Министерство науки и высшего образования Российской Федерации Калужский филиал

федерального государственного бюджетного

образовательного учреждения высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(КФ МГТУ им Н.Э. Баумана)

**ЛАБОРАТОРНЫЙ ПРАКТИКУМ**

**«Программирование микроконтроллера Миландр K1986BE1QI»**

Калуга, 2023 г.

[Цель работы: 3](#_Toc129528846)

[**Теоретические сведения** 4](#_Toc129528847)

[**I. Архитектура микроконтроллеров семейства K1986BE1** 4](#_Toc129528848)

[**II. Отладочная плата LDM – K1986BE1QI** 11](#_Toc129528849)

[**III. Меры безопасности при работе с лабораторным стендом** 15](#_Toc129528850)

[**ЛАБОРАТОРНАЯ РАБОТА №1. Знакомство со средой программирования Keil μVision** 16](#_Toc129528851)

[**ЛАБОРАТОРНАЯ РАБОТА № 2. Программирование портов ввода-вывода** 27](#_Toc129528852)

[**ЛАБОРАТОРНАЯ РАБОТА № 3. Таймеры общего назначения** 40](#_Toc129528853)

[**ЛАБОРАТОРНАЯ РАБОТА № 4. Модуль универсального асинхронного приемопередатчика** 52](#_Toc129528854)

[**Рекомендованная литература** 62](#_Toc129528855)

## Цель работы:

Изучение технологии применения микроконтроллеров в системах управления техническими объектами и технологическими процессами, проектирования систем управления на базе микроконтроллеров, разработки их аппаратно-программного обеспечения и методов контроля.

**Задачи:**

1. Приобретение базовых знаний для понимания принципов создания и функционирования микроконтроллеров
2. Изучение принципов построения, функциональных возможностей и архитектурных решений микроконтроллеров.
3. Освоение методики составления и отладки программ для микроконтроллеров
4. Формирование навыков самостоятельного проектирования фрагментов программного обеспечения микроконтроллера

# **Теоретические сведения**

Современная микропроцессорная техника является важнейшим средством при решении самых разнообразных задач в области сбора и обработки данных, систем автоматического управления и др.

Микроконтроллеры представляют собой ряд широко известных представителей микропроцессорной техники. Они объединяют на одном кристалле высокопроизводительный процессор, память, периферийные устройств и позволяют широкий спектр систем управления как объектами, так и процессами с минимальными затратами реализовать.

В рамках обучения принципам работы и архитектуры микроконтроллеров студентам будет предложено познакомиться микроконтроллерами серии 1986ВЕ1QI, основанных на ядре архитектуре, производства одного из лидера разработки отечественной микроэлектронной элементной базы АО «ПКК МИЛАНДР».

# **I. Архитектура микроконтроллеров семейства K1986BE1**

В одной микросхеме микроконтроллер включает в себя микропроцессор, память программ (обычно на основе ПЗУ), память данных (обычно на основе ОЗУ), устройство ввода/вывода, генератор тактовых сигналов, аппаратную поддержку интерфейсов I2C, SPI и многое другое. Структурная схема микросхемы представлена на рисунке 1.

Микроконтроллер К1986BE1QI имеет встроенную флэш-память программ и построен на базе высокопроизводительного процессорного RISC ядра. Микроконтроллер работает на тактовой частоте до 144 МГц и содержит 128 Кбайт флэш-памяти программ и 48 Кбайт ОЗУ.

Периферия включает в себя контроллер USB интерфейса со встроенным аналоговым приемопередатчиком со скоростью передачи 12 Мбит/с (Full Speed) и 1,5 Мбит/с (Low Speed), стандартные интерфейсы UART и SPI, интерфейсы по ГОСТ 18977-79 и ГОСТ Р52070-2003, контроллер внешней системной шины, позволяющий работать с внешними микросхемами статического ОЗУ и ПЗУ, NAND флэш-памятью и другими периферийными устройствами. Также периферия микросхем с функцией Ethernet содержит цифровой интерфейс Ethernet со скоростью передачи 10/100 Мбит, интерфейсом MII и со встроенным аналоговым приемопередатчиком. Микроконтроллер содержит четыре 32-разрядных таймера с 4 каналами схем захвата и ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки. Также микроконтроллер содержит системный 24-разрядный таймер и два сторожевых таймера. Микроконтроллер содержит 12-разрядный высокоскоростной (до 512 Квыб/с) АЦП с возможностью оцифровки информации с 8 каналов, встроенного датчика температуры и опорного напряжения. В микроконтроллере имеются два 12-разрядных ЦАП.

Встроенные RC генераторы HSI (8 МГц) и LSI (40 кГц), внешние генераторы HSE (2…16 МГц) и LSE (32 кГц) и две схемы умножения тактовой частоты PLL для ядра и USB интерфейса позволяют гибко настраивать скорость работы микроконтроллера. Архитектура системы памяти за счет матрицы системных шин позволяет минимизировать возможные конфликты при работе системы и повысить общую производительность. Контроллер DMA позволяет ускорить обмен информацией между ОЗУ и периферией без участия процессорного ядра.

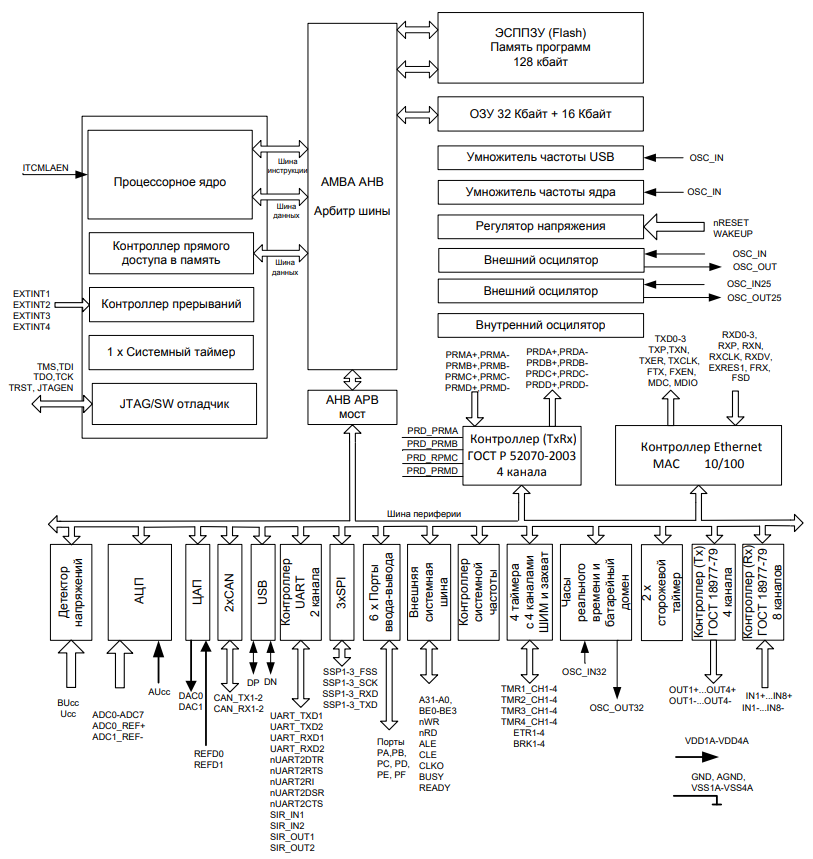


Рисунок 1 – Структурная схема микросхемы

Встроенный регулятор, предназначенный для формирования питания внутренний цифровой части, формирует напряжение 1,8 В и не требует дополнительных внешних элементов. Таким образом, для работы микроконтроллера достаточно одного внешнего напряжения питания в диапазоне от 3,0 до 3,6 В. Также в микроконтроллерах реализован батарейный домен, работающий от внешней батареи, который предназначен для обеспечения функций часов реального времени и сохранения некоторого объёма данных при отсутствии основного питания. Встроенный детектор напряжения питания может отслеживать уровень внешнего основного питания и уровень напряжения питания на батарее. Аппаратные схемы сброса по просадке питания позволяют исключить сбои в работе микросхемы при выходе уровня напряжения питания за допустимые пределы.

Для успешной работы с микропроцессорами серии 1986ВЕ1х необходимо хорошо представлять себе схему питания. В частности, следует знать, что они имеют несколько типов выводов питания. Структурная блок-схема подачи питания приведена на рисунке 2.

Ucc выводы: основное питание микросхемы включает питание пользовательских выводов, встроенного регулятора напряжения, USB PHY, генераторов, компаратора и PLL. Входное напряжение должно быть в пределах от 3,0 до 3,6 В.

BUcc вывод: питание батарейного домена используется при отсутствии основного питания Ucc для питания батарейного домена и LSE генератора. Переключение с основного питания на батарейное происходит автоматически при снижении уровня Ucc ниже 2,0 В. Переключение с батарейного питания на основное происходит автоматически спустя примерно 4 мс после превышения уровнем Ucc порога в 2,0 В. Входное напряжение должно быть в пределах от 1,8 до 3,6 В. Если в системе не требуется батарейное питание, то вывод BUcc должен быть объединен с Ucc.

AUcc вывод: питание аналоговых блоков АЦП, ЦАП выведено на отдельные выводы для уменьшения помех, создаваемых работой других блоков. Если на данные выводы подается напряжение с того же источника, что и Ucc, на печатной плате должны быть применены меры по снижению помех. Если для питания АЦП и ЦАП используется отдельный источник, его выходное напряжение не должно отличаться от Ucc более чем на ±0,2 В. Для корректной работы АЦП входное напряжение должно быть в пределах от 2,4 до 3,6 В.

VDDxA выводы: питание аналогового блока PHY выведено на отдельные выводы для уменьшения помех, создаваемых работой других блоков. Входное напряжение должно быть в пределах от 3,0 до 3,6 В.

GND выводы: «земля» цифровой части микросхемы.

AGND вывод: «земля» аналоговой части микросхемы. Данный вывод должен соединяться с GND, но при этом на печатной плате должны быть применены меры по снижению помех.

VSSxA выводы: «земля» аналогового блока PHY.

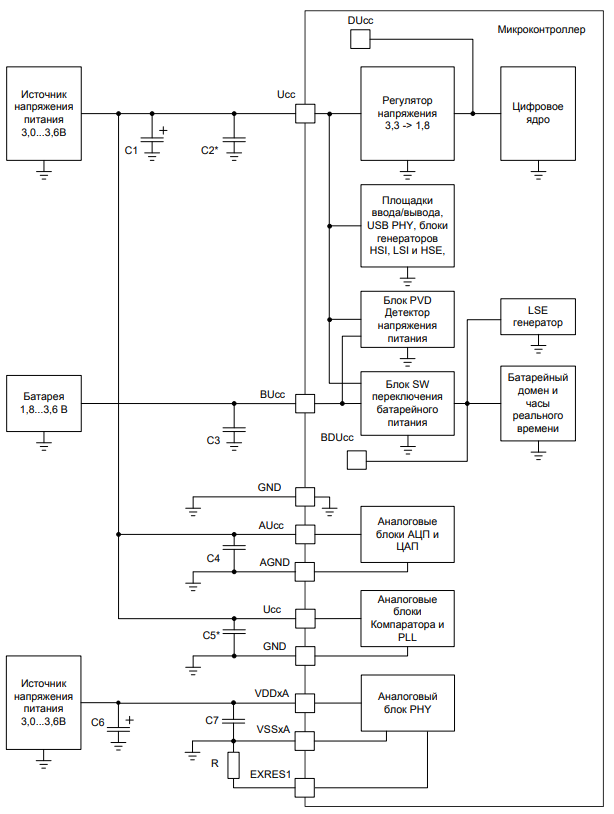


Рисунок 2 - Структурная блок-схема подачи питания

Следует также обратить внимание на организацию структурной памяти (Рисунок 3).

