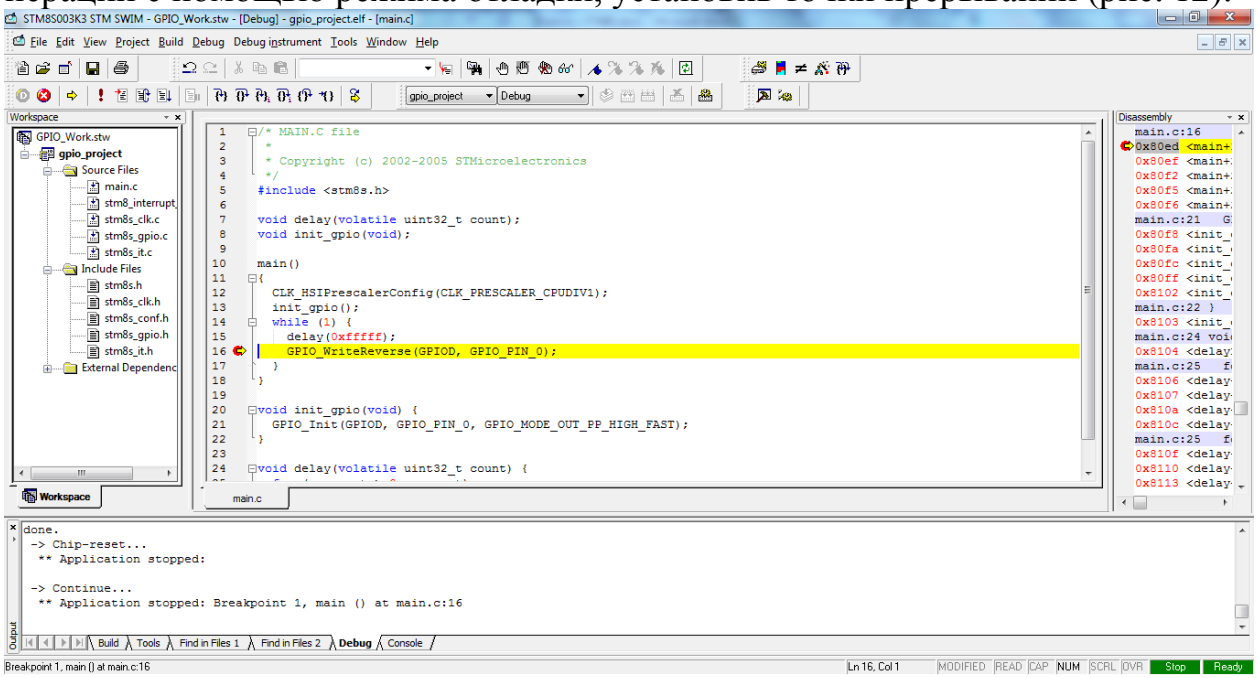

Рис. 12. Среда разработки в режиме отладки

Пример программы

Далее рассмотрим пример самой простой программы для начала работы с отладочной платой – управление состоянием светодиода на плате. Программа, представленная в качестве примера, является очень простой, но в ней можно

увидеть основные моменты, которые необходимы для дальнейшей работы (принцип организации программы, именование функций в библиотеке и др.). Рассмотрим детальнее код программы.

```
#include <stm8s.h>

void delay(volatile uint32_t count);
void init_gpio(void);

main()
{
	CLK_HSIPrescalerConfig(CLK_PRESCALER_CPUDIV1);
	init_gpio();
	while (1) {
		delay(0xfffff);
		GPIO_WriteReverse(GPIOD, GPIO_PIN_0);
	}
}

void init_gpio(void) {
	GPIO_Init(GPIOD, GPIO_PIN_0, GPIO_MODE_OUT_PP_HIGH_FAST);
}

void delay(volatile uint32_t count) {
	for ( ; count > 0; --count);
}
```

В заголовочном файле подключен лишь один заголовочный файл: подключение остальных файлов происходит через него. Далее выполняем описание функций инициализации и задержки. Их реализация выполнена после функции main. В главной функции вначале выполняется настройка делителя внутреннего источника тактовой частоты (High Speed Internal – HSI).

