Лабораторная работа № 1

Контроллеры ARM: порты GPIO

Цель работы: изучение устройства и принципов программирования портов общего назначения у микроконтроллеров с архитектурой ARM7.

1. Устройство портов ввода/вывода

Порт ввода/вывода общего назначения (General purpose input/output - GPIO) является программно-управляемым элементом микроконтроллера. На аппаратном уровне порт объединяет несколько выводов микроконтроллера в группу, работа которой программируется общим набором регистров. В STM32F порты имеют 16 выводов и являются 16-ти разрядными, т.е. с помощью управляющего регистра можно одной командой микроконтроллера изменить сигналы на всех выводах одного порта. Обычно названия портов обозначаются латинскими буквами A, B, C и т.д. Выводы (линии) порта нумеруются цифрами 0, 1, 2 и т.д. Причем нуль используется для обозначения вывода, который соответствует самой младшей группе бит управляющего регистра, и далее по старшинству. Так 4-ый вывод порта B на схеме микроконтроллера обозначают – PB4. Аналогичным образом обозначают остальные выводы портов ввода/вывода общего назначения (рис. 1).

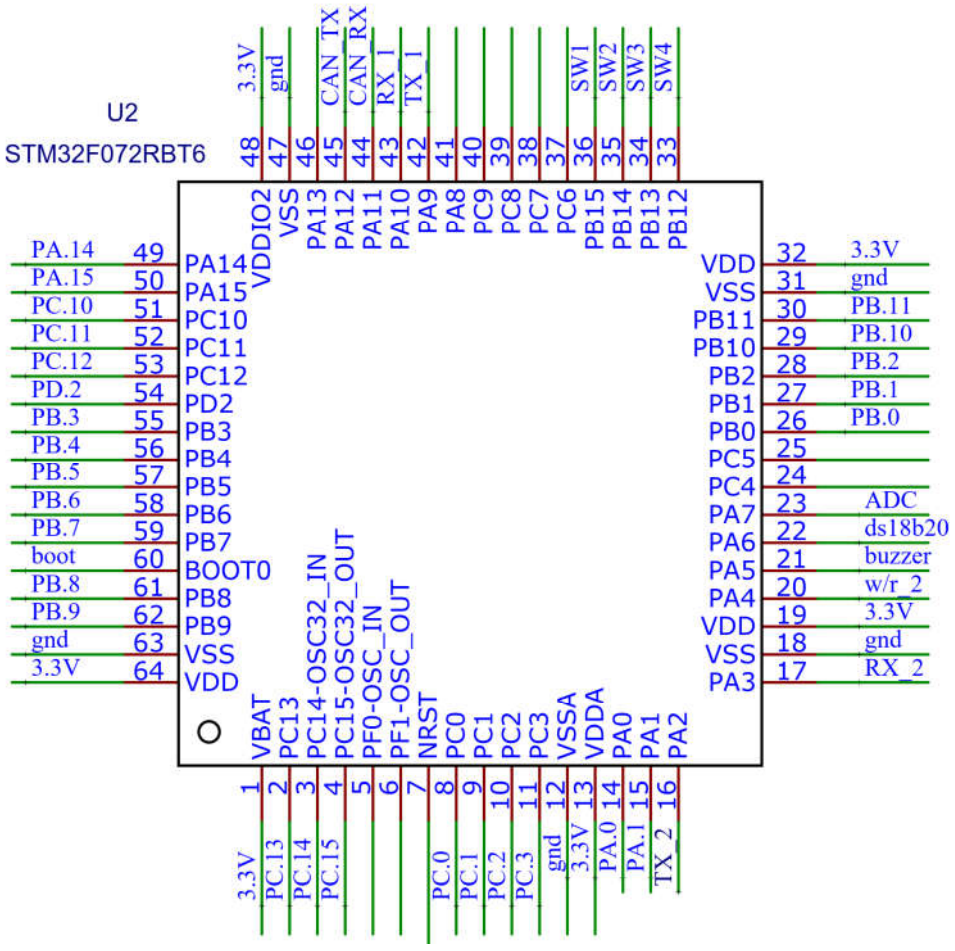


Рис. 1. Назначение выводов микроконтроллера STM32F072RBT6

В программной модели микроконтроллера каждый управляющий регистр имеет уникальное имя, которое используется для работы с ним. Полное описание всех регистров STM32F можно найти в технической документации (на английском языке) – **Reference manual**. В ней описываются общие принципы работы всех периферийных устройств, в том числе портов GPIO, а также назначение каждого управляющего регистра.

Включение/выключение тактирования портов GPIO

За каждый порт GPIO отвечает несколько регистров. Однако перед началом настройки управляющих регистров порта нужно включить их тактирование, разрешив передачу сигнала системного таймера в блок порта. В технологии КМОП потребление энергии происходит только при переключении состояния. Без тактирования нет переключений. Поэтому для экономии энергии после перезагрузки микроконтроллера тактирование его периферии отключено.

За управление тактированием элементов микроконтроллера отвечает система регистров RCC. Для управления тактированием периферии используется регистр **RCC_AHBENR** (программный доступ: **RCC-> AHBENR**)(рис. 2).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCEN	Res.	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	FLITFEN	Res.	SRAMEN	Res.	DMAEN
									rw		rw		rw		rw

Рис. 2. Назначение бит регистра RCC_AHBENR

За включение/выключение тактирования портов GPIO отвечают 17 – 22 бит этого регистра. Значение «1» разрешает тактирование соответствующего порта, а «0» – запрещает. Для упрощения программирования этого регистра используются специально именованные константы, которые содержат единицу в соответствующем бите. Например, для включения тактирования порта A и B в программе на языке C++ следует указать команду:

```
RCC->AHBENR |= RCC_AHBENR_GPIOAEN | RCC_AHBENR_GPIOBEN;
```

А для выключения тактирования только порта B нужно указать команду:

```
RCC->AHBENR &= ~RCC_AHBENR_GPIOBEN;
```

Настройка режима работы порта GPIO

Каждая линия порта GPIO имеет непростую систему управления, которая позволяет определять направление потока данных при реализации основной функции, а также переключаться на альтернативную функцию (подключать к выводу другой блок микроконтроллера) или включать аналоговый режим работы (вывод обеспечивает передачу сигнала до/от АЦП, ЦАП, компаратора) (рис. 2).

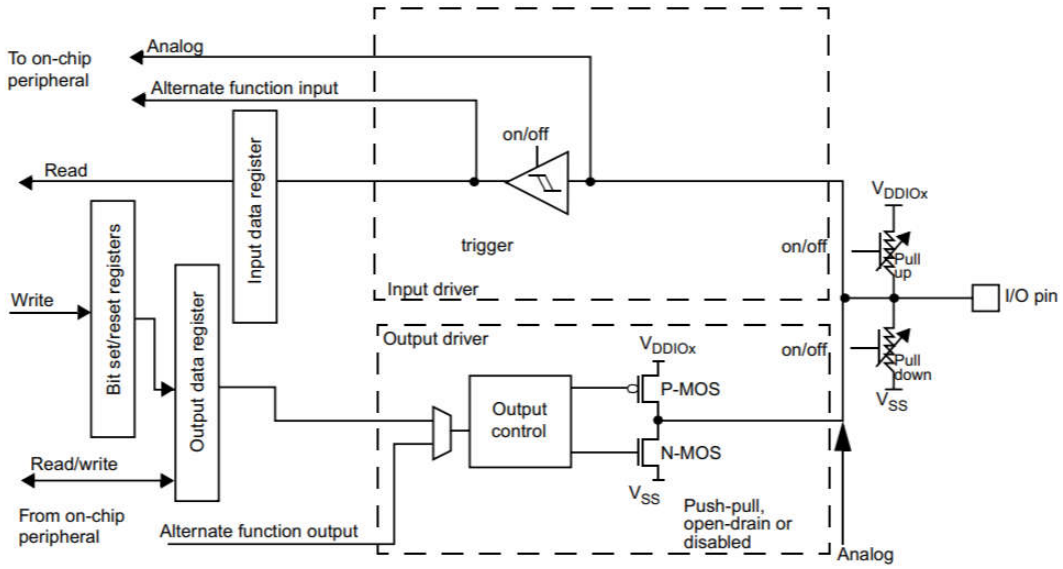


Рис. 2. Блок-схема одной линии порта ввода/вывода микроконтроллера STM32F

Таким образом, для управления каждый порт ввода-вывода общего назначения имеет четыре 32-битных регистра конфигурации (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR и GPIOx_PUPDR), два 32-битных регистра данных (GPIOx_IDR и GPIOx_ODR) и 32-битный регистр установки / сброса (GPIOx_BSRR). Порты A и B также имеет 32-битный регистр блокировки (GPIOx_LCKR) и два 32-битных регистры выбора альтернативных функций (GPIOx_AFRH и GPIOx_AFRL).

