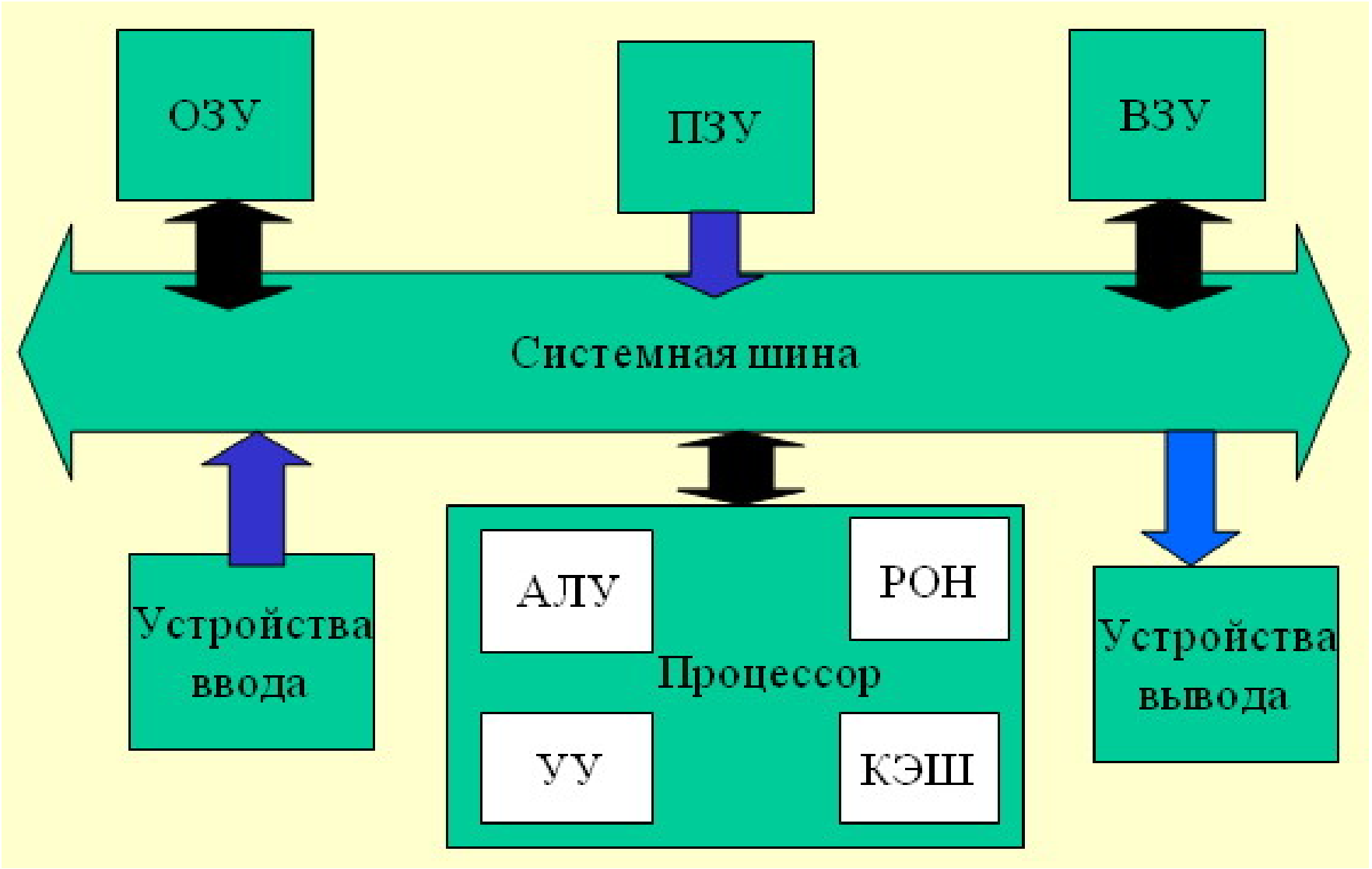
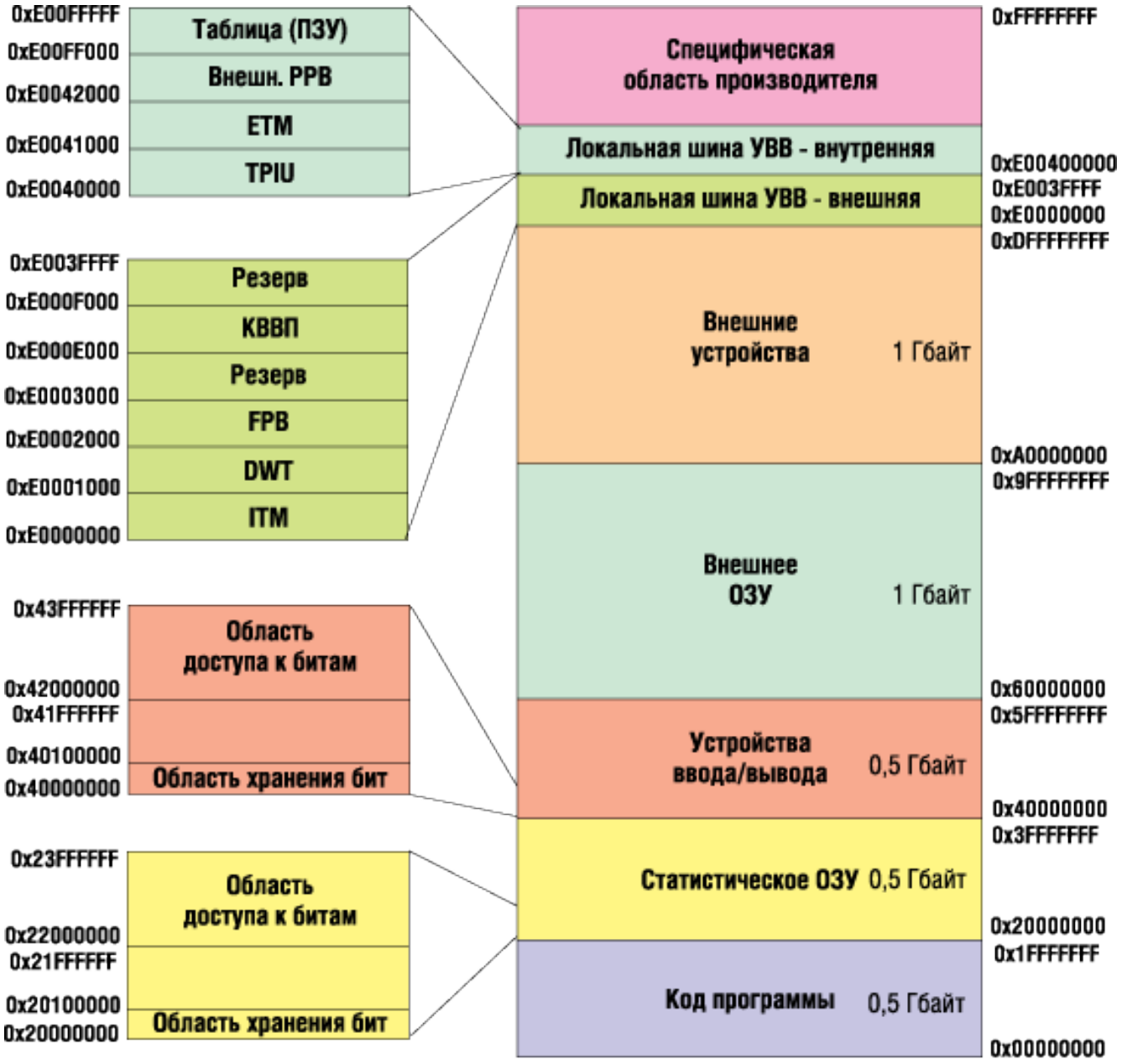
Проектирование устройств на программируемой элементной базе (ПУ на ПЭБ)

Лекция 2

Шинная связь ФМ МК



Адресное пространство (4Гб) ARM МК CM3



Управление любым ФМ

<u>Управление, ввод, вывод осуществляется посредством записи в регистры ФМ</u>

- Регистры можно упрощённо воспринимать как ячейки памяти, расположенные в ФМ, от отдельных битов которых выведены электрические связи, позволяющие:

➢ управлять ФМ,

➢ выводить сигнал на выводы МК

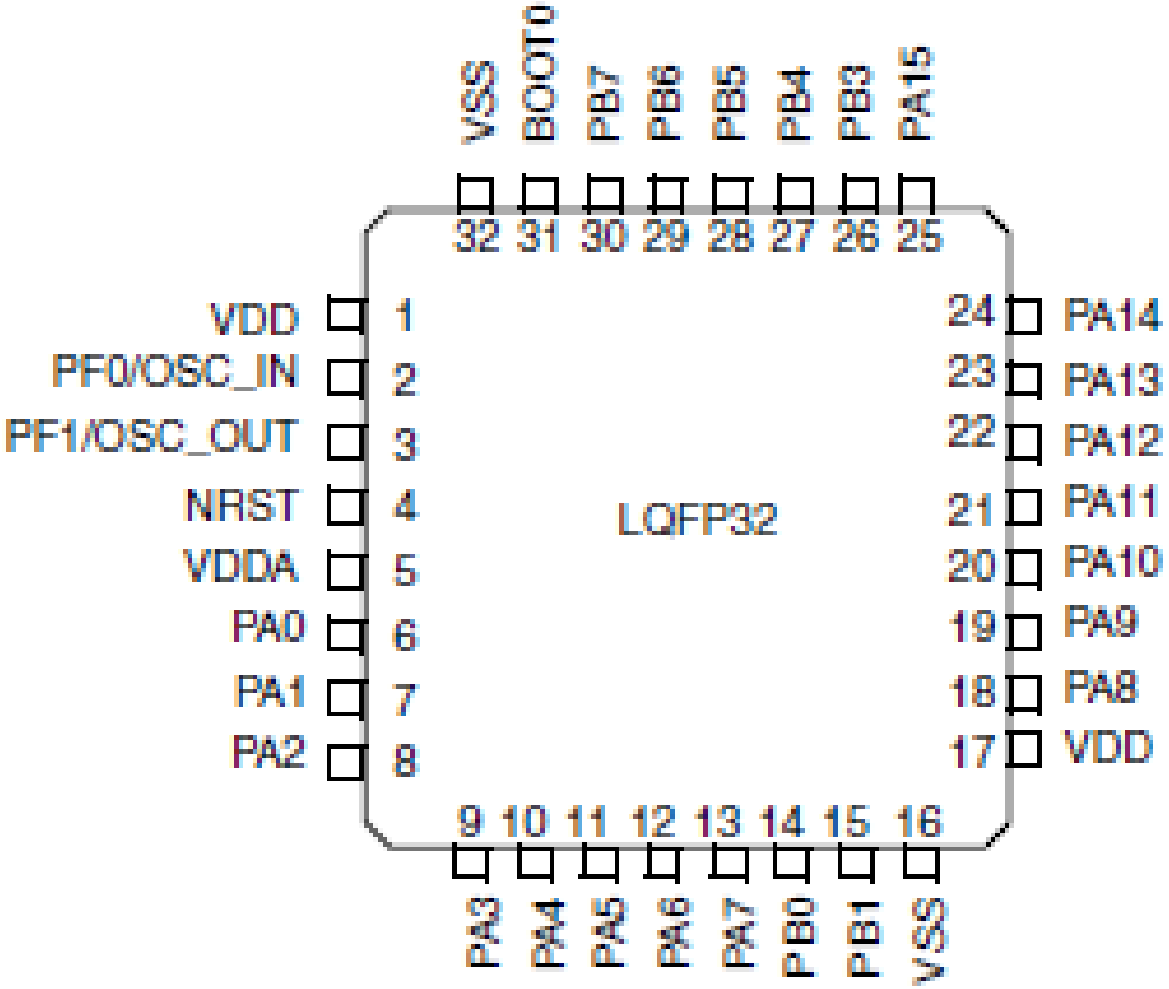
➢ Вводить внешние сигналы с вывода МК в бит регистра

ФМ «ПОРТ ввода вывода»

- PORT, GPIO , GP – модуль МК, позволяющий выводить и вводить логические (дискретные, Не аналоговые) сигналы из внешней среды через выводы МК
- Исторически это первый и поэтому простейший и основной (general) периферийный модуль ввода/вывода
- Несколько (у STM 16 сигналов) сгруппированы в один 16- битный порт
- Таких портов может быть несколько, чем больше выводов, тем больше портов:

PORTA, PORTB …….PORTF, PORTG, ….

Распределение портов в 32 выводном корпусе



Документация на МК:

Stm32f051.pdf- datasheet файл- основные характеристики, корпус, условия эксплуатации

PMxxxxx- progam manual-описания системы команд, проц. Ядра, прерываний

RMxxx-referens manual- описание

Anxxxx-application note- рекомендации по применению отдельных ФМ

Errata- описание ошибок и методов их обхода

Описание структуры МК

- В заголовочном файле xxxxxx.h, подключаемым директивой пред-процессора
- #include stm32f0xx.h
- #include stm32f030x8.h

В среде Кейл имеется описание сотен АРМ МК различных производителей, в том числе и отечественных фирм, например Миландр-К1986ВЕ91….ВЕ93, воронежского НИИЭТ К1921ВК01

Описание группы регистров Порта GPIO как структуры в Си

```
typedef struct
{
__IO uint32_t MODER;    /*!< GPIO port mode register,       Address offset: 0x00 */
__IO uint16_t OTYPER;   /*!< GPIO port output type register,      Address offset: 0x04 */
uint16_t RESERVED0;     /*!< Reserved,                              0x06 */
__IO uint32_t OSPEEDR;  /*!< GPIO port output speed register,      Address offset: 0x08 */
__IO uint32_t PUPDR;     /*!< GPIO port pull-up/pull-down register,   Address offset: 0x0C */
__IO uint16_t IDR;       /*!< GPIO port input data register,              Address offset: 0x10 */
uint16_t RESERVED1;     /*!< Reserved,                                    0x12 */
__IO uint16_t ODR;       /*!< GPIO port output data register,        Address offset: 0x14 */
uint16_t RESERVED2;     /*!< Reserved,                                    0x16 */
__IO uint32_t BSRR;      /*!< GPIO port bit set/reset registerBSRR,       Address offset: 0x18 */
__IO uint32_t LCKR;      /*!< GPIO port configuration lock register,      Address offset: 0x1C */
__IO uint32_t AFR[2];    /*!< GPIO alternate function low register,  Address offset: 0x20-0x24 */
__IO uint16_t BRR;       /*!< GPIO bit reset register,                    Address offset: 0x28 */
  uint16_t RESERVED3;   /*!< Reserved,                                    0x2A */
}  GPIO_TypeDef;
```

Определение портов

```
#define GPIOA     ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB     ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC     ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD     ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE     ((GPIO_TypeDef *) GPIOE_BASE)
#define GPIOF     ((GPIO_TypeDef *) GPIOF_BASE)
```

Определение адресов портов

- #define PERIPH_BASE ((uint32_t)0x40000000) /*!< Peripheral base address */
- ……
- #define GPIOA_BASE (AHB2PERIPH_BASE + 0x00000000)
- #define GPIOB_BASE (AHB2PERIPH_BASE + 0x00000400)
- #define GPIOC_BASE (AHB2PERIPH_BASE + 0x00000800)
- #define GPIOD_BASE (AHB2PERIPH_BASE + 0x00000C00)
- #define GPIOE_BASE (AHB2PERIPH_BASE + 0x00001000)
- #define GPIOF_BASE (AHB2PERIPH_BASE + 0x00001400)

9.4.5 GPIO port input data register (GPIOx_IDR) (x = A..F)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

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

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

Bits 15:0 **IDRy:** Port input data bit (y = 0..15)

These bits are read-only. They contain the input value of the corresponding I/O port.

9.4.6 GPIO port output data register (GPIOx_ODR) (x = A..F)

Address offset: 0x14

Reset value: 0x0000 0000

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

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

Bits 15:0 **ODRy:** Port output data bit (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/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A..F).