Процессорное ядро имеет три системные шины:

− ITCM Bus – шина выборки инструкций и данных, расположенных в коде программы;

− DTCM Bus – шина выборки данных, расположенных в области ОЗУ;

− AHB-Lite – шина выборки инструкций и данных из внешнего адресного пространства.

Также в микроконтроллере реализован контроллер прямого доступа в память (DMA), осуществляющий выборку через шину DMA Bus.

Все адресное пространство микроконтроллера единое и имеет максимальный объем 4 Гбайт. В данное адресное пространство отображаются различные модули памяти и периферии.

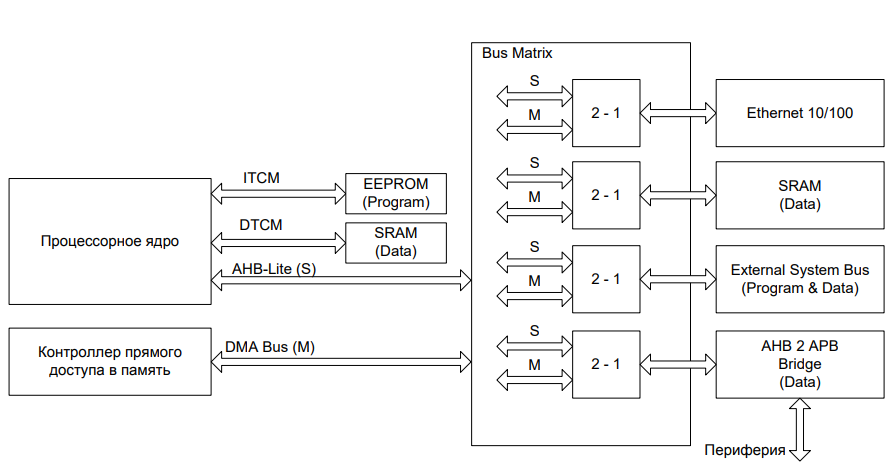


Рисунок 3 - Структурная схема организации памяти

Таким образом, микропроцессор способен выполнять множество операций, что определяется программой, управляющей информацией, которая представляет собой набор команд. При этом его быстродействие, гибкость, удобство использования определяется структурой и объемом системы команд процессора. Процессор может иметь десятки и даже сотни команд, при этом система команд может быть рассчитана на узкий круг решаемых задач или на максимально широкий круг задач в зависимости от того, с каким типом процессора мы имеем дело. Коды команд в каждом случае имеют различное количество разрядов, и каждая команда имеет свое время выполнения. Следовательно, время выполнения всей программы зависит как от количества команд в той или иной программе, так и от конкретных используемых команд.

# **II. Отладочная плата LDM – K1986BE1QI**

Отладочная плата LDM-K1986BE1QI создана на базе российского микроконтроллера фирмы АО «ПКК Миландр» К1986ВЕ1QI с встроенным Ethernet MAC+PHY. На плате может быть установлен микроконтроллер в пластиковом корпусе LQFP144 или в металлокерамическом корпусе 4229.132-3.

Плата предназначена для изучения основ программирования микросхемы К1986ВЕ1QI.

Общий вид отладочной платы представлены на рисунке 4, 5.

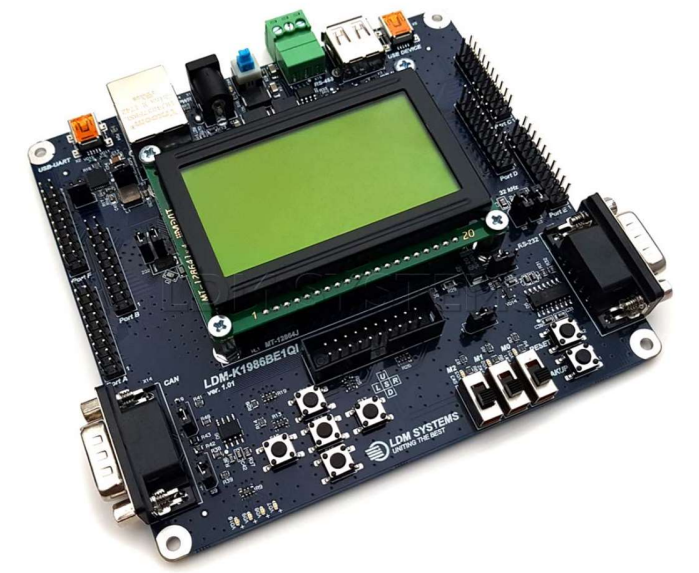


Рисунок 4 – Общий вид отладочной платы LDM-K1986BE1QI

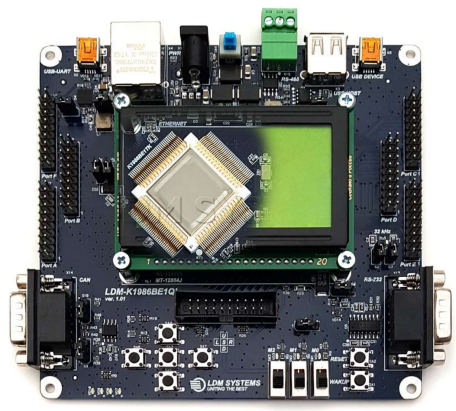


Рисунок 5 – Общий вид отладочной платы LDM-K1986BE1QI в металлокерамическом корпусе 4229.132-3

В таблице 1 представлены основные технические характеристики отладочной платы.

Таблица 1. Основные технические характеристики

|  |  |
| --- | --- |
| Параметр | Значение |
| Тип | Отладочная плата |
| Архитектура контроллера | RISC 32 бит |
| Маркировка контроллера | K1986BE1QI |
| Габаритные размеры (ДxШxВ) | 138x120x22 мм |
| Макетное поле (шаг 2.54 мм) | Нет |
| Корпус контроллера | LQFP144 |
| Количество линий I/O | 36 |
| Кварцевый резонатор | 16 МГц |
| Напряжение питания платы | +5…15 В ± 5% |
| Встроенный программатор | USB-UART |
| FLASH - программ | 128 Кб |
| SRAM | 48 Кб |
| Интерфейс USB - UART | 1 |
| Интерфейс USB | Device и HOST FS (до 12 Мбит/с) |
| Ethernet | 10/100 MAC+PHY Transceiver |
| RS-232 | 1 |
| RS-485 | 1 |
| CAN | 1 |
| ЖКИ дисплей | MT-12864J |
| АЦП | 12 бит, 2 канала |
| ЦАП | 12 бит, 2 канала |

Питание отладочной платы LDM-K1986BE1QI осуществляется от miniUSB разъемов Х2, X3 (+5 В, 0.3÷0.5 А), подключенного к порту USB персонального компьютера. Плату можно запитать от внешнего источника напряжения через разъемы Х1 (+9…15 В, 0.5÷1 А). В таблице 2 приведены режимы включения джамперов, переключателей и их функции.

Таблица 2. Режимы включения джамперов, переключателей и их функции

|  |  |  |
| --- | --- | --- |
| Джампер | Положение | Функционал |
| X1 | - | Разъем питания +9…15 В, 0.5÷1 А |
| X2 | - | Разъем USB c мостом USB-UART на микросхеме CP2102 (может использоваться для загрузки данных во FLASH программ K1986BE1QI) |
| X3 | - | Разъем интерфейса USB HOST |
| X4 | - | Разъем интерфейса Ethernet |
| X5 | - | Разъем интерфейса USB DEVICE |
| X6 | - | Разъем интерфейса JTAG |
| X7…X12 | - | Разъемы GPOI портов A, B, C, D, E, F |
| X13 | - | Разъем интерфейса RS-232 |
| X14 | - | Разъем интерфейса CAN |
| X15 | - | Разъем интерфейса RS-485 |
| HL1 | - | Место установки ЖКИ дисплея |
| SW1…SW3 | - | Пользовательские перемычки. Режим загрузки через USB-UART соответствует включению переключателей SW3…SW1 или (M2…M1) = 100 |
| SW4\* | - | Кнопка включения питания на плате |
| SA1 | - | Кнопка WAKEUP |
| SA2 | - | Кнопка RESET |
| SA3…SA7 | - | Пользовательские кнопки |
| S1 | - | Переключатель выбора BOOT MEMORY |
| S2, S3 | - | Разрыв линий, идущих к резонатору 32 кГц |
| SA4 | 1-2 | Часы реального времени запитаны от линии +3.3 В |
| 2-3 | Часы реального времени запитаны от батареи B1 |
| S5, S6 | - | Разрыв линий, идущих к резонатору 25 МГц |
| S7, S8 | - | Разрыв линий, идущих к USB-UART мосту |
| SA9 | 1-2 | Выбор скорости CAN 500 кбит/с |
| 2-3 | Выбор скорости CAN 125 кбит/с |
| SA10 | 1-2 | Шунт 120 Ом на линии CAN |
| 2-3 | Шунт 62 Ом на линии CAN |
| SA11 | - | Перемычка подключения шунта 120 Ом для RS-485 |
| B1 | - | Батарейный отсек |

\* - Не допускается периодическое нажатие с интервалом менее 1 сек.

# **III. Меры безопасности при работе с лабораторным стендом**

Отладочный макет предназначен для использования в качестве средства разработки аппаратуры и программного обеспечения в условиях учебной лаборатории. Для облегчения использования компоненты платы и соединительные проводники открыты для пользователя и окружающей среды.

При работе с отладочным макетом может произойти разряд статического электричества, что может повлечь повреждение компонентов платы. Поэтому, устройство должно иметь постоянную защиту от электростатического разряда. В дополнение к изложенным выше, необходимо соблюдать следующие рекомендации:

– Пока питание платы включено, не изменяйте подключение электронных компонентов на разъёмах зоны макетирования.

– Пока питание платы включено, не касайтесь открытых проводников и электронных компонентов.

– Будьте осторожны при манипуляциях с переключателями, кнопками, ручками управления при включённом питании платы.

– Перед переноской устройства или начала работы с ним уравняйте потенциалы Вашего тела и платы касанием снятия заряда статического электричества.

# **ЛАБОРАТОРНАЯ РАБОТА №1.** **Знакомство со средой программирования Keil μVision**

**Цель работы:** сформировать навыки программирования микроконтроллера K1986BE1QI с использованием программного обеспечения Keil μVision.

**Задание:** познакомиться с устройством микроконтроллера K1986BE1QI компании Миландр, его спецификацией, с отладочной платой и средой программирования Keil μVision. Создать простейший проект, сконфигурировать среду разработки и запустить программу на отладочной плате.

