1. Указание мер безопасности.

К работе допускаются лица, изучившие настоящее описание, инструкцию по технике безопасности при работе с измерительными приборами, а также прошедшие инструктаж по безопасности труда на рабочем месте.

Прежде чем приступить к работе, внешним осмотром оборудования убедитесь в отсутствии механических повреждений его элементов, отсутствии торчащих и оборванных проводов, следов горения электрооборудования. О выявленных недостатках сообщите преподавателю.

Приступайте к работе только после разрешения и в присутствии преподавателя.

Not merely a chip of the old block, but the old block itself.
Edmund Burke[1]

Лабораторная работа №1

Тема: Создание нового проекта в Keil µVision5.

Цель: Ознакомится с основными приемами работы с документацией при составлении программ для микроконтроллеров.

Постановка задачи: создать новый проект в Keil µVision5 и разработать программу для микроконтроллера (МК) STM32F200, которая включает и выключает светодиод.

Минздрав предупреждает: Курение вредит Вашему здоровью!

Материал для предварительного изучения:

1. Понятие и организация системных ресурсов.
2. Организация карты распределения адресов.
3. Средства языка. С для работы с адресами и данными.

Краткие теоретические сведения

Оценочная плата MCBSTM32F200, оснащенная микроэлектронным прибором STM32F207IGH, содержит все необходимые аппаратные компоненты на одном кристалле системы STM32x.

[1]Непросто копия отца, а сам живой отец (Эдмунд Бёрк).

Блок-схема аппаратных средств на рис. 1 показывает конфигурацию входов, системы питания и пользовательского ввода/вывода платы. Оценочная плата MCBSTM32F200 очень гибкая и легко конфигурируется под Ваши цели. В этой лабораторной работе нам понадобится порт светодиодов, показанный на рис. 2.

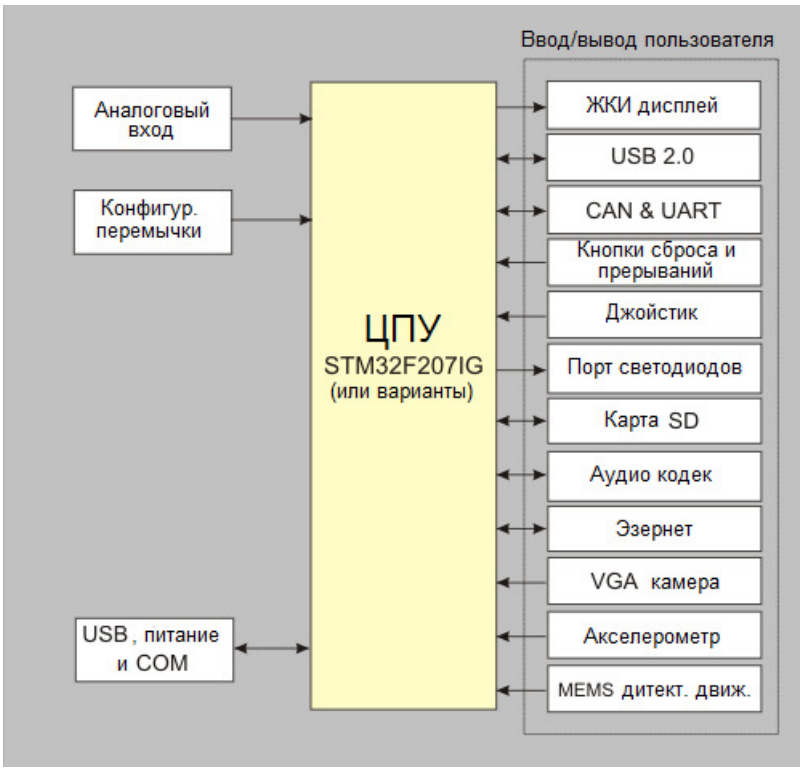


Рис. 1 Блок-схема аппаратных средств

Для работы с оценочной платой MCBSTM32F200 мы будем использовать мощную и простую среду программирования Keil µVision5.

