**Лабораторная работа № 2**

**Цель работы:** Изучение основных особенностей программирования для микроконтроллеров (МК) ARM на примере программы мигания светодиодами.

**Приборы и материалы:**

1. Отладочная плата MDR1986VE91T Rev 4
2. Программатор J-Link ARM
3. Блок питания 5В, 1.4А
4. ПК с установленной средой программирования Keil uVision

**Ход работы**

1. Подключить программатор к порту JTAG-B платы (рис. 1).
2. Установить переключатели SW1, SW2 и SW3 в положение 0 (рис. 1).
3. Подключить блок питания к плате (рис. 1).

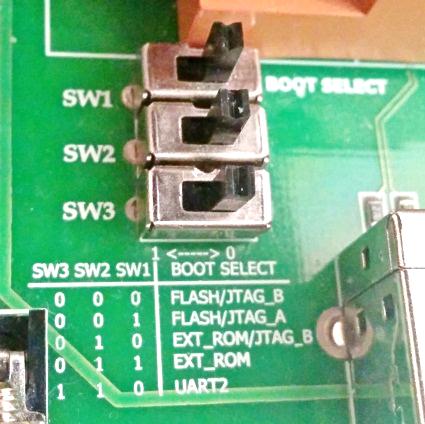
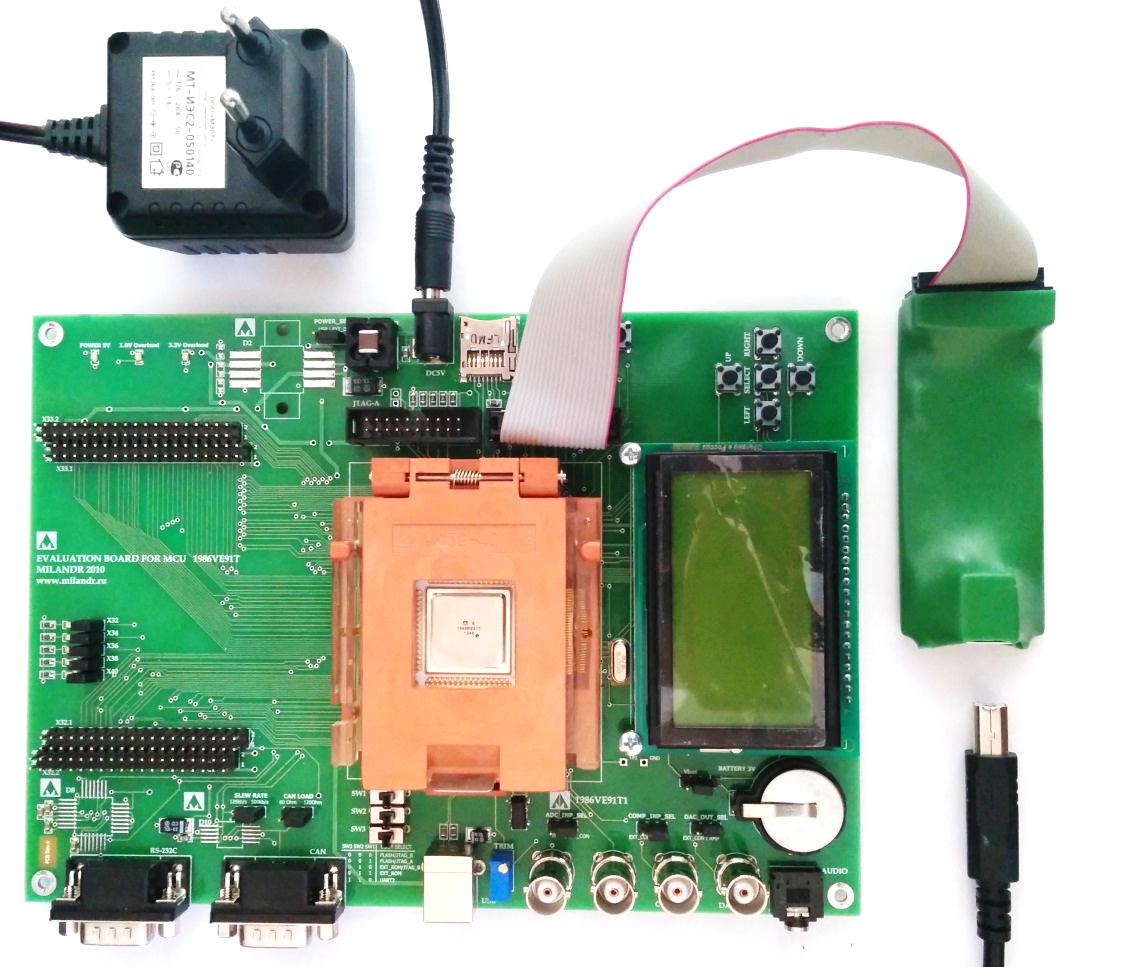


