

## Пример 1 Отображение состояния кнопки SA1 светодиодом VD7

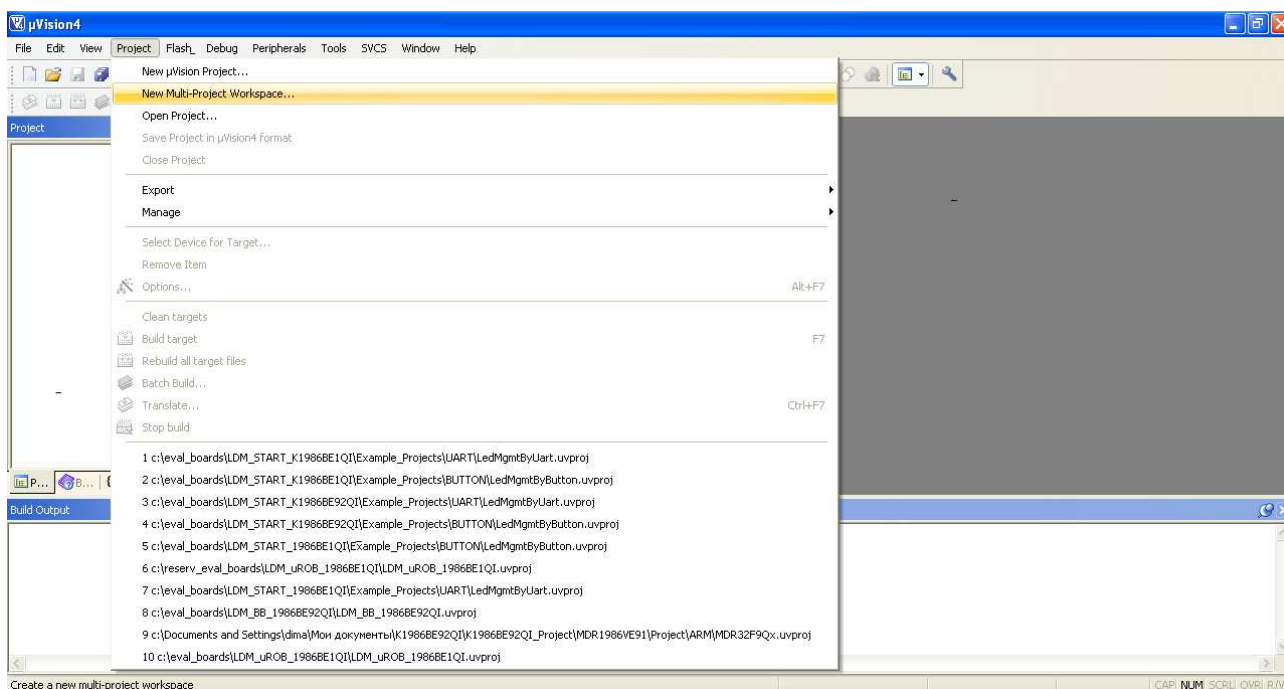
В данном примере последовательно будет показано создание проекта в среде keil uVision. Также приведены настройки среды разработки для работы с МК K1986BE92QI. На конечном этапе будет произведена загрузка получившегося в результате компиляции .hex файла в микроконтроллер через утилиту 1986WSD.

Таблица 1 – Минимальный набор аппаратного и программного обеспечения для примера 1.

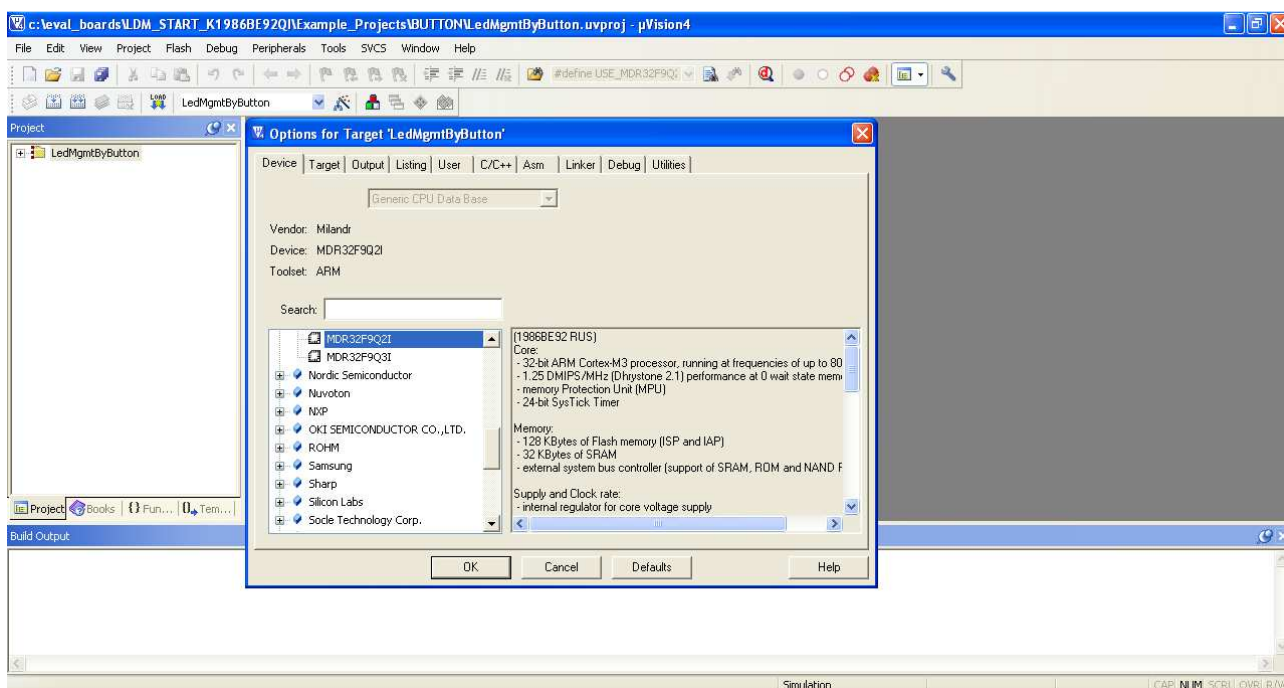
Инструментальная база	Наименование	Вэб-адрес источника
Отладочный комплект	LDM-uROB-K1986BE92QI	<a href="http://www.ldm-systems.ru">www.ldm-systems.ru</a>
Драйвер преобразователя USB/RS-232	Silicon Laboratories CP210x VCP Drivers for Windows XP/2003 Server/Vista/7	<a href="http://www.silabs.com">www.silabs.com</a>
Среда разработки ПО для МК	Keil uVision 4.74	<a href="http://www.keil.com">www.keil.com</a>
Утилита прошивки МК через UART-загрузчик	1986WSD	<a href="http://forum.milandr.ru">forum.milandr.ru</a>

## 1.1 Создание проекта и настройка среды разработки Keil uVision ver. 4.74

1 Создаём новый проект. Для этого на вкладке «Project» программы, выбираем «New uVision Project...»



2 Даем проекту название (LedMgmtByButton), сохраняем проект к примеру по указанному пути C:\LDM\_uROB\_K1986BE92Q1\Example\_projects\BUTTON\. Далее в появившемся окне выбираем устройство MDR32F9Q2I, нажимаем «ОК».



3 На следующем этапе заходим в опции проекта «Option for Target...».

На вкладке Target в разделе «Read/Only Memory Areas» устанавливаем диапазоны адресов: IROM1 Start = 0x8000000, Size = 0x20000; IRAM1 Start = 0x20000000, Size = 0x8000.