**Теоретические сведения**

При выполнении цикла лабораторных работ на микроконтроллере будет использоваться интегрированная среда разработки (ИСР) (Integrated Development Enviroment - IDE) компании Keil (An ARM Company). IDE Keil uVision поддерживает разработку программ для различных ARMмикроконтроллеров (версия uVision 3.0 и выше), в том числе для микроконтроллеров серии 1986 фирмы «Миландр» (версия uVision 4.2 и выше). Укомплектована C/C++ компилятором, ассемблером, отладчиком, средствами для трассировки и внутрисхемного программирования, а также поддерживает USB JTAG адаптеры J-Link и ULINK2.

**Установка и первый запуск ИСР Keil uVision.**

Среду программирования Keil uVision можно найти на сайте разработчика по адресу https://www.keil.com/download/product/. Здесь необходимо скачать и установить MDK-ARM (требуется регистрация). Для поддержки микроконтроллеров серии 1986ВЕ9х необходимо скачать и установить последнюю версию библиотеки стандартной периферии от ПКК Миландр https://ic.milandr.ru/products/mikrokontrollery\_i\_protsessory/32\_razryadnye\_mikrokontrollery/1986vekh\_avia/k1986ve1t/#tests\_tab.

После установки всех компонентов среды программирования Keil uVision можно приступать к созданию первого проекта. Для этого необходимо создать новый проект Project – New uVision Project… (рисунок 6) и выбрать директорию для нового проекта (в названии пути к директории не должно быть русских символов).

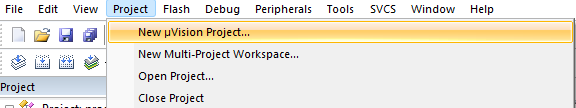


Рисунок 6 – Создание нового проекта

Далее в списке предложенных микроконтроллеров необходимо выбрать тот, для которого будет написана программа. В нашем случае – MDR32F1QI.

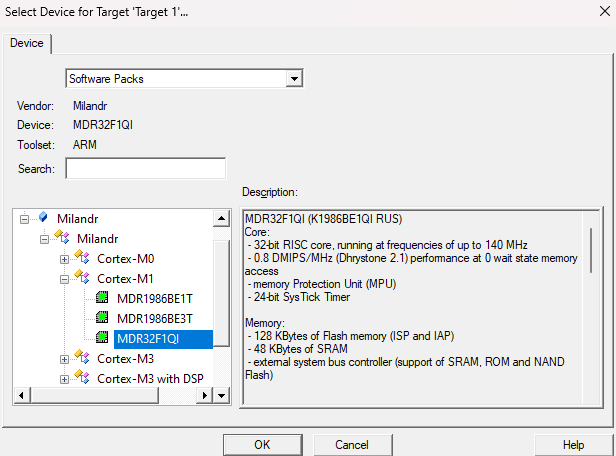


Рисунок 7 – Создание нового проекта

После выбора целевого микроконтроллера необходимо определить периферию и окружение компиляции. Для первого проекта выберем (рисунок 8) Device\Startup\_MDR32F1QI (поддержка ядра микроконтроллера серии MDR32FxQI), Drivers\PORT (порты ввода/вывода), Drivers\RST\_CLK (сигналы тактовой частоты).

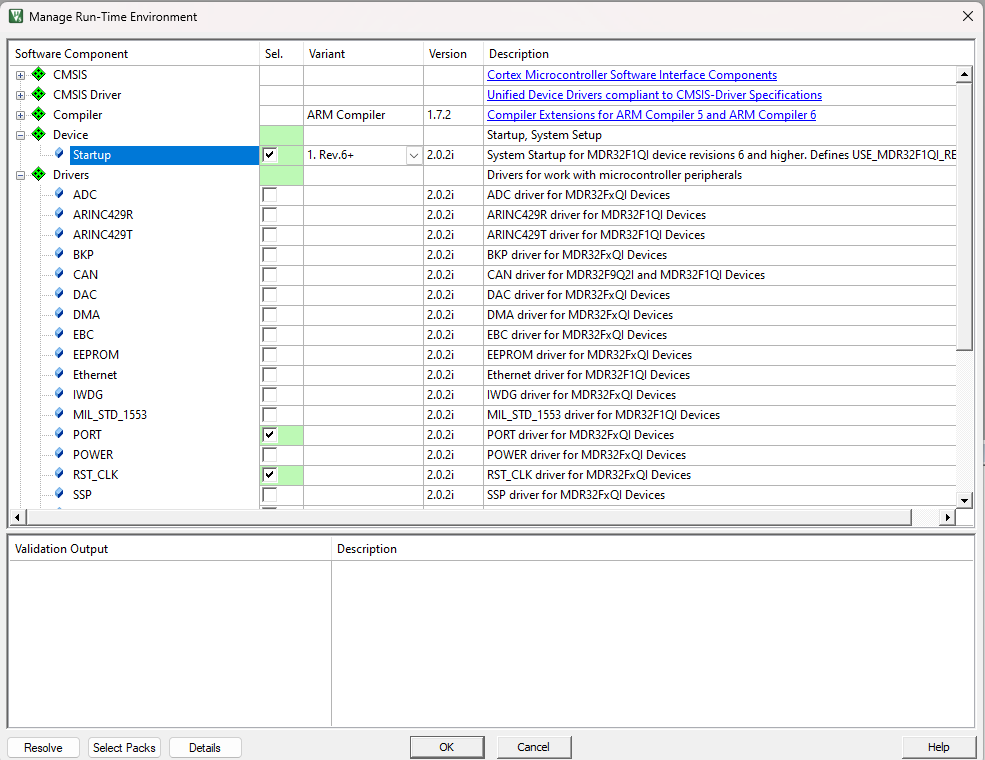


Рисунок 8 – Выбор необходимых драйверов

Далее необходимо настроить проект. Для этого нажимаем правой кнопкой мыши по разделу «Target 1» и выбираем пункт «Options» (рисунок 9).

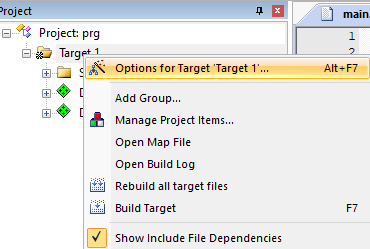


Рисунок 9 – Выбор раздела с параметрами

В открывшемся окне переходим в раздел «Output», где отмечаем галочкой пункт «Create HEX File» (данный параметр отвечает за создание HEX – файла, который понадобится для прошивки микроконтроллера).

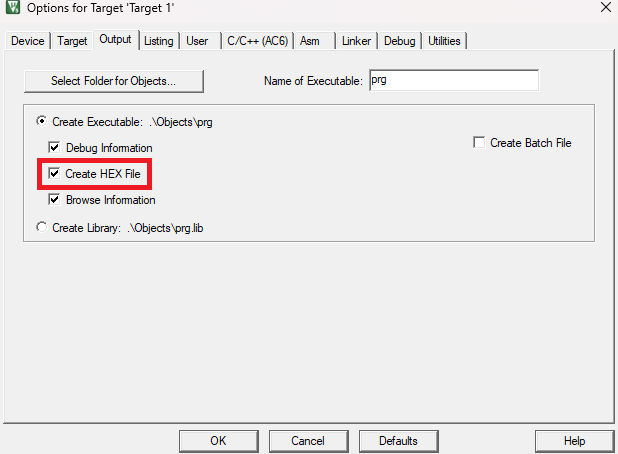


Рисунок 10 – Настройка проекта

Далее переходим в раздел C/C++ (AC6). Здесь необходимо выбрать версию языка c99 (по умолчанию будет стоять параметр c90). Библиотеки Milandr написаны на версии c99, поэтому использование версии c90 будет вызывать множество ошибок при компиляции.

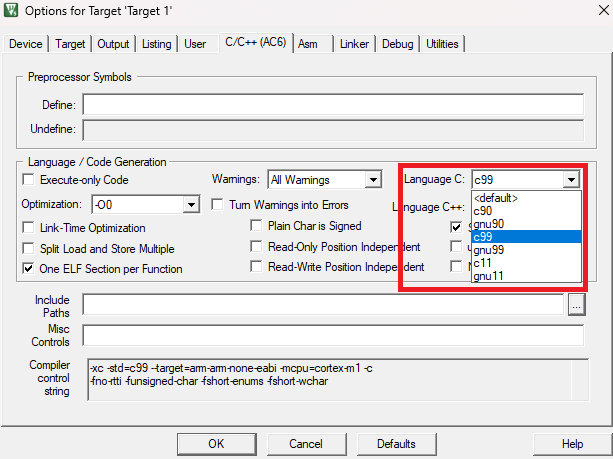


Рисунок 11 – Настройка проекта

Начальная настройка проекта закончена. Далее следует создать сам файл с программой. Для этого нажимаем правой кнопкой мыши по разделу «Source group 1» и выбираем «Add new…». Выбираем тип файла «С File» и даем ему название (на латинице).

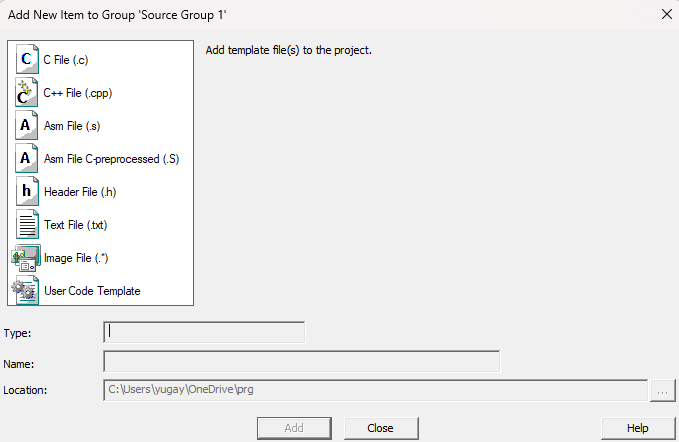


Рисунок 12 – Создание C File

После этого появится поле файла, в которое можно писать код программы. Вставьте предложенный листинг программы в поле файла C. В данной программе происходит мигание светодиодом VD7.

Пример программы:

#include "MDR32FxQI\_config.h"

#include "MDR32FxQI\_port.h"

#include "MDR32FxQI\_rst\_clk.h"

#define DELAY(T) for (i = T; i > 0; i--)

int i;

int main()

{

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA, ENABLE);

PORT\_InitTypeDef Config;

Config.PORT\_Pin = PORT\_Pin\_9;

Config.PORT\_OE = PORT\_OE\_OUT;

Config.PORT\_FUNC = PORT\_FUNC\_PORT;

Config.PORT\_MODE = PORT\_MODE\_DIGITAL;

Config.PORT\_SPEED = PORT\_SPEED\_SLOW;

PORT\_Init (MDR\_PORTA, &Config);

while (1)

{

PORT\_SetBits(MDR\_PORTA, PORT\_Pin\_9);

DELAY(100000);

PORT\_ResetBits(MDR\_PORTA, PORT\_Pin\_9);

DELAY(100000);

}

}

Данная программа демонстрирует мигание светодиодом VD7 с определенной задержкой.