Рисунок 1 – Подключение платы 1986VE91T

1. Подключить программатор J-Link к USB порту компьютера и дождаться окончания установки драйверов.
2. Открыть проект MDRProject в среде программирования Keil uVision.
3. Скомпилировать проект, нажав кнопку «*Build*». При успешной компиляции, в окне **Build Output** появится надпись *«0 Error(s), 0 Warnings»* (рисунок 2).

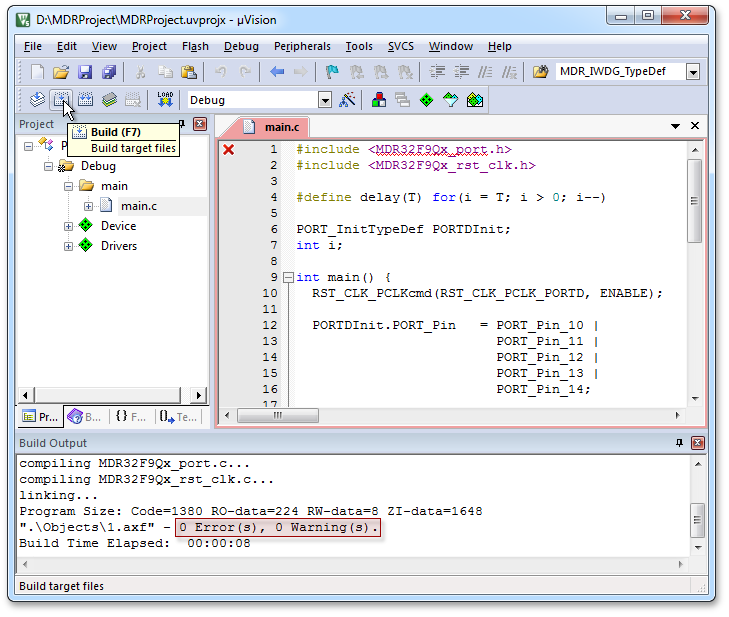


Рисунок 2 – Компиляция проекта

1. Подать питание на плату, вставив блок питания в сеть 220В.
2. Загрузить микропрограмму на микроконтроллер с помощью кнопки *Download* (рисунок 3).
3. При первой отладке, драйвер программатора J-Link выдаст уведомление о том, что устройство “MDR1986BE91” ему не известно и предложит выбрать устройство вручную. Чтобы проигнорировать уведомление, нажмите кнопку «No» (рисунок 4). Успешная загрузка микропрограммы обозначается строчкой «*Verify OK*» в окне **Build Output**.

