Пример 1
Отображение состояния кнопки SA3 светодиодом VD2

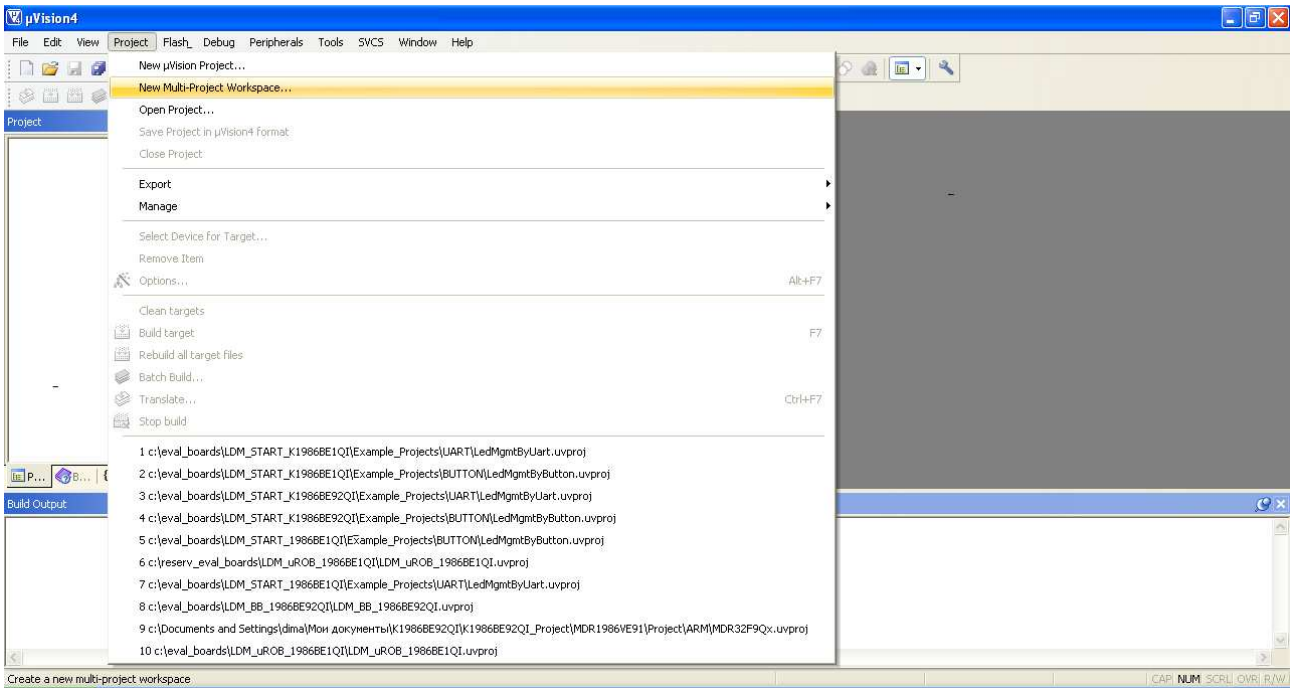
В данном примере последовательно будет показано создание проекта в среде keil uVision. Также приведены настройки среды разработки для работы с МК K1986BE1QI. На конечном этапе будет произведена загрузка получившегося в результате компиляции .hex файла в микроконтроллер через утилиту 1986VE1WSD.

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

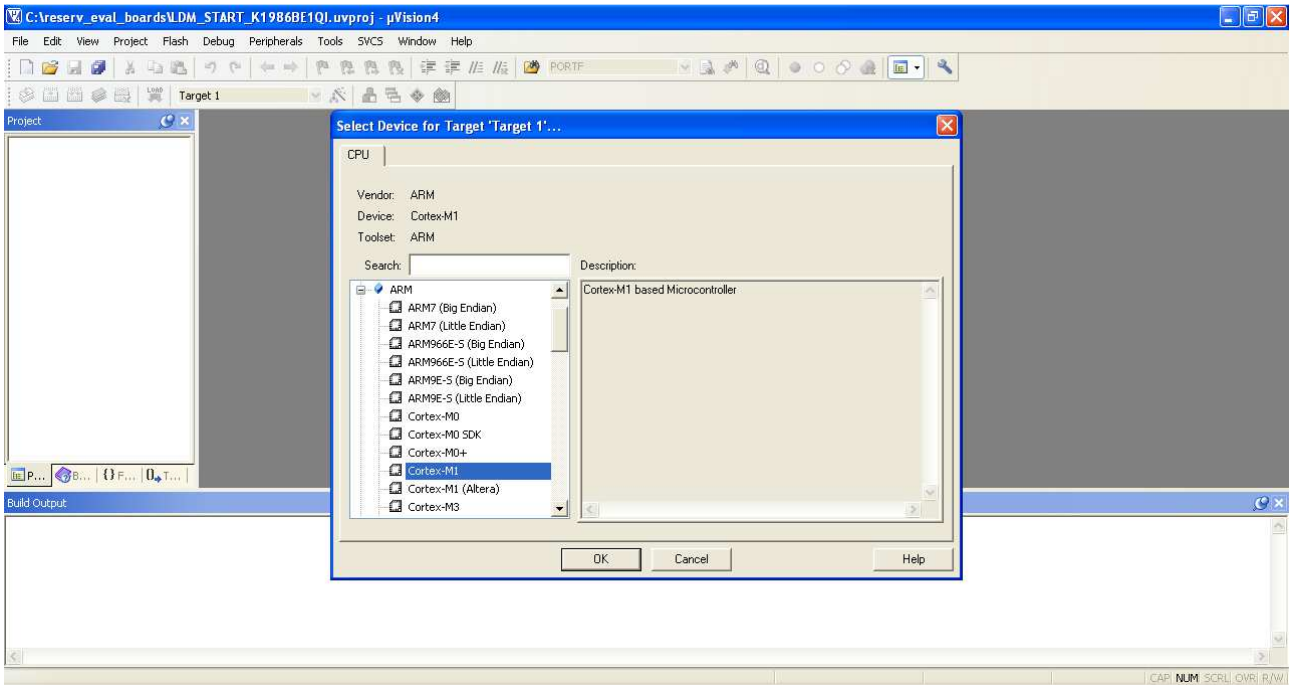
Инструментальная база	Наименование	Вэб-адрес источника
Отладочный комплект	LDM-START-K1986BE1QI	www.ldm-systems.ru
Драйвер преобразователя USB/RS-232	Silicon Laboratories CP210x VCP Drivers for Windows XP/2003 Server/Vista/7	www.silabs.com
Среда разработки ПО для МК	Keil uVision 4.74	www.keil.com
Утилита прошивки МК через UART-загрузчик	1986VE1WSD	forum.milandr.ru