Регистры конфигурации совместно определяют режимы работы линий порта (рис. 3).

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output 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	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

Рис. 3. Таблица конфигурации линий порта (x – бит не имеет значения)

Регистр GPIOx_MODER (программный доступ: **GPIOx->MODER**) (x – буква порта, изменяется от A до F) используется для переключения между вводом/выводом бинарных данных (основная функция), альтернативной функцией и аналоговым режимом работы линий (рис. 4).

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

Рис. 4. Структура регистра GPIOx_MODER

Линии порта в регистре обозначаются **MODERx**, где x – номер линии порта, который изменяется от 0 до 15. Для настройки режима работы каждой линии порта используется два бита, определяющие 4 возможных комбинации:

- 00 – режим ввода бинарных данных (основная функция);
- 01 – режим вывода бинарных данных (основная функция);
- 10 – режим альтернативной функции;
- 11 – аналоговый режим работы линии.

Все биты регистра **GPIOx_MODER** можно как читать в программе (r), так и устанавливать новые значения (w). Например, для включения линии 3 порта B в режим альтернативной функции следует указать команду:

```
GPIOB->MODER |= 0x00000080;
```

или

```
GPIOB->MODER |= GPIO_MODER_MODER3_1; //Используя специальную константу
```

Переключить линию 3 порта A в режим ввода бинарных данных можно командой:

```
GPIOA->MODER &= ~0x000000C0;
```

или

```
GPIOA->MODER &= ~GPIO_MODER_MODER3; //Используя специальную константу
```

Регистр GPIOx_OTYPER (программный доступ: **GPIOx->OTYPER**) (x – буква порта, изменяется от A до F) используется для указания режима вывода бинарных данных (рис. 5).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 5. Структура регистра GPIOx_OTYPER

Линии порта в регистре обозначаются **OTx**, где x – номер линии порта, который изменяется от 0 до 15. Для настройки режима работы каждой линии порта используется один бита, определяющий 2 состояния:

- 0 – вывод данных в режиме «push-pull» (**PP**)(Обычный активный выход. При низком логическом уровне напряжение на выводе равно 0, при высоком – напряжение близко к напряжению питания микроконтроллера, обычно +3В. Режим используется по умолчанию после перезагрузки микроконтроллера);
- 1 – вывод данных в режиме «open-drain»(**OD**)(Работа выхода в режиме «открытый сток»).

Все биты регистра **GPIOx_OTYPER** можно как читать в программе (r), так и устанавливать новые значения (w). Например, для включения линии 2 порта B в режим **OD** следует указать команду:

GPIOB->OTYPER |= 0x00000004;

или

GPIOB->MODER |= GPIO_OTYPER_OT2; //Используя специальную константу

Переключиться обратно в режим **PP** можно командой:

GPIOB->OTYPER &=~0x00000004;

или

GPIOB->OTYPER &=~GPIO_OTYPER_OT2; //Используя специальную константу

Регистр GRIOx_OSPEEDR (программный доступ: **GPIOx->OSPEEDR**) (x – буква порта, изменяется от A до F) используется для настройки частоты установки обновленных данных на выводе порта (рис. 6).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [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
OSPEEDR7 [1:0]		OSPEEDR6 [1:0]		OSPEEDR5 [1:0]		OSPEEDR4 [1:0]		OSPEEDR3 [1:0]		OSPEEDR2 [1:0]		OSPEEDR1 [1:0]		OSPEEDR0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 6. Структура регистра GRIOx_OSPEEDR

Линии порта в регистре обозначаются **OSPEEDRx**, где x – номер линии порта, который изменяется от 0 до 15. Для настройки режима работы каждой линии порта используется два бита, определяющие 4 возможных комбинации:

- 00 или 10 – низкая скорость (частота синхронизации 2 МГц);
- 01 – средняя скорость (частота синхронизации 10 МГц);
- 11- высокая скорость (частота синхронизации 50 МГц).

Все биты регистра **GPIOx_OSPEEDR** можно как читать в программе (r), так и устанавливать новые значения (w). Например, для настройки режима тактирования выхода по линии 4 порта A «высокая скорость» следует указать команду:

GPIOA->OSPEEDR |= 0x00000300;

или

GPIOA->OSPEEDR |= GPIO_OSPEEDR_OSPEEDR4; //Используя спец. константу

Переключить эту линию с высокой скорости на среднюю можно командой:

GPIOA->OSPEEDR &=~0x00000100;

или

GPIOA->OSPEEDR &=~ GPIO_OSPEEDR_OSPEEDR4_0; //Используя спец. константу

Регистр GPIOx_PUPDR (программный доступ: **GPIOx->PUPDR**) (x – буква порта, изменяется от A до F) используется для подключения/отключения подтягивающих резисторов к выводу микроконтроллера (см. рис. 2). Структура регистра показана на рисунке 7.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 7. Структура регистра GPIOx_PUPDR

Линии порта в регистре обозначаются **PUPDRx**, где x – номер линии порта, который изменяется от 0 до 15. Для настройки режима работы каждой линии порта используется два бита, определяющие 4 возможных комбинации:

- 00 – подтягивающие резисторы отключены;
- 01 – к выводу контроллера подключается резистор «Pull Up» (рис. 2);
- 10 – к выводу контроллера подключается резистор «Pull Down» (рис. 2);
- 11 – значение зарезервировано (не используется).

Все биты регистра **GPIOx_PUPDR** можно как читать в программе (r), так и устанавливать новые значения (w). Например, для подтяжки входа на линии 1 порта A к напряжению питания нужно сначала перевести эту линии в режим работы «на вход», а затем указать команду:

GRIOA->PUPDR |=0x00000004;

или

GPIOA->PUPDR |= GPIO_PUPDR_PUPDR1_0; //Используя специальную константу

Работа с данными порта ввода/вывода

Регистр GPIOx_IDR (программный доступ: **GPIOx->IDR**) (x – буква порта, изменяется от A до F) используется для ввода (параллельного) данных с линий порта, работающих в режиме «на вход» (рис. 8). Линии порта в регистре обозначаются **IDRx**, где x – номер линии порта, который изменяется от 0 до 15.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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. 8. Структура регистра GPIOx_IDR

Биты регистра **GPIOx_IDR** можно только читать в программе (r). Например, для ввода данных со всех линий порта B можно использовать такой код:

```
uint32_t Data;
Data=GPIOB->IDR;
```

Регистр GPIOx_ODR (программный доступ: **GPIOx->ODR**) (x – буква порта, изменяется от A до F) используется для вывода (параллельного) данных на линии порта, работающих в режиме «на выход» (рис. 9). Линии порта в регистре обозначаются **ODRx**, где x – номер линии порта, который изменяется от 0 до 15.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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. 9. Структура регистра GPIOx_ODR

Все биты регистра **GPIOx_ODR** можно как читать в программе (r), так и устанавливать новые значения (w). Операции чтения и записи в этот регистр не являются атомарными, т.е. они могут быть приостановлены в процессе обработки микроконтроллером прерывания. Пример: обновления данных на выводах порта C можно добиться командой:

```
uint32_t Data;
Data=0xAC002F19;
GPIOC->ODR=Data;
```

Регистр GPIOx_BSRR (программный доступ: **GPIOx->BSRR**) (x – буква порта, изменяется от A до F) используется для установки и сброса отдельных бит регистра **GPIOx_ODR** с помощью атомарной операции записи. Регистр разделен на две части. Биты с 0 по 15 используются для установки в «1» битов регистра GPIOx_ODR. При этом соответствующие биты регистра **GPIOx->BSRR** обозначаются **BSx**, где x – номер линии порта, который изменяется от 0 до 15. Биты с 16 по 31 используются для сброса в «0» битов регистра GPIOx_ODR. При этом соответствующие биты регистра **GPIOx->BSRR** обозначаются **BRx**, где x – номер линии порта, который изменяется от 0 до 15 (рис. 10).