Далее следует зайти в файл MDR32FxQI\_config.h (раздел Device в древе проекта) и раскомментировать 60 или 61 строку в зависимости от ревизии отладочной платы (здесь происходит выбор микроконтроллера).

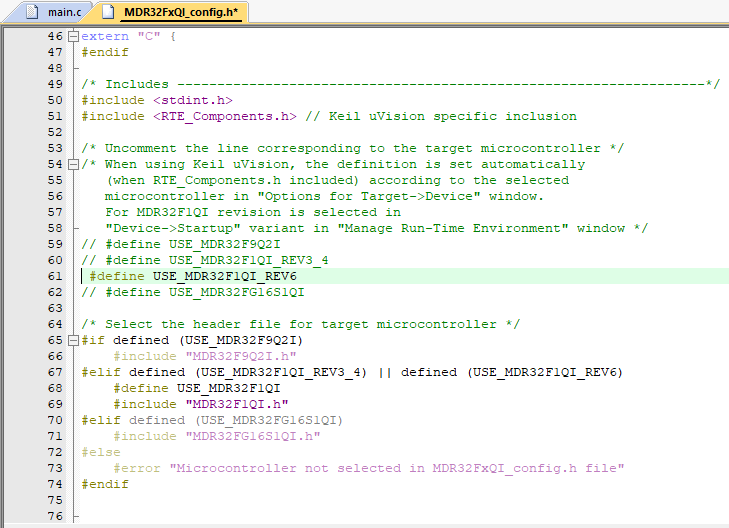


Рисунок 13 – Настройка config.h

Для того, чтобы узнать ревизию отладочной платы, необходимо зайти в раздел «Manage Run-Time Environment» (рисунок 14)

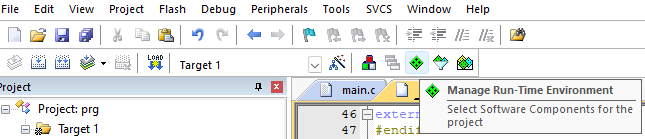


Рисунок 14 – Выбор раздела MRTE

Во вкладке Device -> Startup в поле Variant будет указана ревизия устройства. В данном случае ревизия 6, поэтому необходимо раскомментировать 61 строку в файле MDR32FxQI\_config.h.

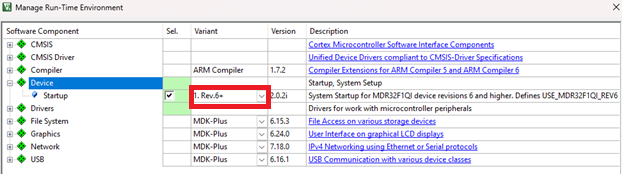


Рисунок 15 – Проверка ревизии устройства

После сохранения всех файлов, можно начинать сборку проекта. Для этого следует нажать на кнопку «Rebuild».

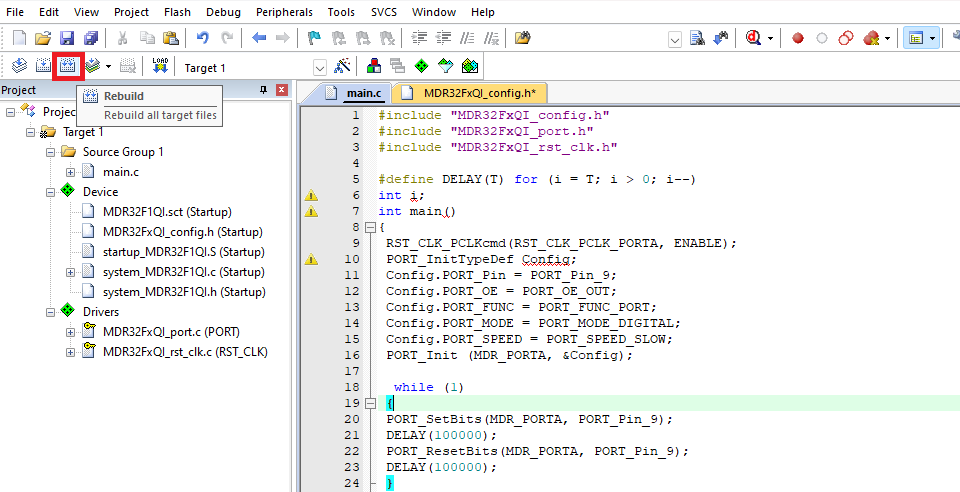
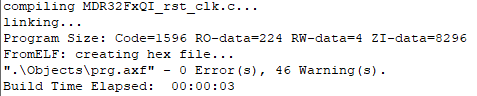


Рисунок 16 – Сборка проекта

Если проект был успешно собран, то в консоли будет следующее сообщение:



Осталось загрузить программу во Flash микроконтроллера. Отладочная плата имеет встроенный программатор USB-UART, с помощью него будет происходить прошивка.

Далее следует подключить отладочную плату к ПК через USB-UART (порт X2). Перед включением отладочной платы, необходимо выставить режим «Загрузка через UART» на самой отладочной плате. Для этого переключатель М2 (SW3) следует установить в положение 1, два других – в 0 (таблица 3) (Изменение режима работы происходит только после включения). Также необходимо замкнуть порты S8 и S7 (располагаются рядом с портом USB-UART) с помощью джамперов на отладочной плате, если это не сделано. После всех этих действий, отладочную плату можно включать (синяя кнопка SW4).

Таблица 3. Режимы работы отладочной платы

|  |  |  |
| --- | --- | --- |
| **MODE**  **(М2, М1, М0)** | **Режим работы** | **Описание** |
| 000 | Режим отладки | Процессор начинает выполнять программу из внутренней FLASH памяти программ. |
| 100, 110 | UART - загрузчик | Микроконтроллер через интерфейс UART1 получает код программы в ОЗУ для исполнения. |

После этого необходимо запустить программу «1986VE1WSD». С помощью нее можно загрузить и запустить программу. Выбираем разработанную ранее программу из папки Objects с расширением .hex, а также COM – порт, в который подключена отладочная плата. Все остальные настройки следует выставить как на рисунке 17.

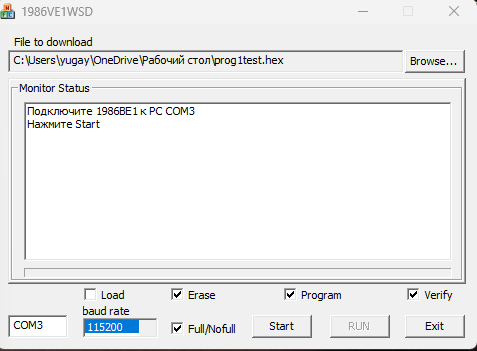


Рисунок 17 – Настройка UART – загрузчика

Нажимаем кнопку «Start». Если все сделано верно, то в статус-баре будет отображена следующая информация:

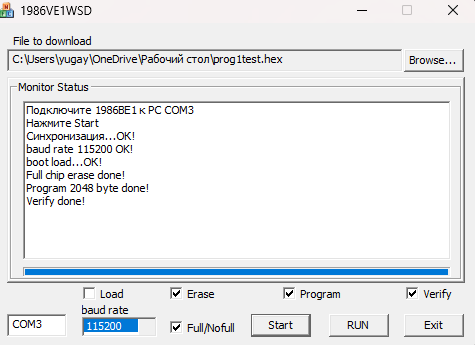


Рисунок 18 – Успешная загрузка программы

Для запуска программы можно нажать на кнопку «RUN». Но для корректной работы отладочной платы следует изменить режим работы на «Режим отладки» (таблица 3) и после этого перезагрузить устройство.

**Ход работы:**

1. Изучить теоретический материал

2. Создать проект на основе теоретического материала

3. Сконфигурировать среду разработки

4. Загрузить программу во FLASH – память микроконтроллера с помощью UART - загрузчика

5. Проверить правильность работы программы на отладочной плате

**Требования к отчету:**

1. Цель работы.

2. Задание.

3. Описание хода выполнения работы

4. Листинг программы.

5. Выводы по результатам выполнения работы

**Вопросы для самоконтроля:**

1. Каково назначение интегрированной среды разработки Keil μVision?

2. Как создать проект в среде разработки Keil μVision?

3. Как изменить частоту миганий светодиодов в приведенном примере программы?

# **ЛАБОРАТОРНАЯ РАБОТА № 2.** **Программирование портов ввода-вывода**

**Цель работы:** сформировать навыки программирования портов ввода-вывода микроконтроллера K1986BE1QI с использованием программного обеспечения Keil μVision.

**Задание:** познакомиться с устройством портов ввода-вывода микроконтроллера K1986BE1QI компании Миландр, их спецификацией и программированием. Настроить порт вывода на индикацию светодиодом согласно индивидуальному заданию.

**Теоретические сведения**

Микроконтроллер имеет 6 портов ввода/вывода. Порты 16-разрядные и их выводы мультиплексируются между различными функциональными блоками, управление для каждого вывода отдельное. Для того, чтобы выводы порта перешли под управление того или иного периферийного блока необходимо задать для нужных выводов выполняемую функцию и настройки. Для использования порта ввода/вывода необходимо разрешить тактирование этого порта в регистре PER\_CLOCK.

Таблица 4. Порты ввода/вывода

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вывод** | **Аналоговая функция**  **ANALOG\_EN=0** | | **Цифровая функция** | | | | | | | |
| **Порт IO**  **MODE[1:**  **0]=00 ANALOG**  **\_EN=1** | **Основная**  **MODE[1:0]=01**  **ANALOG\_EN=1** | | | **Альтернативная**  **MODE[1:0]=00 ANALOG\_EN=1** | | **Переопределенная**  **MODE[1:0]=11**  **ANALOG\_EN=1** | |
| **Порт А** | | | | | | | | | | |
| 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 В** | | | | | | | | | | |
| 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 |
| PA12 | - | PA12 | 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 С** | | | | | | | | | | |
| PС0 | - |  | PС0 | nWR | | 1 | ETR1 | 3 | BRK1 | 3 |
| PС1 | - | PС1 | nRD | | ETR2 | 13 | BRK2 | 13 |
| PС2 | - | PС2 | ALE | | CLKO | 1 | BRK3 | 10 |
| PС3 | - | PС3 | UART\_TXD1 | | 8 | CLE | SIROUT0 | 9 |
| PС4 | - | PС4 | UART\_RXD1 | | BUSY | SIRIN0 |
| PС5 | - | PС5 | EXTINT1 | | 8 | SSP1\_TXD | 14 | SSP1\_RXD | 14 |
| PС6 | - | PС6 | EXTINT2 | | SSP1\_RXD | SSP1\_TXD |
| PС7 | - | PС7 | EXTINT3 |  | | SSP1\_SCK |  | FXEN | 15 |
| PС8 | - | PС8 | EXTINT4 | SSP1\_FSS | FTX |
| PС9 | - | PС9 | SSP2\_TXD | 11 | | BE0 | 1 | CAN\_RX1 | 17 |
| PС10 | - | PС10 | SSP2\_RXD | BE1 | CAN\_TX1 |
| PС11 | - | PС11 | SSP2\_SCK | BE2 | CAN\_RX2 | 18 |
| PС12 | - | PС12 | SSP2\_FSS | BE3 | CAN\_TX2 |
| PС13 | - | PС13 | PRMA+ | 4 | | A30 | 1 | UART\_TXD2 | 12 |
| PС14 | - | PС14 | PRMA- | A31 | UART\_RXD2 |
| PС15 | - | PС15 | 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 | A5 |
| PD7 | ADC0\_REF | 5 | PD7 | SSP2\_TXD | 11 | | nUART2DCD | 12 | 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 | REFD0 | 6 | PD15 | OUT3+ | 2 | | A13 | 1 | FSD | 15 |
| **Порт Е** | | | | | | | | | | |
| PE0 | REFD1 | 6 | PE0 | OUT4+ | 2 | | A14 | 1 | MDC | 15 |
| PE1 | DAC0 | PE1 | OUT3- | A15 | nUART2RI | 12 |
| PE2 | DAC1 | 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 |