Тем, кто программирует на языке C++, следует обратить внимание на то, что если функция не принимает ни одного параметра, то для нее обязательно необходимо указывать на месте void входных параметров. Еще одной особенностью является необходимость объявить переменные в функции до появлений любой управляющей инструкции внутри нее. Это связано с тем, что используется один из предыдущих стандартов языка C – C89. В случае наличия таких ошибок компилятор укажет на необходимость их исправить. Еще одной особенностью, которую можно увидеть является то, что для функции можно не указывать тип возвращаемого значения. По вышеупомянутому стандарту, компилятор считает, что такая функция возвращает значение типа `int`.

Если говорить про часть программы, которая относится к портам ввода/вывода, то с помощью функции `GPIO_Init` указывается порт, его выход и режим, в котором он работает. Установка режима работы вывода задается с помощью константы, которая имеет очень долгое имя, тем не менее, она позволяет задать сразу 4 параметра работы вывода. Другая использованная функция – `GPIO_WriteReverse` – позволяет изменить логический уровень сигнала на противоположный.

Относительно стандартной библиотеки, которая используется, то следует сказать, что она содержит большое количество директив, которые отвечают за инициализацию и изменение параметров, и имеют очень длинные имена, а функции – большое количество параметров. Часто эти директивы и функции

находятся рядом, поэтому можно на основе примера изменить настройки в соответствии с требованиями задания и получить необходимый результат, поэтому это следует активно использовать, поскольку, во-первых, это повышает читабельность написанного, во-вторых, позволяет быстро ознакомиться с возможностями для написания программ.

Задание

1. На основе примера и предыдущих указаний создать и настроить проект в предложенной среде разработке.
2. Продемонстрировать работу примера.
3. Продемонстрировать работу в режиме отладки.

Лабораторная работа №2. Использование внешних прерываний

Цель работы: ознакомится с возможностью использования внешних прерываний.

Оборудование и программное обеспечение: отладочная плата STM8S Value line discovery, набор перемычек, плата для подключения семисегментного индикатора, среда разработки STVD.

Пример программы

Работа с прерываниями – это то, с чем приходится иметь дело каждому разработчику программного обеспечения для микроконтроллеров, именно в работе с ними заключается основная часть разработки. Прерывания – сигнал, который свидетельствует о появлении некоторого важного события в работе контроллера. Примером прерывания можно назвать переполнение счетчика таймера или завершение передачи данных с помощью UART.

Особенностью работы с прерываниями при условии использования функций стандартной библиотеки является то, что обработчики прерываний уже присутствуют в файле *stm8_it.c*. Разработчику следует лишь разрешить прерывания для всего контроллера и настроить конкретный вид прерываний.

Внешние прерывания являются самым простым видом прерываний, поскольку используют лишь порты ввода/вывода и возникают вследствие логического изменения уровня сигнала (какое именно изменение приведет к прерыванию, зависит от конфигурации).

Следующая программа демонстрирует использование внешнего прерывания по нажатию кнопки. При наступлении события прерывания выполняется изменение состояния светодиода на плате. Далее приведено содержание файла *main.c*.

```
#include «stm8s.h»

void init(void);

main()
{
	enableInterrupts();
	init();
	while (1);
}

void init(void) {
	CLK_HSIPrescalerConfig(CLK_PRESCALER_CPUDIV1);
	GPIO_Init(GPIOD, GPIO_PIN_0, GPIO_MODE_OUT_PP_HIGH_FAST);
	GPIO_Init(GPIOB, GPIO_PIN_7, GPIO_MODE_IN_FL_IT);
	EXTI_SetExtIntSensitivity(EXTI_PORT_GPIOB, EXTI_SENSITIVITY_FALL_LOW);
}
```

В файле *stm8s_it.h* найдем готовый обработчик и добавим в него следующий код.

```
INTERRUPT_HANDLER(EXTI_PORTB_IRQHandler, 4)
{
	GPIO_WriteReverse(GPIOD, GPIO_PIN_0);
}
```

Задание

1. На основе примера и предыдущих указаний создать и настроить проект в предложенной среде разработке.
2. Продемонстрировать работу примера.
3. Продемонстрировать работу в режиме отладки.