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. 10. Структура регистра GPIOx_BSRR

Регистр можно использовать только на запись значений (w). Например, для установки «1» на 12 линии порта B, работающей в режиме «на выход», нужно выполнить команду:

GPIOB->BSRR |= 0x00001000;

А сброс в ноль сигнала на этой линии выполнит команда:

GPIOB->BSRR |= 0x10000000;

В команде:

GPIOB->BSRR |= 0x10001000;

приоритет будет отдан «установке» и после нее на 12-ом выводе порта B зафиксируется «1».

Регистр GPIOx_BRR (программный доступ: **GPIOx->BRR**) (x – буква порта, изменяется от A до F) используется для сброса отдельных бит регистра **GPIOx_ODR** с помощью атомарной операции записи. Линии порта в регистре обозначаются **BRx**, где x – номер линии порта, который изменяется от 0 до 15. (рис. 11).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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. 11. Структура регистра GPIOx_BRR

Регистр можно использовать только на запись значений (w). Работа с ним аналогична работе с регистром GPIOx_BSRR за исключением возможности установки бит регистра GPIOx_ODR.

Выбор альтернативных функции GPIO и блокировка конфигурирования порта

Регистр GPIOx_AFRL (программный доступ: **GPIOx->AFRL**) (x – буква порта, изменяется от A до F) используется для выбора одной из 8 альтернативных функций на выводе порта для линий с 0 по 7 (рис. 12).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
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
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 12. Структура регистра GPIOx_AFRL

Для настройки режима работы каждой линии порта используется 4 бита, определяющие 16 возможных комбинации, но для выбора доступно только 8 альтернативных функций, поэтому сочетания 1xxx зарезервированы и не используются.

Регистр GPIOx_AFRH (программный доступ: **GPIOx->AFRH**) (x – буква порта, изменяется от A до F) используется для выбора одной из 8 альтернативных функций на выводе порта для линий с 9 по 15 (рис. 13).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
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
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 13. Структура регистра GPIOx_AFRH

Для настройки режима работы каждой линии порта используется 4 бита, определяющие 16 возможных комбинации, но для выбора доступно только 8 альтернативных функций, поэтому сочетания 1xxx зарезервированы и не используются.

Регистр GPIOx_LCKR (программный доступ: **GPIOx-> LCKR**) (x – буква порта, изменяется от A до F) используется для блокировки конфигурации соответствующих линий микроконтроллера.

2. Структура стенда

Учебный лабораторный комплекс STM_01 (рис. 14) предназначен для изучения принципов организации микропроцессорных систем, структуры и функционирования базовых компонентов (памяти, контроллеров ввода-вывода, подсистемы памяти и т.д.), получения навыков программирования встраиваемых систем различного назначения. Отличительными чертами STM_01 является высокая производительность процессорного ядра, ориентация на одну из наиболее динамично развивающихся архитектур - ARM7, большое разнообразие периферии: энкодер, клавиатура 3x4, семисегментный индикатор, цифровой датчик температуры, зуммер, порты CAN 2.0 и RS-485.

В состав учебного стенда входят:

- Микроконтроллер STMicroelectronics STM32F072RBT6 на базе ядра STM32 ARM Cortex M0 RISC 128KB Flash;
- Подсистема интерфейса человек-машина, состоящая из семисегментного индикатора, клавиатуры 3x4, звукового излучателя, набора микропереключателей;
- Сетевая подсистема, содержащая один порт CAN-2.0 и один канал RS-485;
- Подсистема ввода-вывода включающая в себя 1 канал АЦП, подключенный к переменному резистору, и 12 каналов дискретного ввода-вывода.

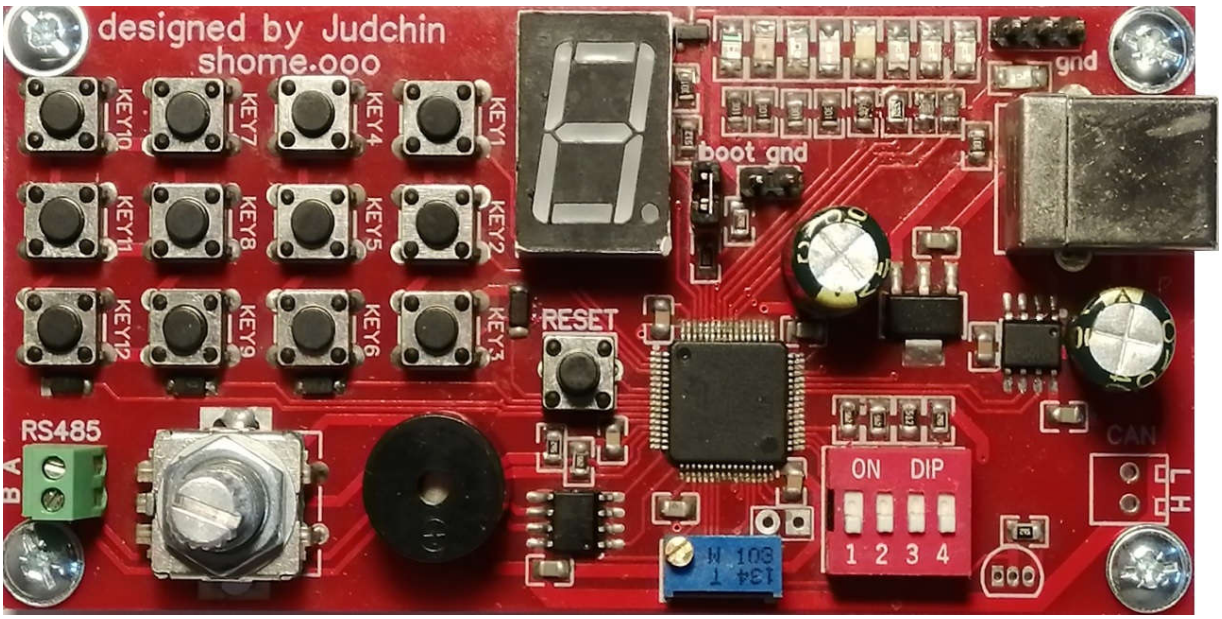


Рис. 14. Стенд STM_01 для изучения микроконтроллера с ARM архитектурой

Схемы подключения к микроконтроллеру внешних цепей, используемых в настоящей лабораторной работе, представлены ниже. На рисунке 15 изображена схема подключения светодиодов к выводам микроконтроллера, работающим в режиме «на выход» («Output»). На рисунке 16 показаны схемы подключения к подобным выводам микроконтроллера семисегментного индикатора и зуммера.