Определение констант

Для удобства определения отдельных бит каждого регистра в файле определения структуры МК определены символьные константы для каждого регистра, например для 16 битного регистра ODR для бита 5:

15…12 11…. 8 7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 b

0 0 2 0 hex=0x0020

define GPIO_ODR_5 0x0020

- /***** Bit definition for GPIO_ODR register *********/
- #define GPIO_ODR_0 ((uint32_t)0x00000001U)
- #define GPIO_ODR_1 ((uint32_t)0x00000002U)
- #define GPIO_ODR_2 ((uint32_t)0x00000004U)
- #define GPIO_ODR_3 ((uint32_t)0x00000008U)
- #define GPIO_ODR_4 ((uint32_t)0x00000010U)
- #define GPIO_ODR_5 ((uint32_t)0x00000020U)
- #define GPIO_ODR_6 ((uint32_t)0x00000040U)
- #define GPIO_ODR_7 ((uint32_t)0x00000080U)
- #define GPIO_ODR_8 ((uint32_t)0x00000100U)
- #define GPIO_ODR_9 ((uint32_t)0x00000200U)
- #define GPIO_ODR_10 ((uint32_t)0x00000400U)
- #define GPIO_ODR_11 ((uint32_t)0x00000800U)
- #define GPIO_ODR_12 ((uint32_t)0x00001000U)
- #define GPIO_ODR_13 ((uint32_t)0x00002000U)
- #define GPIO_ODR_14 ((uint32_t)0x00004000U)
- #define GPIO_ODR_15 ((uint32_t)0x00008000U)

- /******** Bit definition for GPIO_IDR register *******/
- #define GPIO_IDR_0 ((uint32_t)0x00000001U)
- #define GPIO_IDR_1 ((uint32_t)0x00000002U)
- #define GPIO_IDR_2 ((uint32_t)0x00000004U)
- #define GPIO_IDR_3 ((uint32_t)0x00000008U)
- #define GPIO_IDR_4 ((uint32_t)0x00000010U)
- #define GPIO_IDR_5 ((uint32_t)0x00000020U)
- #define GPIO_IDR_6 ((uint32_t)0x00000040U)
- #define GPIO_IDR_7 ((uint32_t)0x00000080U)
- #define GPIO_IDR_8 ((uint32_t)0x00000100U)
- #define GPIO_IDR_9 ((uint32_t)0x00000200U)
- #define GPIO_IDR_10 ((uint32_t)0x00000400U)
- #define GPIO_IDR_11 ((uint32_t)0x00000800U)
- #define GPIO_IDR_12 ((uint32_t)0x00001000U)
- #define GPIO_IDR_13 ((uint32_t)0x00002000U)
- #define GPIO_IDR_14 ((uint32_t)0x00004000U)
- #define GPIO_IDR_15 ((uint32_t)0x00008000U)

9.4.1 GPIO port mode register (GPIOx_MODER) (x =A..F)

Address offset:0x00

Reset values:

- 0x2800 0000 for port A
- 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+1:2y **MODERy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O mode.

00: Input mode (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

```
#define GPIO_MODER_MODER0 ((uint32_t)0x00000003U)
#define GPIO_MODER_MODER0_0       ((uint32_t)0x00000001U)
#define GPIO_MODER_MODER0_1       ((uint32_t)0x00000002U)
#define GPIO_MODER_MODER1        ((uint32_t)0x0000000CU)
#define GPIO_MODER_MODER1_0       ((uint32_t)0x00000004U)
#define GPIO_MODER_MODER1_1       ((uint32_t)0x00000008U)
……..
#define GPIO_MODER_MODER15       ((uint32_t)0xC0000000U)
#define GPIO_MODER_MODER15_0      ((uint32_t)0x40000000U)
#define GPIO_MODER_MODER15_1      ((uint32_t)0x80000000U)
```

Запись в регистр порта

PORTA-> MODE= 1; // перевести 0й бит порта А в режим выхода, остальные-входы

PORTA->ODR=0x11;// выставить в 1 0-й и 4-й биты выходного регистра, **остальные в «0»**

Битовые операции

PORTA->ODR=PORTA->ODR | 0x01 /* ИЛИ*/

Выставить в 1 0-й бит, остальные без изменения

Равнозначная запись в СИ:

PORTA->ODR |= 0x01

PORTA->ODR=PORTA->ODR & 0x01 /* И */

Сбросить в 0 все кроме 0-го бита

Равнозначная запись в СИ:

PORTA->ODR |= 0x01

Плохо-Три проц. инструкции: читать+ И/Или+ запись

В ARM МК есть Спец регистры модификации ODR позволяющие исп. одну инструкцию

BSRR – Bit Set & Reset Register- в 1 и/или 0

- Запись 1 в младш 16 бит (0-15) выставляет в 1 соответствующий бит выходного регистра ODR-
- Запись 1 в старшие 16 бит (16-32) сбрасывает в 0 соответствующий бит выходного регистра ODR (0-15)

BRR-Bit Reset Register – только в 0

- Запись 1 в старшие 16 бит (16-32) сбрасывает в 0 соответствующий бит выходного регистра ODR (0-15)

ОСТАЛЬНЫЕ БИТЫ ОСТАЮТСЯ БЕЗ ИЗМЕНЕНИЙ

9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..F)

Address offset: 0x18

Reset value: 0x0000 0000

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

9.4.11 GPIO port bit reset register (GPIOx_BRR) (x =A..F)

Address offset: 0x28
Reset value: 0x0000 0000

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

Bits 31:16 Reserved

Bits 15:0 **BRy:** Port x Reset bit y (y= 0..15)

These bits are write-only. A read to these bits returns the value 0x0000

0: No action on the corresponding ODx bit

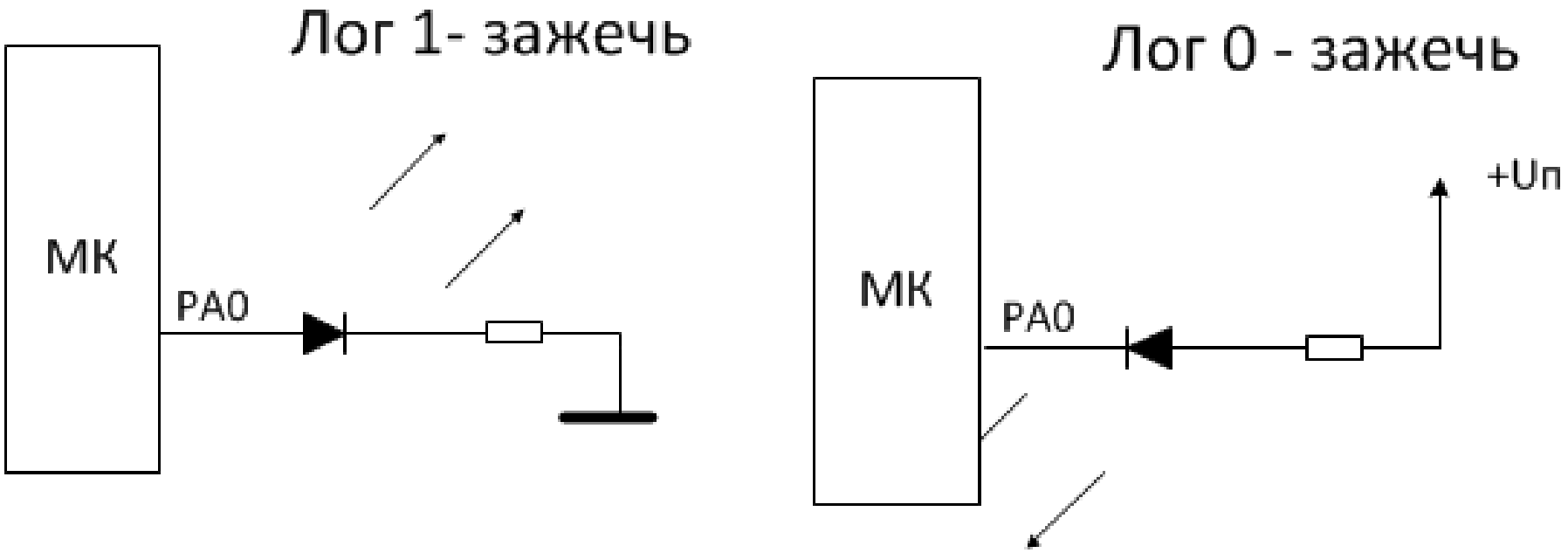
1: Reset the corresponding ODx bit

Применение BSRR и BRR

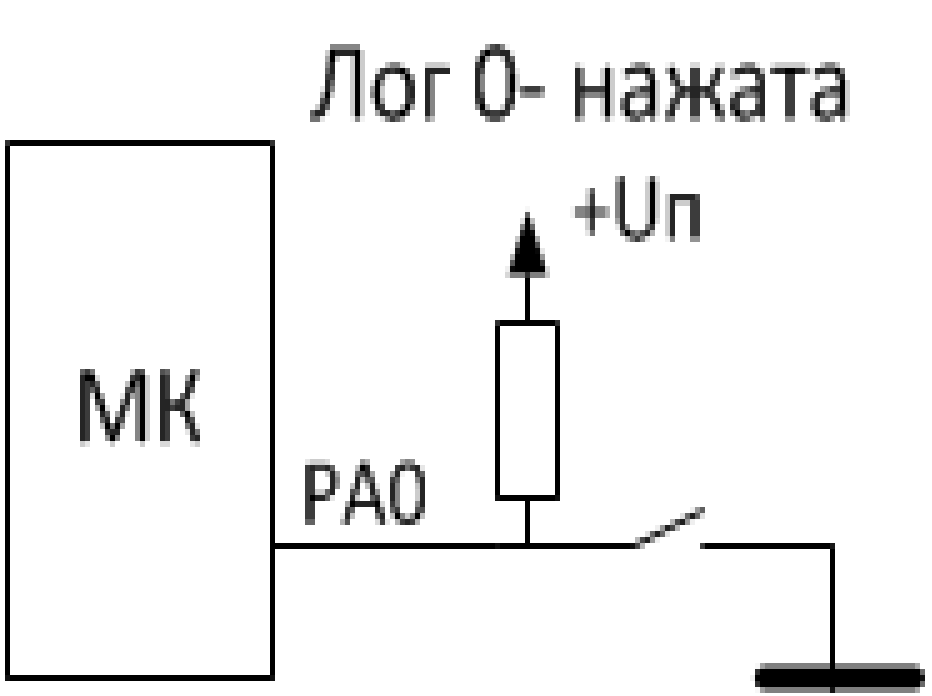
PORTA->BSRR=0x11;// в 1 0-й и 4-й биты, остальные без изменений