Лабораторная работа №3. Использование таймеров. Организация задержек

Цель работы: овладеть навыками организации программных задержек с использованием таймеров.

Типичной задачей при написании программ для микроконтроллера является организация задержек, проверка определенных параметров, состояний, ожидание определенного события, например, завершения обработки данных. Выполнение этих задач в главном блоке программы с использованием циклов приведет к тому, что программа станет сложной для понимания и внесения дальнейших изменений, а также к тому, что процессорное время используется полностью, поэтому возможны сбои, получение неожиданных результатов.

Одним из основных вариантов решения данной проблемы является использование таймеров. Задачи проверки состояний, ожидания можно вынести в блок обработки прерываний по завершению счета таймера, предварительно настроив таймер и разрешив это прерывание.

В состав микроконтроллера на отладочном модуле входит 6 таймеров, 4 из которых являются 16-разрядными. Каждый таймер содержит регистр счетчика и регистр предделителя, регистр значения для загрузки. С их помощью реализуется логика работы таймера.

Пример программы

Для примера рассмотрим программу, в которой выполняется реализация включения/выключения светодиода через интервал времени, который равен 1 с. Частота тактирования устройства при использовании внутреннего источника тактирования составляет 16 МГц. Соответственно, необходимо организовать управление таймером так, чтобы событие обновления возникало 1 раз на 16 млн. тактов. Для этого установим конечное значение счетчика таймера в 999, а значение предделителя – 15999.

```
#include «stm8s.h»

void init(void);

main()
{
	init();
	while (1);
}

void init(void) {
	enableInterrupts();
	CLK_HSIPrescalerConfig(CLK_PRESCALER_CPUDIV1);
	GPIO_Init(GPIOD, GPIO_PIN_0, GPIO_MODE_OUT_PP_HIGH_FAST);
	TIM1_TimeBaseInit(999, TIM1_COUNTERMODE_UP, 15999, 0);
	TIM1_ITConfig(TIM1_IT_UPDATE, ENABLE);
	TIM1_Cmd(ENABLE);
}
```

В блоке обработки прерываний в файле *stm8s_it.c* таймера TIM1 выполняем следующий код.

```
INTERRUPT_HANDLER(TIM1_UPD_OVF_TRG_BRK_IRQHandler, 11)
{
	if (TIM1_GetITStatus(TIM1_IT_UPDATE)) {
		TIM1_ClearITPendingBit(TIM1_IT_UPDATE);
		GPIO_WriteReverse(GPIOD, GPIO_PIN_0);
	}
}
```

Задание

1. Настроить проект и проверить работу программного кода из примера.
2. Реализовать временную задержку в соответствии с вариантом:

№ варианта	Длина задержки, с
1	1
2	0.5
3	5
4	2
5	3

3. Продемонстрировать работу задержки и проведенные расчеты.

Лабораторная работа №4. Использование таймеров Режим широтно-импульсной модуляции (ШИМ)

Цель работы: исследовать возможности использования режима ШИМ таймера.

Другой возможностью, которую предоставляют таймеры, является возможность генерации сигнала ШИМ. В составе микроконтроллера отладочного модуля такую возможность предоставляет лишь таймер TIM1. Режим ШИМ таймера относится к группе режимов работы захвата/сравнения.

ШИМ часто используется для управления уровнем напряжения на низкочастотной нагрузке, например, для регулировки скорости вращения двигателя или для настройки яркости свечения светодиода.

Пример программы

В качестве примера рассмотрим программу, которая позволяет генерировать сигнал ШИМ на одном из выходов с коэффициентом заполнения в 50%. Код программы представлен ниже.

```
#include "stm8s.h"

void init(void);

main()
{
	init();
	while (1);
}

void init(void) {
	CLK_HSIPrescalerConfig(CLK_PRESCALER_CPUDIV1);
	TIM1_TimeBaseInit(99, TIM1_COUNTERMODE_UP, 99, 0);
	TIM1_OC1Init(TIM1_OCMODE_PWM1,
			TIM1_OUTPUTSTATE_ENABLE,
			TIM1_OUTPUTNSTATE_ENABLE,
			49,
			TIM1_OCPOLARITY_LOW,
			TIM1_OCNPOLARITY_HIGH,
			TIM1_OCIDLESTATE_SET,
			TIM1_OCIDLESTATE_RESET);
	TIM1_Cmd(ENABLE);
	TIM1_CtrlPWMOutputs(ENABLE);
}
```