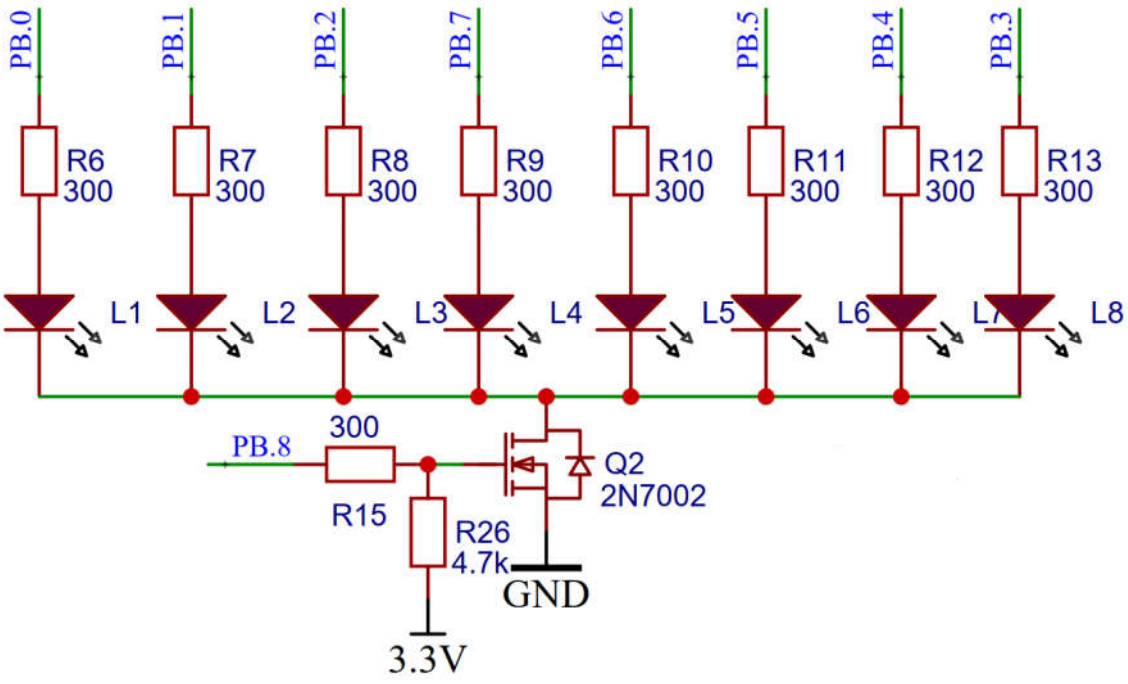
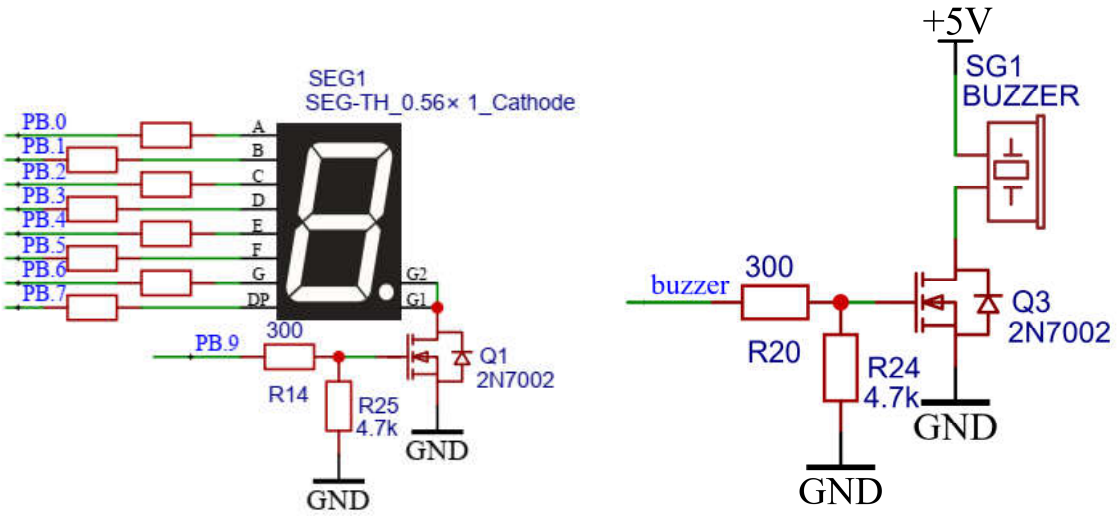


Рис. 15. Схема подключения светодиодов



(а) (б)

Рис. 16. Схема подключения семисегментного индикатора (а), зуммера (б)

На рисунке 17 приведена схема подключения микропереключателей к выводам микроконтроллера, работающим в режиме «на вход» («Input»).

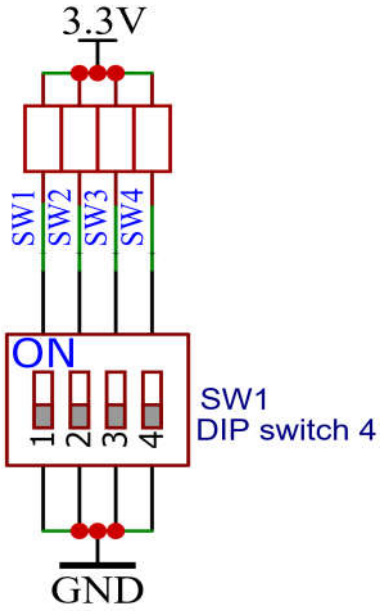


Рис. 17. Схема подключения микропереключателей

3. Задание

1. Создайте папку «LabHard(<фамилия на латинице>-<номер группы>)-1». Например, студент Иванов из группы 1191б должен создать папку: LabHard(Ivanov-1191)-1.
2. Внутри папки «LabHard…» создайте папки: «Task1_1», «Task1_2» и «Task1_3». В них далее следует размещать проекты среды Keil µVision, соответственно предназначенные для решения: обучающей, самостоятельной и индивидуальных частей задания.

3.1. Обучающая часть

Создание проекта

1. Запустите приложение Keil µVision.
2. Выберите пункт меню «Project – New µVision Project…».
3. В проявившемся диалоговом окне выберите папку Task1_1 и введите название проекта: «<инициалы ФИО на латинице>1_1». Например, проект Сидорова Ивана Петровича должен иметь название: **sip1_1**. Нажмите кнопку «Ok».
4. Далее в диалоговом окне «Select Device…» раскройте список «STMicroelectronics» и выберите микроконтроллер «STM32F072RBTx». Нажмите «Ok».
5. В следующем диалоговом окне «Manager Run-Time Environment» отметьте элементы, показанные на рисунке 18. Нажмите «Ok». Это добавит в Ваш проект библиотеки, обеспечивающие выдачу диагностических сообщений для Cortex-M и поддержку программной модели STM32F072RBTx.

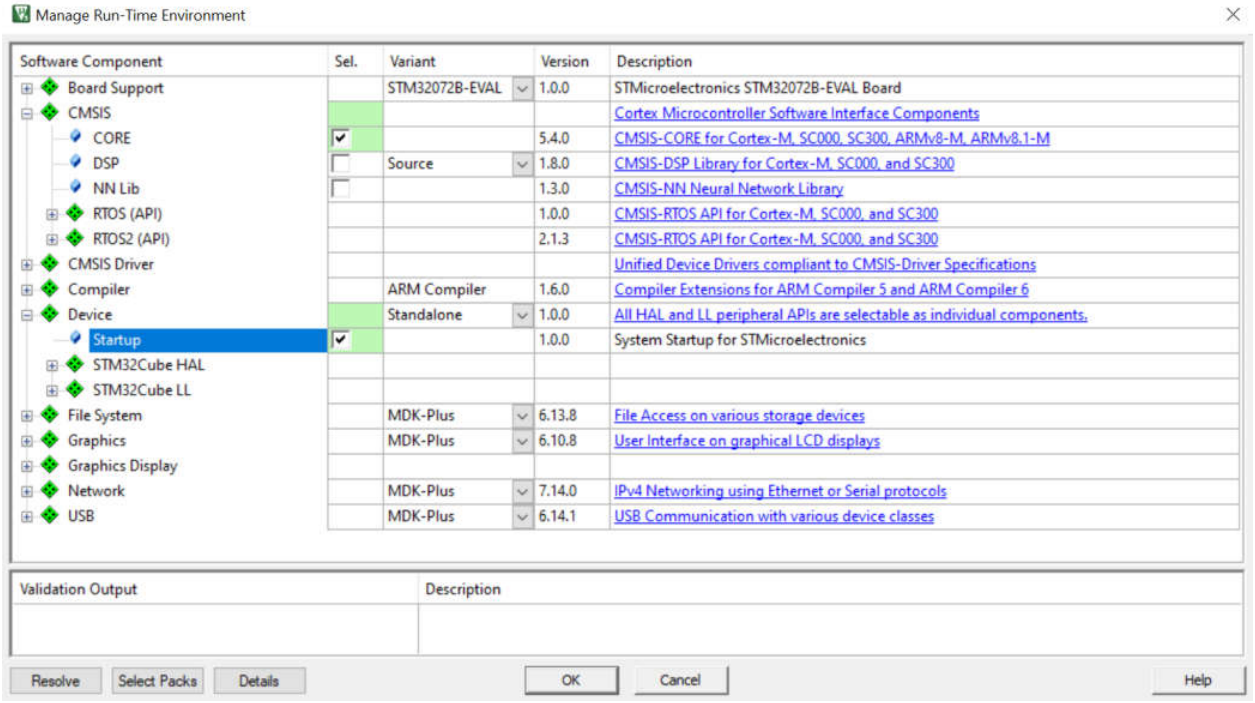


Рис. 18. Настройка среды разработки

Функция main

6. Начальный проект создан. В окне «Project» раскройте пункт «Target 1» и щелкните правой кнопкой мыши по папке «Source Group 1». В появившемся меню выберите пункт «Add new item…».
7. В диалоговом окне «Add New Item…» выберите шаблон «C File (.c)», задайте название файла: **main** и нажмите кнопку «Add».
8. Должно открыться окно текстового редактора для файла «main.c». Если этого не произошло, то в окне «Project» раскройте пункт «Source Group 1» и щелкните по пункту «main.c».