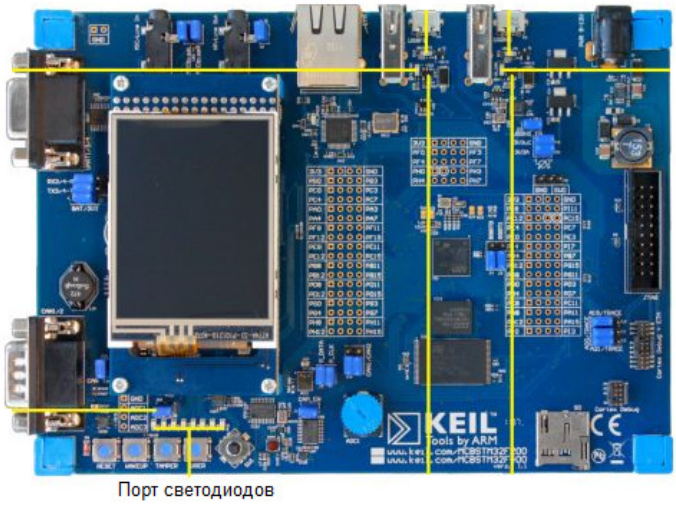


Рис. 2 Порт светодиодов оценочной платы

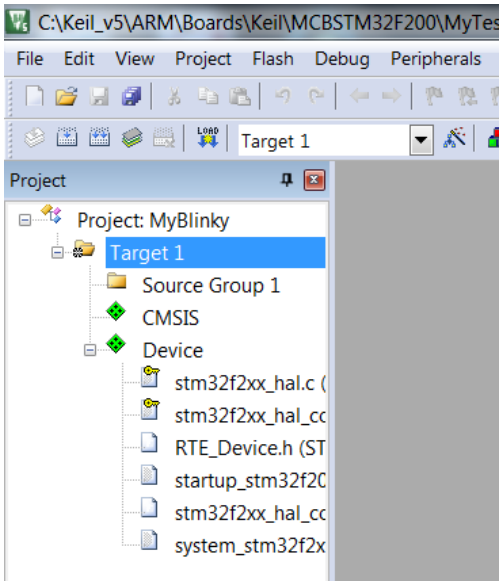
µVision представляет собой платформу развития программного обеспечения, объединяющую робастный современный редактор с менеджером проектов. µVision интегрирует в себе весь инструментарий необходимый для развития вложенных приложений, включая компилятор C++, макро ассемблер, компоновщик/позиционер и генератор шестнадцатеричных файлов.Цикл разработки программного обеспечения в µVision примерно такой же, как и в любом другом инструменте разработки программного обеспечения.

1. Создать проект, выбрать из базы данных чип, являющийся целевым для разработки, и сконфигурировать параметры инструментария.
2. Создать файлы источников на C++ или ассемблере.
3. С помощью менеджера проектов построить приложение.
4. Исправить ошибки в файлах источниках.
5. Протестировать скомпонованное приложение.

Программа работы

A. Создание нового проекта под названием MyBlinky

1. Запустите µVision и не в режиме отладки выберите Project/New µVision Project.
2. В открывшемся окне Create New Project откройте папку C:\Keil\ARM\Boards\Keil\MCBSTM32F200.
3. Щелкните правой кнопкой мыши и при помощи контекстного меню создайте папку MyTest.
4. Левой кнопкой мыши дважды щелкните по новой созданной папке для того, чтобы войти в нее.
5. В окне File name задайте имя проекта, например, MyBlinky.
6. В появившейся закладке Select Device for Target 1 в левом нижнем окне выберите папку STMicroelectronics и в этой папке тип микросхемы STM32F207IGHx.
7. В раскрывшемся окне Manage Run-Time Environment откройте папку CMSIS и поставьте отметку в окошке CORE (поддержка ядра), в папке Device (устройство) поставьте отметку в окошке Startup (запуск), затем в папке STM32Cube Framework … отметьте окошко Classic, а в папке STM32Cube HAL отметьте окошки Common и Cortex. Если некоторые из отмеченных окошек не окрашены в зеленый цвет, нажмите кнопку Resolve в нижнем левом углу. Затем кнопку OK.
8. Раскройте все папки в рабочем пространстве Project Workspace, показанном на рисунке ниже, щелкнув по значку + подле каждой папки