Затем на вкладке Output нажимаем кнопку «Select folder for objects...», создаем в корне проекта папку obj и выбираем ее для сохранения результатов работы компилятора. Ставим галку «Create HEX file».

Переходим на вкладку Listings нажимаем кнопку «Select folder for listings...», создаем в корне проекта папку lst и также выбираем данную папку.

В настройках компилятора «C/C++» указываем пути «Include Paths» к компонентам библиотек CMSIS (Cortex Microcontroller Software Interface Standard) и Standard Peripheral Library MDR32F9x:

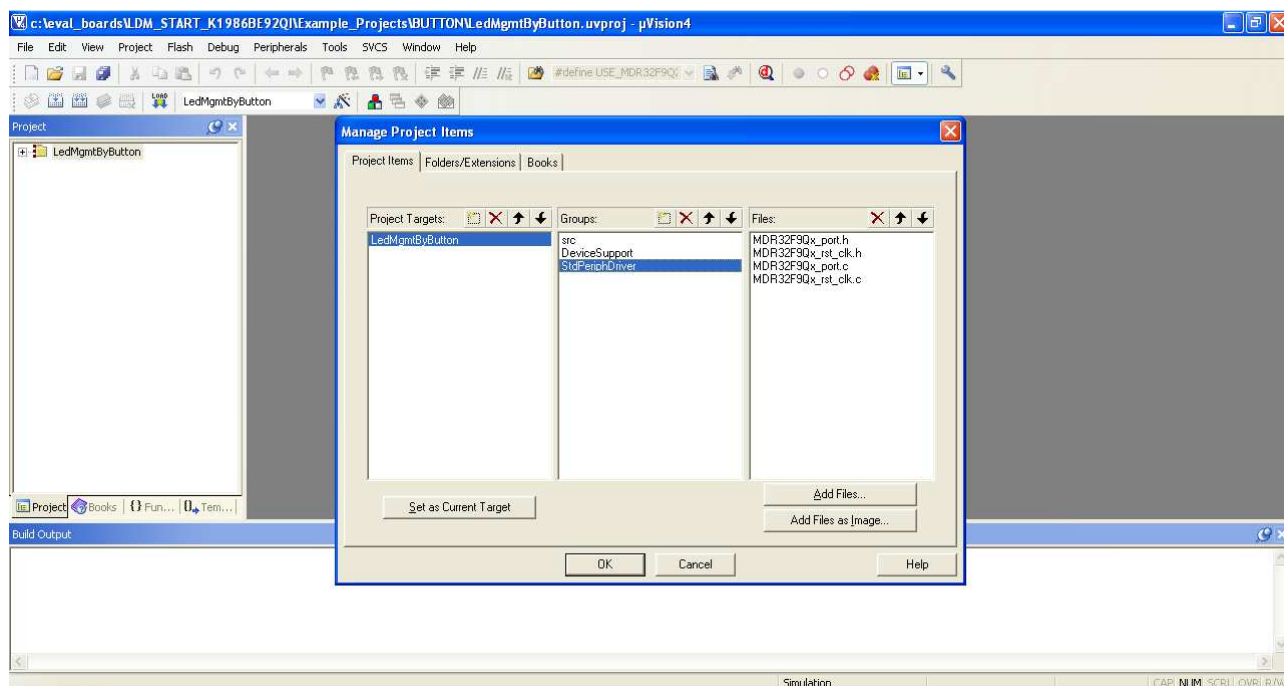
```
..\..\lib\CMSIS\CM3\DeviceSupport\MDR32F9Qx\startup;  
..\..\lib\CMSIS\CM3\DeviceSupport\MDR32F9Qx\inc;  
..\..\lib\CMSIS\CM3\DeviceSupport\MDR32F9Qx\startup\arm;  
..\..\lib\MDR32F9Qx_StdPeriph_Driver\inc;..\..\lib\Config
```

Сама библиотека поставляется в комплекте с платой. Её можно также взять на форуме компании «ЗАО «ПКК Миландр» <http://forum.milandr.ru> в разделе:

*Интегральные микросхемы ЗАО "ПКК Миландр" → 32-разрядные микроконтроллеры (1986BE9x, 1986BE1T, 1986BE2x, 1986BE3T, 1986BE4V, 1986BE8T) → Standard Peripherals Library MDR32F9x, VE1, VE3, VE4, VC1*

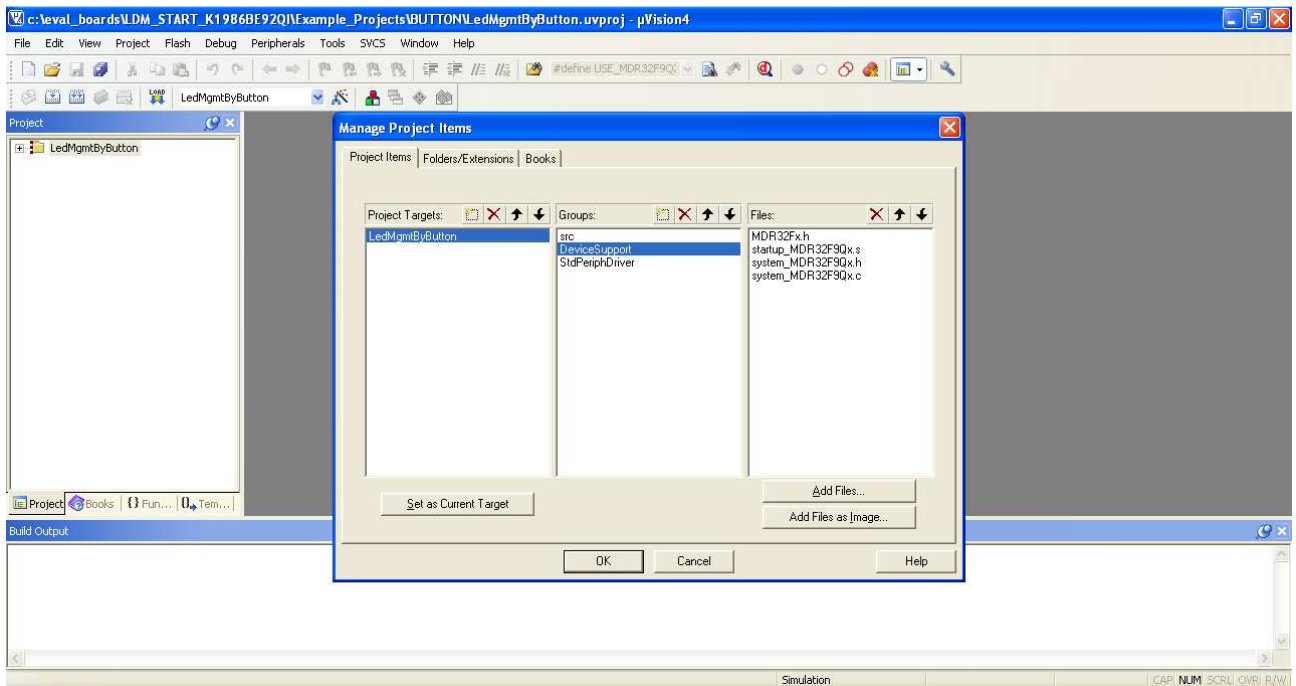
Компоненты библиотеки в папках CMSIS и MDR32F9Qx\_StdPeriph\_Driver после загрузки необходимо разместить по пути C:\LDM\_uROB\_K1986BE92QI\ в папке lib.