9. В окне «main.c» щелкните правой кнопкой мыши и во всплывающем меню выберите пункт «Insert '#include file' – stm32f0xx.h». Эта команда подключит библиотеку с программной моделью микроконтроллера stm32f0xx к Вашему проекту.

10. Добавьте в редакторе следующий код на языке С:

```
int main(void)
{
}
//и еще одна строка. Без нее среда будет выдавать предупреждение
```

Функция main() является обязательной для запуска на исполнение пользовательского кода. При сбросе микроконтроллера из-за нажатия кнопки «Reset» или после включения питания сначала заработает код из библиотеки «Startup», которую Вы подключили к проекту. Он инициализирует основные регистры микроконтроллера для правильной работы функции Debug и симулятора среды Keil µVision. На финальном этапе работы стартового кода управление микроконтроллером передается функции main().

11. В окне «Project» щелкните правой кнопкой мыши по пункту «Target 1» и выберите во всплывающем меню пункт «Option for Target…». В результате появится диалоговое окно «Option for Target…». Выберите вкладку «Output». На ней поставьте отметку в пункте «Create HEX File». Эти действия направлены на активацию построения средой Keil µVision бинарного файла для загрузки на микроконтроллер.

12. Выполните построение проекта, выбрав пункт меня «Project – Build Target» или нажав кнопку «F7». В нижней части экрана в окне «Build Output» можно увидеть сообщения компилятора и компоновщика рабочей среды. Если ошибки отсутствуют – их количество равно нулю, то в папке «Task1_1\Objects» Вы найдете файл с именем проекта и расширением «hex». Его нужно загрузить на микроконтроллер.

<u>Подключение стенда к ПК и его подготовка для загрузки исполняемого кода</u>

13. С помощью кабеля USB подключите лабораторный стенд к ПК. На стенде рядом с USB-разъемом должен загореться светодиод.

14. Со стороны ПК соединение со стендом STM_01 отображается на виртуальный COM-порт. Используя «Диспетчер устройств» определите имя COM-порта, которое появляется в списке устройств с подключением стенда. Сведения об устройстве должны содержать: «…USB-Serial … CH340…».

15. На стенде найдите штырьковый разъем с красной колодкой (рис. 19) и аккуратно установите на него перемычку.

16. Найдите на стенде, нажмите до щелчка и отпустите кнопку Reset (рис. 19). Микроконтроллер готов к загрузке файла с исполняемым кодом.

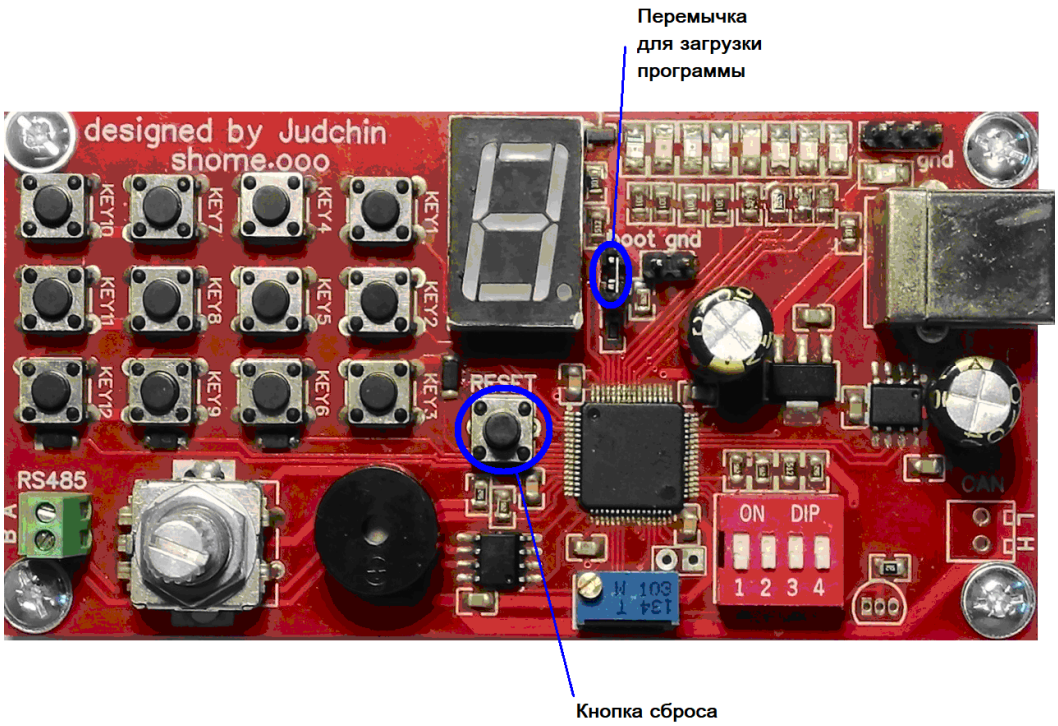


Рис. 19. Ключевые элементы стенда STM_01 для загрузки программы

Загрузка исполняемого кода на микроконтроллер

17. Запустите приложение «Demonstrator GUI» и с помощью выпадающего списка «Port Name» укажите имя COM-порта, к которому подключен лабораторный стенд. Нажмите кнопку «Next».
18. Если на следующей странице диалогового окна появиться светофор, то связь со стендом STM_01 установлена и можно нажать кнопку «Next». При сообщении об ошибке нужно отключить стенд от ПК и повторить процесс инициализации соединения, начиная с пункта 12.
19. На третьей странице диалога «Demonstrator GUI» должны быть представлены сведения: «Target» (тип целевого микроконтроллера заполняется автоматически), «PID», «BID», «Version» и «Flash mapping». На этой странице ничего изменять не нужно. Нажмите кнопку «Next».
20. На четвертой странице диалога «Demonstrator GUI» нажмите радиокнопку «Download to device». В панели «Download from file» укажите путь к файлу с расширением «hex» и выберите опцию «Erase necessary pages» (рис. 20). Нажмите кнопку «Next».
21. Пятая страница диалога содержит сведения о микроконтроллере, загружаемом файле и отображает индикатор загрузки. По завершении загрузки нажмите кнопку «Close».
22. На стенде STM_01 удалите перемычку, установленную в пункте 14 (рис. 19) и нажмите кнопку «Reset». После перезагрузки микроконтроллера заработает код функции **main**, который был написан Вами в проекте Keil µVision.

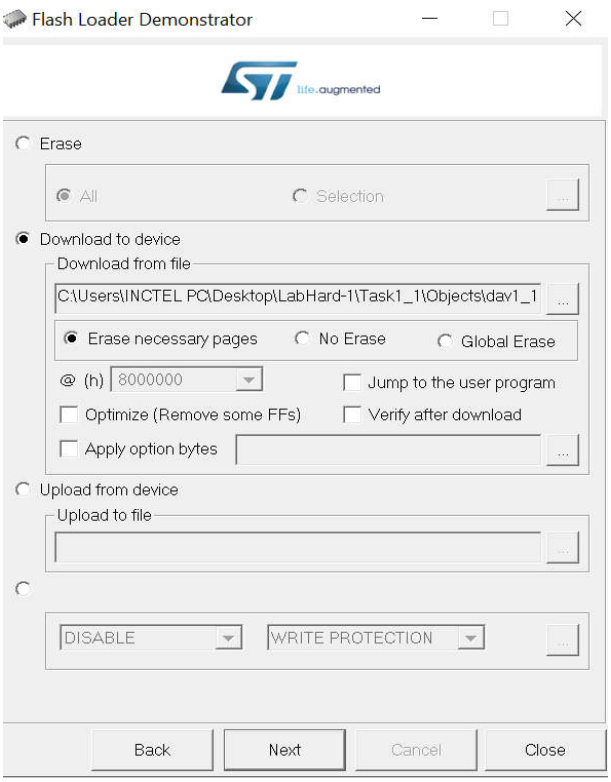


Рис. 20. Выбор расположения hex-файла в приложении «Demonstrator GUI»

Замечание. Если тело main() отсутствует, то микроконтроллер никаких пользовательских инструкций (загруженных на микроконтроллер подпрограмм и функций) выполнять не будет.