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…»

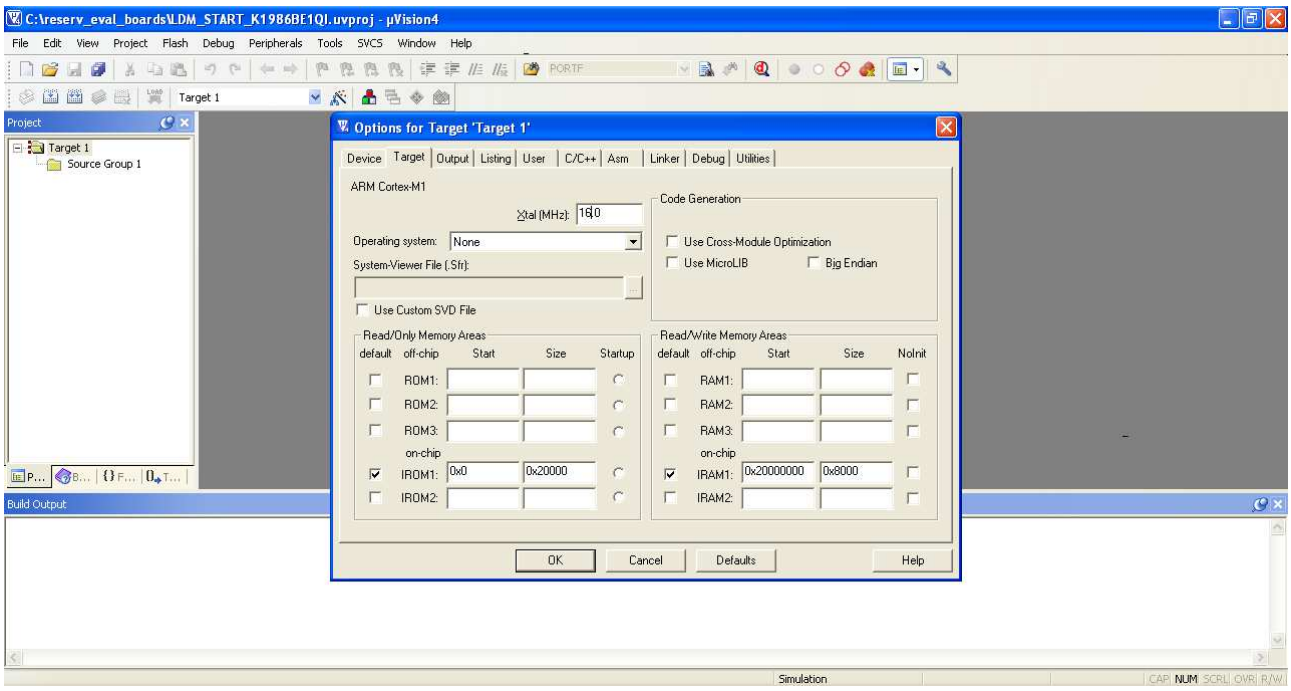


2 Даем проекту название (LedMgmtByButton), сохраняем проект к примеру по указанному пути C:\LDM_START_K1986BE1QI\Example_projects\BUTTON\. Далее в появившемся окне выбераем устройство ARM - Cortex — M1, нажимаем «ОК».



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 = 0x00, Size = 0x2000; IRAM1 Start = 0x20000000, Size = 0x8000.



Затем на вкладке Output нажимаем кнопку «Select folder 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 папаку.

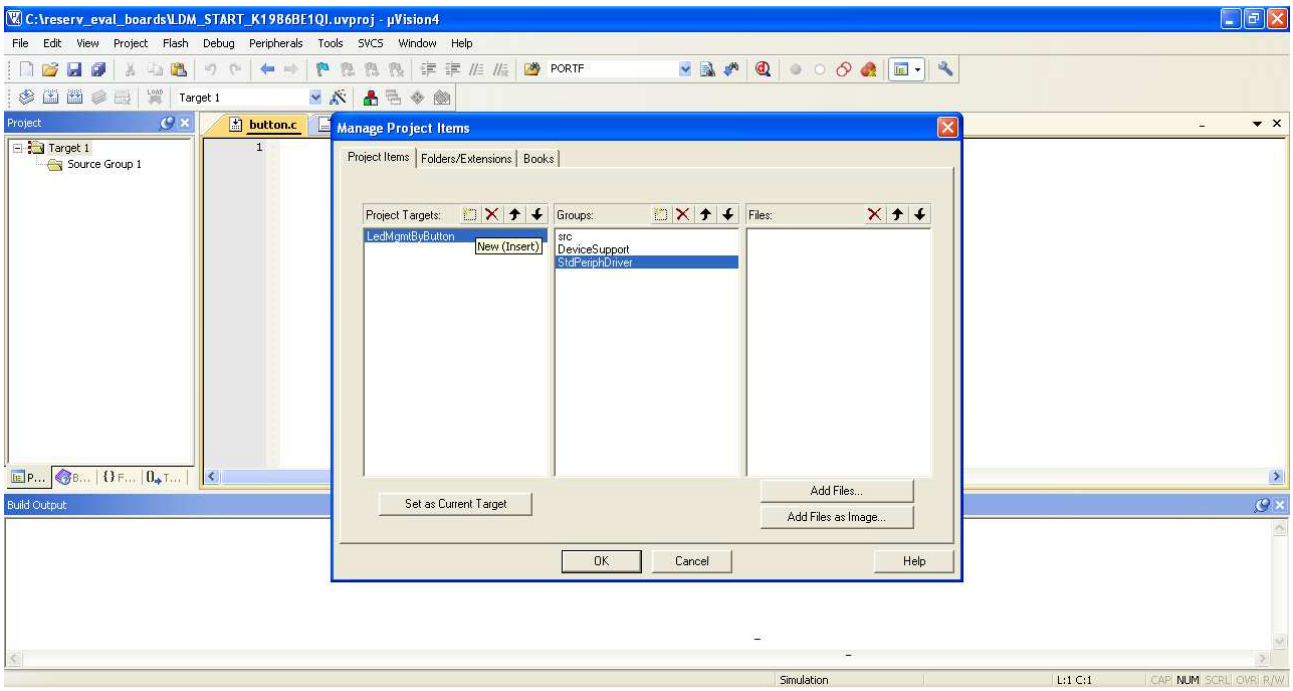
В настройках компилятора «C/C++» указываем пути «Include Paths» к компонентам библиотек CMSIS (Cortex Microcontroller Software Interface Stardard) и Standard Peripheral Library MDR32F9x:

..\..\lib\CMSIS\CM1\CoreSupport;
..\..\lib\CMSIS\CM1\DeviceSupport\MDR1986VE1T\inc;
..\..\lib\CMSIS\CM1\DeviceSupport\MDR1986VE1T\startup\arm;
..\..\lib\MDR32F9Qx_StdPeriph_Driver\inc;

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, 1986BE4У, 1986BE8T)→Standard Peripherals Library MDR32F9x, VE1, VE3, VE4, VC1

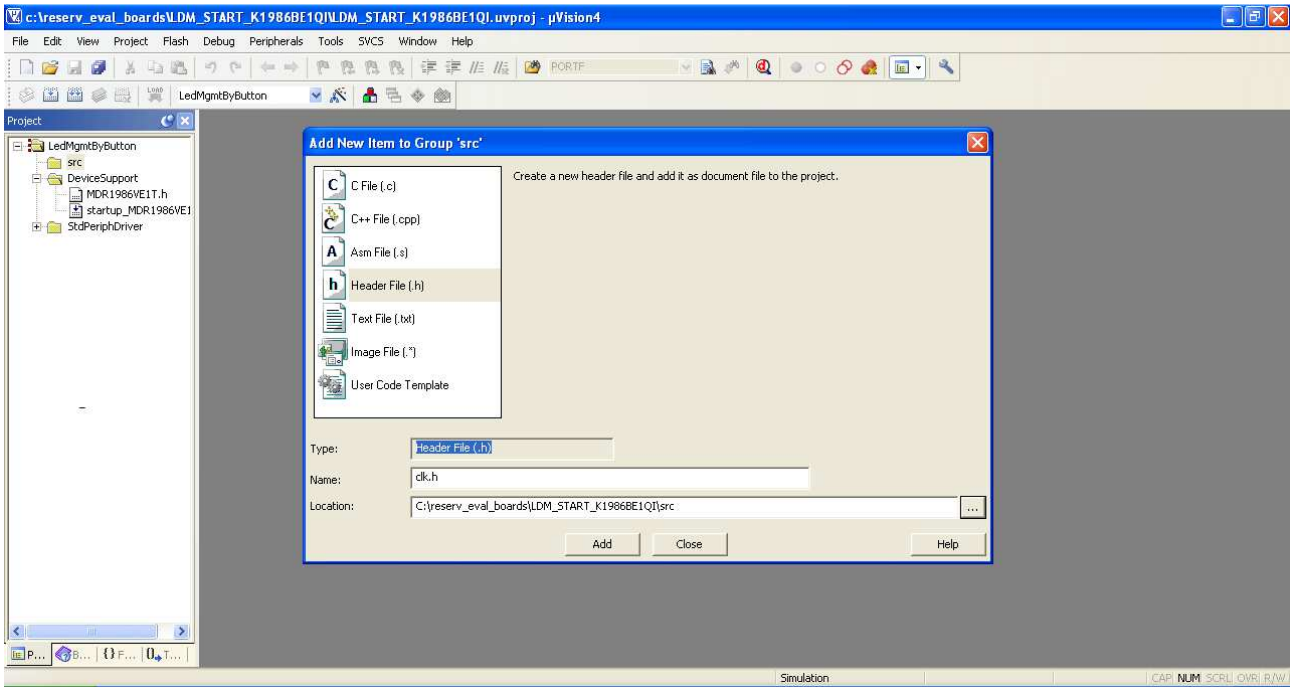
Компоненты библиотеки в папках *CMSIS* и *MDR32F9Qx_StdPeriph_Driver* после загрузки необходимо разместить по пути C:\LDM_START_K1986BE1QI\ в папке lib.

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 MDR1986VE1T.h (определение регистров МК) и startup_MDR1986VE1.s (настройки старта, адресного пространства и векторов прерываний). Нажимаем «OK».


C:\reserv_eval_boards\LDM_START_K1986BE1QI.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PORTF
Target 1
Project
Target 1
Source Group 1
button.c
Manage Project Items
Project Items
Folders/Extensions
Books
Project Targets:
Groups:
Files:
LedMgmtByButton
New (Insert)
src
DeviceSupport
StdPeriphDriver
Set as Current Target
Add Files...
Add Files as Image...
OK
Cancel
Help
Build Output
Simulation
L:1 C:1

1.2. Написание функционального кода примера по отображению состояния кнопки SA3 светодиодом VD2 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 коментариями представлены ниже. Приведенный код производит подключение и настройку внешнего тактирования микропроцессора.