4 Переходим через панель быстрого доступа в менеджер управления элементами проекта, где производим формирование архитектуры проекта и добавление файлов используемых библиотек. Необходимо сформировать три группы в рамках проекта – src, DeviceSupport и StdPeriphDriver. В категорию StdPeriphDriver с помощью «Add Files...» добавляем из библиотеки файлы: MDR32F9Qx\_rst\_clk.h, MDR32F9Qx\_port.h, MDR32F9Qx\_rst\_clk.c, MDR32F9Qx\_port.c.



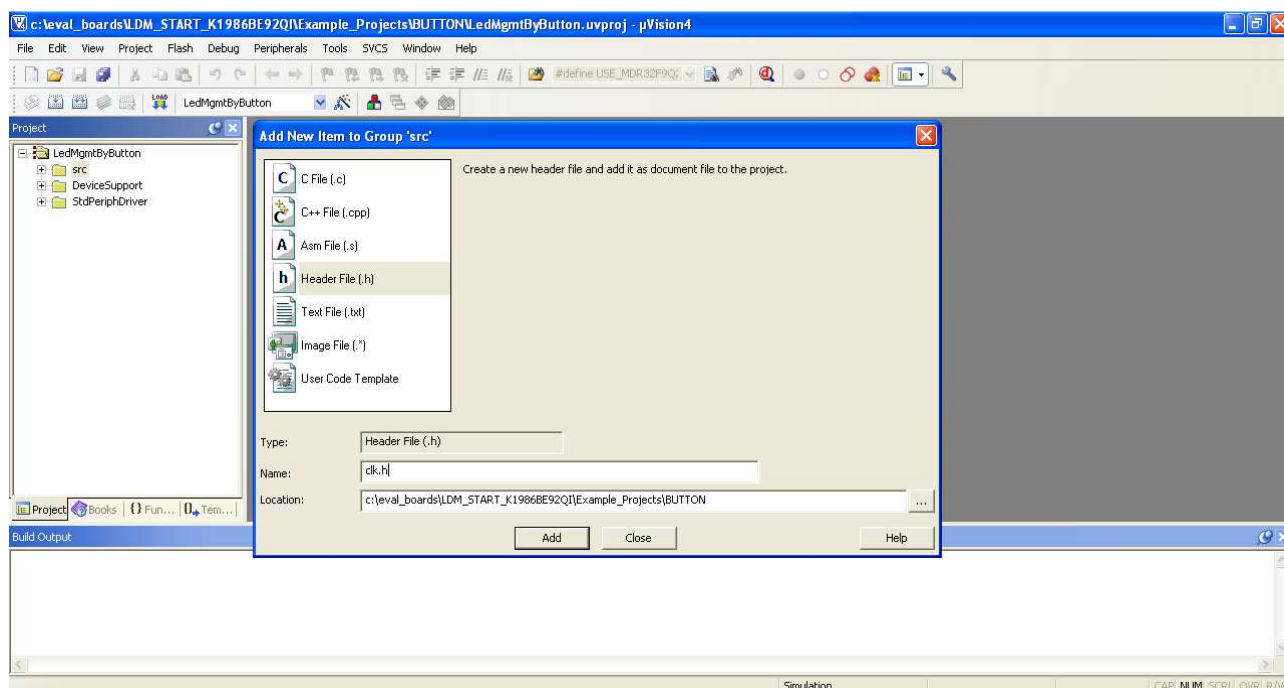
В категорию DeviceSupport добавляем 2 файла MDR32Fx.h (определение регистров МК) и startup\_MDR32F9Qx.s (настройки старта, адресного пространства и векторов прерываний). Также дополнительно добавляем 2 файла system\_MDR32F9Qx.c и

system\_MDR32F9Qx.c, которые содержат определение переменных и настройки системы тактирования микропроцессора. Нажимаем «ОК».



## 1.2 Написание функционального кода примера по отображению состояния кнопки SA1 светодиодом VD7 и добавление в проект

После этого переходим к наполнению проекта. Для этого необходимо правой кнопкой мыши нажать в разделе Project на группе src «Add new Item to Group...», выбрать создание Header File (.h), ввести имя clk.h и определить местоположение в src для нового файла. Затем также создаем файл C File (c.) с названием clk.c. Для добавления файлов в проект нажимаем правой кнопкой мыши на папку src и в выпадающем меню выбираем «Add Existing Files to Group src», где в окошке из папки src производим добавление путем нажатия на «Add». Также данную операцию можно воспроизвести через Manage Project Items как было описано ранее для файлов библиотек. Листинги с исходным кодом и комментариями представлены ниже. Приведенный код производит подключение и настройку внешнего тактирования микропроцессора.



### Листинг 1. файл clk.h

```
#ifndef __CLK_H__
#define __CLK_H__

#include "type_custom.h"

void clk_CoreConfig(void);

#endif
```

## Листинг 2. файл clk.c

```
//Функция настройки тактовой частоты МК
#include "clk.h"

//Функция настройки тактовой частоты МК
void clk_CoreConfig(void)
{
    //Реинициализация настроек тактирования
    RST_CLK_DeInit();
    //Включение тактирования от внешнего источника HSE (High Speed External)
    RST_CLK_HSEconfig(RST_CLK_HSE_ON);
    //Проверка статуса HSE
    if (RST_CLK_HSEstatus() == ERROR) while (1);
    //Настройка делителя/умножителя частоты CPU_PLL(фазовая подстройка
частоты)
    RST_CLK_CPU_PLLconfig(RST_CLK_CPU_PLLsrcHSEdiv1, RST_CLK_CPU_PLLmul5);
    //Включение CPU_PLL
    RST_CLK_CPU_PLLcmd(ENABLE);
    //Проверка статуса CPU_PLL
    if (RST_CLK_CPU_PLLstatus() == ERROR) while (1);
    //Коммутация выхода CPU_PLL на вход CPU_C3
    RST_CLK_CPU_PLLuse(ENABLE);
    //Выбор источника тактирования ядра процессора
    RST_CLK_CPUclkSelection(RST_CLK_CPUclkCPU_C3);
    //Подача тактовой частоты на PORTF, PORTE
    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTF, ENABLE);
    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTE, ENABLE);
}
```

Необходимо отметить, что в Листинге 2 файла clk.c производится настройка тактирования микропроцессора через внешний источник (кварцевый резонатор HSE), а также включение тактирования периферийных модулей, в нашем случае порта «F» и порта «E». Для более подробного ознакомления открываем раздел «Сигналы тактовой частоты» в спецификации на серию микроконтроллеров 1986BE9x, K1986BE9x, K1986BE92QI, K1986BE92QS, K1986BE91H4 (спецификация расположена на сайте производителя).

На следующем этапе требуется настроить порты ввода-вывода для работы с кнопкой SA1 и светодиодом VD7. При этом модули работы с кнопкой и светодиодом опишем в разных файлах. Для программного описания функционала работы с кнопкой потребуется создать заголовочный файл button.h (Листинг 3), где объявлены функции инициализации и считывания состояния кнопки SA1. Функциональный код приведен ниже в файле button.c (Листинге 4). Процедуру создания файлов необходимо повторить по ранее приведенному алгоритму.