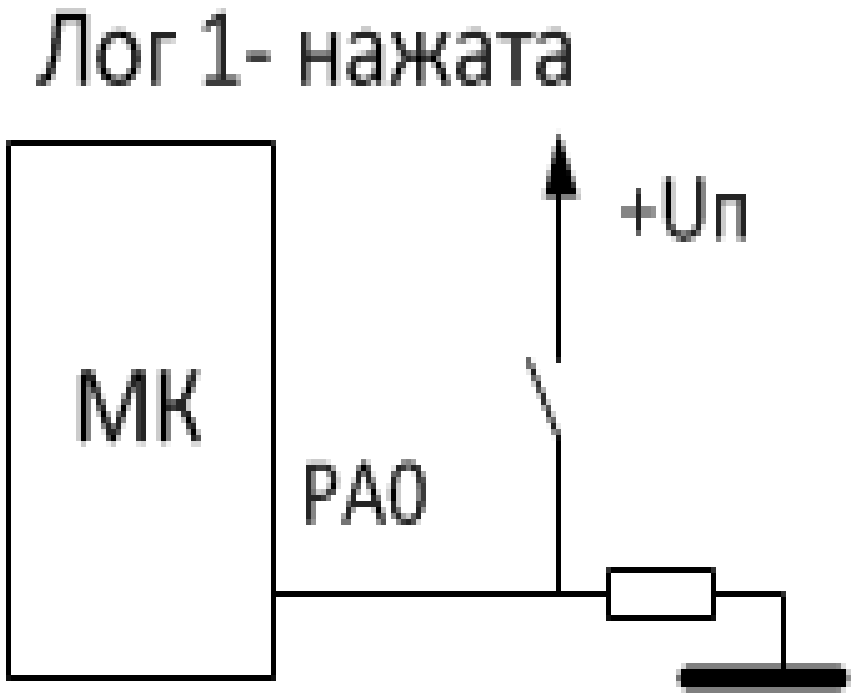
PORTA->BRR=0x11;// в 0 0-й и 4-й биты, остальные без изменений

Учитывать схемотехнику выхода



Чтение входов





Чтение лог входов

A=GPIOA->IDR;// в переменную A все 16 бит

If(GPIO->IDR & 4); //если бит 3 ==1 (1,2,4,8…..)

If(GPIO->IDR & 3);// 3=1+2 если бит 0 или 1 ==1

Описание схемотехники устройства

Определять свою схему через свои #define, которые можно сконцентрировать в отдельном файле, подключаемый директивой include – это даст возможность определить своё железо в одном месте и не менять при необходимости по всему тексту программы.

```
#include  moyaSchema.h
……
If(PORTA->IDR & BUTTON1)……
```

Файл moyaSchema.h:

```
#define  LED1    GPIO_ODR_0  /*бит0*/
#define LED1    1 /* аналог строки выше*/

#defne BUTTON1 GPIO_IDR_1 /*бит 1-вход*/
#defne BUTTON1 GPIO_ODR_1 /*бит 1-вход*/
#defne BUTTON1   2 /*бит 1-вход*/
```

```
#define  BUT2   (PORTA->IDR & GPIO_IDR_2 )
/*бит 2-вход*/

If( BUT2 )
```

Язык Си

- Керниган и Ричи- авторы языка . Их книга-описание языка -классика по языку Си.
- развитие языка: Си++, C# (Си шарп), Visual C++, ….

Указатели и структуры

```
struct time {
int hour;
int minute;
int second;}
struct time watch, *pt ; /* описана структура watch ( типа time)  и указатель на структуру такую же структуру  pt*/
watch.minute=5;
pt=&watch;  /* в pt положить адрес  watch (указатель на watch)*/
pt->hour=1; /* в структуре watch  в часы записать 5 */
```

Арифметические операторы

Оператор	Действие
-	Вычитание, унарный минус
+	Сложение
*	Умножение
/	Деление
%	Деление по модулю
--	Декремент
++	Инкремент

Операторы отношений и логические операторы

>	**Больше**
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно

Логические операторы

&&	**И**
||	ИЛИ
!	НЕ

Побитовые и сдвиговые операторы

&	**И**
|	ИЛИ
^	Исключающее ИЛИ
~	Инверсия
>>	Сдвиг вправо
<<	Сдвиг влево

Операторы присвоения

=, *=, /=, +=, -=, %=, <<=, >>=, &=, ^=, |=

X = X * 25	**X *= 25**
X = X / 25	X /= 25
X = X + 25	X += 25
X = X - 25	X -= 25
X = X % 25	X %= 25
X = X << 25	X <<= 25
X = X >>25	X >>= 25
X = X & 0xFF	X &= 0xFF
X = X ^ 0xFF	X ^= 0xFF
X = X | 0xFF	X |= 0xFF

Среда разработки (IDE)
Integrated Development Environment

- Keil- бесплатно для STM32F0xx (CM0)+ демо до 32кБ загрузочного кода любого МК
- IAR-
- Atollic True Stutio for STM32- free
- Coocox (Eclipse)- free
- Atmel Studo (для Arduino) для ARM МК CM0 от ATMEL(было фирмы ATMEL теперь Mirochip)- free

IDE (среда) Кейл ARM microVision (μν==uv)

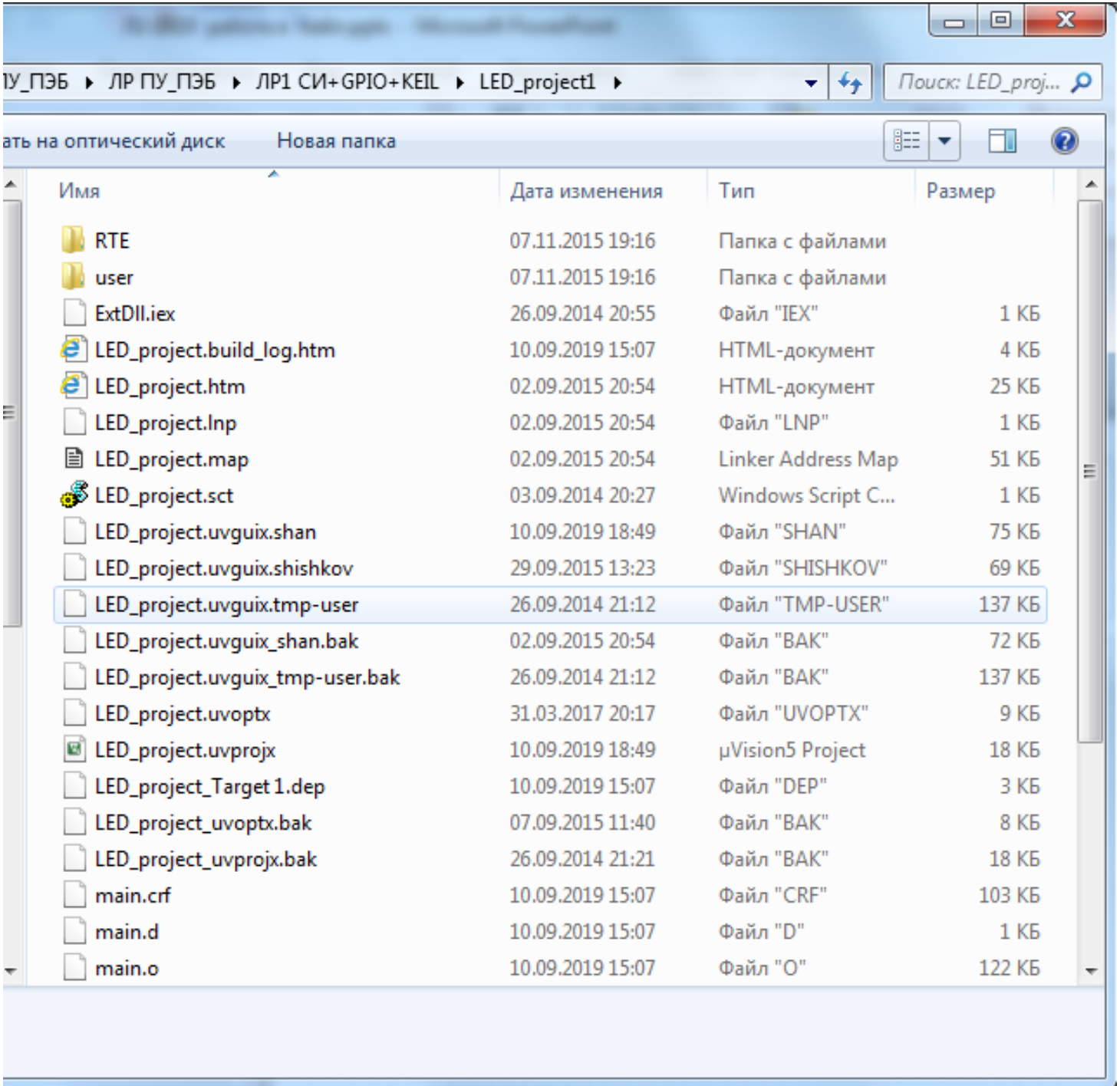
- Папка проекта- содержит все файлы:

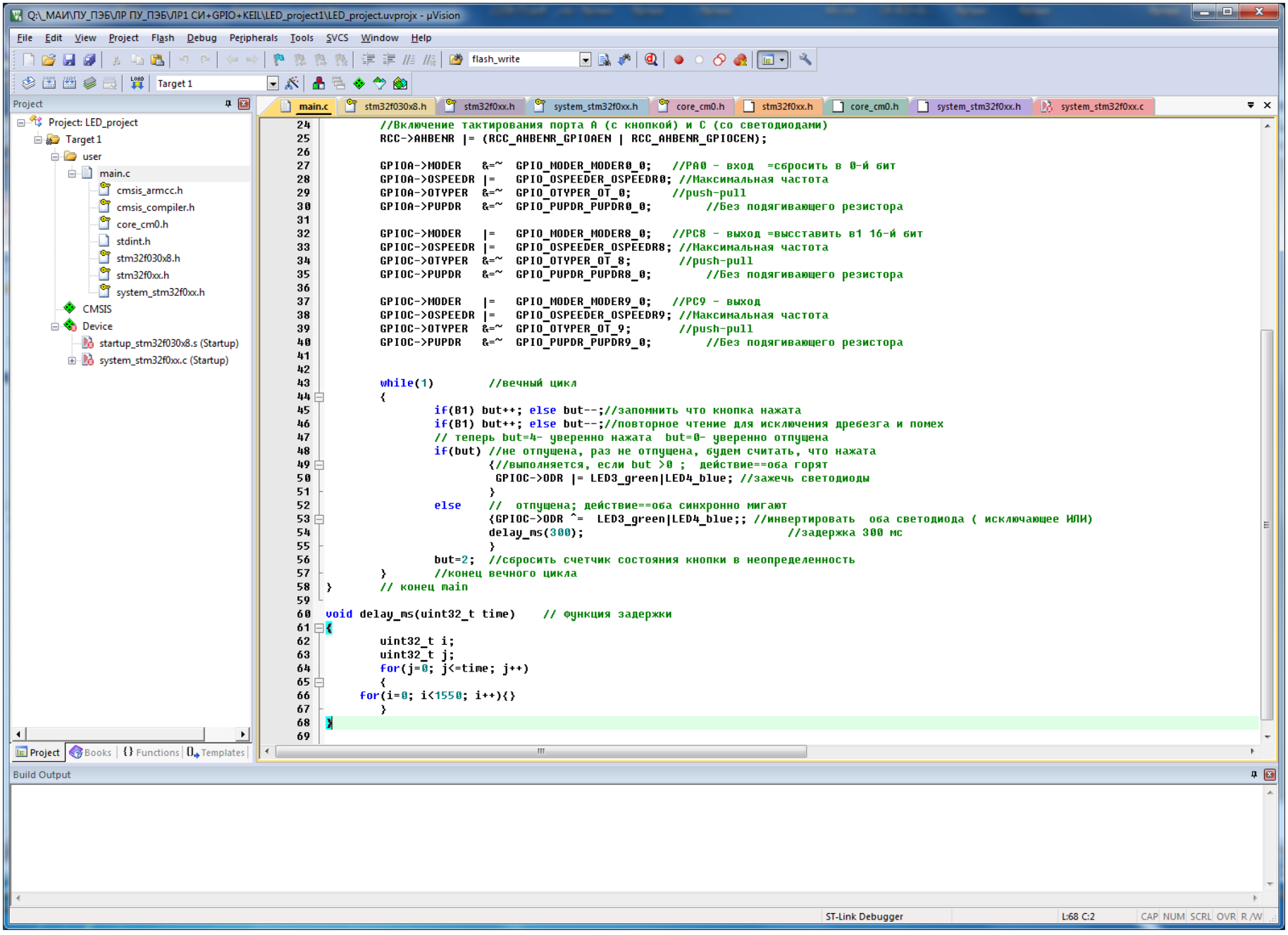
  - исходный текст; .c(си) , .s (асемблер)

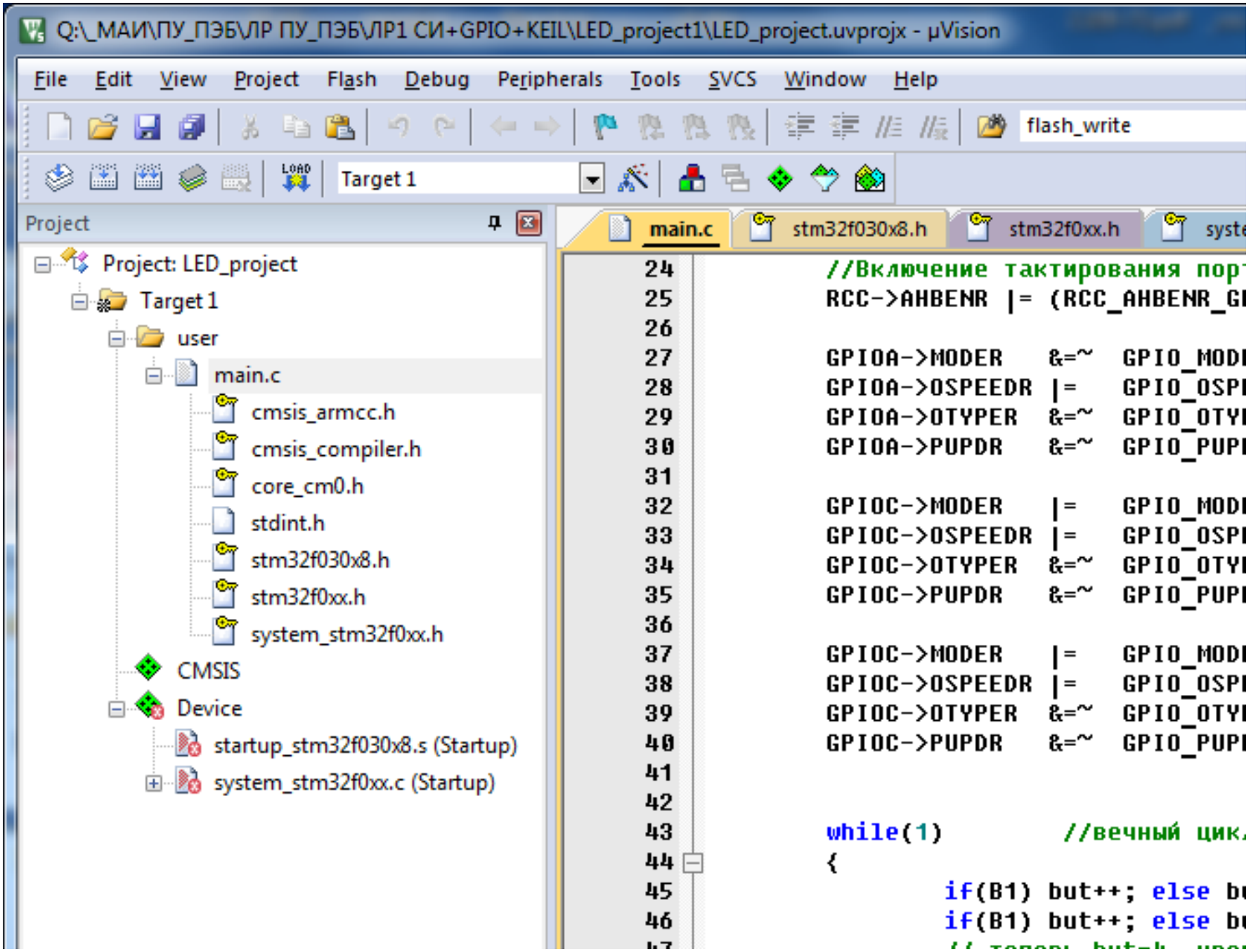
  - оттранслированные файлы (обьектный код) . Obj, ,o