9. Осторожно щелкните по имени папки Target 1 и переименуйте ее в MCBSTM32F200, это позволит легко различать проекты разных поддерживаемых Keil_v5 микроконтроллеров.
10. Создайте главный исполняемый файл Си. Выберите из меню File пункт New.
11. Сохраните файл, например, под именем BlinkyLed.c.
12. Добавляем сохраненный файл в проект. Для чего на расположенной слева вкладке Project щелкаем правой кнопкой мыши по папке Source Group 1 и в появившемся контекстном окне выбираем Add Existent Files to Group «Source Group 1.
13. В появившемся окне выбираем добавляемый в проект файл BlinkyLed.c и щёлкаем ADD. В результате слева на вкладке Project в списке файлов проекта появится добавленный файл BlinkyLed.c. Нажимаем Clouse.
14. Набрать основное тело программы, используя синтаксис С.

Побитовые операции и работа с памятью в С

Предположим, что по адресу 0x60004012 тридцати двух битного регистра необходимо записать число 0x3B H. Согласно синтаксису языка С следует сделать следующее преобразование типов данных:

```
(unsigned long*)(0x60004012)
```

Таким образом, получен указатель на элемент. Теперь, в этот элемент необходимо записать требуемое значение. Последнее выполняется разыменовыванием указателя, при этом команда принимает вид:

```
*(unsigned long*)(0x60004012) = 0x3B;
```

* – говорит здесь о косвенной адресации.

1. Установка произвольных бит в 1. Пусть в регистре с адресом 0x60004012 следует установить в «1» *седьмой* и *первый* биты, причем значение всех остальных бит в регистре должно остаться неизмененным. Для выполнения этого необходимо использовать булеву операцию | (ИЛИ):

```
*(unsigned long*)(0x60004012) |= 0x82;
```

Отметим, что данная операция занимает 3 такта – чтение – модификация – запись.

2. Установка произвольных бит в 0. Пусть в регистре с адресом 0x60004012 следует установить в «0» *седьмой* и *первый* биты, причем значение всех остальных бит в регистре должно остаться неизмененным. Для выполнения этого необходимо использовать булеву операцию & (И):

```
*(unsigned long*)(0x60004012) &= 0xFFFFFF7D;
```

Той же цели можно достичь и с помощью более простой записи, воспользовавшись булевой операцией ~ (НЕ):

```
*(unsigned long*)(0x60004012) &= (~0x82);
```

Работа над программой для микроконтроллера (МК) всегда начинается с чтения документации на него, поскольку для удовлетворительного программирования необходимо знать аппаратную часть программируемого МК. Ниже приводятся некоторые сведения о портах ввода/вывода.

Порты ввода/вывода общего назначения

Каждый порт ввода/вывода общего назначения обладает четырьмя 32-битовыми регистрами конфигурации (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR и GPIOx_PUPDR), двумя 32-битовыми регистрами данных (GPIOx_IDR и GPIOx_ODR), 32-битовым регистром установки/сброса (GPIOx_BSRR), 32 битным регистром захвата (GPIOx_LCKR) и двумя 32-битовыми регистрами выбора альтернативной функции (GPIOx_AFRH и GPIOx_AFRL)