### Листинг 3. файл button.h

```
#ifndef __BUTTON_H__
#define __BUTTON_H__

#include "type_custom.h"

void button_Init(void);
uint8_t button_State(void);
#endif
```

### Листинг 4. файл button.c

```
#include "button.h"

//Функция инициализации кнопки SA1
void button_Init(void)
{
    //Создание структуры для инициализации порта
    PORT_InitTypeDef PORT_InitStructure;

    //Настройки порта: ввод, функция ввода/вывода, цифровой режим,
    минимальная скорость, Pin6
    PORT_InitStructure.PORT_OE = PORT_OE_IN;
    PORT_InitStructure.PORT_FUNC = PORT_FUNC_PORT;
    PORT_InitStructure.PORT_MODE = PORT_MODE_DIGITAL;
    PORT_InitStructure.PORT_SPEED = PORT_SPEED_SLOW;
    PORT_InitStructure.PORT_Pin = (PORT_Pin_6);

    PORT_Init(MDR_PORTF, &PORT_InitStructure);
}

//Функция считывания текущего состояния кнопки SA1
uint8_t button_State(void)
{
    return PORT_ReadInputDataBit(MDR_PORTF, PORT_Pin_6);
}
```

В разделе спецификации «Порты ввода-вывода» приведены выводы доступные для микроконтроллера. В приведенной ниже таблице видно, что у МК есть аналоговая и цифровая функции. Цифровая функция порта в свою очередь разделена на несколько возможных видов: основной, альтернативный и переопределённый. Каждый из них отвечает за взаимодействие внутренних периферийных компонентов с выводами МК.

Таблица 2 – Описание выводов микроконтроллеров серии 1986BE9x

Выход	Контактная площадка кристалла	Тип корпуса				Дополнительные функции вывода				
		429x32-3	Н1864-1В	Н1648-1В	1QFP94	Аналог.	Основ.	Альтер.	Переопр.	
<b>Порт А</b>										
PA0	137	130	55	41	63	-	DATA0	EXT INT1	-	
PA1	136	129	54	40	62	-	DATA1	TMR1_CH1	TMR2_CH1	
PA2	135	128	53	39	61	-	DATA2	TMR1_CH1N	TMR2_CH1N	
PA3	134	127	52	38	60	-	DATA3	TMR1_CH2	TMR2_CH2	
PA4	133	126	51	37	59	-	DATA4	TMR1_CH2N	TMR2_CH2N	
PA5	132	125	50	36	58	-	DATA5	TMR1_CH3	TMR2_CH3	
PA6	131	124	49	35	57	-	DATA6	CAN1_TX	UART1_RXD	
PA7	130	123	48	34	56	-	DATA7	CAN1_RX	UART1_TXD	
PA8	129	122	-	-	-	-	DATA8	TMR1_CH3N	TMR2_CH3N	
PA9	128	121	-	-	-	-	DATA9	TMR1_CH4	TMR2_CH4	
PA10	125	119	-	-	-	-	DATA10	nUART1DTR	TMR2_CH4N	
PA11	124	118	-	-	-	-	DATA11	nUART1RTS	TMR2_BLK	
PA12	123	117	-	-	-	-	DATA12	nUART1RI	TMR2_ETR	
PA13	122	115	-	-	-	-	DATA13	nUART1DCD	TMR1_CH4N	
PA14	121	114	-	-	-	-	DATA14	nUART1DSR	TMR1_BLK	
PA15	120	113	-	-	-	-	DATA15	nUART1CTS	TMR1_ETR	
<b>Порт В</b>										
PB0	97	92	35	25	43	-	DATA16	TMR3_CH1	UART1_TXD	
JA_TDO										
PB1	98	93	36	26	44	-	DATA17	TMR3_CH1N	UART2_RXD	
PB2	99	94	37	27	45	-	DATA18	TMR3_CH2	CAN1_TX	
JA_TCK										
PB3	100	95	38	28	46	-	DATA19	TMR3_CH2N	CAN1_RX	
JA_TDI										
PB4	101	96	39	29	47	-	DATA20	TMR3_BLK	TMR3_ETR	
JA_TRST										
PB5	107	102	42	32	50	-	DATA21	UART1_TXD	TMR3_CH3	
PB6	108	103	43	33	51	-	DATA22	UART1_RXD	TMR3_CH3N	
PB7	109	104	44	-	52	-	DATA23	nSIRIOUT1	TMR3_CH4	
PB8	110	105	45	-	53	-	DATA24	COMP_OUT	TMR3_CH4N	
PB9	111	106	46	-	54	-	DATA25	nSIRIN1	EXT_INT4	
PB10	112	107	47	-	55	-	DATA26	EXT_INT2	nSIRIOUT1	
PB11	113	108	-	-	-	-	DATA27	EXT_INT1	COMP_OUT	
PB12	114	109	-	-	-	-	DATA28	SSP1_FSS	SSP2_FSS	
PB13	115	110	-	-	-	-	DATA29	SSP1_CLK	SSP2_CLK	
PB14	116	111	-	-	-	-	DATA30	SSP1_RXD	SSP2_RXD	
PB15	119	112	-	-	-	-	DATA31	SSP1_TXD	SSP2_TXD	
<b>Порт С</b>										
PC0	96	91	34	23	42	-	READY*	SCL1	SSP2_FSS	
PC1	95	90	33	-	41	-	OE	SDA1	SSP2_CLK	
PC2	94	89	31	-	40	-	WE	TMR3_CH1	SSP2_RXD	
PC3	93	88	-	-	-	-	BE0	TMR3_CH1N	SSP2_TXD	
PC4	92	87	-	-	-	-	BE1	TMR3_CH2	TMR1_CH1	
PC5	91	86	-	-	-	-	BE2	TMR3_CH2N	TMR1_CH1N	
PC6	90	85	-	-	-	-	BE3	TMR3_CH3	TMR1_CH2	
PC7	89	84	-	-	-	-	CLOCK	TMR3_CH3N	TMR1_CH2N	
PC8	88	83	-	-	-	-	CAN1_TX	TMR3_CH4	TMR1_CH3	
<b>Порт D</b>										
PD0	70	65	23	17	31	-	ADC0_R	TMR1_CH1N	UART2_RXD	
JB_TMS										
PD1	71	66	24	18	32	-	ADC1_R	TMR1_CH1	UART2_TXD	
JB_TCK										
PD2	72	67	25	19	33	-	ADC2	BUSY1	SSP2_RXD	
PD2	72	67	25	19	33	-	ADC2	BUSY1	SSP2_RXD	
JB_TRST										
PD3	73	68	26	20	34	-	ADC3	-	SSP2_FSS	
JB_TDI										
PD4	69	64	22	-	30	-	ADC4	TMR1_ETR	nSIRIOUT2	
JB_TDO										
PD5	74	69	27	-	35	-	ADC5	CLE	SSP2_CLK	
PD6	75	70	28	-	36	-	ADC6	ALE	SSP2_TXD	
PD7	68	63	21	-	29	-	ADC7	TMR1_BLK	nSIRIN2	
PD8	67	62	-	-	-	-	ADC8	TMR1_CH4N	TMR2_CH1	
PD9	76	71	-	-	-	-	ADC9	CAN2_TX	TMR2_CH1N	
PD10	66	61	-	-	-	-	ADC10	TMR1_CH2	TMR2_CH2	
PD11	65	60	-	-	-	-	ADC11	TMR1_CH2N	TMR2_CH2N	
PD12	64	59	-	-	-	-	ADC12	TMR1_CH3	TMR2_CH3	
PD13	63	58	-	-	-	-	ADC13	TMR1_CH3N	TMR2_CH3N	
PD14	62	57	-	-	-	-	ADC14	TMR1_CH4	TMR2_CH4	
PD15	61	56	-	-	-	-	ADC15	CAN2_RX	BUSY2	
<b>Порт E</b>										
PE0	56	53	18	14	26	-	DAC2_O	ADDR16	TMR2_CH1	
UT										
PE1	55	52	17	-	25	-	DAC2_R	ADDR17	TMR2_CH1N	
EF										
PE2	48	45	14	11	22	-	COMP_I	ADDR18	TMR2_CH3	
N1										
PE3	47	44	13	10	21	-	COMP_I	ADDR19	TMR2_CH3N	
N2										
PE4	45	42	-	-	-	-	COMP_R	ADDR20	TMR2_CH4N	
EF+										
PE5	44	41	-	-	-	-	COMP_R	ADDR21	TMR2_BLK	
EF-										
PE6	36	33	8	6	16	-	OSC_IN3	ADDR22	CAN2_RX	
2										
PE7	35	32	7	-	15	-	OSC_OU	ADDR23	CAN2_TX	
T32										
PE8	46	43	-	-	-	-	COMP_I	ADDR24	TMR2_CH4	
N3										
PE9	54	51	-	-	-	-	DAC1_O	ADDR25	TMR2_CH2	
UT										
PE10	53	50	-	-	-	-	DAC1_R	ADDR26	TMR2_CH2N	
UT										
<b>Питание</b>										
Vcc	1,2,31,32,7,78,103,104	1,28,29,72,73,98,99	4,29,40,57	5,21,30,43	1, 12, 38, 48	-	Основное питание 2.2...3.6В			
AUcc	59	55	20	16	28	-	Аналоговое питание АЦП, ЦАП и Comparator 2.4...3.6В			
AUcc1	51,52	48,49	16	13	24	-	Аналоговое питание PLL 2.2...3.6В			
BUcc	33	30	5	-	13	-	Батарейное питание 1.8...3.6В			
GND	29,30,79,105,139	26,27,74,100,132	3,30,41,56	4,22,31,42	11, 39, 49, 64	-	Общий			
AGND	57	54	19	15	27	-	Общий			
AGND1	49,50	46,47	15	12	23	-	Общий			
<b>Порт F</b>										
PF0	3	2	58	44	2	-	ADDR0	SSP1_TXD	UART2_RXD	
PF1	4	3	59	45	3	-	ADDR1	SSP1_CLK	UART2_TXD	
PF2	5	4	60	46	4	-	ADDR2	SSP1_FSS	CAN2_RX	
PF3	6	5	61	47	5	-	ADDR3	SSP1_RXD	CAN2_TX	
PF4	7	6	62	48	6	-	ADDR4	-	-	
MODE[0]										
PF5	8	7	63	1	7	-	ADDR5	-	-	
MODE[1]										
PF6	9	8	64	-	8	-	ADDR6	TMR1_CH1	-	
MODE[2]										
PF7	10	9	-	-	-	-	ADDR7	TMR1_CH1N	TMR3_CH1	
PF8	11	10	-	-	-	-	ADDR8	TMR1_CH2	TMR3_CH1N	
PF9	12	11	-	-	-	-	ADDR9	TMR1_CH2N	TMR3_CH2	
PF10	13	12	-	-	-	-	ADDR10	TMR1_CH3	TMR3_CH2N	
PF11	14	13	-	-	-	-	ADDR11	TMR1_CH3N	TMR3_ETR	
PF12	15	14	-	-	-	-	ADDR12	TMR1_CH4	SSP2_FSS	
PF13	16	15	-	-	-	-	ADDR13	TMR1_CH4N	SSP2_CLK	
PF14	17	16	-	-	-	-	ADDR14	TMR1_ETR	SSP2_RXD	
PF15	18	17	-	-	-	-	ADDR15	TMR1_BLK	SSP2_TXD	
<b>Системное управление</b>										
RESET	40	37	10	7	18	-	Сигнал внешнего сброса 0 – сброс 1 – разрешена работа			
WAKEUP	38	35	9	-	17	-	Сигнал внешнего выхода из режима StandBy 0 – выход из режима StandBy 1 – нет сигнала выхода			
STANDBY	34	31	6	-	14	-	Флаж режима StandBy 0 – микросхема не в режиме StandBy 1 – микросхема в режиме StandBy			
OSC_IN	41	38	11	8	19	-	Вход генератора HSE			
OSC_OUT	42	39	12	9	20	-	Выход генератора HSE			
<b>USB интерфейс</b>										
DP	22	21	1	2	9	-	Шина USB D+			
DN	25	22	2	3	10	-	Шина USB D-			