  - загрузочный код (скомпонованный проект это сборка нескольких обьектных+ библиотечные модули)- чаще формат .hex

- Файл проекта имя.uvprojx для Кейл

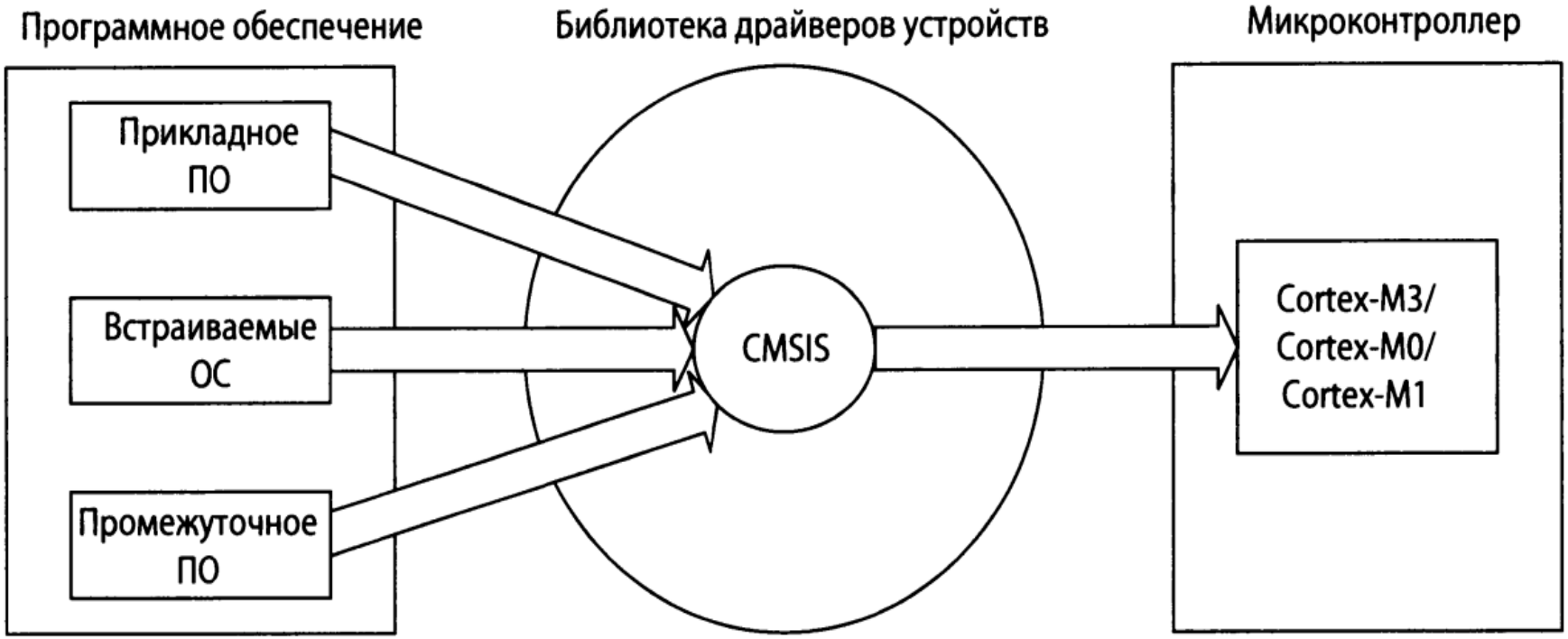

ПУ_ПЭБ ▸ ЛР ПУ_ПЭБ ▸ ЛР1 СИ+GPIO+KEIL ▸ LED_project1 ▸
Поиск: LED_proj...
ать на оптический диск
Новая папка
Имя
Дата изменения
Тип
Размер
RTE 07.11.2015 19:16 Папка с файлами
user 07.11.2015 19:16 Папка с файлами
ExtDll.iex 26.09.2014 20:55 Файл "IEX" 1 КБ
LED_project.build_log.htm 10.09.2019 15:07 HTML-документ 4 КБ
LED_project.htm 02.09.2015 20:54 HTML-документ 25 КБ
LED_project.lnp 02.09.2015 20:54 Файл "LNP" 1 КБ
LED_project.map 02.09.2015 20:54 Linker Address Map 51 КБ
LED_project.sct 03.09.2014 20:27 Windows Script C... 1 КБ
LED_project.uvguix.shan 10.09.2019 18:49 Файл "SHAN" 75 КБ
LED_project.uvguix.shishkov 29.09.2015 13:23 Файл "SHISHKOV" 69 КБ
LED_project.uvguix.tmp-user 26.09.2014 21:12 Файл "TMP-USER" 137 КБ
LED_project.uvguix_shan.bak 02.09.2015 20:54 Файл "BAK" 72 КБ
LED_project.uvguix_tmp-user.bak 26.09.2014 21:12 Файл "BAK" 137 КБ
LED_project.uvoptx 31.03.2017 20:17 Файл "UVOPTX" 9 КБ
LED_project.uvprojx 10.09.2019 18:49 µVision5 Project 18 КБ
LED_project_Target 1.dep 10.09.2019 15:07 Файл "DEP" 3 КБ
LED_project_uvoptx.bak 07.09.2015 11:40 Файл "BAK" 8 КБ
LED_project_uvprojx.bak 26.09.2014 21:21 Файл "BAK" 18 КБ
main.crf 10.09.2019 15:07 Файл "CRF" 103 КБ
main.d 10.09.2019 15:07 Файл "D" 1 КБ
main.o 10.09.2019 15:07 Файл "O" 122 КБ


Q:_МАИ\ПУ_ПЭБ\ЛР ПУ_ПЭБ\ЛР1 СИ+GPIO+KEIL\LED_project1\LED_project.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
flash_write
Target 1
Project
Project: LED_project
Target 1
user
main.c
cmsis_armcc.h
cmsis_compiler.h
core_cm0.h
stdint.h
stm32f030x8.h
stm32f0xx.h
system_stm32f0xx.h
CMSIS
Device
startup_stm32f030x8.s (Startup)
system_stm32f0xx.c (Startup)
main.c
stm32f030x8.h
stm32f0xx.h
system_stm32f0xx.h
core_cm0.h
stm32f0xx.h
core_cm0.h
system_stm32f0xx.h
system_stm32f0xx.c
24 //Включение тактирования порта A (с кнопкой) и C (со светодиодами)
25 RCC->AHBENR |= (RCC_AHBENR_GPIOAEN | RCC_AHBENR_GPIOCEN);
26
27 GPIOA->MODER &=~ GPIO_MODER_MODER0_0; //PA0 - вход =сбросить в 0-й бит
28 GPIOA->OSPEEDR |= GPIO_OSPEEDER_OSPEEDR0; //Максимальная частота
29 GPIOA->OTYPER &=~ GPIO_OTYPER_OT_0; //push-pull
30 GPIOA->PUPDR &=~ GPIO_PUPDR_PUPDR0_0; //Без подягивающего резистора
31
32 GPIOC->MODER |= GPIO_MODER_MODER8_0; //PC8 - выход =выссставить в1 16-й бит
33 GPIOC->OSPEEDR |= GPIO_OSPEEDER_OSPEEDR8; //Максимальная частота
34 GPIOC->OTYPER &=~ GPIO_OTYPER_OT_8; //push-pull
35 GPIOC->PUPDR &=~ GPIO_PUPDR_PUPDR8_0; //Без подягивающего резистора
36
37 GPIOC->MODER |= GPIO_MODER_MODER9_0; //PC9 - выход
38 GPIOC->OSPEEDR |= GPIO_OSPEEDER_OSPEEDR9; //Максимальная частота
39 GPIOC->OTYPER &=~ GPIO_OTYPER_OT_9; //push-pull
40 GPIOC->PUPDR &=~ GPIO_PUPDR_PUPDR9_0; //Без подягивающего резистора
41
42
43 while(1) //вечный цикл
44 {
45 if(B1) but++; else but--;//запомнить что кнопка нажата
46 if(B1) but++; else but--;//повторное чтение для исключения дребезга и помех
47 // теперь but=4- уверенно нажата but=0- уверенно отпущена
48 if(but) //не отпущена, раз не отпущена, будем считать, что нажата
49 {//выполняется, если but >0 ; действие==оба горят
50 GPIOC->ODR |= LED3_green|LED4_blue; //зажечь светодиоды
51 }
52 else // отпущена; действие==оба синхронно мигают
53 {GPIOC->ODR ^= LED3_green|LED4_blue;; //инвертировать оба светодиода (исключающее ИЛИ)
54 delay_ms(300); //задержка 300 мс
55 }
56 but=2; //сбросить счетчик состояния кнопки в неопределенность
57 } //конец вечного цикла
58 } // конец main
59
60 void delay_ms(uint32_t time) // функция задержки
61 {
62 uint32_t i;
63 uint32_t j;
64 for(j=0; j<=time; j++)
65 {
66 for(i=0; i<1550; i++){}
67 }
68 }
69
Project Books Functions Templates
Build Output
ST-Link Debugger
L:68 C:2
CAP NUM SCRL OVR R/W