Листинг 1. файл clk.h

```
#ifndef __CLK_H__
#define __CLK_H__

#include "MDR1986VE1T.h"
#include "MDR32F9Qx_rst_clk.h"

//Сигнатура функции настройки тактовой частоты МК
void clk_CoreConfig(void);

#endif
```

Листинг 2. 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 PORTA
        RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTA, ENABLE);
}
```

Необходимо отметить, что в Листинге 2 файла clk.c производится настройка тактирования микропроцессора через внешний источник (кварцевый реонатор HSE), а также включение тактирвания периферийных модулей, в нашем случае порта «А». Для более подробного ознакомления открываем раздел «Сигналы тактовой частоты» в спецификации на серию микроконтроллеров 1986BE1T, К1986BE1T, К1986BE1H4 (спецификация расположена на сайте производителя).

На следующем этапе требуется настроить порты ввода-вывода для работы с кнопкой SA3 и светодиодом VD2. При этом модули работы с кнопокой и светодиодом опишим в разных файлах. Для программного описания функционала работы с кнопкой потребуется создать заголовочный файл button.h (Листинг 3), где объявлены функции инициализации и считвания состояния кнопки SA3. 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 "MDR32F9Qx_port.h"

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

Листинг 4. файл button.c

```
#include "button.h"

//Функция инициализации кнопки SA3
void button_Init(void)
{
        //Создание структуры для инициализации порта
        PORT_InitTypeDef PORT_InitStructure;

        //Настройки порта: ввод, функция ввода/вывода, цифровой режим,
минимальная скорость, Pin3
        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_3);

        PORT_Init(MDR_PORTA, &PORT_InitStructure);
}

//Функция считывания текущего состояния кнопки SA3
uint8_t button_State(void)
{
        return PORT_ReadInputDataBit(MDR_PORTA, PORT_Pin_3);
}
```

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 96 – Порты ввода-вывода

Вывод	Аналоговая функция	Цифровая функция							
		Порт IO	Основная		Альтернативная		Переопределенная		
	ANALOG_EN=0	MODE[1:0]=00 ANALOG_EN=1	MODE[1:0]=01 ANALOG_EN=1		MODE[1:0]=10 ANALOG_EN=1		MODE[1:0]=11 ANALOG_EN=1		
Порт A									
PA0	-	PA0	D0	1	EXTINT1	8	ETR1	3	
PA1	-	PA1	D1		EXTINT2		ETR2	13	
PA2	-	PA2	D2		EXTINT3		ETR3	10	
PA3	-	PA3	D3		EXTINT4		BRK1	3	
PA4	-	PA4	D4		BRK2	13	FRX	15	
PA5	-	PA5	D5		BRK3	10	FSD		
PA6	-	PA6	D6		TMR4_CH1	16	FXEN		
PA7	-	PA7	D7		TMR4_CH1N		FTX		
PA8	-	PA8	D8		TMR4_CH2		PRMC+	4	
PA9	-	PA9	D9		TMR4_CH2N		PRMC-		
PA10	-	PA10	D10		TMR4_CH3		PRMD+		
PA11	-	PA11	D11		TMR4_CH3N		PRMD-		
PA12	-	PA12	D12		TMR4_CH4		PRDC+		
PA13	-	PA13	D13		TMR4_CH4N		PRDC-		
PA14	-	PA14	D14		BRK4		PRDD+		
PA15	-	PA15	D15		ETR4		PRDD-		
Порт B									
PB0	-	PB0	D16	1	IN1+	2	TMR3_CH1	10	
PB1	-	PB1	D17		IN1-		TMR3_CH1N		
PB2	-	PB2	D18		IN2+		TMR3_CH2		
PB3	-	PB3	D19		IN2-		TMR3_CH2N		
PB4	-	PB4	D20		IN3+		TMR3_CH3		
PB5	-	PB5	D21		IN3-		TMR3_CH3N		
PB6	-	PB6	D22		IN4+		TMR3_CH4		
PB7	-	PB7	D23		IN4-		TMR3_CH4N		
PB8	-	PB8	D24		IN5+		TMR1_CH1N	3	
PB9	-	PB9	D25		IN5-		TMR2_CH1N	13	
PB10	-	PB10	D26		IN6+		TMR1_CH2N	3	
PB11	-	PB11	D27		IN6-		TMR2_CH2N	13	
PB12	-	PB12	D28		IN7+		TMR1_CH3N	3	