Во время инициализации указывается максимальное значение, до которого происходит отсчет (99) и значение предделителя (99). При конфигурации линии сравнения таймера указываются уровни, которые отвечают состояниям сигнала, а также значение, после которого происходит переключение из нулевого уровня на уровень логической единицы – 49.

Результирующий график сигнала, полученный на экране осциллографа, показан на рисунке 13. Из рисунка хорошо видно, как именно выглядит выходной сигнал ШИМ при значении коэффициента заполнения 50%.

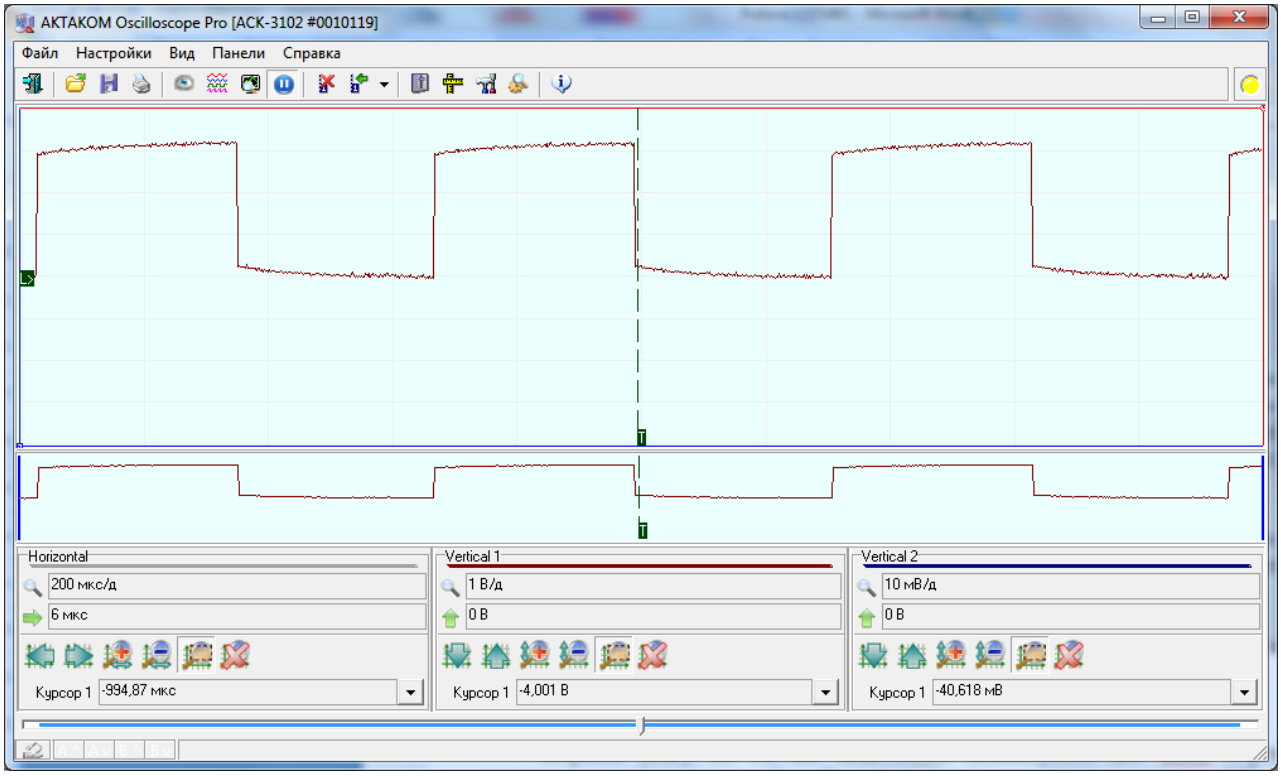

Рис. 13. График сигнала

Задание

1. За основу выполнения задания взять проект с приведенным выше кодом.
2. Настроить 2 других канала на генерацию сигнала ШИМ (могут выступать выходы PC2-PC4)
3. Получить макетную плату, собранную по следующей схеме:

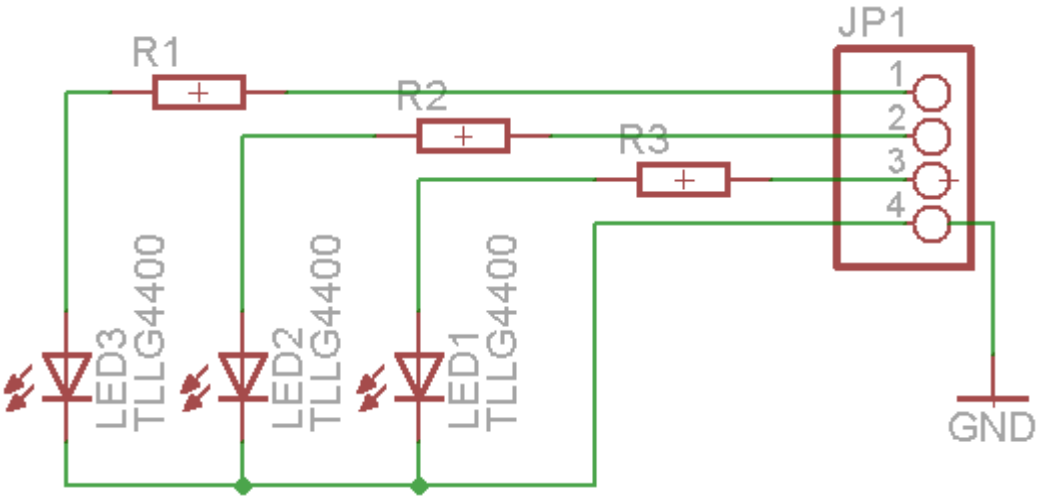

Рис. 13. Схема макетной платы для выполнения задания

4. Реализовать световой эффект на основе изменения яркости светодиодов, подключенных к выходам ШИМ (поочередное угасание, включение/выключение, смена эффектов и др.).

Лабораторная работа №5. Работа с UART

Цель работы: освоить принципы работы с универсальным приемо-передатчиком (UART).

Оборудование: отладочная плата STM8S Value Line Discovery, любая другая плата с возможностью приема/передачи данных через UART, набор перемычек.

Универсальный асинхронный приемо-передатчик (англ. Universal asynchronous receiver-transmitter, сокр. UART) является наиболее распространенным интерфейсом передачи данных в микроконтроллерах. Данный интерфейс является последовательным, а для его работы необходимо всего две линии – одна для передачи, а другая для приема. При подключении двух устройств с помощью UART две линии разных устройств подключаются к разноименным линиям. Таким образом, можно, в том числе, передавать и принимать данные с помощью одного устройства с UART, подключив линию приема к линии передачи, что полезно для отладки работы программы.

Пример программы

В качестве примера рассмотрим программу, которая использует UART для передачи данных. В главном цикле программы организована временная задержка, за которой следует передача данных (символ “a”). В параметрах настройки UART указывается скорость передачи и формат посылки (длина данных, количество стоп-бит, наличие бита парности), режим работы в качестве приемника и передатчика одновременно.

```
#include «stm8s.h»
#include "stm8s_uart1.h"
#include "stm8s_it.h"

void init_UART(void);
void delay(uint16_t count);

main()
{
	int count;
	CLK_HSIPrescalerConfig(CLK_PRESCALER_CPUDIV1);
	init_UART();
	count = 0;
	while (1) {
		if (count == 10) {
			UART1_SendData8('a');
			count = 0;
		}
		delay(0xffff);
		++count;
	}
}

void delay(uint16_t count) {
	while(count != 0)
		count--;
}
```

```
void init_UART() {
	UART1_DeInit();
	UART1_Init((uint32_t)9600, UART1_WORDLENGTH_8D, UART1_STOPBITS_1,
                                                UART1_PARITY_NO,
                                                UART1_SYNCMODE_CLOCK_DISABLE,
                                                UART1_MODE_TXRX_ENABLE);
}
```

Задание

1. Проверить работу программы из примера.
2. Организовать последовательный прием и передачу данных между платами с интервалом в 1 с. Одна из плат подает сигнал начала и первой отсылает данные, другая плата принимает эти данные и через 1 с отсылает собственную посылку. Размер посылки – 1 байт. Другие характеристики формата можно выбрать самостоятельно.