Примечания:

1 – выводы управляются системной шиной EXT\_BUS

2 – выводы управляются контроллером интерфейса по ГОСТ 18977-79

3 – выводы управляются Таймером 1

4 – выводы управляются контроллером интерфейса по ГОСТ Р52070-2003

5 – выводы используются АЦП

6 – выводы используются ЦАП

7 – выводы используются генератором LSE

8 – выводы используются контроллером прерываний

9 – выводы управляются контроллером интерфейса UART1

10 – выводы управляются Таймером 3

11 – выводы управляются контроллером интерфейса SSP2

12 – выводы управляются контроллером интерфейса UART2

13 – выводы управляются Таймером 2

14 – выводы управляются контроллером интерфейса SSP1

15 – выводы управляются контроллером интерфейса Ethernet 10/100 (для микросхем с функцией Ethernet)

16 – выводы управляются Таймером 4

17 – выводы управляются контроллером интерфейса CAN1

18 – выводы управляются контроллером интерфейса CAN2

19 – выводы управляются контроллером интерфейса SSP3

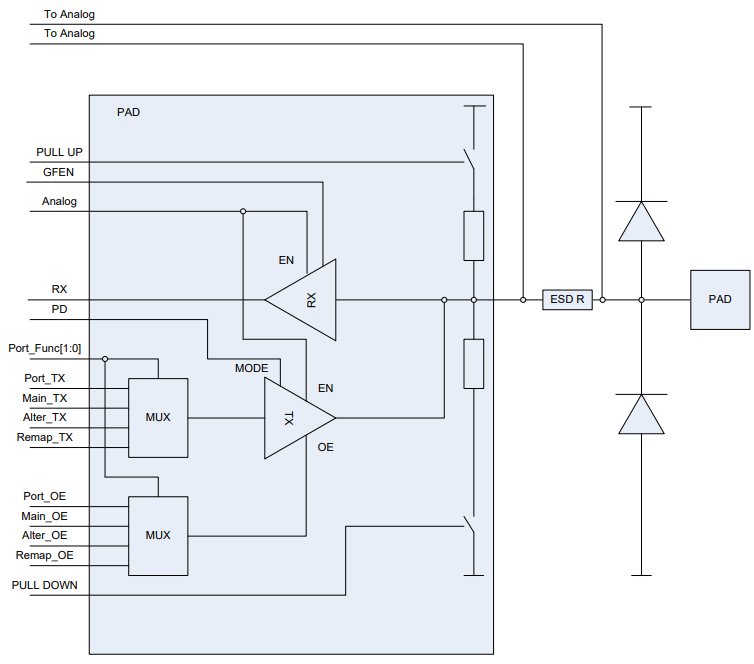


Рисунок 19 – Порты ввода/вывода

Таблица 5. Описание регистров портов ввода-вывода

|  |  |  |
| --- | --- | --- |
| **Базовый адрес** | **Название** | **Описание** |
| 0x400A\_8000 | GPIO1 | Порт А |
| 0x400B\_0000 | GPIO2 | Порт B |
| 0x400B\_8000 | GPIO3 | Порт C |
| 0x400C\_0000 | GPIO4 | Порт D |
| 0x400C\_8000 | GPIO5 | Порт E |
| 0x400E\_8000 | GPIO6 | Порт F |
| **Смещение** |  |  |
| 0x00 | PORT\_RXTX[15:0] | Данные порта |
| 0х04 | PORT\_OE[15:0] | Направление порта |
| 0x08 | PORT\_FUNC[31:0] | Режим работы порта |
| 0x0C | PORT\_ANALOG[15:0] | Режим работы выводов порта (аналоговый/цифровой) |
| 0x10 | PORT\_PULL[31:0] | Подтяжка порта |
| 0x14 | PORT\_PD[31:0] | Режим работы выходного драйвера |
| 0x18 | PORT\_PWR[31:0] | Режим мощности передатчика |
| 0x1C | PORT\_GFEN[15:0] | Режим работы входного фильтра |
| 0х20 | PORT\_SETTX[15:0] | Регистр SET\_TX записью 1 устанавливает 1 в регистре PORT\_RXTX |
| 0x24 | PORT\_CLRTX[15:0] | Регистр CLR\_TX записью 1 устанавливает 0 в регистре RXTX |
| 0x28 | PORT\_RDTX | Регистр позволяет читать то, что записано в выходной регистр порта |

Для работы с портами ввода/вывода используются библиотека MDR32FxQI\_port.h, которая описывает следующие регистры:

– MDR\_PORTA

– MDR\_PORTB

– MDR\_PORTC

– MDR\_PORTD

– MDR\_PORTE

– MDR\_PORTF

Для инициализации используется структура типа PORT\_InitTypeDef с полями:

– PORT\_OE – направление передачи данных

– PORT\_FUNC – режим работы вывода порта

– PORT\_MODE – режим работы контроллера

– PORT\_SPEED – скорость фронта вывода

– PORT\_Pin – выбор выводов для инициализации

Для грамотного написания программного кода необходимо (не обязательно) соблюдать структуру (по убыванию):

1. Подключение библиотек

2. Для инициализации различных элементов микроконтроллера (UART, Timer, порты и т.п.) следует писать отдельные функции (в примере portInitVD7, portInitSA3)

3. Функции с настройками и инициализацией следует объявлять в отдельной функции (в примере cpuInit), вызов которой прописан в основном теле программы main

Пример программы:

#include "MDR32FxQI\_port.h" // Milandr::Drivers:PORT

#include "MDR32FxQI\_rst\_clk.h" // Milandr::Drivers:RST\_CLK

#include "MDR32FxQI\_config.h" // Milandr::Device:Startup

void portInitVD7(void) //инициализация порта со светодиодом VD7

{

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA, ENABLE); //Тактирование портаА

PORT\_InitTypeDef Config; //Создание структуры

Config.PORT\_OE = PORT\_OE\_OUT;

Config.PORT\_FUNC = PORT\_FUNC\_PORT;

Config.PORT\_MODE = PORT\_MODE\_DIGITAL;

Config.PORT\_SPEED = PORT\_SPEED\_SLOW;

Config.PORT\_Pin = PORT\_Pin\_9;

PORT\_Init(MDR\_PORTA, &Config); // Инициализация порта А с объявленной структурой

}

void portInitSA3(void) //инициализация порта с кнопкой SA3

{

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTD, ENABLE);

PORT\_InitTypeDef Config;

Config.PORT\_OE = PORT\_OE\_IN;

Config.PORT\_FUNC = PORT\_FUNC\_PORT;

Config.PORT\_MODE = PORT\_MODE\_DIGITAL;

Config.PORT\_SPEED = PORT\_SPEED\_SLOW;

Config.PORT\_Pin = PORT\_Pin\_2;

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

}

void cpuInit(void)

{

portInitVD7();

portInitSA3();

}

int main(void)

{

cpuInit();

while(1)

{

if (PORT\_ReadInputDataBit(MDR\_PORTD, PORT\_Pin\_2)==1)

{

PORT\_WriteBit(MDR\_PORTA, PORT\_Pin\_9, RESET);

}

else

{

PORT\_WriteBit(MDR\_PORTA, PORT\_Pin\_9, SET);

}

}

}

В данной программе происходит включение светодиода VD7 по нажатию кнопки SA3.

Для нахождения порта с требуемым электрическим элементом (PA3(VD7), PD2(SA3) и т.д.), следует просмотреть принципиальную схему отладочной платы, представленную в PDF – формате.

Также для поиска методов и переменных, используемых при работе с портами, следует просмотреть библиотеку MDR32FxQI\_port.h.

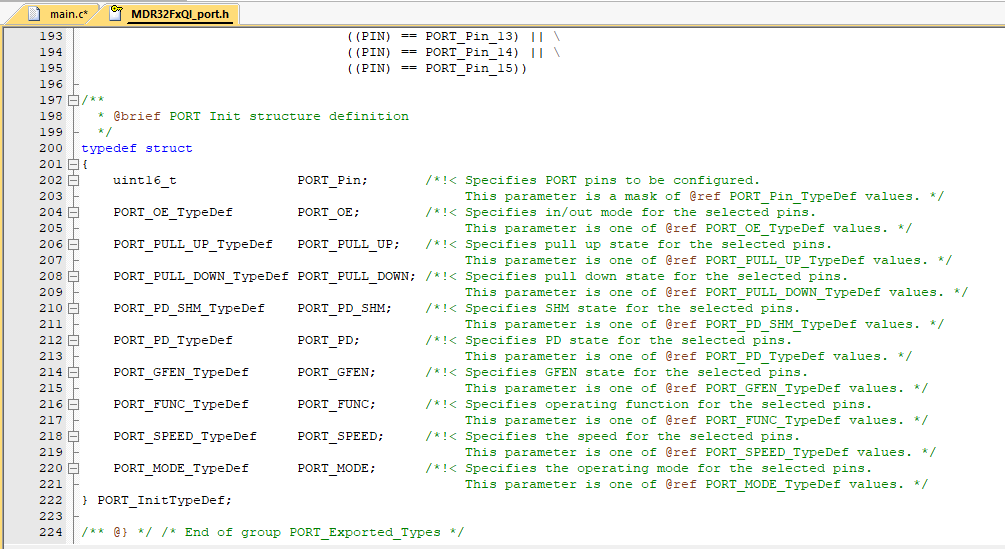
******

Рисунок 20 – Описание полей структуры инициализации порта в библиотеке MDR32FxQI\_port.h