1. Основные характеристики портов ввода/вывода общего назначения

- Под управлением находится до 16 входов/выходов
- Состояния выходов: тяни-толкай (двухтактный выход) или с открытым стоком + подтягивание/приспускание
- Выходные данные из регистра выходных данных (GPIOx_ODR) или периферии (выход альтернативной функции)
- Выбор скорости для каждого входа/выхода
- Входные состояния: плавающее, с подтягиванием/приспусканием, аналоговое
- Входные данные в регистр входных данных (GPIOx_IDR) или периферию (вход альтернативной функции)
- Регистр установки и сброса (GPIOx_BSRR) для доступа к непосредственной записи в GPIOx_ODR
- Механизм захвата (GPIOx_LCKR) обеспечивающий замораживание конфигурации вход/выход
- Аналоговые функции

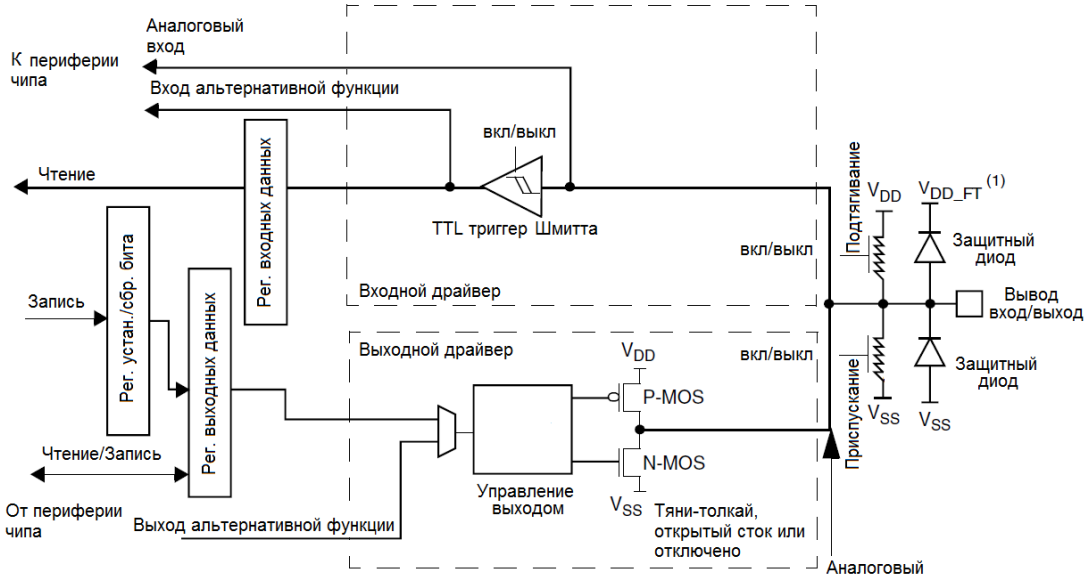
- Регистр выбора альтернативных функций вход/выход (самое большое 16 альтернативных функций на вход/выход)
- Быстрые способности к переключению в каждые два периода тактовой частоты
- Очень гибкое мультиплексирование выводов позволяющее использовать входные/выходные выводы в качестве портов ввода/ вывода общего назначения или в качестве одной из нескольких периферийных функций

1. Функциональное описание портов ввода/вывода общего назначения

Благодаря особенным характеристикам аппаратной части каждого порта ввода/вывода, приведенным в справочном листке, каждый бит портов общего назначения может быть индивидуально сконфигурирован программным обеспечением на работу в нескольких режимах:

- Плавающий вход
- Подтянутый вход
- Приспущенный вход
- Аналоговый
- Выход, открытый сток с возможностью подтягивания или приспускания
- Выход, тяни-толкай с возможностью подтягивания или приспускания
- Альтернативная функция, тяни-толкай с возможностью подтягивания или приспускания
- Альтернативная функция, открытый сток с возможностью подтягивания или приспускания

Каждый бит порт ввода/вывода свободно программируется, однако доступ к регистрам порта ввода вывода может быть осуществлен 32-разрядным словом, полусловом или байтом. Целью регистра GPIOx_BSRR является обеспечение доступа атомарного чтения/модификации любого из регистраторов GPIO. В таком способе доступа отсутствует риск возникновения IRQ между чтением и модификацией. На рис. 1а показана основная структура выдерживающего 5В бита порта ввода/вывода. В таблице 1а приведены возможные конфигурации бита порта.