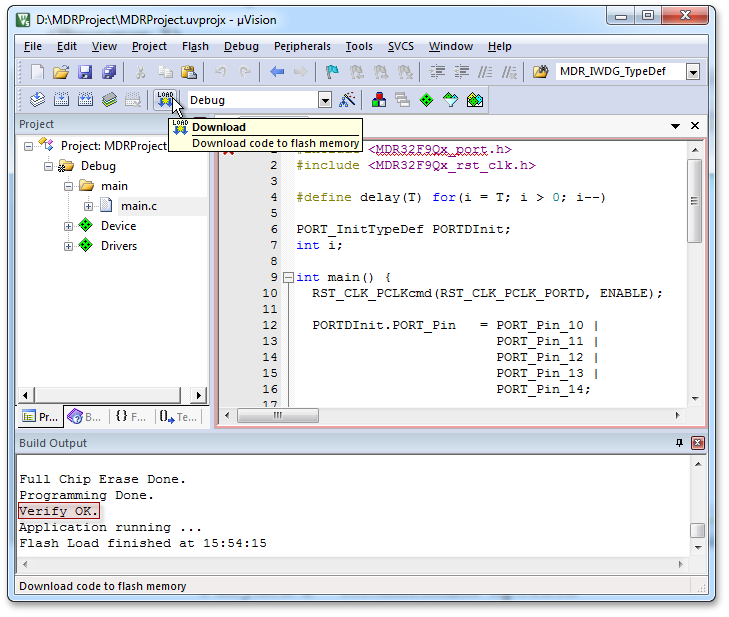


Рисунок 3 – Загрузка микропрограммы в устройство

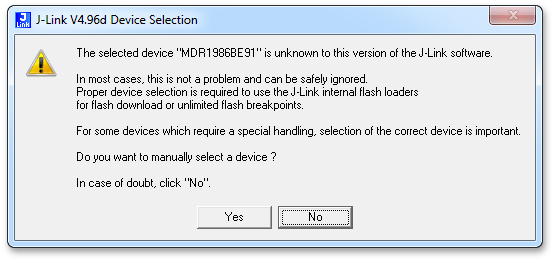


Рисунок 4 – Уведомление о неизвестном устройстве

1. Если программатору не удалось загрузить микропрограмму, попробуйте перевести переключатель SW2 (рисунок 1) в положение 1, активировав режим загрузки с внешнего носителя данных (EXT\_ROM/JTAG\_B) и еще раз нажать кнопку *Download*. При успешной загрузке микропрограммы, необходимо вернуть SW2 в положение 0 и перезагрузить микроконтроллер. Микропрограмма должна начать исполняться и мигать светодиодами VD7 - VD11.

**Теоретические сведения**

Микроконтроллер ­– это программно-аппаратный комплекс, позволяющий решать определенный круг узкоспециальных задач без применения компьютера. По сути, МК – это полноценный компьютер с ПЗУ, ОЗУ, интерфейсами ввода-вывода информации и, разумеется, процессором. Однако, в отличие от компьютера, МК обычно не содержит операционной системы и программируется пользователем на исполнение одного необходимого алгоритма. Обычно, этот алгоритм производит управление одним или несколькими устройствами с совершенной произвольной целью. Это может быть сбор данных с нескольких датчиков, поддержание определенного режима чего-либо на основе собранных данных, микроконтроллер может организовывать интерфейс между оконечной аппаратной частью устройства (например, рулевыми машинками самолета) и высокоуровневыми устройствами, например, смартфоном. Область применения МК ограничена только идеями разработчика.

В данном методическом комплексе, будет рассмотрена задача использования самых важных элементов 32-разрядного микроконтроллера на базе процессорного ядра ARM Cortex-M3 (обычно используется в смартфонах) компании Миландр – **1986ВЕ91**. Структурная схема МК приведена на рисунке 5. Более подробную информацию можно узнать из официальной спецификации для данной серии МК