Лабораторная работа №6. Работа с SPI

Цель работы: изучить возможности передачи данных с помощью SPI с помощью отладочного модуля.

Оборудование: отладочная плата STM8S Value Line Discovery, любая другая плата с возможностью приема/передачи данных через UA SPI RT, набор перемычек.

SPI (Serial Peripheral Interface) – последовательный интерфейс периферии. Один из наиболее распространенных интерфейсов передачи данных для микроконтроллеров. Фактически представляет собой два сдвиговых регистра, которые передают данные один одному.

Для работы SPI используются 4 линии:

– CLK – линия, на которой генерируется частота передачи.

– MOSI – линия передачи данных относительно устройства, которое передает данные;

– MISO – линия приема данных относительно устройства, которое передает данные;

– SS – линия выбора устройства для передачи (1 – устройство не выбрано, в таком случае приемник данных не получает, 0 – устройство выбрано; таких линий может быть несколько помимо аппаратной – для этого можно использовать выходы портов ввода/вывода).

Пример программы

В качестве примера рассмотрим программу, которая позволяет принимать данные из любого другого устройства с SPI, которое передает данные на частоте, с которой может работать модуль SPI. Модуль SPI микроконтроллера в таком случае работает в режиме slave. Прием данных ведется в обработчике прерывания SPI. Для начала приема необходимо выполнить настройки SPI. В таком случае указываем, что:

– первым передается старший бит;

– частота берется с предделителем 2 (как понятно, этот параметр не влияет на работу устройства, поскольку используется тактовый сигнал другого устройства);

– устройство работает в режиме slave;

– по полярности и фазе устройство работает в режиме 0 (по полярности и фронтом приема);

– управление сигналом выбора устройства осуществляется программно.

Обратите внимание на то, что дополнительных настроек для пинов порта ввода/вывода не требуется. Также мы разрешаем прерывание для SPI и включаем устройство.

```
#include "stm8s.h"

/**
	MISO - C7
	MOSI - C6
	CLK  - C5
	SS   - E5
*/

void init(void);

main()
{
	init();
	while (1);
}

void init(void) {
	enableInterrupts();
	CLK_HSIPrescalerConfig(CLK_PRESCALER_CPUDIV1);
	SPI_Init(SPI_FIRSTBIT_MSB, SPI_BAUDRATEPRESCALER_2,
		SPI_MODE_SLAVE, SPI_CLOCKPOLARITY_LOW, SPI_CLOCKPHASE_1EDGE,
		SPI_DATADIRECTION_2LINES_FULLDUPLEX, SPI_NSS_SOFT, 0);
	SPI_ITConfig(SPI_IT_RXNE, ENABLE);
	SPI_Cmd(ENABLE);
}
```

В обработчике прерываний мы выполняем очистку бита в регистре статуса (так следует делать для многих прерываний, поскольку если этот бит установлен, то сложно отследить дальнейший ход программы) и считываем полученные данные.

```
INTERRUPT_HANDLER(SPI_IRQHandler, 10)
{
	int res;
	SPI_ClearITPendingBit(SPI_IT_RXNE);
	res = SPI_ReceiveData();
	++res; // робота з прийнятими даними
}
```

Задание

1. Проверить работу программы из примера.
2. Организовать последовательный прием и передачу данных между платами с интервалом в 1 с. Одна из плат подает сигнал начала и первой отсылает данные, другая плата принимает эти данные и через 1 с отсылает собственную посылку. Размер посылки – 1 байт. Другие характеристики формата можно выбрать самостоятельно.

Лабораторная работа №7. Работа с аналогово–цифровым преобразователем

Цель работы: исследовать функциональность встроенного АЦП микроконтроллера.