Q:_МАИ\ПУ_ПЭБ\ЛР ПУ_ПЭБ\ЛР1 СИ+GPIO+KEIL\LED_project1\LED_project.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
flash_write
Target 1
Project
Project: LED_project
Target 1
user
main.c
cmsis_armcc.h
cmsis_compiler.h
core_cm0.h
stdint.h
stm32f030x8.h
stm32f0xx.h
system_stm32f0xx.h
CMSIS
Device
startup_stm32f030x8.s (Startup)
system_stm32f0xx.c (Startup)
main.c
stm32f030x8.h
stm32f0xx.h
24 //Включение тактирования порт
25 RCC->AHBENR |= (RCC_AHBENR_GI
26
27 GPIOA->MODER &=~ GPIO_MODI
28 GPIOA->OSPEEDR |= GPIO_OSPI
29 GPIOA->OTYPER &=~ GPIO_OTYI
30 GPIOA->PUPDR &=~ GPIO_PUPI
31
32 GPIOC->MODER |= GPIO_MODI
33 GPIOC->OSPEEDR |= GPIO_OSPI
34 GPIOC->OTYPER &=~ GPIO_OTYI
35 GPIOC->PUPDR &=~ GPIO_PUPI
36
37 GPIOC->MODER |= GPIO_MODI
38 GPIOC->OSPEEDR |= GPIO_OSPI
39 GPIOC->OTYPER &=~ GPIO_OTYI
40 GPIOC->PUPDR &=~ GPIO_PUPI
41
42
43 while(1) //вечный цик
44 {
45 if(B1) but++; else bu
46 if(B1) but++; else bu

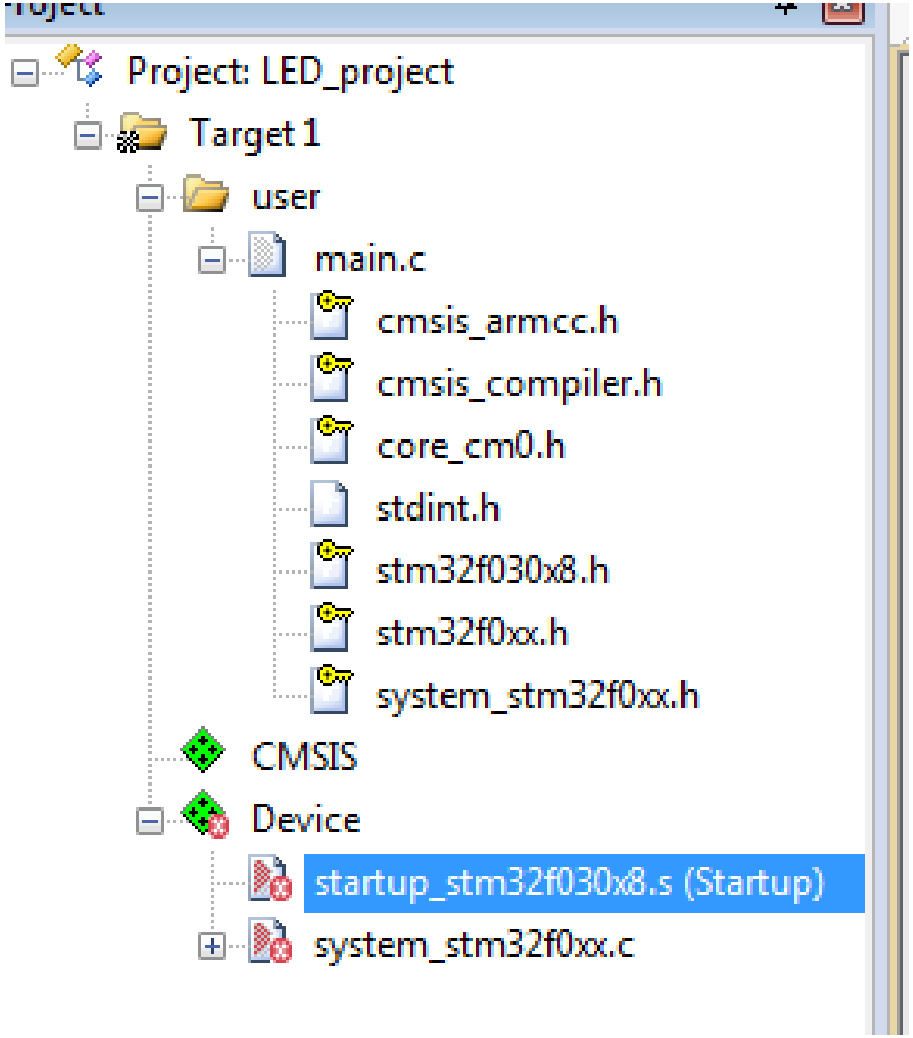
SMSIS , CUBE MX , HAL

CMSIS-Cortex Microcontroller Software Interface Standard библиотека, созданная компанией ARM на языке Си для микроконтроллеров, работающих на ядре Cortex



core_m0.c – описание встроенных функций;
core_m0.h – доступ к периферии ядра;
stm32f0xx.h –заголовочный файл. Здесь объявлены указатели на структуры и сами структуры для программирования регистров периферии и всех модулей микроконтроллера.
system_stm32f0xx.c – описание функций начальной инициализации периферии, в том числе и функция инициализации системы тактирования;
system_stm32f0xx.h - объявление функций начальной инициализации периферии, в том числе и функция инициализации системы тактирования;
startup_stm32f030.s – вектора прерываний;

При определении МК



При определении МК Кейл автоматом вставляет 2 стартовых файла в Подпапку RTE->Device-> папка МК и добавляет их в проект