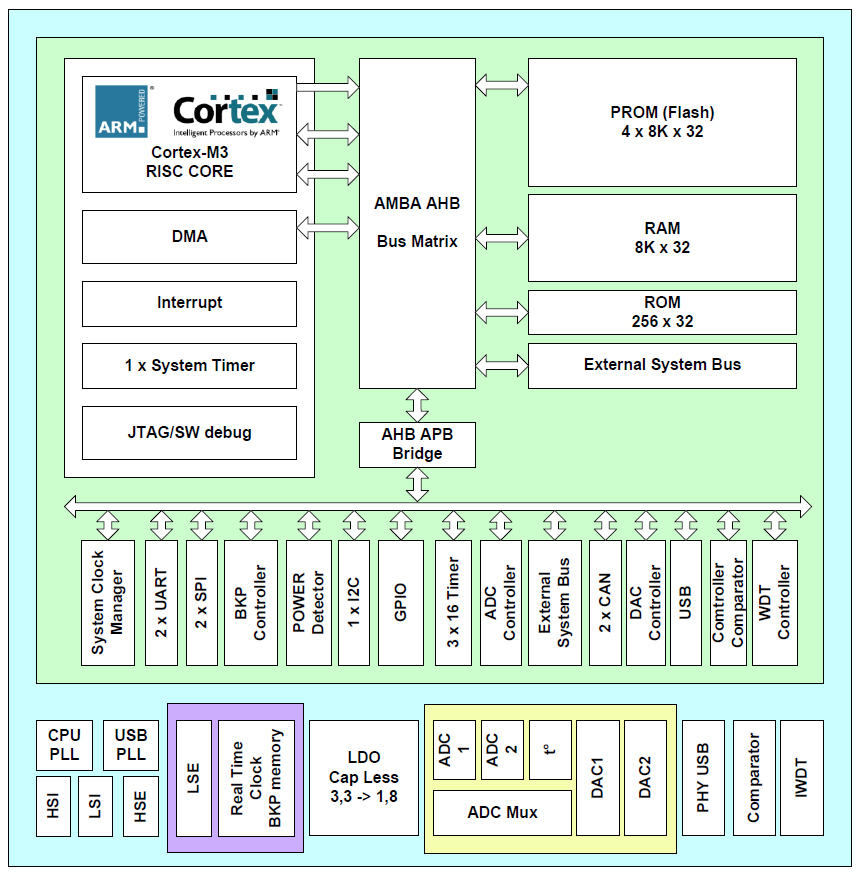


Рисунок 5 – Структурная блок-схема микроконтроллера 1986ВЕ9х

Программирование для микроконтроллеров обычно производится на языке Си, однако, при создании микропрограмм для МК архитектуры ARM, повсеместно используются библиотеки, позволяющие отвязать код от конкретного МК и значительно улучшить его переносимость. Библиотеки дают возможность управлять функциями МК с помощью инвариантного программного кода на разных устройствах. Каждая функция МК содержится в своей библиотеке, и они подключаются по мере необходимости.

Для управления подключенными к проекту библиотеками, необходимо воспользоваться кнопкой «*Manage Run-Time Environment*» (рисунок 5). Минимальный набор необходимых библиотек состоит из основной системной библиотеки **Device** / **Startup\_MDR1986E9x** и библиотеки управления тактованием **Device** / **RST\_CLK**. Однако, используя только функции этих двух библиотек, можно лишь включить МК, а чтобы что-либо сделать с его помощью, понадобится еще хотя бы одна библиотека, например, **Device** / **PORT** для управления портами ввода/вывода. Таким образом, видеокадр окна с подключенными библиотеками начального проекта **MDRProject** представлен на рисунке 5.