Из схематехнического документа для отладочного комплекта (доступного на сайте производителя <https://ldm-systems.ru/product/21102>, Схема LDM-uROB-K1986BE92QI.pdf) следует, что для взаимодействия с кнопкой потребуется вывод PF6 порта «F». В структуре инициализации необходимо произвести конфигурирование порта на вход в режиме «ввода-вывода» на минимальной скорости в цифровом режиме. Для взаимодействия с регистром и определения состояния кнопки требуется функция API драйвера PORT\_ReadInputDataBit, где в качестве входных параметров передается номер вывода и порт (PORTF, PF6).



Для настройки светодиода необходимо добавить в проект файлы led.h и led.c. По аналогии с описанием порта ввода-вывода кнопки производим конфигурирование вывода PE0 порта «E» для светодиода .

### Листинг 5. файл led.h

```
#ifndef __LED_H__
#define __LED_H__

#include "type_custom.h"

void led_Init(void);
void led_Write(bool on_off);

#endif
```

### Листинг 6. файл led.c

```
#include "led.h"

//Функция инициализации светодиода VD7
void led_Init(void)
{
    //Создание структуры для инициализации порта
    PORT_InitTypeDef PORT_InitStructure;

    //Настройки порта: вывод, функция ввода/вывода, цифровой режим,
    //максимальная скорость, Pin0
    PORT_InitStructure.PORT_OE = PORT_OE_OUT;
    PORT_InitStructure.PORT_FUNC = PORT_FUNC_PORT;
    PORT_InitStructure.PORT_MODE = PORT_MODE_DIGITAL;
    PORT_InitStructure.PORT_SPEED = PORT_SPEED_MAXFAST;
    PORT_InitStructure.PORT_Pin = (PORT_Pin_0);

    PORT_Init(MDR_PORTE, &PORT_InitStructure);
}

//Функция записи состояния (1:0) светодиода VD7
void led_Write(bool on_off)
{
    PORT_WriteBit(MDR_PORTE, PORT_Pin_0, on_off ? Bit_SET : Bit_RESET);
}
```

В структуре инициализации необходимо произвести конфигурирование порта на вывод с функцией «ввода-вывода» на максимальной скорости в цифровом режиме. Для взаимодействия с регистром и записи состояния светодиода VD7 потребуется функция API драйвера PORT\_WriteBit, где в качестве входных параметров передается номер вывода, порт, а также состояние вывода 1 или 0 (PORTE, PE0, state).

Для определения пользовательских числовых и булевых типов потребуется их описание в заголовочном файле type\_custom.h. Также необходимо произвести подключение заголовочных файлов с драйверами, которые используются в проекте.

## Листинг 7. файл type\_custom.h.

```
#ifndef __TYPE_CUSTOM_H__
#define __TYPE_CUSTOM_H__

#include "MDR32F9Qx_config.h"
#include "MDR32F9Qx_board.h"
#include "MDR32F9Qx_rst_clk.h"
#include "MDR32F9Qx_port.h"

typedef unsigned char bool;
typedef unsigned char uint8_t;
typedef unsigned short int uint16_t;
typedef unsigned int uint32_t;

#define true 1
#define false 0

#endif
```

После этого можно переходить к созданию файла main.c, который является основной точкой входа для работы программы микроконтроллера. Данный файл должен содержать основной алгоритм работы программы. Сперва, необходимо подключить заголовочные файлы ранее созданных модулей «led.h», «button.h», «clk.h». В теле main требуется инициализировать программные модули тактирования, кнопки и светодиода. Затем в бесконечном цикле while(1) добавляем чтение состояния кнопки SA1 с вывода (PF6) порта (PORTF) и его дальнейшую запись на вывод (PE0) порта (PORTE) светодиода VD7.