PB13	-	PB13	D29		IN7-		TMR2_CH3N	13	
PB14	-	PB14	D30		IN8+		TMR1_CH4N	3	
PB15	-	PB15	D31		IN8-		TMR2_CH4N	13	
Порт C									
PC0	-	PC0	nWR	1	ETR1	3	BRK1	3	
PC1	-	PC1	nRD		ETR2	13	BRK2	13	
PC2	-	PC2	ALE		CLKO	1	BRK3	10	
PC3	-	PC3	UART_TXD1	9	CLE		SIROUT0	9	
PC4	-	PC4	UART_RXD1		BUSY		SIRIN0		
PC5	-	PC5	EXTINT1	8	SSP1_TXD	14	SSP1_RXD	14	
PC6	-	PC6	EXTINT2		SSP1_RXD		SSP1_TXD		
PC7	-	PC7	EXTINT3		SSP1_SCK		FXEN	15	
PC8	-	PC8	EXTINT4		SSP1_FSS		FTX		
PC9	-	PC9	SSP2_TXD	11	BE0	1	CAN_RX1	17	
PC10	-	PC10	SSP2_RXD		BE1		CAN_TX1		
PC11	-	PC11	SSP2_SCK		BE2		CAN_RX2	18	
PC12	-	PC12	SSP2_FSS		BE3		CAN_TX2		
PC13	-	PC13	PRMA+	4	A30	1	UART_TXD2	12	
PC14	-	PC14	PRMA-		A31		UART_RXD2		
PC15	-	PC15	PRMB+		BUSY		TMR2_CH1	13	
Порт D									
PD0	-	PD0	PRMB-	4	ALE	1	A16	1	
PD1	-	PD1	PRDA+		CLE		A15		
PD2	-	PD2	PRDA-		SSP1_TXD	14	A14		
PD3	-	PD3	PRDB+		SSP1_RXD		A13		
PD4	-	PD4	PRDB-		SSP1_SCK		A7		
PD5	-	PD5	PRD_PRMA		SSP1_FSS		A6		
PD6	-	PD6	PRD_PRMB		nUART2RI	12	A5		
PD7	ADC0_REF+	5 PD7	SSP2_TXD	11	nUART2DCD		A4		
PD8	ADC1_REF-	PD8	SSP2_RXD		nUART2DTR		A3		
PD9	ADC2	PD9	SSP2_SCK		nUART2DSR		A2		
PD10	ADC3	PD10	SSP2_FSS		nUART2RTS		A1		
PD11	ADC4	PD11	A0	1	nUART2CTS		FRX	15	
PD12	ADC5	PD12	SSP3_TXD	19	ETR3	10	SSP3_RXD	19	
PD13	ADC6	PD13	UART_TXD2	12	OUT1+	2	SIROUT1	12	
PD14	ADC7	PD14	UART_RXD2		OUT1-		SIRIN1		
PD15	DAC1_REF	6 PD15	OUT3+	2	A13	1	FSD	15	
Порт E									
PE0	DAC2_REF	6 PE0	OUT4+	2	A14	1	MDC	15	
PE1	DAC1_OUT	PE1	OUT3-		A15		nUART2RI	12	
PE2	DAC2_OUT	PE2	OUT4-		A16		MDIO	15	
PE3	-	PE3	TMR1_CH1	3	A17		TXD[0]		
PE4	-	PE4	TMR1_CH2		A18		TXD[1]		
PE5	-	PE5	TMR1_CH3		A19		TXD[2]		
PE6	OSC_IN32	7 PE6	TMR1_CH4		A20		TXD[3]		
PE7	OSC_OUT32	PE7	TMR2_CH1	13	A21		RXD[0]		
PE8	-	PE8	TMR2_CH2		A22		RXD[1]		
PE9	-	PE9	TMR2_CH3		A23		RXD[2]		
PE10	-	PE10	TMR2_CH4		A24		RXD[3]		
PE11	-	PE11	CAN_RX1	17	A25		TXEN		
PE12	-	PE12	CAN_TX1		A26		TXER		
PE13	-	PE13	CAN_RX2	18	A27		TXCLK		
PE14	-	PE14	CAN_TX2		A28		RXCLK		
PE15	-	PE15	PRD_PRMD	4	A29		RXDV		
Порт F									
PF0	OSC_IN25	PF0	PRD_PRMA	4	READY	1	RXER	15	
PF1	OSC_OUT25	PF1	PRD_PRMB		A30		CRS		
PF2	-	PF2	READY/PRD_PRMC	1/4	A31		COL		
PF3	-	PF3	PRMC+	4	A0		TMR1_CH1	3	
PF4	-	PF4	PRMC-		A1		TMR1_CH2		
PF5	-	PF5	PRMD+		A2		TMR1_CH3		
PF6	-	PF6	PRMD-		A3		TMR1_CH4		
PF7	-	PF7	PRDC+		A4		OUT4+	2	
PF8	-	PF8	PRDC-		A5		OUT4-		
PF9	-	PF9	PRDD+		A6		OUT3+		
PF10	-	PF10	PRDD-		A7		OUT3-		
PF11	-	PF11	PRD_PRMC		A8		OUT2+		
PF12	-	PF12	PRD_PRMD		A9		OUT2-		
PF13	-	PF13	OUT2+	2	A10		SSP3_FSS	19	
PF14	-	PF14	OUT2-		A11		SSP3_SCK		
PF15	-	PF15	SSP3_RXD	19	A12		SSP3_TXD		

Из схемотехнического документа для отладочного комплекта (доступного на сайте производителя https://ldm-systems.ru/product/21003, Схема LDM-START-K1986BE1QI.pdf) следует, что для взаимодействия с кнопкой потребуется вывод PA3 порта «A». 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 потребуется функция API драйвера PORT_ReadInputDataBit, где в качестве входных параметров передается номер вывода и порт (PA3, PORTA).

Для настройки светодиода необходимо добавить в проект файлы led.h и led.c. По аналогии с описанием порта ввода-вывода кнопки производим конфигурирование вывода PA4 порта «A» для светодиода .

Листинг 5. файл led.h