**Варианты заданий**

Таблица 6. Варианты заданий

|  |  |
| --- | --- |
| **№ варианта** | **Задание** |
| 1 | По нажатию кнопки SA6(LEFT) включается индикация светодиодов VD7 и VD10. По нажатию кнопки SA4(UP) включается индикация VD8 и VD9. По нажатию кнопки SA3(SEL) вся индикация прекращается. |
| 2 | По нажатию кнопки SA5(DOWN) включается индикация светодиодов VD7 и VD8. По нажатию кнопки SA7(RIGHT) включается индикация VD9 и VD10. По нажатию кнопки SA3(SEL) вся индикация прекращается. |
| 3 | По нажатию кнопки SA4(UP) включается индикация светодиодов VD7 и VD9. По нажатию кнопки SA6(LEFT) включается индикация VD8 и VD10. По нажатию кнопки SA5(DOWN) вся индикация прекращается. |
| 4 | По нажатию кнопки SA6(LEFT) включается индикация светодиодов VD10 и VD9. По нажатию кнопки SA7(RIGHT) включается индикация VD7 и VD8. По нажатию кнопки SA3(SEL) включается индикация VD8 и VD9, а вся остальная индикация прекращается. |
| 5 | По нажатию кнопки SA4(UP) включается индикация светодиодов VD10 и VD9. По нажатию кнопки SA5(DOWN) включается индикация VD7 и VD8. По нажатию кнопки SA3(SEL) включается индикация VD10 и VD7, а вся остальная индикация прекращается. |
| 6 | По нажатию кнопки SA3(SEL) включается индикация светодиодов VD10 и VD8. По нажатию кнопки SA5(DOWN) включается индикация VD7 и VD9. По нажатию кнопки SA6(LEFT) включается индикация VD9 и VD8, а вся остальная индикация прекращается. |
| 7 | По нажатию кнопки SA4(UP) включается индикация светодиодов VD10, VD9, VD8, а VD7 прекращает индикацию (если был включен). По нажатию кнопки SA5(DOWN) включается индикация VD7, VD8, VD9, а VD10 прекращает индикацию (если был включен). По нажатию кнопки SA3(SEL) вся индикация прекращается. |
| 8 | По нажатию кнопки SA6(LEFT) включается индикация светодиодов VD10, VD9, VD7, а VD8 прекращает индикацию (если был включен). По нажатию кнопки SA7(RIGHT) включается индикация VD7, VD8, VD10, а VD9 прекращает индикацию (если был включен). По нажатию кнопки SA3(SEL) вся индикация прекращается. |
| 9 | По нажатию кнопки SA6(LEFT) включается индикация светодиодов VD10 и VD8, а VD7 и VD9 прекращают индикацию (если были включены). По нажатию кнопки SA7(RIGHT) включается индикация VD7 и VD9, а VD10 и VD8 прекращают индикацию (если были включены). По нажатию кнопки SA3(SEL) вся индикация включается. |
| 10 | По нажатию кнопки SA4(UP) включается индикация светодиодов VD10 и VD9, а VD7 и VD8 прекращают индикацию (если были включены). По нажатию кнопки SA5(DOWN) включается индикация VD7 и VD8, а VD9 и VD10 прекращают индикацию (если были включены). По нажатию кнопки SA3(SEL) вся индикация включается. |
| 11 | При удерживании кнопки SA6(LEFT) включается индикация светодиодов VD10 и VD9, при отпускании они выключаются. При удерживании кнопки SA7(RIGHT) включается индикация светодиодов VD7 и VD8, при отпускании они выключаются. При удержании кнопки SA3(SEL) все индикация светодиодов включается, при отпускании – выключается. |
| 12 | При удерживании кнопки SA4(UP) включается индикация светодиодов VD7 и VD8, при отпускании они выключаются. При удерживании кнопки SA5(DOWN) включается индикация светодиодов VD9 и VD10, при отпускании они выключаются. При удержании кнопки SA3(SEL) все индикация светодиодов включается, при отпускании – выключается. |
| 13 | При удерживании кнопки SA6(LEFT) включается индикация светодиодов VD7, VD8 и VD9, при отпускании они выключаются. При удерживании кнопки SA7(RIGHT) включается индикация светодиодов VD9, VD10 и VD8, при отпускании они выключаются. При удержании кнопки SA3(SEL) все индикация светодиодов включается, при отпускании – выключается. |
| 14 | При удерживании кнопки SA5(DOWN) включается индикация светодиодов VD7 и VD9, при отпускании они выключаются. При удерживании кнопки SA7(RIGHT) включается индикация светодиодов VD8 и VD10, при отпускании они выключаются. При удержании кнопки SA3(SEL) все индикация светодиодов включается, при отпускании – выключается. |
| 15 | При удерживании кнопки SA4(UP) включается индикация светодиодов VD7 и VD10, при отпускании они выключаются. При удерживании кнопки SA6(LEFT) включается индикация светодиодов VD8 и VD9, при отпускании они выключаются. При удержании кнопки SA3(SEL) все индикация светодиодов включается, при отпускании – выключается. |
| 16 | При удерживании кнопки SA4(UP) включается индикация светодиода VD7, при отпускании он выключается. При удерживании кнопки SA5(DOWN) включается индикация светодиода VD10, при отпускании он выключается. При удержании кнопки SA3(SEL) включается индикация светодиодов VD8 и VD9, при отпускании они выключаются. |
| 17 | При удерживании кнопки SA6(LEFT) включается индикация светодиода VD8, при отпускании он выключается. При удерживании кнопки SA7(RIGHT) включается индикация светодиода VD9, при отпускании он выключается. При удержании кнопки SA3(SEL) включается индикация светодиодов VD7 и VD10, при отпускании они выключаются. |
| 18 | При удерживании кнопки SA3(SEL) включается индикация светодиода VD7, при отпускании он выключается. При удерживании кнопки SA7(RIGHT) включается индикация светодиода VD8, при отпускании он выключается. При удержании кнопки SA4(UP) включается индикация светодиодов VD7, VD8 и VD9, при отпускании они выключаются. |

**Ход работы:**

1. Изучить теоретический материал

2. Создать и настроить новый проект в среде Keil uVision

3. Написать код программы согласно своему варианту

4. Загрузить программу во FLASH – память микроконтроллера с помощью UART - загрузчика

5. Проверить правильность работы программы на отладочной плате

**Требования к отчету:**

1. Цель работы.

2. Задание.

3. Описание хода выполнения работы

4. Листинг программы.

5. Выводы по результатам выполнения работы

**Вопросы для самоконтроля:**

1. Что представляет собой порт микроконтроллера?

2. Сколько портов имеет микроконтроллер K1986BE1QI?

3. Какие программируемые параметры имеет порт микроконтроллера?

4. Какую библиотеку используют для работы с портами ввода/вывода?

# **ЛАБОРАТОРНАЯ РАБОТА № 3.** **Таймеры общего назначения**

**Цель работы:** сформировать навыки программирования таймера общего назначения микроконтроллера K1986BE1QI с использованием программного обеспечения Keil μVision.

**Задание:** познакомиться с устройством таймера микроконтроллера K1986BE1QI компании Миландр, его спецификацией и программированием. Реализовать задержку или функцию delay () на основе таймера и применить ее согласно индивидуальному заданию.

**Теоретические сведения**

Все блоки таймеров выполнены на основе 32-битного перезагружаемого счетчика, который синхронизируется с выхода 16-битного предделителя. Перезагружаемое значение хранится в отдельном регистре. Счет может быть прямой, обратный или двунаправленный (сначала прямой до определенного значения, а затем обратный).

Каждый из четырех таймеров микроконтроллера содержит 32-битный счетчик, 16- битный предделитель частоты и 4-канальный блок захвата/сравнения. Их можно синхронизировать системной синхронизации, внешними сигналами или другими таймерами.

Помимо составляющего основу таймера счетчика, в каждый блок таймера также входит четырехканальный блок захвата/сравнения. Данный блок выполняет, как стандартные функции захвата и сравнения, так и ряд специальных функций. Таймеры имеют в своем составе 4 канала схем захвата, ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки. Каждый из таймеров может генерировать прерывания и запросы DMA.

Особенности:

− 32-битный вверх, вниз, вверх/вниз счетчик;

− 16-разрядный программируемый предварительный делитель частоты;

− до четырех независимых 32-битных каналов захвата на один таймер. Каждый из каналов захвата может захватить (скопировать) текущее значение таймера при изменении некоторого входного сигнала. В случае захвата имеется дополнительная возможность генерировать прерывание и/или запрос DMA;

− четыре 32-битных регистра сравнения (совпадения), которые позволяют осуществлять непрерывное сравнение, с дополнительной возможностью генерировать прерывание и/или запрос DMA при совпадении;

− имеется до четырёх внешних выводов, соответствующих регистрам совпадения со следующими возможностями:

- сброс в НИЗКИЙ уровень при совпадении;

- установка в ВЫСОКИЙ уровень при совпадении;

- переключение (инвертирование) при совпадении;

- при совпадении состояние выхода не изменяется;

- переключение при некотором условии.

**Функционирование**

Таймер предназначен для того, чтобы подсчитывать циклы периферийной тактовой частоты Fdts или какие-либо внешние события и произвольно генерировать прерывания, запросы DMA или выполнять другие действия. Значения таймера, при достижении которых будут выполнены те или иные действия, задаются восьмью регистрами совпадения. Кроме того, в микроконтроллере имеются четыре входа захвата, чтобы захватить значение таймера при изменении некоторого входного сигнала, с возможностью генерировать прерывание или запрос DMA.

Структурная схема таймера представлена на рисунке 21.

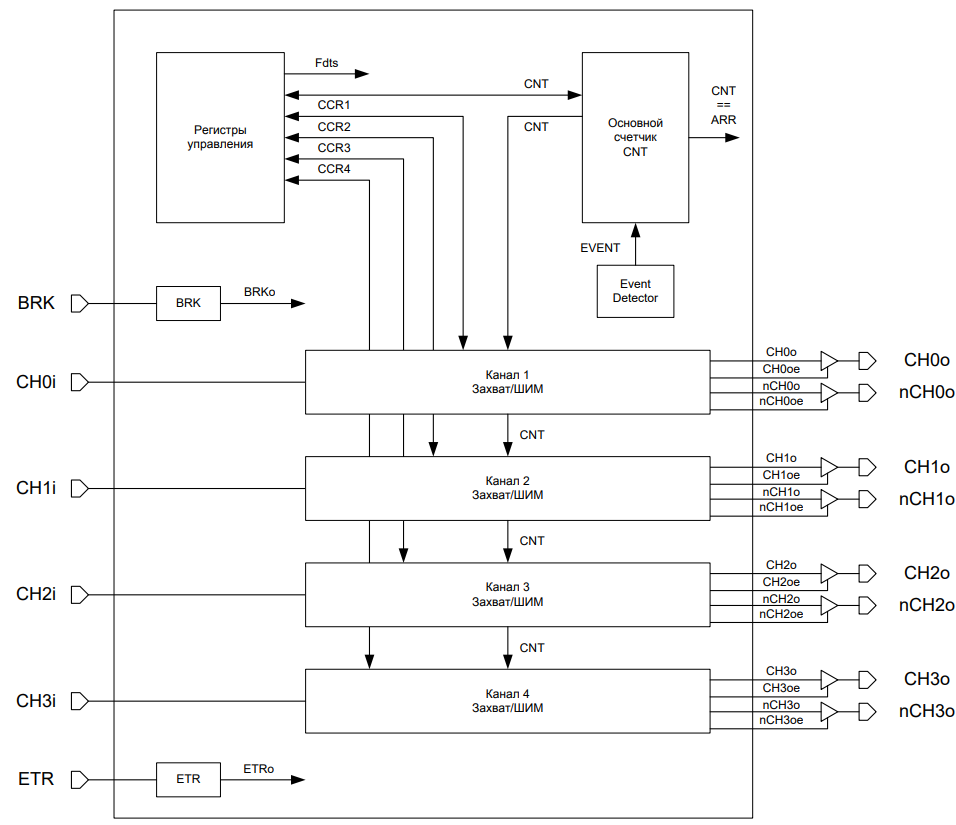


Рисунок 21 – Структурная схема таймера

Для работыс таймерами используется структура TIMER\_CntInitTypeDef с полями:

TIMER\_Prescaler – значение величины предделителя

TIMER\_Period – период таймера

TIMER\_CounterMode – режим счета

TIMER\_CounterDirection – направление счета

TIMER\_EventSource – источник событий для таймера

TIMER\_FilterSampling – указывает фильтр событий

TIMER\_ARR\_UpdateMode – режим сброса счетчика

TIMER\_ETR\_FilterConf – задает параметры выхода ETR

TIMER\_ETR\_Prescaler – задает параметры предделителя фильтра выхода ETR

TIMER\_ETR\_Polarity – задает полярностьвыхода ETR

TIMER\_BRK\_Polarity – задает полярностьвыхода BRK Структура

TIMER\_ChnInitTypeDef с полями:

TIMER\_CH\_Mode – задает режим работы таймера

TIMER\_CH\_REF\_Format – формат выработки сигнала REFв режима ШИМ

TIMER\_CH\_Number – номер канала таймера Структура

TIMER\_ChnOutInitTypeDef с полями:

TIMER\_CH\_DirOut\_Polarity – полярность выхода CHx таймера

TIMER\_CH\_DirOut\_Source – задает сигнал на выходе CHx таймера

TIMER\_CH\_DirOut\_Mode – задает сигнал на выходе CHx таймера

TIMER\_CH\_NegOut\_Polarity – полярность инверсного выхода CHx таймера

TIMER\_CH\_NegOut\_Source – задает сигнал на инверсном выходе CHx таймера

TIMER\_CH\_NegOut\_Mode – задает сигнал на инверсном выходе CHx таймера

TIMER\_CH\_Number – номер канала

Многие из этих параметров можно указать «по умолчанию», поэтому прописывать в коде программы не имеет особого значения.

Однако некоторые параметры при инициализации таймера следует корректировать всегда:

**Настройка делителя тактовой частоты** осуществляется с помощью параметра TIMER\_HCLK, который участвует в выражении:

*TIMER\_BRGInit(MDR\_TIMER1, TIMER\_HCLKdiv1);*

Данный параметр может принимать следующие значения:

− TIMER\_HCLKdiv1: нет деления входной частоты;

− TIMER\_HCLKdiv2: делитель входной частоты на 2 (т.е. замедлит переключение таймера в 2 раза);

− TIMER\_HCLKdiv4: делитель входной частоты на 4;

− TIMER\_HCLKdiv8: делитель входной частоты на 8;

− TIMER\_HCLKdiv16: делитель входной частоты на 16;

− TIMER\_HCLKdiv32: делитель входной частоты на 32;

− TIMER\_HCLKdiv64: делитель входной частоты на 64;

− TIMER\_HCLKdiv128: делитель входной частоты на 128.

**Коэффициент деления предделителя** тактовой частоты задается:

*TIM1Init.TIMER\_Prescaler = 8000;*

Данный параметр изменяет частоту переключения таймера (может принимать значение от 1 до 65535), т.е. таймер будет работать с частотой: тактовая частота (в данному случае 8МГц) / 8000 (указанный коэффициент деления) = 1 мс.

**Период срабатывания таймера** задается выражением**:**

*TIM1Init.TIMER\_Period = 1000;*

Данный параметр изменяет период работы таймера (может принимать значение от 1 до 65535), т.е. таймер досчитает до 1 мс. \*1000 мс. = 1000 мс. и обнулится.

**Выбор таймера,** для которого будут применятся настройки, задается выражением:

*TIMER\_CntInit(MDR\_TIMER1, &TIM1Config);*

То есть настройки, записанные в структуре TIM1Config, будут применены в таймере №1.

Пример программы:

#include "MDR32FxQI\_port.h" // Milandr::Drivers:PORT

#include "MDR32FxQI\_rst\_clk.h" // Milandr::Drivers:RST\_CLK

#include "MDR32FxQI\_config.h" // Milandr::Device:Startup

#include "MDR32FxQI\_timer.h" // Milandr::Drivers:TIMER

void portInitVD7(void)

{

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA, ENABLE);

PORT\_InitTypeDef Config;

Config.PORT\_OE = PORT\_OE\_OUT;

Config.PORT\_FUNC = PORT\_FUNC\_PORT;

Config.PORT\_MODE = PORT\_MODE\_DIGITAL;

Config.PORT\_SPEED = PORT\_SPEED\_SLOW;

Config.PORT\_Pin = PORT\_Pin\_9;

PORT\_Init(MDR\_PORTA, &Config);

}

void timerInit(void)

{

TIMER\_CntInitTypeDef TIM1Config; //Создание структуры

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_TIMER1, ENABLE); //Тактирование

TIMER\_CntStructInit(&TIM1Config); //Заполнение структуры значениями по умолчанию

TIMER\_BRGInit(MDR\_TIMER1, TIMER\_HCLKdiv1);

TIM1Config.TIMER\_Prescaler = 8000;

TIM1Config.TIMER\_Period = 10000;

TIMER\_CntInit(MDR\_TIMER1, &TIM1Config);

TIMER\_Cmd(MDR\_TIMER1, ENABLE); //Включение таймера 1

}

void cpuInit(void)

{

portInitVD7();

timerInit();

}

int main(void)

{

cpuInit();

while(1)

{

PORT\_WriteBit(MDR\_PORTA, PORT\_Pin\_9, SET);

\*функция задержки(delay)\*;

PORT\_WriteBit(MDR\_PORTA, PORT\_Pin\_9, RESET);

\*функция задержки(delay)\*;

}

}

В данной программе происходит мигание светодиода VD7 с определенной задержкой.

Для поиска методов и переменных, используемых при работе с таймером, следует просмотреть библиотеку MDR32FxQI\_timer.h.

Для работы с таймером необходимо при создании проекта (либо уже в созданном проекте) подключить драйвер – TIMER во вкладке Manage Run-Time Environment.

**Варианты заданий**

Таблица 7. Варианты заданий

|  |  |  |
| --- | --- | --- |
| **№ задания** | **№ варианта** | **Задание** |
| 1 | Необходимо реализовать функцию delay() (задержка), используя функции TIMER\_SetCounter() и TIMER\_GetCounter() из библиотеки MDR32FxQI\_timer.h. | |
| 2 | 1 | По нажатию кнопки SA6(LEFT) индикация светодиодов VD7 и VD10 происходит 3 раза с задержкой в 1 секунду. По нажатию кнопки SA4(UP) индикация VD8 и VD9 происходит 5 раз с задержкой в 500 мс. По нажатию кнопки SA3(SEL) индикация VD7-VD10 включается. |
| 2 | По нажатию кнопки SA7(RIGHT) индикация светодиодов VD8 и VD9 происходит 2 раза с задержкой в 2 секунды. По нажатию кнопки SA5(DOWN) индикация VD10 и VD7 происходит 5 раз с задержкой в 250 мс. По нажатию кнопки SA3(SEL) индикация VD7-VD10 включается. |
| 3 | По нажатию кнопки SA4(UP) индикация светодиодов VD9 и VD10 происходит 4 раза с задержкой в 500 мс. По нажатию кнопки SA5(DOWN) индикация VD8 и VD7 происходит 5 раз с задержкой в 100 мс. По нажатию кнопки SA3(SEL) индикация VD7-VD10 происходит 2 раза с задержкой 1 секунду. |
| 4 | По нажатию кнопки SA6(LEFT) индикация светодиодов VD7 и VD8 происходит 2 раза с задержкой в 1 секунду. По нажатию кнопки SA7(RIGHT) индикация VD9 и VD10 происходит 4 раза с задержкой в 200 мс. По нажатию кнопки SA3(SEL) индикация VD8 и VD9 происходит 2 раза с задержкой 1 секунду. |
| 5 | По нажатию кнопки SA6(LEFT) индикация светодиода VD7 происходит 3 раза с задержкой в 800 мс. По нажатию кнопки SA4(UP) индикация VD10 происходит 5 раз с задержкой в 500 мс. По нажатию кнопки SA3(SEL) индикация VD8 и VD9 происходит 3 раза с задержкой в 800 мс. |
| 6 | По нажатию кнопки SA7(RIGHT) индикация светодиода VD8 происходит 4 раза с задержкой в 900 мс. По нажатию кнопки SA5(DOWN) индикация VD9 происходит 3 раза с задержкой в 700 мс. По нажатию кнопки SA3(SEL) индикация VD7 и VD10 происходит 5 раз с задержкой в 400 мс. |
| 7 | По нажатию кнопки SA6(LEFT) индикация светодиода VD10 происходит 5 раз с задержкой в 500 мс. По нажатию кнопки SA7(RIGHT) индикация VD7 происходит 5 раз с задержкой в 500 мс. По нажатию кнопки SA3(SEL) индикация VD10 и VD7 происходит 5 раз с задержкой в 500 мс. |
| 8 | По нажатию кнопки SA4(UP) индикация светодиода VD9 происходит 4 раза с задержкой в 600 мс. По нажатию кнопки SA5(DOWN) индикация VD8 происходит 5 раз с задержкой в 400 мс. По нажатию кнопки SA3(SEL) индикация VD9 и VD8 происходит 3 раза с задержкой в 1 секунду. |
| 9 | По нажатию кнопки SA6(LEFT) индикация светодиода VD10 происходит 5 раз с задержкой в 500 мс. По нажатию кнопки SA4(UP) индикация VD10 происходит 3 раза с задержкой в 2 раза больше. По нажатию кнопки SA3(SEL) индикация VD7-VD10 происходит 5 раз с задержкой в 400 мс. |
| 10 | По нажатию кнопки SA7(RIGHT) индикация светодиода VD7 происходит 6 раз с задержкой в 400 мс. По нажатию кнопки SA5(DOWN) индикация VD7 происходит 4 раза с задержкой в 2 раза больше. По нажатию кнопки SA3(SEL) индикация VD7-VD10 происходит 2 раза с задержкой в 1 секунду. |
| 11 | По нажатию кнопки SA6(LEFT) индикация светодиода VD8 происходит 4 раза с задержкой в 600 мс. По нажатию кнопки SA7(RIGHT) индикация VD8 происходит 5 раз с задержкой в 2 раза меньше. По нажатию кнопки SA3(SEL) индикация VD7-VD10 происходит 2 раза с задержкой в 900 мс. |
| 12 | По нажатию кнопки SA4(UP) индикация светодиода VD9 происходит 3 раза с задержкой в 800 мс. По нажатию кнопки SA5(DOWN) индикация VD9 происходит 4 раза с задержкой в 2 раза меньше. По нажатию кнопки SA3(SEL) индикация VD7-VD10 происходит 5 раз с задержкой в 500 мс. |
| 13 | По нажатию кнопки SA6(LEFT) индикация светодиодов VD7 и VD10 происходит 4 раза с задержкой в 1 секунду. По нажатию кнопки SA4(UP) индикация VD8 и VD9 происходит 5 раз с задержкой в 600 мс. По нажатию кнопки SA3(SEL) индикация VD7 и VD10 происходит 3 раза с задержкой в 800 мс. |
| 14 | По нажатию кнопки SA7(RIGHT) индикация светодиодов VD8 и VD9 происходит 4 раза с задержкой в 800 секунды. По нажатию кнопки SA5(DOWN) индикация VD10 и VD7 происходит 5 раз с задержкой в 300 мс. По нажатию кнопки SA3(SEL) индикация VD7 и VD10 происходит 3 раза с задержкой в 550 мс. |
| 15 | По нажатию кнопки SA4(UP) индикация светодиодов VD9 и VD10 происходит 3 раза с задержкой в 700 мс. По нажатию кнопки SA5(DOWN) индикация VD8 и VD7 происходит 5 раз с задержкой в 500 мс. По нажатию кнопки SA3(SEL) индикация VD7-VD10 происходит 3 раза с задержкой 900 мс. |
| 16 | По нажатию кнопки SA6(LEFT) индикация светодиодов VD7 и VD8 происходит 2 раза с задержкой в 2 секунды. По нажатию кнопки SA7(RIGHT) индикация VD9 и VD10 происходит 4 раза с задержкой в 400 мс. По нажатию кнопки SA3(SEL) индикация VD8 и VD9 происходит 3 раза с задержкой 1 секунду. |
| 17 | По нажатию кнопки SA6(LEFT) индикация светодиодов VD7 и VD8 происходит 4 раза с задержкой в 600 мс. По нажатию кнопки SA5(DOWN) индикация VD7 и VD8 происходит 3 раза с задержкой в 750 мс. По нажатию кнопки SA3(SEL) индикация VD7 и VD8 включается |
| 18 | По нажатию кнопки SA7(RIGHT) индикация светодиодов VD9 и VD10 происходит 5 раз с задержкой в 350 мс. По нажатию кнопки SA4(UP) индикация VD9 и VD10 происходит 4 раза с задержкой в 550 мс. По нажатию кнопки SA3(SEL) индикация VD9 и VD10 включается |