Включение светодиода на лабораторном стенде

23. Управление светодиодами стенда STM_01 выполняется с помощью линий микроконтроллера, относящихся к порту B. Поэтому первый шаг, который следует сделать, включить тактирование порта B. Добиться этого можно, если в 18 бит регистра **RCC_AHBENR** поместить значение «1». Добавьте в main() строчку кода:

    RCC->AHBENR | = RCC_AHBENR_GPIOBEN; //Включение тактирования порта B

    Константа *RCC_AHBENR_GPIOBEN* определена в файле 'stm32f072xb.h' макрокомандой #define. Чтобы увидеть соответствующую строчку кода в файле 'stm32f072xb.h' щелкните правой кнопкой мыши в main() над константой и во всплывающем меню выберите пункт «Go To Definition of 'RCC_AHBENR_GPIOBEN'». Действие соответствующей макрокоманды #define эквивалентно коду на языке C: *RCC_AHBENR_GPIOBEN =0x00040000;* Константа *RCC_AHBENR_GPIOBEN* содержит единичный бит только в 18 разряде. Поэтому включение тактирования порта B можно выполнить альтернативной командой:

    RCC->AHBENR | = 0x00040000; //Включение тактирования порта B (альтернатива)

Указанные выше команды изменяют только один бит в управляющем регистре. Они устанавливают 18 бит в «1» с помощью операции «ИЛИ», но не затрагивают значения других бит, оставляя их прежними.

24. Включим светодиод, соответствующий линии PB.0 (рис. 15). Чтобы добиться этого нужно выводы 0 и 8 порта B перевести в режим работы «на выход» («Output») и установить их в состояние логической «1».

25. Для включение режима «Output» 0 и 8 линий порта B требуется в регистре **GPIOB_MODER** битам 0-1 и 16-17 присвоить значение 01_2. Чтобы реализовать это действие добавьте в конец тела функции main следующую строку:

    GPIOB->MODER |= GPIO_MODER_MODER0_0 | GPIO_MODER_MODER8_0;

    эквивалент команды без использования констант:

    GPIOB->MODER |=0x00010001; //Установить «1» в 0 и 16 битах регистра

26. Чтобы подать логическую «1» на выводы 0 и 8 добавьте в конец тела функции main следующую строку:

    GPIOB->ODR |=0x101; //Установить «1» в 0 и 8 битах регистра

27. Нажмите кнопку «F7», чтобы построить hex-файл обновленного проекта.

28. Загрузите на микроконтроллер обновленный hex-файл, выполнив действия пунктов 17 – 22. После удаления перемычки и перезагрузки микроконтроллера на стенде должен загореться светодиод, подключенный к выводу PB.0.

Добавление заголовочного файла и реализация функция задержки

29. Добавим в проект заголовочный файл с расширением «.h». Для этого в окне «Project» щелкните правой кнопкой мыши на пункте «Target 1 – Source Group 1» и во всплывающем меню выберите пункт «Add New Item to Group…». В диалоговом окне левой кнопкой мыши щелкните на пункте списка «Header File (.h)», в поле «Name» укажите: «main», и нажмите кнопку «Add».

30. Откройте окно редактора «main.h». Из файла «main.c» вырежьте строку кода:

    #include "stm32f0xx.h"

    и вставьте ее в начало файла «main.h».

31. Задекларируйте в файле «main.h» функцию задержки **delay**. Для этого поместите в конец файла строку кода:

    void delay(uint32_t);

32. В начале файле «main.c» добавьте строку кода:

    #include "main.h"

33. В конец файла «main.c» добавьте код реализации **delay()**:

```
void delay(uint32_t count)
{
    for (uint32_t i=0;i<count;i++);
}
```

34. Измените функцию **main** следующим образом:

```
int main(void)
{
    uint32_t half_period;

    RCC->AHBENR|=RCC_AHBENR_GPIOBEN;
    GPIOB->MODER|=GPIO_MODER_MODER0_0 | GPIO_MODER_MODER8_0;
    GPIOB->ODR|=0x100;
    half_period = 50000;
    while(1)
    {
      GPIOB->BSRR=0x1;
      delay(half_period);
      GPIOB->BSRR=0x10000;
      delay(half_period);
    }
}
```

35. Нажмите кнопку «F7», чтобы построить hex-файл обновленного проекта.

36. Загрузите на микроконтроллер новый hex-файл. После удаления перемычки и перезагрузки микроконтроллера светодиод, подключенный к выводу PB.0, должен мигать.

37. Разберитесь самостоятельно каким образом в цикле **while** выполняется выключение и включение светодиода. Как в программе можно изменять частоту мигания светодиода?

Регулировка частоты мигания светодиода с помощью микропереключателей

38. Используем микропереключатели SW4 и SW3 стенда STM_01 для управления величиной периода мигания светодиода. С помощью 4-х различных комбинаций микропереключателей станем задавать степень коэффициента задержки между сменами состояния светодиода («горит» или «не горит»). Коэффициент задержки будем рассчитывать по формуле: $K=2^n$, а длительность задержки между изменениями состояния светодиода по формуле: T=K*half_period.

39. Микропереключатели SW4 и SW3 присоединены соответственно к выводам 12 и 13 порта B. Для считывания состояния микропереключателей настройте PB.12 и PB.13 на режим работы «Input», добавив в main() выделенную строку кода:

```
…
GPIOB->MODER|=GPIO_MODER_MODER0_0 | GPIO_MODER_MODER8_0;
GPIOB->MODER&=~( GPIO_MODER_MODER12 | GPIO_MODER_MODER13);
…
```

Эта строка сбрасывает в ноль биты регистра GPIOB_MODER, отвечающие за выбор режима работы линий PB.12 и PB.13 (см. рис. 4).

40. Добавьте в main() определение переменной, хранящей степень коэффициента задержки, с помощью выделенной строки кода:

```
…
uint32_t half_period;
uint32_t n;
…
```

41. Модифицируйте цикл while в main() следующим образом:

```
…
while(1)
{
    n=((GPIOB->IDR)&0x3000)>>12;
    GPIOB->BSRR=0x1;
    delay(half_period<<n);
    GPIOB->BSRR=0x10000;
    delay(half_period<<n);
}
…
```

42. В первой выделенной строке п.41 с помощью GPIOB->IDR выполняется чтение входных сигналов на выводах порта B, работающих в режиме «Input». Конструкция (GPIOB->IDR)&0x3000 позволяет оставить в качестве значащих бит только те, которые соответствуют PB.12 и PB.13, а остальные биты принятого числа обнулить. После наложения маски производится сдвиг вправо на 12 разрядов. В результате переменная **n** сможет принимать только одно из значений множества: {0, 1, 2, 3}. Таким образом, микропереключатели SW3 и SW4 задают степень коэффициента задержки.

43. Расчет длительности задержки между переключениями светодиода выполняется во второй и третьей выделенных строках п. 41 с помощью конструкции: **half_period<<n**. Эта операция осуществляет сдвиг целого значения базовой задержки на n бит влево. Сдвиг на 1 бит влево эквивалентен умножению целого числа на 2. Таким образом, сдвиг на 0, 1, 2 или 3 бита эквивалентен умножению значения **half_period** на 1, 2, 4 или 8 соответственно.

44. Нажмите кнопку «F7», чтобы построить hex-файл обновленного проекта.

45. Загрузите на микроконтроллер новый hex-файл. После удаления перемычки и перезагрузки микроконтроллера частота мигания светодиода должна зависеть от положения микропереключателей SW3 и SW4. Протестируйте работоспособность программы на стенде STM_01, изменяя положение этих микропереключателей.

46. Добавьте комментарии к программному коду в соответствие с требованиями, которые описаны в **разделе 4**. Сохраните проект для последующей демонстрации его работоспособности преподавателю.