```
#ifndef __LED_H__
#define __LED_H__

#include "type_custom.h"
#include "MDR32F9Qx_port.h"

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

```
#endif
```

Листинг 6. файл led.c

```
#include "led.h"

//Функция инициализации светодиода VD2
void led_Init(void)
{
        //Создание структуры для инициализации порта
        PORT_InitTypeDef PORT_InitStructure;

        //Настройки порта: вывод, функция ввода/вывода, цифровой режим,
максимальная скорость, Pin4
        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_4);

        PORT_Init(MDR_PORTA, &PORT_InitStructure);
}

//Функция записи состояния (1:0) светодиода VD2
void led_Write(bool on_off)
{
        PORT_WriteBit(MDR_PORTA, PORT_Pin_4, 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 VD2 потребуется функция API драйвера PORT_WriteBit, где в качестве входных параметров передается номер вывода, порт, а также состояние вывода 1 или 0 (PA4, PORTA, state).

Для определения пользовательских числовых и булевых типов потребуется их описание в заголовочном файле type_custom.h.

Листинг 7. файл type_custom.h.

```
#ifndef __TYPE_CUSTOM_H__
#define __TYPE_CUSTOM_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
```

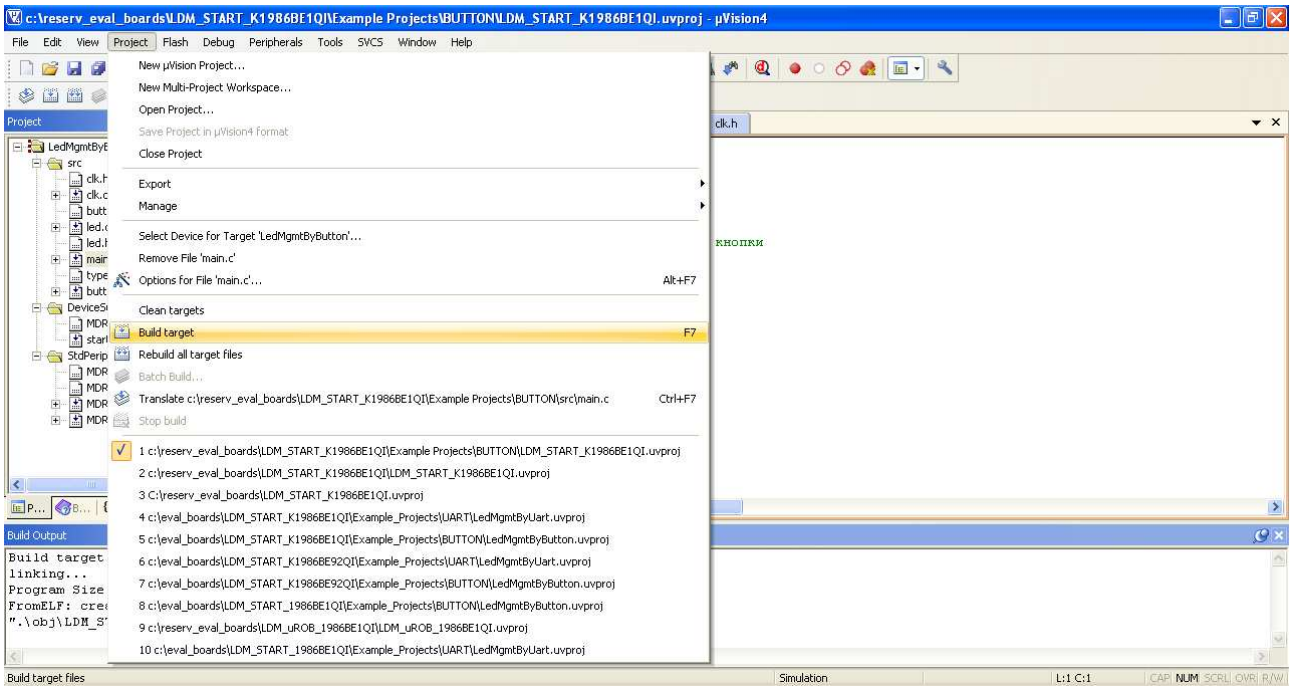
После этого можно переходить к созданию файла main.c, который является основной точкой входа для работы программы микроконтроллера. Данный файл должен содержать основной алгоритм работы программы. Сперва, необходимо подключить заголовочные файлы ранее созданных модулей «led.h», «button.h», «clk.h». В теле main требуется инициализировать программные модули тактирования, кнопки и светодиода. Затем в бесконечном цикле while(1)

добавляем чтение состояния вывода (PA3) порта (PORTA) кнопки SA3 и его дальнейшую запись на вывод (PA4) порта (PORTA) светодиода VD2.

Листинг 8. файл main.c

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

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

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

  //Алгоритм работы программы в «вечном цикле»
  while (1){
                //Считывание текущего состояния кнопки SA3
                state = button_State();
                //Запись состояния кнопки на вывод светодиода VD2
                led_Write(!state);
        }
}
```

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



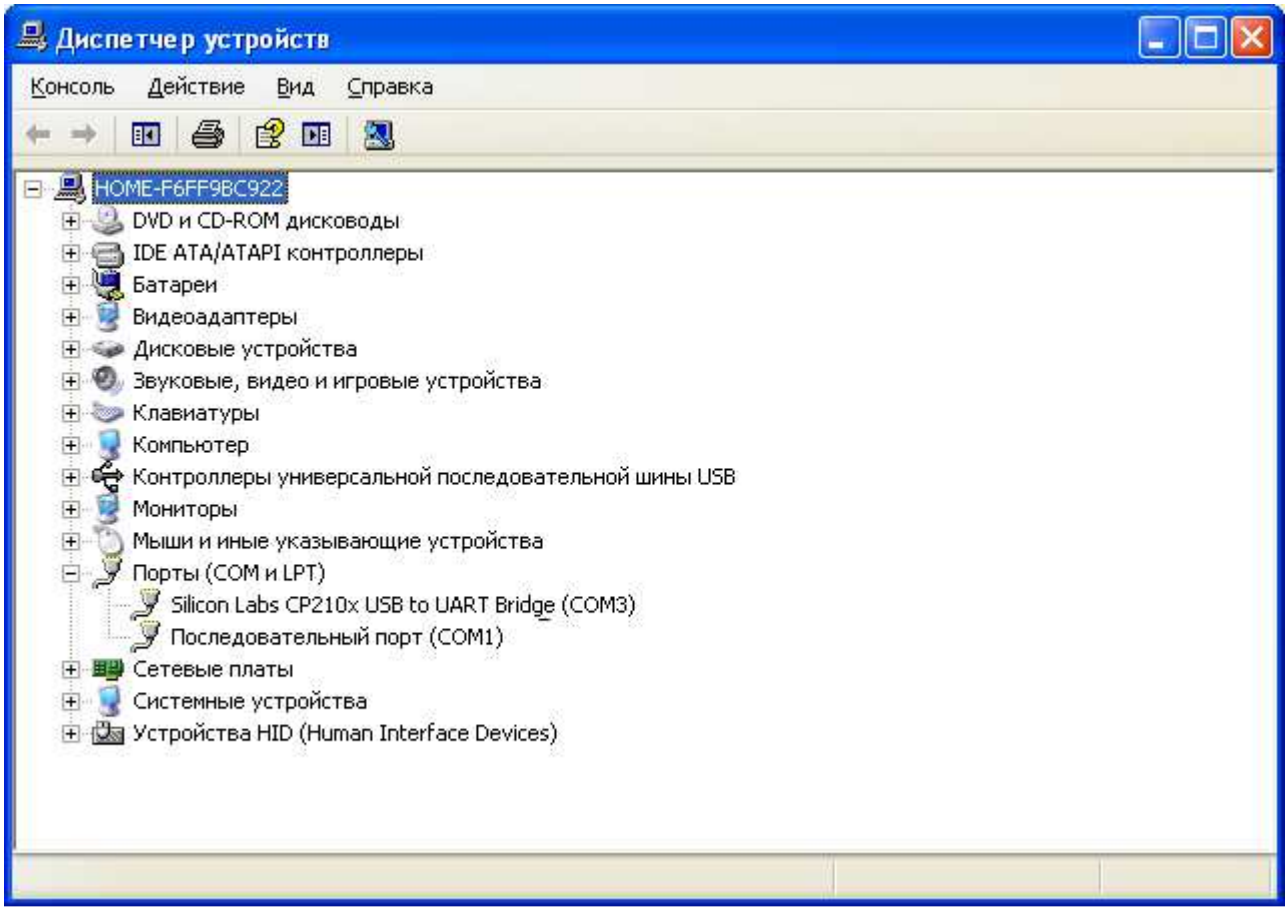
Можно увидеть, что в папке проекта, в случае отсутствия ошибок или предупреждений в окне «Build Output», в директории obj появился файл с расширением .hex.