1. V_{DD_FT} является потенциалом выдерживающего 5В входа/выхода и отличается от V_{DD}

Рис. 1а

Таблица 1а **Конфигурация бита порта**(1)

MODER(i) [1:0]	**OTYPER(i)**	**OSPEEDR(i) [B:A]**		**PUPDR(i) [1:0]**		**Конфигурация вход/выход**	
01	0	SPEED [B:A]		0	0	выход GP	PP
	0			0	1	выход GP	PP + PU
	0			1	0	выход GP	PP + PD
	0			1	1	Зарезервировано	
	1			0	0	выход GP	OD
	1			0	1	выход GP	OD+PU
	1			1	0	выход GP	OD+PD
	1			1	1	Зарезервировано (выход GP с OD)	
10	0	SPEED [B:A]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Зарезервировано	
	1			0	0	AF	OD
	1			0	1	AF	OD+PU
	1			1	0	AF	OD+PD
	1			1	1	Зарезервировано	
00	X	X	X	0	0	Вход	Плавающий
	X	X	X	0	1	Вход	PU
	X	X	X	1	0	Вход	PD
	X	X	X	1	1	Зарезервировано (плавающий вход)	
11	X	X	X	0	0	Вход/Выход	Аналоговый
	X	X	X	0	1	Зарезервировано	
	X	X	X	1	0		
	X	X	X	1	1		

1. GP = общего назначения, PP = тяни-толкай, PU = подтянутый, PD = приспущенный, OD = с открытым стоком, AF = альтернативной функции.

Более подробные сведения приведены в справочном руководстве в файле CD00225773.pdf.

2. Постановка задачи. Настало время определиться с тем, что будет делать наша программа и с какими блоками нам предстоит работать.

Пусть первым нашим проектом будет ***мигающий светодиод***. После изучения фрагмента принципиальной схемы оценочной платы, показанного на рис. 2а, остановим свой выбор на светодиоде PG7.