## Листинг 8. файл main.c

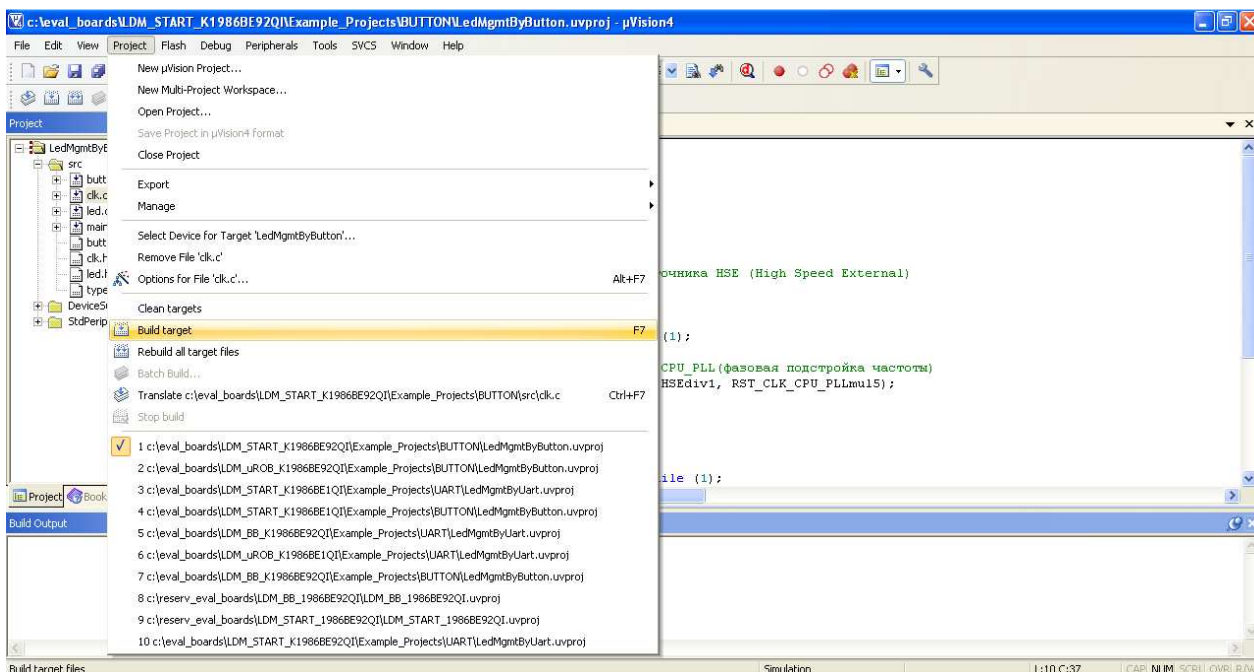
```
#include "clk.h"
#include "led.h"
#include "button.h"

int main(void)
{
    //Создание переменной для записи текущего состояния кнопки
    uint8_t state = 0;

    //Инициализация модулей
    clk_CoreConfig();
    led_Init();
    button_Init();

    while (1){
        //Считывание текущего состояния кнопки SA1
        state = button_State();
        //Запись состояния кнопки на вывод светодиода VD7
        led_Write(!state);
    }
}
```

Теперь, необходимо полностью откомпилировать проект. Для этого в меню «Project» выбираем «Build Target» (F7).



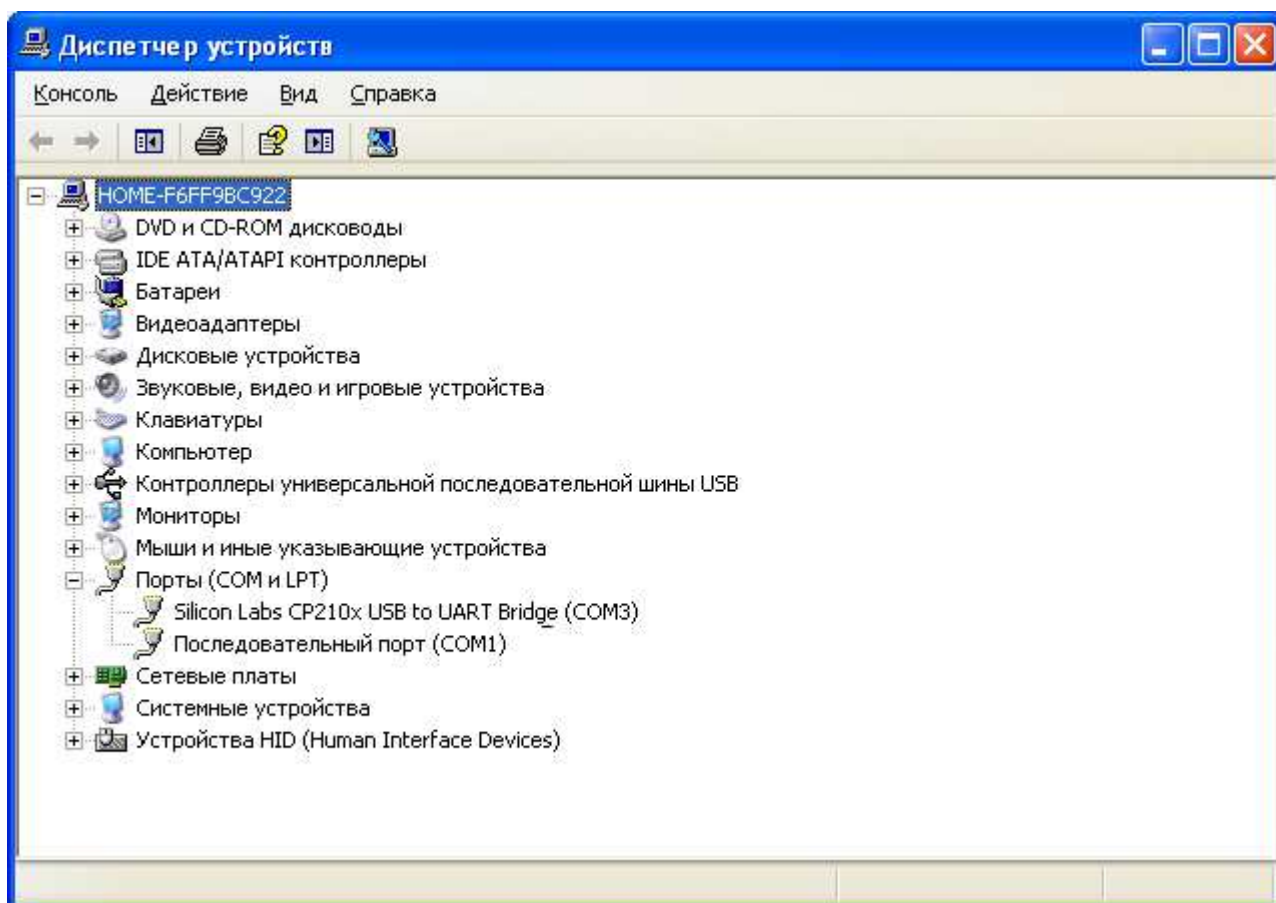
Можно увидеть, что в папке проекта, в случае отсутствия ошибок или предупреждений в окне «Build Output», в директории obj появился файл с расширением .hex.