3.2. Самостоятельная работа

1. В проекте Task1_2 реализовать два режима работы светодиодов:
   - В первом режиме сначала светодиоды последовательно загораются слева направо, а потом последовательно слева направо гаснут;
   - Во втором режиме слева направо «бежит» один огонек.

   Для выбора режима работы используйте микропереключатель SW1.
2. Изучить назначение регистра GPIOx_OSPEEDR и установить частоту переключения выводов PB.0 – PB.7 на уровне 10 МГц.
3. С помощью микропереключателя SW2 организуйте реверсные режимы работы светодиодов, описанные в пункте 1.
4. Используйте микропереключатели SW3 и SW4 для управлением временем задержки между изменением состояний светодиодов.

3.3. Индивидуальное задание

1. В файле «Назначение вариантов (группа).pdf» определить номер Вашего вариант для выполнения индивидуального задания.
2. В **Приложении 1** находятся варианты индивидуальных заданий.
3. В соответствие с вариантом задания разработать проект Task1_3, в котором программа микроконтроллера должна выполнять следующие действия:
   - На основе константных и варьируемых бит (значения варьируемых бит задается положением микропереключателей SW1 – SW4) определять значение числа в десятичной системе счисления;
   - Преобразовывать число из десятичной системы счисления в систему с основанием, которое определено вариантом задания;
   - Кодировать каждую цифру числа символом для семисегментного индикатора;
   - Выводить все цифры числа на семисегментном индикаторе, начиная со старшей цифры. Каждая цифра должна отображаться 2 секунды. Перед выводом следующей цифры индикатор должен гаснуть на 0.5 секунды. По завершении вывода всех цифр числа на индикаторе в течении 5 секунд должен гореть символ «точка», а затем вывод числа должен начаться заново.
4. Для преобразования из двоичной в десятичную систему счисления использовать формулу:

$$N_{10} = \sum_i a_i \cdot 2^i ,$$

   где N_{10} - число в десятичной системе счисления; a_i - значение бита в i-ом разряде двоичного числа; i изменяется от 0 до 7.
5. Для преобразования целого числа N из десятеричной системы счисления в систему с основание B использовать следующий алгоритм:
   а). Положить: $i = 0$, $Z_i = N$.

b). Определить остаток от деления Z_i на B. Это будет цифра нулевого разряда d_0 числа в системе счисления с основанием B;

c). Увеличить i на 1;

d). Определить целую часть Z_i при делении Z_{i-1} на B. Если $Z_i < B$, то $d_i = Z_i$, где d_i - цифра i-ого разряда числа в системе счисления с основанием B, а преобразование является завершенным. Если $Z_i \geq B$, то остаток от деления Z_i на B даст i-ый разряд числа в системе счисления с основанием B (d_i), и далее следует перейти к пункту c).

Пример: 159_{10} перевести в систему счисления с основание 7, использующей цифры: 0, 1, 2, 3, 4, 5, 6. Шаги алгоритма: d_0=остаток(159/7)=5; Z_1 = целая часть(159/7)=22; $Z_1 > 7$, поэтому d_1 = остаток(22/7)=1; Z_2 = целая часть(22/7)=3; $Z_2 < 7$, поэтому $d_2 = Z_2$ и преобразование завершено. Результат: $159_{10}=315_7$. Проверка: $3 \cdot 7^2 + 1 \cdot 7^1 + 5 \cdot 7^0 = 159_{10}$

6. Для преобразования цифр числа в системе счисления с основанием B к символам семисегментного индикатора использовать таблицу кодировок, приведенную в **Приложении 2**.

4. Подготовка отчета, представление и оценка работы

Комментарии в программном коде в качестве «Отчета о выполнении задания»

1. Комментарии в программном коде каждого проекта выступают в роли отчета и оцениваются в процессе защиты работы.
2. Структура комментариев, описывающих программный код должна быть следующей:
   - В начале файла «main.c» должны присутствовать сведения о разработчике (ФИО - группа), цели выполнения работы и задачах, которые решались в ходе создания программы;
   - Перед реализацией каждой функции следует указать ее назначение и привести список входных/выходных параметров с описанием;
   - Внутри тела функции следует выделить отдельные блоки, реализующие единую комплексную операцию обработки данных, и вынести в их заголовок сведения о назначении блока;
   - Каждая элементарная операция в строке кода должна быть снабжена комментарием;
   - В конце файла «main.c» нужно привести комментарии, содержащие краткое руководство использования программы на стенде STM_01.