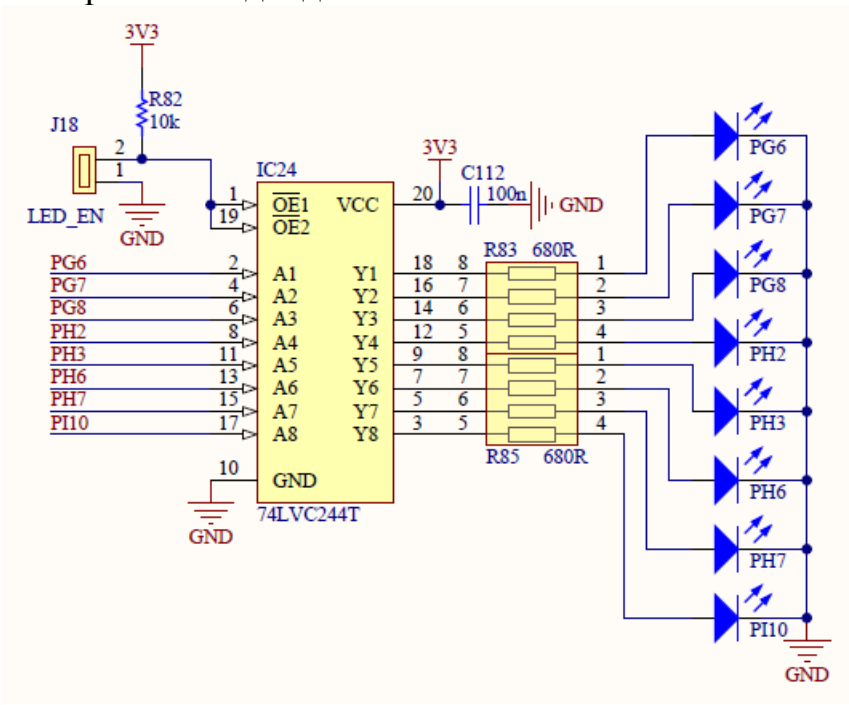


Рис. 2а

Из схемы следует, что для того, что бы «зажечь» светодиод необходимо на время латентного периода глаза (около 20 миллисекунд) перевести *седьмой* разряд порта GPIOG в состояние «1» (верхний уровень значения которой в данном МК не превышает 3,3 В), а для того чтобы «потушить» светодиод необходимо *седьмой* разряд порта GPIOG перевести в состояние «0» (верхний уровень значения «0» не превышает 0,4 В).

3. Управление тактированием периферийных блоков. С целью снижения энергопотребления МК после подачи на него питания практически все его периферийные блоки выключены. Включение/выключение блока производится *подачей/прекращением подачи* на его вход тактового сигнала. Таким образом, для корректной работы периферийного блока, необходимо сконфигурировать контроллер тактового сигнала МК так, чтобы к необходимому периферийному блоку стал поступать тактовый сигнал.

За включение тактирования периферийных блоков отвечают регистры включения тактирования периферии RCC XXX. Вместо XXX могут стоять шины AHB1, AHB2, AHB3, APB1 и APB2. После внимательного изучения описания соответствующих регистров, приведенное в файле CD00225773.pdf, можно сделать вывод о том, тактирование периферийного блока GPIOG включается установкой *шестого* бита регистра RCC AHB1 peripheral clock enable register (RCC_AHB1ENR) в состояние «1» (см. рис. 3а), т.е. записью в регистр 0x40 Н.

5.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved		DMA2EN	DMA1EN	Reserved		BKPSRAMEN	Reserved	
	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
Reserved			CRCEN	Reserved			GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 6 **GPIOGEN:** IO port G clock enable
Set and cleared by software.
0: IO port G clock disabled
1: IO port G clock enabled

Рис. 3а

Теперь следует определить адрес самого регистра RCC_AHB1ENR.

4. Определение адресов регистров. Для определения адресов регистров обратимся сначала к разделу “Карта памяти” из справочного руководства, приведенного в файле CD00225773.pdf. Здесь каждому блоку выделен свой участок адресного пространства. Например, для блока RCC это участок 0x4002 3800 — 0x4002 3BFF (см. рис. 4а). Для получения адреса регистра RCC_AHB1ENR, необходимо к начальному граничному адресу адресного пространства блока RCC, т.е. к 0x4002 3800 прибавить адрес смещения требуемого регистра, т.е. 0x30 (см. рис. 3а). Таким образом, адресом регистра RCC_AHB1ENR будет 0x4002 3830.

2.3 Memory map

See the datasheet corresponding to your device for a comprehensive diagram of the memory map. *Table 1* gives the boundary addresses of the peripherals available in all STM32F20x and STM32F21x devices.