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*

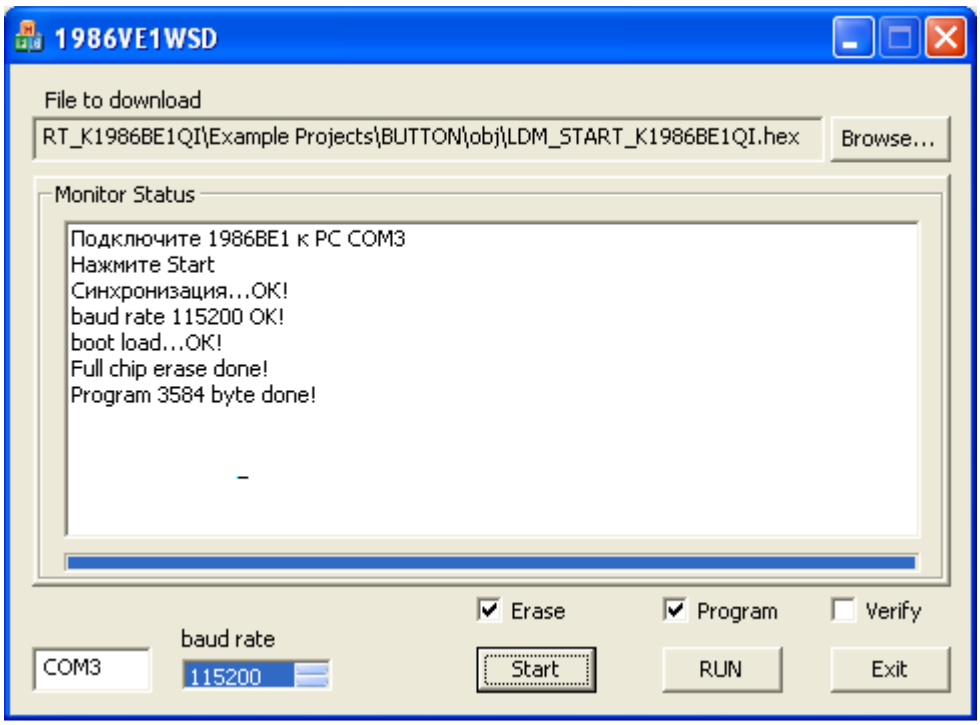
Перед тем, как запустить программу, необходимо установить на компьютер драйвер для микросхемы CP2102. Скачать драйвера можно непосредственно с сайта производителя, компании «Silicon Labs» (http://www.silabs.com).

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 1-3 в положение OFF («0»), а 4 в положение ON («1»). Отключение питания подключенной к USB платы производится по кнопке SA1. Установка «1» на входе M2 производится по кнопке SA5. Таким образом, для получения режима загрузки BOOT (MODE) микроконтроллера M2..M0 = «100» при котором происходит загрузка по UART, необходимо зажать SA5 и коротким нажатием на SA1 сбросить питание, затем отжать SA5 . Подробнее о режимах загрузки микроконтроллера написано в спецификации на микросхему.

Запускаем программу UART-загрузчик «1986VE1WSD.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 окончании операции загрузки.



Последовательность действий для загрузки прошивки в микроконтроллер:

- ✓ Вводим в режим BOOT MODE путем зажатия SA5 и одновременного сброса питания коротким нажатием SA1
- ✓ Запускаем программу 1986VE1WSD.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 SA3, после чего должен загореться светодиод VD2.

Пример 2
Управление светодиодом VD2 по UART1

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 VD2 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 «C» и на UART1 надо дописать строки:

```
//Подача тактовой частоты на порт PORTC
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC, ENABLE);
//Подача тактовой частоты на UART1
RST_CLK_PCLKcmd(RST_CLK_PCLK_UART1, 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 PORTA, PORTC
        RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTA, ENABLE);
        RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC, ENABLE);
        //Подача тактовой частоты на UART1
        RST_CLK_PCLKcmd(RST_CLK_PCLK_UART1, ENABLE);
}
```

Далее для работы с UART1 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 UART1, а также процедура приема байта и последущей интерпритации его в команду.