### 1.3 Загрузка ВПО в микроконтроллер через встроенный UART-загрузчик

Для загрузки ВПО в микроконтроллер необходимо загрузить программу прошивки микроконтроллера через внутренний UART-загрузчик: *Программно-отладочные средства*→*Программное обеспечение*→*Утилиты для прошивки Flash МК 1986 через UART*

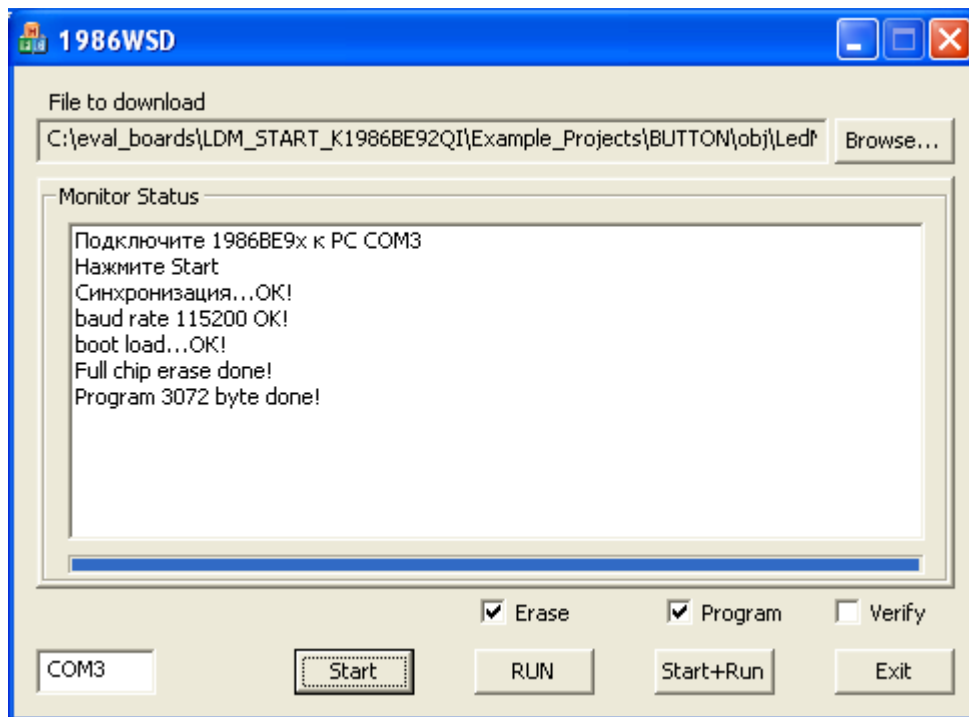
Перед тем, как запустить программу, необходимо установить на компьютер драйвер для микросхемы CP2102. Скачать драйвера можно непосредственно с сайта производителя, компании «Silicon Labs» (<http://www.silabs.com>).

После установки драйвера, в системе, при подключении платы в диспетчере устройств появится виртуальный COM-порт. В данном примере это COM3.



Для работы с ПО-загрузчиком необходимым условием является подключение через COM порты 1-9. Далее необходимо перевести dip - переключатели SA4 1-2 в положение ON («1»), для загрузки данных по UART. Отключение питания, подключенной к USB платы производится по кнопке SA5. Установка «1» на входе M2 производится по кнопке SA1. Таким образом, для получения режима загрузки BOOT (MODE) микроконтроллера M2..M0 = «101» при котором происходит загрузка по UART, необходимо зажать SA1 и коротким нажатием на SA5 сбросить питание, затем отжать SA1. Подробнее о режимах загрузки микроконтроллера написано в спецификации на микросхему.

Запускаем программу UART-загрузчик «1986WSD.exe». Выбираем hex-файл из папки obj созданного проекта. Настраиваем параметры порта, скорости, ставим галочки для операций стирания и программирования. Нажимаем «Start». После чего в окне «Monitor Status» должна появиться соответствующую надпись об успешном выполнении загрузки программы во внутреннюю flash-память микроконтроллера. Можно непосредственно произвести запуск программы, нажатием кнопки «RUN» после окончания операции загрузки.



Последовательность действий для загрузки прошивки в микроконтроллер:

- ✓ Вводим в режим BOOT MODE путем зажатия SA1 и одновременного сброса питания коротким нажатием SA5
- ✓ Запускаем программу 1986WSD.exe, указываем COM-порт, путь к hex-файлу, параметры загрузки (галочки Erase и Program)
- ✓ Нажимаем «Start»
- ✓ Нажимаем «RUN»

Для проверки работоспособности ВПО необходимо нажать на SA1, после чего должен загореться светодиод VD7.

## Пример 2

### Управление светодиодом VD7 по UART2

Для начала работы потребуется создать новый проект с названием «LedMgmtByUart» в папке «Example\_projects\UART\». Далее необходимо воспроизвести последовательность действий из 1го примера раздела «Создание проекта и настройка среды разработки Keil uVision ver. 4.74» для настройки проекта. Таким образом, у нас получился настроенный проект с подключенными библиотечными файлами из папок *MDR32F9Qx\_StdPeriph\_Driver* и *CMSIS*. В группу с драйверами *StdPeriphDriver* следует добавить файлы для обеспечения настройки и взаимодействия по UART — *MDR32F9Qx\_uart.h* и *MDR32F9Qx\_uart.c*.

Перед тем как приниматься создавать функциональный код необходимо скопировать файлы *led.c* и *led.h* в папку *src* из первого примера. Добавить данные файлы можно также в группу *src*. В текущем варианте код для работы со светодиодом VD7 остается неизменным и подходит для реализации данного примера. Файл с пользовательскими типами *type\_costum.h* также требуется скопировать в папку *src* и добавить в проект.

Для упрощения можно взять написанные ранее файлы *clk.c* и *clk.h* с процедурой настройки тактовой частоты микропроцессора. В файл *clk.c* для того, чтобы подать тактовую частоту на порт «D» и на UART2 надо дописать строки:

```
//Подача тактовой частоты на порт PORTD
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTD, ENABLE);
//Подача тактовой частоты на UART2
RST_CLK_PCLKcmd(RST_CLK_PCLK_UART2, ENABLE);
```

В общем виде файл примет вид:

#### Листинг 1. файл *clk.c*

```
#include "clk.h"

//Функция настройки тактовой частоты МК
void clk_CoreConfig(void)
{
    //Реинициализация настроек тактирования
    RST_CLK_DeInit();
    //Включение тактирования от внешнего источника HSE (High Speed External)
    RST_CLK_HSEconfig(RST_CLK_HSE_ON);
    //Проверка статуса HSE
    if (RST_CLK_HSEstatus() == ERROR) while (1);
    //Настройка делителя/умножителя частоты CPU_PLL( фазовая подстройка
частоты)
    RST_CLK_CPU_PLLconfig(RST_CLK_CPU_PLLsrcHSEdiv1, RST_CLK_CPU_PLLmul5);
    //Включение CPU_PLL
    RST_CLK_CPU_PLLcmd(ENABLE);
    //Проверка статуса CPU_PLL
    if (RST_CLK_CPU_PLLstatus() == ERROR) while (1);
    //Коммутация выхода CPU_PLL на вход CPU_C3
    RST_CLK_CPU_PLLuse(ENABLE);
    //Выбор источника тактирования ядра процессора
    RST_CLK_CPUclkSelection(RST_CLK_CPUclkCPU_C3);
    //Подача тактовой частоты на порты PORTE, PORTD
    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTE, ENABLE);
    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTD, ENABLE);
    //Подача тактовой частоты на UART2
    RST_CLK_PCLKcmd(RST_CLK_PCLK_UART2, ENABLE);
}
```

Далее для работы с UART2 создадим и добавим в проект файлы `uart.h` и `uart.c`. В заголовочном файле `uart.h` (Листинг 2) обозначены прототипы функций для инициализации периферийного модуля UART2, а также процедура приема байта и последующей интерпритации его в команду.