Table 1. STM32F20x and STM32F21x register boundary addresses

Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FSMC control register	AHB3	*Section 31.6.9: FSMC register map on page 1300*
0x5006 0800 - 0X5006 0BFF	RNG	AHB2	*Section 20.4.4: RNG register map on page 537*
0x5006 0400 - 0X5006 07FF	HASH		*Section 21.4.8: HASH register map on page 557*
0x5006 0000 - 0X5006 03FF	CRYP		*Section 19.6.11: CRYP register map on page 531*
0x5005 0000 - 0X5005 03FF	DCMI		*Section 12.8.12: DCMI register map on page 299*
0x5000 0000 - 0X5003 FFFF	USB OTG FS		*Section 29.16.6: OTG_FS register map on page 1028*
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1	*Section 30.12.6: OTG_HS register map on page 1172*
0x4002 9000 - 0x4002 93FF	ETHERNET MAC		*Section 28.8.5: Ethernet register maps on page 941*
0x4002 8C00 - 0x4002 8FFF			
0x4002 8800 - 0x4002 8BFF			
0x4002 8400 - 0x4002 87FF			
0x4002 8000 - 0x4002 83FF			
0x4002 6400 - 0x4002 67FF	DMA2		*Section 9.5.11: DMA register map on page 208*
0x4002 6000 - 0x4002 63FF	DMA1		
0x4002 4000 - 0x4002 4FFF	BKPSRAM		
0x4002 3C00 - 0x4002 3FFF	Flash interface register		See Flash programming manual
0x4002 3800 - 0x4002 3BFF	RCC		*Section 5.3.24: RCC register map on page 135*
0x4002 3000 - 0x4002 33FF	CRC		*Section 3.4.4: CRC register map on page 61*
0x4002 2000 - 0x4002 23FF	GPIOI		*Section 6.4.11: GPIO register map on page 156*
0x4002 1C00 - 0x4002 1FFF	GPIOH		
0x4002 1800 - 0x4002 1BFF	GPIOG		
0x4002 1400 - 0x4002 17FF	GPIOF		
0x4002 1000 - 0x4002 13FF	GPIOE		
0X4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		

Рис. 4а

5. Управление режимом работы блока GPIO. Управление режимами работы разрядов портов ввода/вывода общего назначения выполняется с помощью регистра режимов, показанного на рис. 5а (см. файл CD00225773.pdf).

6.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[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
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Рис. 5а

Как видно для перевода *седьмого* разряда блока GPIOG в режим вывода необходимо записать значение 01 в 15-14 биты регистра GPIOG_MODER. Адрес регистра определяем аналогично, как и в случае с регистром RCC_AHB1ENR описанном ранее. Адресом регистра GPIOG_MODER является 0x4002 1800.

6. Настройка параметров работы выходных выводов порта блока GPIO. Дополнительные настройки для выходных выводов блока GPIOG выполняются в регистрах:

- GPIOG_OTYPER – задается тип выхода тяни-толкай или открытый сток
- GPIOG_OSPEEDR – задается скорость работы выхода

Здесь будем использовать параметры по умолчанию (тяни-толкай, низкая скорость).

7. Установка требуемых значений на выводе МК. Для установки требуемого значения на заданном выводе порта МК используем регистр GPIOG_ODR (см. рис 6а).

6.4.6 GPIO port output data register (GPIOx_ODR) (x = A..I)

Address offset: 0x14

Reset value: 0x0000 0000

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
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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I).

Рис. 6а

Запись «0» или «1» в биты 0-15 приводят к соответствующему изменению состояния выводов порта. Для того чтобы установить определенное значение на выходе одного или нескольких выводов МК и не изменить состояния остальных, будем использовать булевы операции специально предназначенные для работы с отдельными битами так, как это уже показывалось выше. Для записи «1» в бит PG7 в регистр следует записать 0x80 H. Адресом регистра GPIOG_ODR является 0x4002 1814.

Данный метод предпочтительнее всего использовать для изменения состояния выхода на противоположное, если его изначальное состояние не известно. В противном случае, можно использовать регистр установки/сброса бита GPIOG_BSRR.