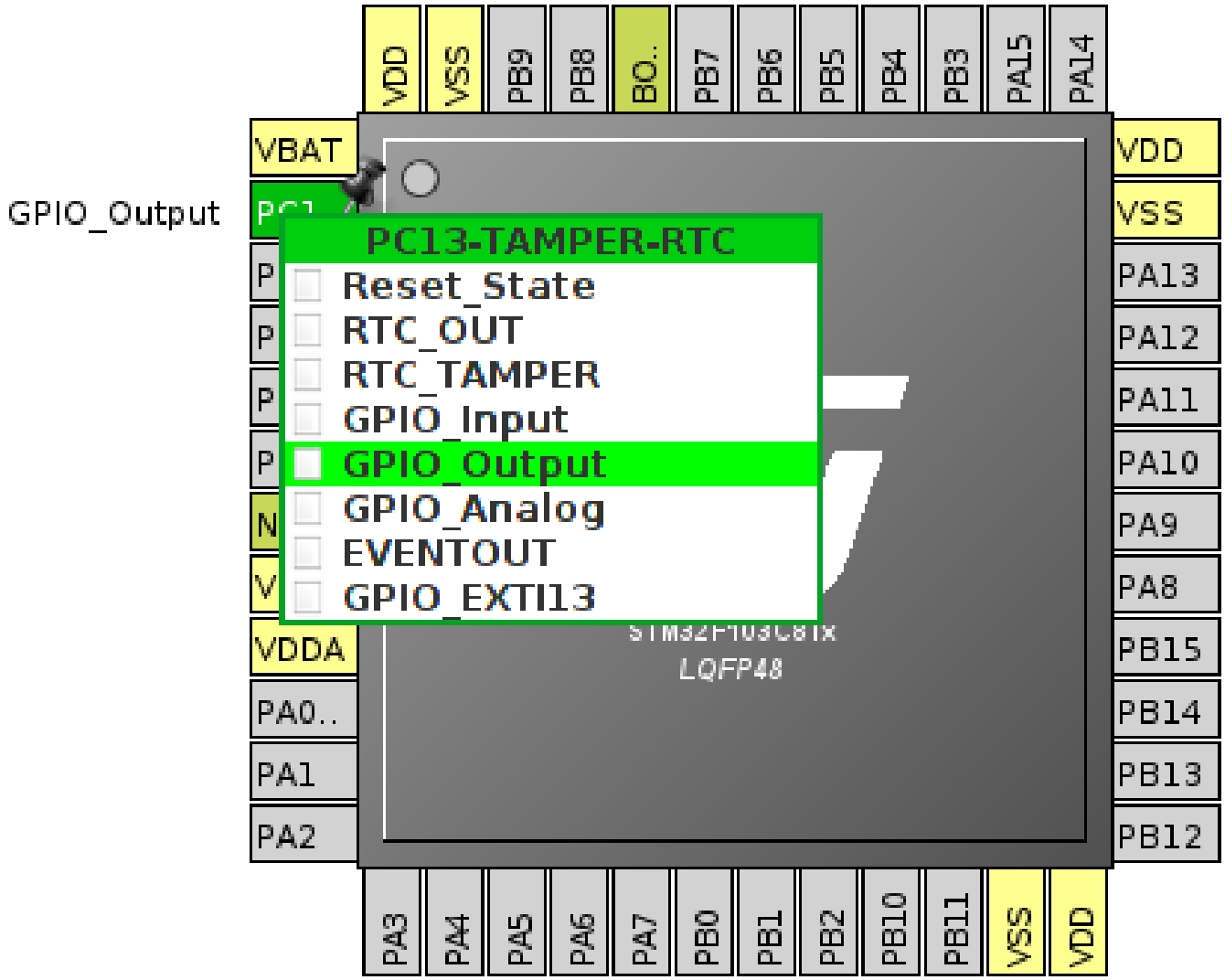
HAL

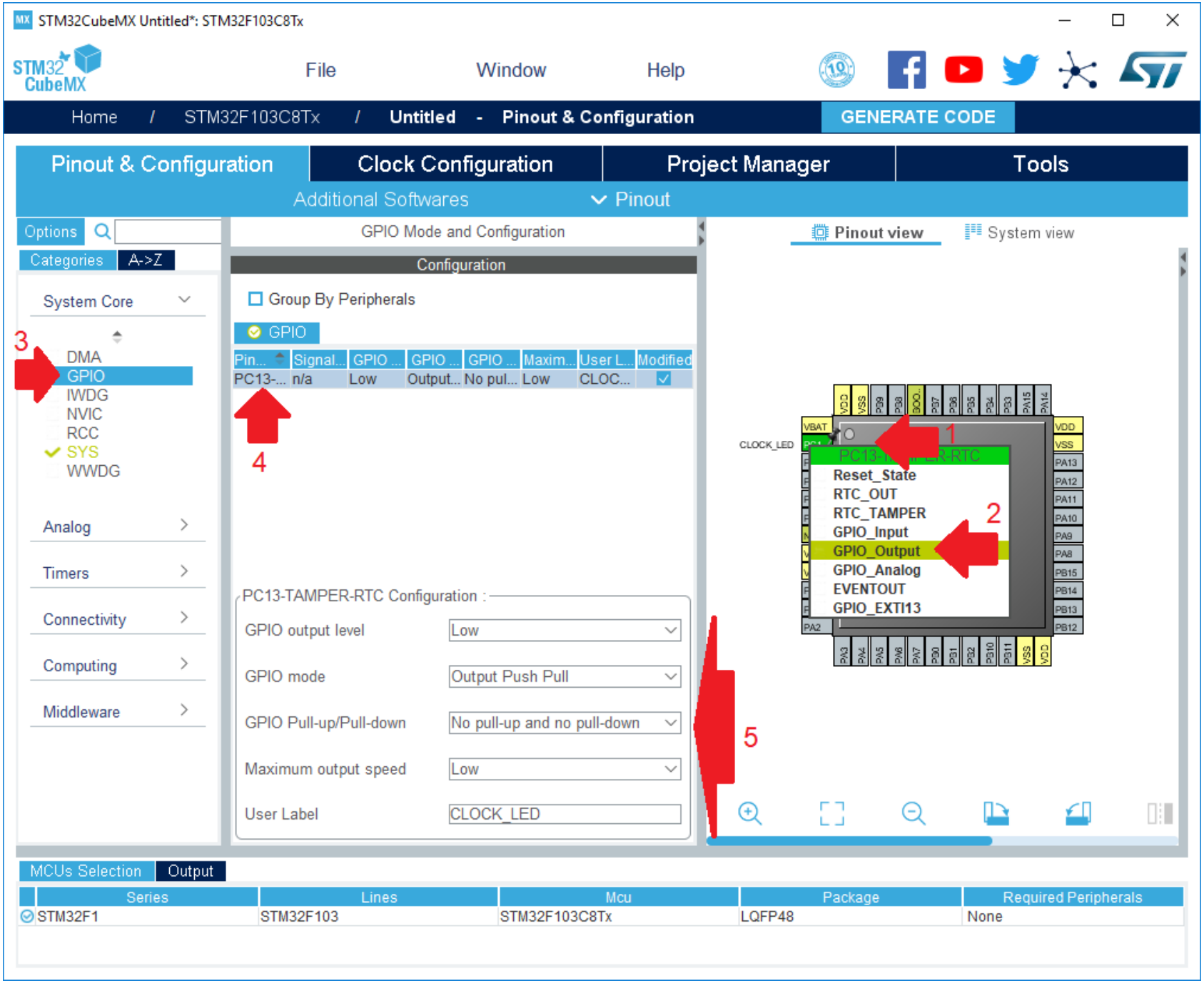
HAL *(Hardware Abstraction Layer)* — это библиотека для создания приложений на stm32, разработанная компанией *ST* в 2014 году.

CubeMX

STM32CubeMX — это программа для предварительной настройки МК и инициализации начального кода для различных сред разработки, - неотъемлемая часть HAL

Определение ножек в CubeMX




STM32CubeMX Untitled*: STM32F103C8Tx
File
Window
Help
Home / STM32F103C8Tx / Untitled - Pinout & Configuration
GENERATE CODE
Pinout & Configuration
Clock Configuration
Project Manager
Tools
Additional Softwares
Pinout
Options
Categories
A->Z
System Core
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
Timers
Connectivity
Computing
Middleware
GPIO Mode and Configuration
Configuration
Group By Peripherals
GPIO
Pin...
Signal...
GPIO ...
GPIO ...
GPIO ...
Maxim...
User L...
Modified
PC13-...
n/a
Low
Output...
No pul...
Low
CLOC...
PC13-TAMPER-RTC Configuration :
GPIO output level
Low
GPIO mode
Output Push Pull
GPIO Pull-up/Pull-down
No pull-up and no pull-down
Maximum output speed
Low
User Label
CLOCK_LED
Pinout view
System view
CLOCK_LED
Reset_State
RTC_OUT
RTC_TAMPER
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI13
1
2
3
4
5
MCUs Selection
Output
Series
Lines
Mcu
Package
Required Peripherals
STM32F1
STM32F103
STM32F103C8Tx
LQFP48
None

Плюсы и минусы CMSIS+ регистры

+ максимально быстрый код, минимальный размер кода;

+ несложно добавить описание своего/нового (росийского) МК;

-большая трудоемкость, возможны ошибки программиста;

- надо знать архитектуру и принцип работы ФМ и их регистров.

+и- HAL

+ простота настроек и работа с регистрами ФМ,

+ почти не требуется знание работы периферии и структуры регистров;

+ ускорение написания сложного кода

+ быстрый и простой перенос на другой кристалл МК

+ минимизация ошибок программиста за счёт применения проверок параметров настроек ФМ

- Избыточный код бОльшего размера чем SMSIS;
- Более медленный код;
- не для всех ARM МК (преимущественно для ST32xxxxx)

Фрагмент программы

```
//Включение тактирования порта C (со светодиодами)
RCC->AHBENR = RCC_AHBENR_GPIOCEN;
/*    Это комментарий, который продолжается до тех пор,
         пока не встретится символ конца комментария
         в виде звездочки и наклонной черты       */

GPIOC->MODER    =   GPIO_MODER_MODER8_0;      //PC8 - выход
   GPIOC->OSPEEDR =   GPIO_OSPEEDER_OSPEEDR8; //Максимальная частота
   GPIOC->OTYPER   = ~  GPIO_OTYPER_OT_8;            //push-pull
   GPIOC->PUPDR      = ~  GPIO_PUPDR_PUPDR8_0;      //Без подягивающего резистора

   GPIOC->ODR = GPIO_ODR_8; // выдать 1 в бит 8, остальные в 0
//зажечь светодиод PC8 если это анод, а катод к  GND
```