   На рисунке 21 приведен пример оформления комментариев к программе.
3. В заголовочных файлах (с расширением «h»), созданных разработчиком проекта, следует прокомментировать все строки кода. Однако можно опустить заголовочный и финальный комментарии со сведениями о проекте и руководстве пользователя.

```
//-----------------------------------------------------------------------------------------------------------------------------------------------------------------
//Разработчик проекта: Долматов Алексей Викторович - ИЦЭ
//
//Цель: продемонстрировать студентам основы программирования в среде Keil uVision портов ввода/вывода общего назначения у микроконтроллеров с архитектурой ARM7
//
//Решаемые проектом задачи:
//    1. Показать основы разработки проекта в среде Keil uVision, структуру программы на языке С и способ загрузки исполяемого файла на микроконтроллер стенда STM_01
//    2. Продемонстрировать элементы настройки портов GPIO для управления внешними электрическими цепями микроконтроллера и мониторинга входных сигналов
//    3. Разработать программу управления частотой мигания светодиода с помощью микропереключателей стенда STM_01
//-----------------------------------------------------------------------------------------------------------------------------------------------------------------

#include "main.h"   //Заголовочный файл с описанием подключаемых библиотечных модулей

//main обязательная функция для исполнения кода пользователя
//После подачи питания или щелчка кнопкой "reset" стартует Загрузчик, которые выполняет начальную настройку основных регистров микроконтроллера
//В завершение работы Загрузчик передает управление микроконтроллером функции main
int main(void)
{
  uint32_t half_period;                                     //Половина периода мигания светодиода
  uint32_t n;                                               //Степень коэффициента базовой задержки: K=2^n
  //Конфигурирование портов GPIO
  RCC->AHBENR|=RCC_AHBENR_GPIOBEN;                          //Включение тактирования порта B: RCC_AHBENR_GPIOBEN=0x00040000
  GPIOB->MODER|=GPIO_MODER_MODER0_0 | GPIO_MODER_MODER8_0;  //Переключение линий 0 и 8 порта B в режим "Output": GPIO_MODER_MODER0_0=0x1; GPIO_MODER_MODER8_0=0x10000
  GPIOB->MODER&=~(GPIO_MODER_MODER12 | GPIO_MODER_MODER13); //Переключение линий 12(SW4) и 13(SW3) порта B в режим "Input":
  GPIOB->ODR|=0x100;                                        //Разрешение работы светодиодов на стенде STM_01 с помощью установки логической "1" на выводе PB.8
  half_period = 50000;                                      //Задание величины базовой задержки
  //Реализация процесса мигания светодиода
  while(1)
  {
    n=((GPIOB->IDR)&0x3000)>>12;                            //Определение степени коэффициента базовой задержки: K=2^n
    GPIOB->BSRR=0x1;                                        //Зажечь светодиод, подключенный к выводу PB.0
    delay(half_period<<n);                                  //Задержка после  включения светодиода
    GPIOB->BSRR=0x10000;                                    //Погасить светодиод, подключенный к выводу PB.0
    delay(half_period<<n);                                  //Задержка после выключения светодиода
  }
}

//Функция задержки: count - количество элементарных периодов задержки с длительностью примерно 2.5 мкс
void delay(uint32_t count)
{
  for (uint32_t i=0;i<count;i++);                           //Выполнение пустых циклов для реализации программной задержки
}
//-----------------------------------------------------------------------------------------------------------------------------------------------------------------
//Руководство пользователя:
//    1. Для запуска загруженной программы удалите перемычку "boot" на стенде и нажмите кнопку "reset"
//    2. В управлении частотой мигания светодиода используйте микропереключатели SW3(старший разряд) и SW4(младший разряд);
//    3. Примерная частота мигания: 00 - 4 Гц; 01 - 2 Гц; 10 - 1 Гц; 11 - 0.5 Гц
//    4. Интервал между переключениями светодиода задается с помощью программной задержки
//-----------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Рис. 21. Пример оформления программного кода в качестве отчета о выполнении задания лабораторной работы

Представление и защита работы

4. Для каждой части **Задания** нужно построить отдельный проект Keil µVision, протестировать и отладить его работу на стенде STM_01, снабдить программный код комментариями в соответствие с п. 4.1- 4.3.
5. Представление работы начинается с демонстрации преподавателю работоспособности программного кода проекта на стенде STM_01. Каждый проект (часть задания лабораторной работы) может быть представлена отдельно. В ходе представления преподаватель оценивает функциональность программы и качество написания программного кода. Студент должен быть готов ответить на вопросы преподавателя по использованным в проекте алгоритмам и их программной реализации.
6. После получения оценки за все проекты лабораторной работы следует создать единый архив (со всеми проектами), и загрузить его на сайт Eluniver.
7. По комментариям к проекту преподаватель оценивает качество отчета по каждому из выполненных заданий.
8. После представления всех заданий лабораторной работы студент получает право ее защиты, которая заключается в ответе на 2 контрольных вопроса из обозначенного в **разделе 5** списка. Каждый контрольный вопрос оценивается отдельно.

Структура оценки лабораторной работы

№	Вид оценки	Максимальный балл
1.	Проект «Обучающая часть»	10
2.	Проект «Самостоятельная работа»	15
3.	Проект «Индивидуальное задание»	25
4.	Отчет «Обучающая часть»	5
5.	Отчет «Самостоятельная работа»	5
6.	Отчет «Индивидуальное задание»	10
7.	Контрольный вопрос 1	15
8.	Контрольный вопрос 2	15
	Итого:	**100**

5. Контрольные вопросы

1. Как расшифровывается аббревиатура GPIO?
2. Какова разрядность портов в микроконтроллерах STM32F? Поясните.
3. Как обозначаются выводы GPIO на принципиальной схеме?
4. Каким образом осуществляется доступ к GPIO в программе?
5. Почему по умолчанию тактирование портов GPIO отключено?
6. Какой регистр отвечает за управление тактированием GPIO? Как осуществляется доступ к нему из программы?
7. Как включить тактирование порта C, используя численное значение?
8. Как отключить тактирование порта B, не затронув управления тактированием других GPIO и блоков микроконтроллера?
9. Какие регистры используются для конфигурирования линий порта?
10. Управляющие регистры порта позволяют задавать единую конфигурацию для всех линий порта или отдельно конфигурировать каждую линию?
11. Каково назначение регистра *GPIOx_MODER*?
12. Можно ли с помощью регистра *GPIOx_MODER* управлять подтягивающими резисторами линии порта?
13. Каково назначение регистра *GPIOx_OTYPER*? Поясните.
14. Какие регистры используются для хранения входных и выходных данных порта? Как осуществляется к ним доступ из программы?
15. Каким образом можно управлять обновлением данных в выводных и входных регистрах порта? Приведите пример изменения режима обновления данных.
16. Что произойдет при выполнении команды: *GRIOA->PUPDR=0x01000020;*
17. Что произойдет при выполнении команды: *GPIOB->IDR=0x0040f080;*
18. В какой регистр попадают данные, если вывод порта работает на вход?
19. Изменяться ли данные регистра *GPIOB->IDR*, если в регистр *GPIOB->ODR* записать новое значение?
20. Каково назначение регистра *GPIOx_BSRR*? Поясните.
21. На вывода порта B необходимо установить значение 0xE46A. Как можно выполнить эту операцию с помощью регистра *GPIOx_BSRR*?
22. В чем различие установки выходных данных с помощью регистров *GPIOB->ODR* и *GPIOx_BSRR*?
23. Чем отличаются регистры *GPIOx_BSRR* и *GPIOx_BRR*?
24. Какой из регистров лучше использовать, когда требуется регулярное обновление данных на выхода порта?
25. Каково назначение регистра *GPIOx_AFRL*? Поясните.
26. О каких альтернативных функциях линий GPIO Вам известно?
27. Какие альтернативные функции имеет PB.10?
28. Какая альтернативная функция соответствует AF4 линии PB.8?
29. Для чего предназначен регистр *GPIOx_LCKR*?
30. Можно ли изменить содержимое регистра *GPIOB->ODR*, если включена блокировка конфигурирования всех линий порта B? Поясните.

Приложение 1. Варианты индивидуальных заданий

№ варианта	Биты исходного числа (1 байт)	Основание системы счисления
1.	0 (SW1) 1 1 (SW2) 0 (SW3) (SW4)	4
2.	0 (SW1) 0 (SW2) 1 (SW3) (SW4) 1	12
3.	(SW1) 0 0 (SW2) 1 (SW3) 0 (SW4)	8
4.	(SW1) 1 (SW2) 0 1 (SW3) (SW4) 0	14
5.	1 0 1 (SW1) (SW2) 0 (SW3) (SW4)	5
6.	0 (SW1) 1 0 (SW2) (SW3) (SW4) 1	9
7.	1 1 (SW1) 1 (SW2) (SW3) (SW4) 0	13
8.	0 1 (SW1) (SW2) 1 (SW3) 0 (SW4)	11
9.	(SW1) (SW2) 0 (SW3) 1 (SW4) 0 1	4
10.	1 0 (SW1) (SW2) (SW3) 0 1 (SW4)	15
11.	1 1 (SW1) 1 (SW2) 1 (SW3) (SW4)	7
12.	(SW1) 0 (SW2) (SW3) 1 (SW4) 1 1	3
13.	0 0 1 (SW1) (SW2) (SW3) 1 (SW4)	6
14.	1 0 1 (SW1) (SW2) 0 (SW3) (SW4)	16
15.	0 0 (SW1) 1 (SW2) (SW3) 0 (SW4)	5
16.	(SW1) (SW2) 0 (SW3) 1 0 (SW4) 1	13
17.	0 (SW1) (SW2) 1 1 0 (SW3) (SW4)	9
18.	1 (SW1) 0 (SW2) 1 (SW3) 0 (SW4)	4
19.	(SW1) 1 (SW2) 1 (SW3) 0 (SW4) 1	11
20.	(SW1) (SW2) (SW3) 0 1 (SW4) 1 0	8
21.	1 (SW1) 0 (SW2) (SW3) 1 1 (SW4)	15
22.	0 0 1 (SW1) (SW2) (SW3) (SW4) 1	6
23.	(SW1) 0 0 (SW2) (SW3) 0 (SW4) 1	12
24.	1 (SW1) (SW2) 0 0 1 (SW3) (SW4)	7
25.	0 1 (SW1) (SW2) (SW3) 1 1 (SW4)	14
26.	1 1 1 (SW1) 0 (SW2) (SW3) (SW4)	9
27.	0 (SW1) (SW2) 1 (SW3) 0 (SW4) 1	16
28.	0 0 0 (SW1) (SW2) 1 (SW3) (SW4)	3
29.	1 (SW1) (SW2) 0 1 (SW3) 1 (SW4)	11
30.	0 1 0 (SW4) 1 (SW2) (SW3) (SW4)	5

Приложение 2. Кодировка цифр для семисегментного индикатора

Таблица 1. Обозначение цифр

Значение цифры в десятичной системе	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Обозначение цифры	0	1	2	3	4	5	6	7	8	9	A	b	C	d	E	F

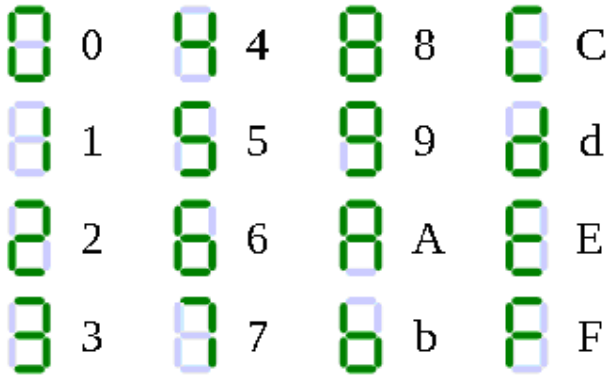


Рис. 22. Отображения цифр семисегментным индикатором

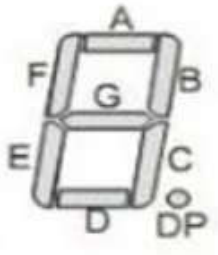


Рис. 23. Обозначение секций индикатора