8. Составление программы. Текст программы представлен ниже:

```
/*-------------------------------------------------------------
 * Name:    Blinky.c
 * Purpose: LED PG7 Flasher for MCBSTM32F200
 *-------------------------------------------------------------*/
/*-------------------------------------------------------------
  Main function
 *-------------------------------------------------------------*/
int main ()
{
//-------Declaration of type of variables--------------------

int i; //counter for get ready delay
unsigned long int  j; //counter for blinky delay

//-------Initialization of variables-------------------------

i=0;
j=0;

//-------Main cycle of algorithm-----------------------------

  *(unsigned long*)(0x40023830) |= 0x40; //Enable port G clocking
```

```
    for(i=0; i<4; i++){} //small delay for GPIOG get ready

    *(unsigned long*)(0x40021800) = (*(unsigned long*)(0x40021800) &
(~0x00008000)) | (0x00004000); //Set PG7 as General purpose output

  while(1)
    {
    *(unsigned long*)(0x40021814) |= 0x80; //Turn LED ON!

      for( j=0; j<2000000 ;j++ ){} //Delay

      *(unsigned long*)(0x40021814) &= ~0x80; //Turn LED OFF

        for(j=0; j<2000000 ; j++){} //Delay
    }
  }
```

15. Постройте проект, нажав F7 или выбрав пункт Build из меню Project.
16. Запустите отладочный режим, выбрав из меню Debug пункт Start/Stop Debug Session.
17. В появившемся окошке Evaluation Mode нажмите кнопку OK.
18. Находясь в режиме отладки, выберите из меню Debug пункт Run.
19. Программа загрузится в оценочную плату и светодиод PG7 начнет мигать.
20. Для остановки работы программы в требуемый момент времени выберите из меню Debug пункт Stop.

B. Модификация программы BlinkyLed

1. Модифицируйте текст программы BlinkyLed.c добавив в нее определения адресов и данных.

Создание определений

Работа с адресами, выраженными в виде чисел неудобна, особенно если программа достаточно велика и в процессе ее составления требуется обращение к большому количеству разных адресов и в различных местах программы.

Составление программы значительно упрощается, если для используемых адресов сделать определения.

Определение делается присвоением числовому значению адреса мнемонического обозначения, например названия регистра которому принадлежит данный адрес. При таком подходе, например, фрагмент программы BlinkyLed.c

```
*(unsigned long*)(0x40023830) |= 0x40; //Enable port G clocking
```

будет выглядеть как:

определение в программе

```
#define RCCAHB1_ENR (unsigned long*)0x40023830
```

строка программы, использующая сделанное определение

```
*RCCAHB1_ENR |= 0x40; // Enable port G clocking
```

Можно пойти еще дальше и присвоить определения часто используемым числовым данным. В этом случае рассматриваемый фрагмент программы будет выглядеть следующим образом:

```
#define RCCAHB1_ENR (*(unsigned long*)0x40023830)
#define RCCAHB1_PORTG 0x40

RCCAHB1_ENR |= RCCAHB1_PORTG; // Enable port G clocking
```

Обратите внимание, что здесь в первой строке одновременно определяется и косвенная адресация к регистру (первая звездочка).

2. Постройте проект MyBlinkyMod, нажав F7 или выбрав пункт Build из меню Project.
3. Запустите отладочный режим, выбрав из меню Debug пункт Start/Stop Debug Session.
4. В появившемся окошке Evaluation Mode нажмите кнопку OK.
5. Находясь в режиме отладки, выберите из меню Debug пункт Run.
6. Программа загрузится в оценочную плату и светодиод PG7 начнет мигать.
7. Для остановки работы программы в требуемый момент времени выберите из меню Debug пункт Stop.

В заключение заметим, что подход, подобный рассмотренному нами, используется и в Keil µVision5. Так обсуждавшийся выше фрагмент программы в Keil µVision5 будет выглядеть следующим образом:

```
RCC->AHB1ENR |= 1<<6; // Enable port G clocking
```

Содержание отчета:

1. Тема и цель работы.
2. Задание на лабораторную работу.
3. Алгоритм программы.
4. Полученные результаты.
5. Выводы по работе с анализом реализованной программы.

Литература:

1. MCBSTM32F200-400 User's Guide. Доступно (2015, Июнь): http://www.keil.com/
2. µVision User's Guide. Доступно (2015, Июнь): https://www.keil.com/
3. Дейтел Г., Дейтел П. Как программировать на C++. М.: Бином, 2004, 1152 с.