Оборудование: отладочная плата, потенциометр, набор перемычек для подключения, мультиметр (вольтметр).

Микроконтроллер STM8S003K3 содержит один встроенный АЦП. АЦП имеет разрядность 10 бит и предоставляет возможность работать с 4 входными каналами для обработки сигналов (такое количество доступно на плате). Время, необходимое на выполнение одного преобразования составляет 14 тактов. На входы можно подавать напряжение в диапазоне от 0 до напряжения питания (может быть 3.3 В или 5 В); подача собственного опорного напряжения невозможна. АЦП может работать в двух режимах:

– режим однократного преобразования, при котором преобразование сигнала в цифровой код выполняется один раз;

– режим постоянного преобразования, при котором следующее преобразование начинается сразу после завершения предыдущего.

Настройки АЦП позволяют также указать сигналы от других устройств как начало преобразования – по сигналу от таймера и от внешнего прерывания. Модуль АЦП может генерировать прерывания по завершению преобразования.

Пример программы

Для демонстрации работы АЦП в составе микроконтроллера рассмотрим самый простой пример.

Нас интересует режим постоянного преобразования, поскольку таким образом можно организовать наблюдение за уровнем сигнала на одном из входов. Кроме того, такой режим удобен для отладки, поскольку можно динамично изменять параметры входного сигнала и просматривать результат изменений непосредственно в коде программы. В функции настройки основным параметром является константа, которая означает режим постоянного преобразования. Также указывается входной канал, с которого берется сигнал (обратите внимание на то, что он предварительно сконфигурирован в качестве входа) и размещение данных в регистре данных АЦП. Результат преобразования наблюдаем в режиме отладки в обработчике прерывания, поэтому позволяем прерывание по завершению преобразования АЦП и включаем модуль. Последним необходимым действием является запуск процесса преобразования функцией `ADC1_StartConversion()`: без этого преобразование не происходит.

```
#include "stm8s.h"

void init(void);

main()
{
	uint16_t res;

	init();
	while (1) {
	}
}

void init(void) {
	enableInterrupts();
	CLK_HSIPrescalerConfig(CLK_PRESCALER_CPUDIV1);
	GPIO_Init(GPIOB, GPIO_PIN_0, GPIO_MODE_IN_FL_NO_IT);
	ADC1_Init(ADC1_CONVERSIONMODE_CONTINUOUS,
				ADC1_CHANNEL_0, ADC1_PRESSEL_FCPU_D2,
				ADC1_EXTTRIG_GPIO, DISABLE,
				ADC1_ALIGN_RIGHT, ADC1_SCHMITTTRIG_CHANNEL0, DISABLE);
	ADC1_ITConfig(ADC1_IT_EOCIE, ENABLE);
	ADC1_Cmd(ENABLE);
	ADC1_StartConversion();
}
```

В обработчике преобразования выполняем очистку бита в регистре статуса и считываем значение кода. В дальнейшем это значение на основе величины опорного напряжения можно использовать для вывода показателей, выполнения определенных операций, передаче данных при достижении определенного уровня сигнала и т.д.

```
INTERRUPT_HANDLER(ADC1_IRQHandler, 22)
 {
	uint16_t res;
	ADC1_ClearITPendingBit(ADC1_IT_EOC);
	res = ADC1_GetConversionValue();
	res++; // робота з прийнятими даними
 }
```

Задание

1. Запустить на выполнение предложенный пример.
2. Продемонстрировать в режиме отладки полученное значение.
3. Подключить к отладочной плате модуль с 2-разрядным семисегментным индикатором, на который вывести значение напряжения с точностью до десятых.
4. Проверить точность показателей при помощи мультиметра (вольтметра).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. STM8S003K3 [Электронный ресурс]. Режим доступа: URL: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00024550.pdf. – Загол с экрана.
2. STM8SVLDISCOVERY [Электронный ресурс]. Режим доступа: URL: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00040833.pdf. – Загол с экрана.
3. STM8S value line discovery [Электронный ресурс]. Режим доступа: URL: http://www.st.com/web/en/resource/technical/document/user_manual/DM00040810.pdf. – Подзагол. с экрана.