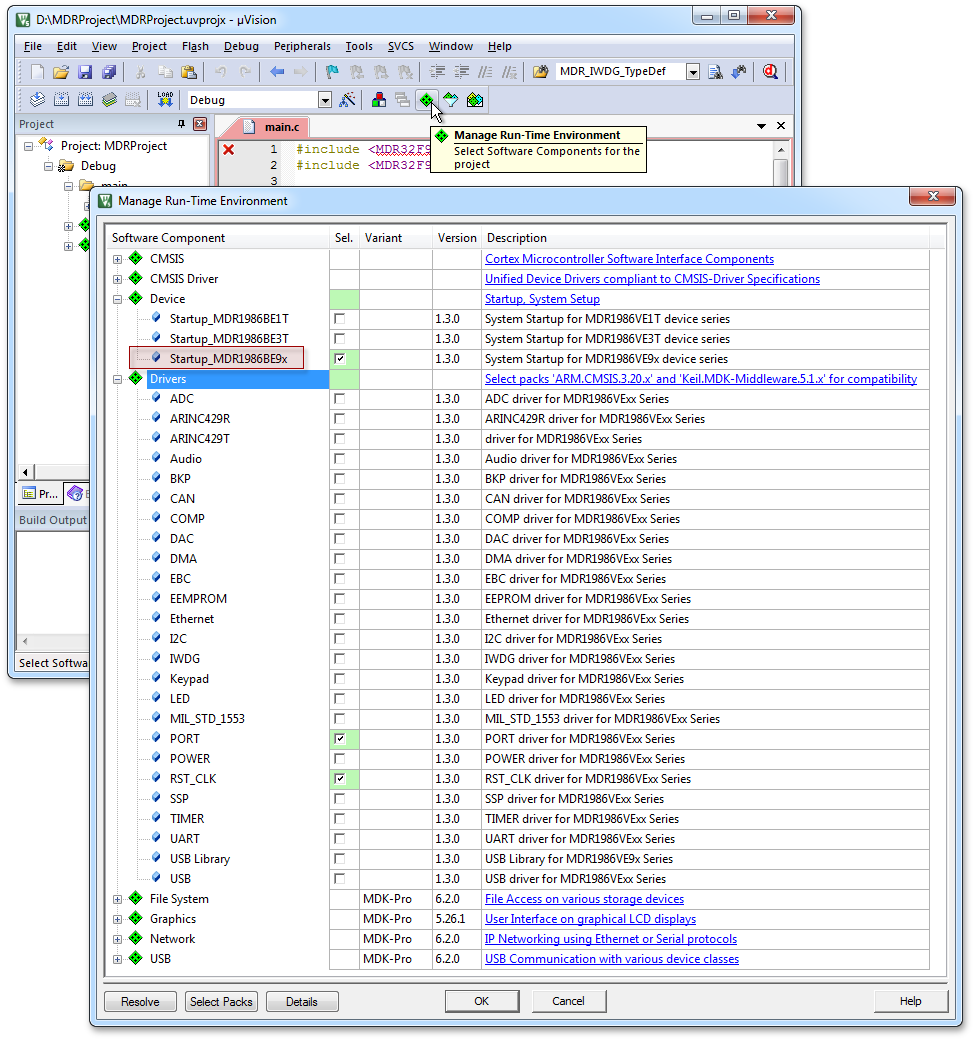


Рисунок 5 – Менеджер библиотек проекта

Рассмотрим программный код, позволяющий мигать светодиодами.

#include <MDR32F9Qx\_port.h>

#include <MDR32F9Qx\_rst\_clk.h>

**// Подключение заголовочных файлов тех библиотек,**

**// которые непосредственно используются в данном файле исходного кода**

#define delay(T) for(i = T; i > 0; i--) **// Определение функции задержки (см. примечение 1)**

PORT\_InitTypeDef PORTDInit; **// Объявление структуры, с помощью которой**

**// будет происходить инициализация порта (см. примечение 2)**

int i; **// Глобальная переменная счетчика, которая используется в функции delay()**

int main() { **// Главная функция, с которой начинается работа программы**

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTD, ENABLE); **// Включение тактования порта D**

PORTDInit.PORT\_Pin = PORT\_Pin\_10 |

PORT\_Pin\_11 |

PORT\_Pin\_12 |

PORT\_Pin\_13 | **// Объявляем пины, которые**

PORT\_Pin\_14;  **// настраиваются данной структурой**

PORTDInit.PORT\_OE = PORT\_OE\_OUT; **// Конфигурация группы выводов как выход**

PORTDInit.PORT\_FUNC = PORT\_FUNC\_PORT; **// Работа а режиме порта ввода-вывода**

PORTDInit.PORT\_MODE = PORT\_MODE\_DIGITAL; **// Цифровой режим**

PORTDInit.PORT\_SPEED = PORT\_SPEED\_SLOW; **// Низкая частота тактования порта**