Листинг 2. файл uart.h

```
#ifndef __UART_H__
#define __UART_H__

#include "MDR32F9Qx_port.h"
#include "MDR32F9Qx_uart.h"

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

В коде инициализации UART1 (Листинг 3) присутствует процедура uart_Init, которая настраивает и инициализирует структурами порт и непосредственно сам интерфейс последовательного приемопередатчика UART. Для настройки выводов порта UART1 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 PC3 настраивается на выход т.к. выполняет функцию передатчика TX, PC4 — на вход, т.к. выполняет функцию приемника RX. Для обеспечения требуемых скоростей блока UART требуется настроить предделитель 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 UART1
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_MAIN;
	PortInit.PORT_SPEED = PORT_SPEED_MAXFAST;
	PortInit.PORT_MODE = PORT_MODE_DIGITAL;
```

```
        //Настойка PORTC pins 3 на вывод (UART1_TX)
        PortInit.PORT_OE = PORT_OE_OUT;
        PortInit.PORT_Pin = PORT_Pin_3;
        PORT_Init(MDR_PORTC, &PortInit);

        //Настойка PORTC pins 4 на ввод (UART1_RX)
        PortInit.PORT_OE = PORT_OE_IN;
        PortInit.PORT_Pin = PORT_Pin_4;
        PORT_Init(MDR_PORTC, &PortInit);

        //Установка предделителя частоты UART1 HCLKdiv = 16
        UART_BRGInit(MDR_UART1, UART_HCLKdiv16 );

        //Настройка параметров UART1: 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 UART1
        UART_Init(MDR_UART1, &UART_InitStructure);

        //Команда включения UART1
        UART_Cmd(MDR_UART1, ENABLE);
}

//Функция приема байта по UART1
uint8_t uart_Work(void)
{
        uint16_t data;
        uint8_t res = 0xFF;

        if(UART_GetFlagStatus(MDR_UART1, UART_FLAG_RXFE) != SET)
        {

                data = UART_ReceiveData(MDR_UART1);

                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 VD2. 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 VD2. После инициализации периферийных модулей тактирования, светодиода, приемопередатчика UART происходит переход в основной бесконечный цикл программы, где происходит постоянный опрос регистра UART1 и при

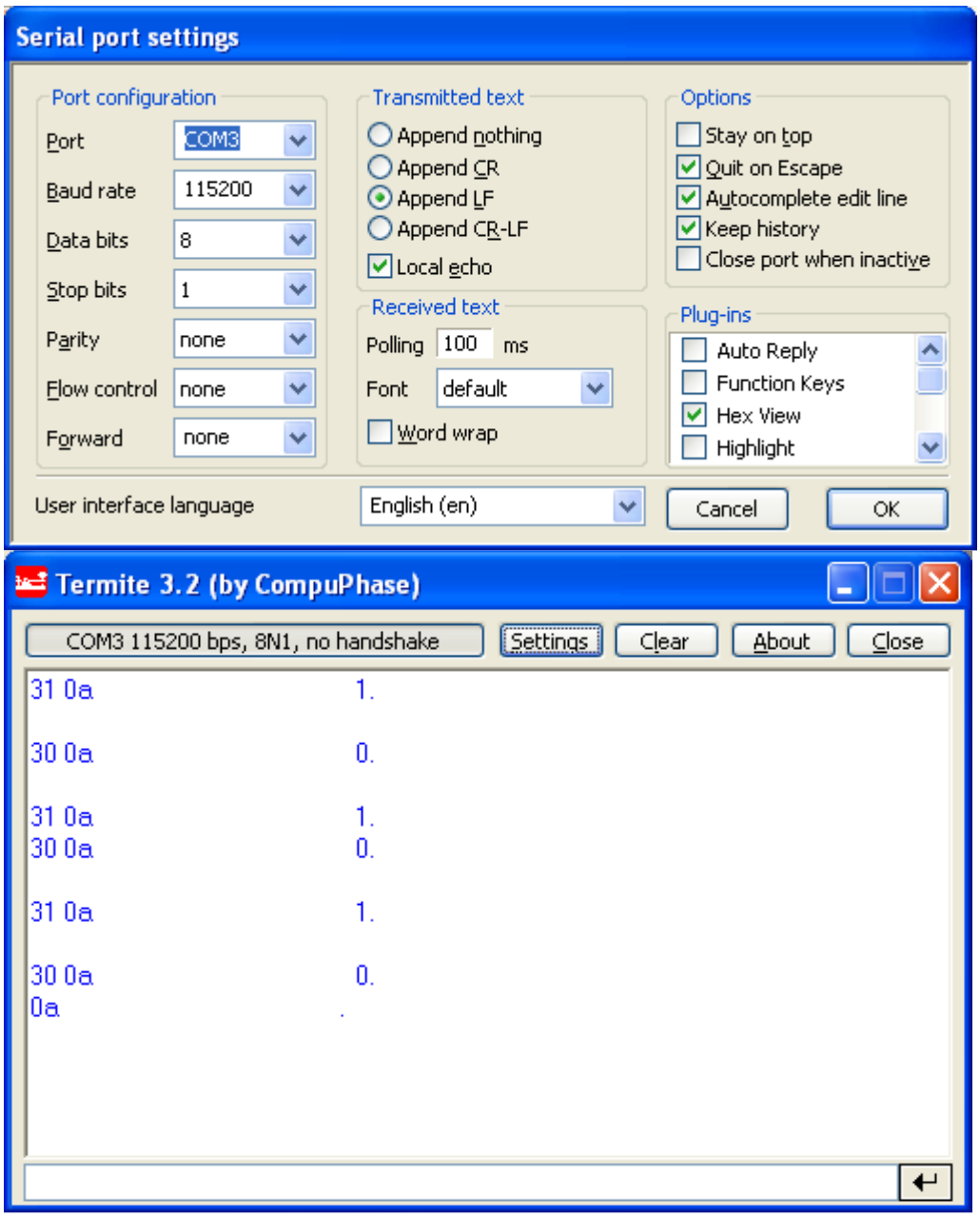
наличии данных — чтение и интерпритация кода ASCII в код команды для записи состояния (1:0) на выходной вывод PA4 порта «A» светодиода VD2.

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 VD2
                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 «1986VE1WSD.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.compuphase.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-START-K1986BE1QI, 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 VD2. Состояние должно меняться в соответсвии с отправленными командами.


Serial port settings
Port configuration
Port
COM3
Baud rate
115200
Data bits
8
Stop bits
1
Parity
none
Flow control
none
Forward
none
Transmitted text
Append nothing
Append CR
Append LF
Append CR-LF
Local echo
Received text
Polling
100
ms
Font
default
Word wrap
Options
Stay on top
Quit on Escape
Autocomplete edit line
Keep history
Close port when inactive
Plug-ins
Auto Reply
Function Keys
Hex View
Highlight
User interface language
English (en)
Cancel
OK
Termite 3.2 (by CompuPhase)
COM3 115200 bps, 8N1, no handshake
Settings
Clear
About
Close
31 0a 1.
30 0a 0.
31 0a 1.
30 0a 0.
31 0a 1.
30 0a 0.
0a .