## Листинг 2. файл `uart.h`

```
#ifndef __UART_H__
#define __UART_H__

#include "type_custom.h"

void uart_Init(void);
uint8_t uart_Work(void);
uint8_t uart_Send(void);
#endif
```

В коде инициализации UART2 (Листинг 3) присутствует процедура `uart_Init`, которая настраивает и инициализирует структурами порт и непосредственно сам интерфейс последовательного приемопередатчика UART. Для настройки выводов порта UART2 производится отключение подтягивающих резисторов, функции триггера Шмидта, фильтрации Гаусса. Функционально порт настраивается в цифровом режиме как основной. Скорость порта задается максимальной для достижения требуемого быстродействия на высоких скоростях. Выходные каскады должны быть сконфигурированы как push-pull драйверы. Соответственно пин PD1 настраивается на выход т.к. выполняет функцию передатчика TX, PD0 — на вход, т.к. выполняет функцию приемника RX. Для обеспечения требуемых скоростей блока UART требуется настроить делитель на 16 (функция `UART_BRGInit`). Параметры UART подобраны из стандартной линейки, где скорость соответствует 115200 бод, длина слова — 8 бит, 1 стоп-бит, отсутствие паритета, аппаратный контроль потока на RX и TX. Завершается инициализация командой включения. API функции позволяют получить доступ к адресам регистров специального назначения микропроцессора и произвести их настройку. Для наглядности можно посмотреть на реализацию функций настройки порта, чтения флагов и приема данных.

## Листинг 3. файл `uart.c`

```
#include "uart.h"

//Функция инициализации UART2
void uart_Init()
{
    //Создание структур для инициализации выводов порта и UART
    PORT_InitTypeDef PortInit;
    UART_InitTypeDef UART_InitStructure;

    //Настройка порта: основная функция, цифровой режим, максимальная
    скорость
    //Отключение: подтягивающих резисторов, функций триггера Шмидта,
    фильтрации Гаусса
    PortInit.PORT_PULL_UP = PORT_PULL_UP_OFF;
    PortInit.PORT_PULL_DOWN = PORT_PULL_DOWN_OFF;
    PortInit.PORT_PD_SHM = PORT_PD_SHM_OFF;
    PortInit.PORT_PD = PORT_PD_DRIVER;
    PortInit.PORT_GFEN = PORT_GFEN_OFF;
    PortInit.PORT_FUNC = PORT_FUNC_ALTER;
    PortInit.PORT_SPEED = PORT_SPEED_MAXFAST;
    PortInit.PORT_MODE = PORT_MODE_DIGITAL;
```

```

//Настройка PORTD pins 1 на вывод (UART2_TX)
PortInit.PORT_OE = PORT_OE_OUT;
PortInit.PORT_Pin = PORT_Pin_1;
PORT_Init(MDR_PORTD, &PortInit);

//Настройка PORTD pins 0 на ввод (UART2_RX)
PortInit.PORT_OE = PORT_OE_IN;
PortInit.PORT_Pin = PORT_Pin_0;
PORT_Init(MDR_PORTD, &PortInit);

//Установка делителя частоты UART2 HCLKdiv = 16
UART_BRGInit(MDR_UART2, UART_HCLKdiv16 );

//Настройка параметров UART2: 115200, 8бит, 1 стоп бит, без паритета,
откл. буфера FIFO, контроль потока RX/TX
UART_InitStructure.UART_BaudRate = 115200;
UART_InitStructure.UART_WordLength = UART_WordLength8b;
UART_InitStructure.UART_StopBits = UART_StopBits1;
UART_InitStructure.UART_Parity = UART_Parity_No;
UART_InitStructure.UART_FIFOMode = UART_FIFO_OFF;
UART_InitStructure.UART_HardwareFlowControl =
UART_HardwareFlowControl_RXE | UART_HardwareFlowControl_TXE;

//Конфигурирование параметров UART2
UART_Init(MDR_UART2, &UART_InitStructure);

//Команда включения UART2
UART_Cmd(MDR_UART2, ENABLE);
}

//Функция приема байта по UART2
uint8_t uart_Work(void)
{
    uint16_t data;
    uint8_t res = 0xFF;

    if(UART_GetFlagStatus(MDR_UART2, UART_FLAG_RXFE) != SET)
    {

        data = UART_ReceiveData(MDR_UART2);

        if((char)(data & 0xFF) == '0')
            res = 0x00;
        else if((char)(data & 0xFF) == '1')
            res = 0x01;
    }

    return res;
}

```

Функция приема байт по UART `uart_Work` представляет собой интерпритатор принятых байт в формате ASCII в команды 0x00 - «0», 0x01 - «1» для светодиода VD7. В процессе приема происходит анализ флага регистра данных приемника. Если он выставляется в «0» значит регистр заполнен и необходимо прочитать принятый байт из регистра UARTx->DR.

Основной файл программы `main.c` представлен ниже (Листинг 4). В данном примере описан процесс получения команд по UART в основном цикле, регистрация и запись состояния на вывод светодиода VD7. После инициализации периферийных модулей тактирования, светодиода, приемопередатчика UART происходит переход в основной бесконечный цикл программы, где происходит постоянный опрос регистра UART2 и при



наличии данных — чтение и интерпритация кода ASCII в код команды для записи состояния (1:0) на выходной вывод PE0 порта «E» светодиода VD7.

#### Листинг 4. файл main.c

```
#include "clk.h"
#include "led.h"
#include "uart.h"

int main(void)
{
    //Переменная для чтения команды по UART
    uint8_t cmd = 0;

    //Инициализация периферии
    clk_CoreConfig();
    led_Init();
    uart_Init();

    while (1){

        //Получение команд по UART
        cmd = uart_Work();

        //Условие сравнения команд и вывода (1:0) на светодиод VD7
        if(cmd == 0x00)
            led_Write(false);
        else if(cmd == 0x01)
            led_Write(true);
    }
}
```

После завершения написания программы необходимо скомпилировать проект. После успешного завершения сборки проекта и создания hex-файла переходим к загрузке полученного ВПО в микроконтроллер через UART-загрузчик. Таким образом, повторяем последовательность действий для ввода в режим BOOT MODE. Запускаем утилиту «1986WSD.exe», указываем hex-файл и необходимые настройки, прошиваем внутреннюю flash-память. Нажимаем «RUN» - итак все готово для тестирования.

Для тестирования потребуется утилита с дружественным интерфейсом под названием Termite — терминал для работы с последовательным COM портом (загружаем и устанавливаем Termite version 3.2 - complete setup с сайта [http://www.compuhase.com/software\\_termite.htm](http://www.compuhase.com/software_termite.htm)). После установки запускаем Termite.exe и переходим в настройки «Settings». Снизу на рисунке показаны настройки для обеспечения связи с отладочной платой LDM-uROB-K1986BE92QI, для примера взят «COM3» (на другом ПК номер порта может отличаться). Далее нажимаем на кнопку «Disconnected — click to connect». При успешном соединении на кнопке отобразятся текущие параметры установленного соединения (как на рисунке см ниже). Затем необходимо в очередности 1 + Enter , 0 + Enter отправлять команды в плату и наблюдать за светодиодом VD7. Состояние должно меняться в соответствии с отправленными командами.