PORT\_Init(MDR\_PORTD, &PORTDInit); **//Инициализация порта D объявленной структурой**

while(1){ **//Главный цикл (см. примечение 3)**

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_10); **// Установка единицы на 10 пине в порту D**

delay(0xFFFF); **// Задержка (см. примечение 1)**

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_10); **// Установка нуля на 10 пине в порту D**

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_11);

delay(0xFFFF);

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_11);

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_12);

delay(0xFFFF);

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_12);

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_13);

delay(0xFFFF);

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_13);

PORT\_SetBits(MDR\_PORTD, PORT\_Pin\_14);

delay(0xFFFF);

PORT\_ResetBits(MDR\_PORTD, PORT\_Pin\_14);

}

}

**Примечание 1**

Поскольку, частота работы процессора очень высокая (8 млн. тактов в секунду), глаз не сможет различить мигание светодиода, если написать строчки, включающие и выключающие светодиод подряд. Для формирования задержки, в простейшем варианте используют пустой цикл с огромным числом итераций. Недостаток данного способа в том, что пока процессор занят отсчетом итераций, он не может делать что-то еще. Однако, классический вид первой программы для МК имеет именно такой вид. Как и программа, которая выводит на экран надпись “Hello World”, мигание светодиодом с задержкой через цикл не имеет практического применения, но позволяет быстро создать хоть что-то работающее.

**Примечание 2**

При программировании микроконтроллеров ARM с использованием библиотек, повсеместно применяется такой элемент языка Си как структура. Структура похожа на массив, но каждый элемент структуры имеет имя вместо номера, а их количество строго определено. Структура в некоторой степени похож на объект класса, у которого есть только поля, и нет методов. Синтаксис обращения к элементам структуры похож на обращение к полям классов в C#:

[имя структуры].[имя элемента]

Порядок инициализации устройств МК следующий:

1. Объявление структуры соответствующего типа
2. Включение тактирования устройства
3. Заполнение элементов структуры требуемыми значениями
4. Ассоциирование структуры с необходимым устройством МК

**Примечание 3**

Очередная особенность программирования для МК заключается в том, что программа никогда не заканчивается. Представьте что вы сделали фонарик с переключением мощности по кнопке и он перестает работать после того как вы один раз переключили все режимы. Правильный вариант состоит в том, чтобы после последнего режима переходить обратно на первый. Таким образом, необходимо делать программу, в которой при любых внешних воздействиях будет выполняться некоторый код. Из этих соображений, в программе для МК существует такое понятие как главный цикл. Это бесконечный цикл внутри функции **main()**. Все инструкции внутри главного цикла бесконечно повторяются, пока подается питание на МК. Все что до главного цикла выполняется лишь однажды при включении.

**Задание 1 (по вариантам)**

1. Изменить программу таким образом, чтобы светодиоды «бегали» в обоих направлениях.
2. Изменить программу таким образом, чтобы светодиоды «бежали» из центра в края.
3. Изменить программу таким образом, чтобы светодиоды «бежали» из краев в центр.
4. Изменить программу таким образом, чтобы светодиоды «бегали» с задержкой в один светодиод (в каждый момент времени должны гореть два светодиода).
5. Изменить программу таким образом, чтобы чётные светодиоды переключались в 2 раза быстрее нечётных.

**Задание 2 (общее)**

Организовать считывание кнопок джойстика на отладочной плате и реагировать на их состояние соответствующими светодиодами, пользуясь следующей информацией:

* Кнопки джойстика располагаются в порту *C*, номера пинов с 10 по 14.
* Константа поля **PORT\_OE** структуры **PORT\_InitTypeDef** для конфигурации группы портов как вход имеет вид **PORT\_OE\_IN**.
* Кнопки на плате при нажатии соединяются с цепью GND (землей).
* Сигнатура функции для чтения состояния пина имеет вид:

**uint8\_t PORT\_ReadInputDataBit( MDR\_PORT\_TypeDef\* PORTx,**

**uint32\_t PORT\_Pin )**