**Ход работы:**

1. Изучить теоретический материал

2. Создать и настроить новый проект в среде Keil uVision

3. Написать код программы согласно своему варианту

4. Загрузить программу во FLASH – память микроконтроллера с помощью UART - загрузчика

5. Проверить правильность работы программы на отладочной плате

**Требования к отчету:**

1. Цель работы.

2. Задание.

3. Описание хода выполнения работы

4. Листинг программы.

5. Выводы по результатам выполнения работы

**Вопросы для самоконтроля:**

1. Что представляет собой таймер микроконтроллера?

2. Сколько таймеров существует в микроконтроллере?

3. Какие программируемые параметры имеет таймер?

4. Какие значения могут принимать коэффициент деления и период срабатывания таймера?

# **ЛАБОРАТОРНАЯ РАБОТА № 4.** **Модуль универсального асинхронного приемопередатчика**

**Цель работы:** сформировать навыки программирования УАПП микроконтроллера K1986BE1QI с использованием программного обеспечения Keil μVision.

**Задание:** познакомиться с устройством УАПП микроконтроллера K1986BE1QI компании Миландр, его спецификацией и программированием. Реализовать передачу и прием данных на ПК через терминал с помощью УАПП.

**Теоретические сведения**

Модуль универсального асинхронного приемопередатчика (UART – Universal Asynchronous Receiver-Transmitter) представляет собой периферийное устройство микроконтроллера.

В состав контроллера включен кодек (ENDEC – Encoder/Decoder) последовательного интерфейса инфракрасной (ИК) передачи данных в соответствии с протоколом SIR (SIR – Serial Infra Red) ассоциации Infrared Data Association (IrDA).

**Основные характеристики модуля UART**

Модуль UART может быть запрограммирован для использования, как в качестве универсального асинхронного приемопередатчика, так и для инфракрасного обмена данными (SIR).

Модуль содержит независимые буферы приема (16x12) и передачи (16x8) типа FIFO (First In First Out – первый вошел, первый вышел), что позволяет снизить интенсивность прерываний центрального процессора.

Программное отключение FIFO позволяет ограничить размер буфера, одним словом.

Есть возможность программно настраивать скорость обмена данными, путем деления тактовой частоты опорного генератора в диапазоне (1x16 – 65535x16). Допускается использование нецелых коэффициентов деления частоты, что позволяет использовать любой опорный генератор с частотой более 3,6864 МГц.

Модулем поддерживаются стандартные элементы асинхронного протокола связи – стартового и стопового бит, а также бита контроля четности, которые добавляются перед передачей и удаляются после приема.

Независимо могут быть маскированы прерывания от буфера FIFO передатчика, буфера FIFO приемника, по таймауту приемника, по изменению линий состояния модема, а также в случае обнаружения ошибки.

Модуль UART позволяет использовать DMA-контроллер для организации обмена данными и снижения нагрузки на ядро микроконтроллера.

Также модуль обеспечивает обнаружение ложных стартовых бит. Формирование и обнаружения сигнала разрыва линии.

Функция управления модемом (линии CTS, DCD, DSR, RTS, DTR и RI) будет поддерживаться с ревизии 3.

Возможность организации аппаратного управления потоком данных.

Полностью программируемый асинхронный последовательный интерфейс имеет следующие характеристики:

− данные длиной 5, 6, 7 или 8 бит;

− формирование и контроль четности (проверочный бит выставляется по четности, нечетности, имеет фиксированное значение, либо не передается);

− формирование 1 или 2 стоповых бит;

− скорость передачи данных – от 0 до UART\_CLK/16 Бод.

**Программируемые параметры**

Следующие ключевые параметры могут быть заданы программно:

− скорость передачи данных;

– целая и дробная часть числа;

− количество бит данных;

− количество стоповых бит;

− режим контроля четности;

− разрешение или запрет использования буферов FIFO (глубина очереди данных – 16 элементов или один элемент, соответственно);

− порог срабатывания прерывания по заполнению буферов FIFO (1/8, 1/4, 1/2, 3/4 и 7/8);

− частота внутреннего тактового генератора (номинальное значение – 1,8432 МГц) может быть задана в диапазоне 1,42 – 2,12 МГц для обеспечения возможности формирования бит данных с укороченной длительностью в режиме пониженного энергопотребления (для ИК-обмена);

− режим аппаратного управления потоком данных.

**Функциональные возможности**

Устройство выполняет следующие функции:

− преобразование данных, полученных от периферийного устройства, из последовательной в параллельную форму;

− преобразование данных, передаваемых на периферийное устройство, из параллельной в последовательную форму.

Процессор читает и записывает данные, а также управляющую информацию и информацию о состоянии модуля. Прием и передача данных буферизуются с помощью внутренней памяти FIFO, позволяющей сохранить до 16 байтов независимо для режимов приема и передачи.

Модуль приемопередатчика:

− содержит программируемый генератор, формирующий тактовый сигнал одновременно для передачи и для приема данных на основе внутреннего тактового сигнала UART\_CLK;

− обеспечивает возможности, сходные с возможностями индустриального стандарта – контроллера UART 16C650;

− позволяет осуществлять обмен информацией с максимальной скоростью:

- в режиме IrDA – до 460800 бит/с;

- в режиме IrDA с пониженным энергопотреблением – до 115200 бит/с.

Устройство может формировать следующие сигналы:

− независимые маскируемые прерывания от приемника (в том числе по таймауту), передатчика, а также по изменению состояния модема и в случае обнаружения ошибки;

− общее прерывание, возникающее в случае, если возникло одно из независимых немаскированных прерываний;

− сигналы запроса на прямой доступ к памяти (DMA) для совместной работы с контроллером DMA.

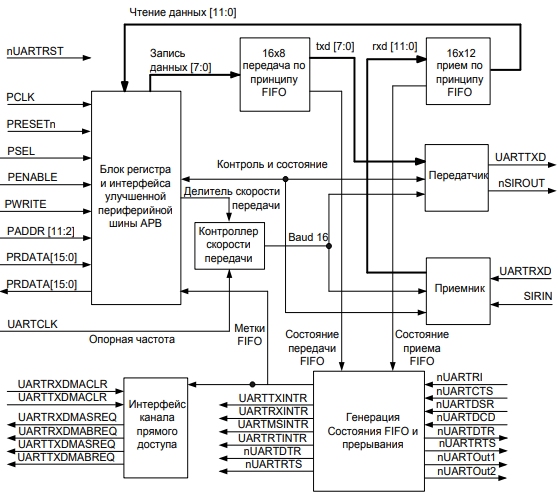


Рисунок 22 – Блок-схема UART

**Генератор тактового сигнала приемопередатчика**

Генератор содержит счетчики без цепи сброса, формирующие внутренние тактовые сигналы Baud16 и IrLPBaud16.

Сигнал Baud16 используется для синхронизации схем управления приемником и передатчиком последовательного обмена данными. Он представляет собой последовательность импульсов с шириной, равной одному периоду сигнала UART\_CLK и частотой, в 16 раз выше скорости передачи данных.

Сигнал IrLPBaud16 предназначен для синхронизации схемы формирования импульсов с длительностью, требуемой для ИК обмена данными в режиме с пониженным энергопотреблением.

**Буфер FIFO передатчика**

Буфер передатчика имеет ширину 8 бит, глубину 16 слов, схему организации доступа типа FIFO («первый вошел, первый вышел»). Данные от центрального процессора, записанные через шину APB, сохраняются в буфере до тех пор, пока не будут считаны логической схемой передачи данных. Существует возможность запретить буфер FIFO передатчика, в этом случае он будет функционировать как однобайтовый буферный регистр.

**Буфер FIFO приемника**

Буфер приемника имеет ширину 12 бит, глубину 16 слов, схему организации доступа типа FIFO («первый вошел, первый вышел»). Принятые от периферийного устройства данные и соответствующие коды ошибок сохраняются логикой приема данных в нем до тех пор, пока не будут считаны центральным процессором через шину APB. Буфер FIFO приемника может быть запрещен, в этом случае он будет действовать как однобайтовый буферный регистр.

**Блок передатчика**

Логические схемы передатчика осуществляют преобразование данных, считанных из буфера передатчика, из параллельной в последовательную форму. Управляющая логика выдает последовательный поток бит в порядке: стартовый бит, биты данных, начиная с младшего значащего разряда, бит проверки на четность, и, наконец, стоповые биты, в соответствии с конфигурацией, записанной в регистре управления.

**Блок приемника**

Логические схемы приемника преобразуют данные, полученные от периферийного устройства, из последовательной в параллельную форму после обнаружения корректного стартового импульса. Кроме того, производятся проверки на: переполнение буфера, ошибки контроля четности, ошибки в структуре сигнала и на разрыв линии. Признаки обнаружения этих ошибок также сохраняются в выходном буфере.

**Блок и регистры синхронизации**

Контроллер поддерживает как асинхронный, так и синхронный режимы работы тактовых генераторов CPU\_CLK и UART\_CLK. Регистры синхронизации и логика квитирования постоянно находятся в активном состоянии. Это практически не отражается на характеристиках устройства и занимаемой площади. Синхронизация сигналов управления осуществляется в обоих направлениях потока данных, то есть как из области действия CPU\_CLK в область действия UART\_CLK, так и наоборот.

Для работы с блоком UART будет использована библиотека MDR32FxQI\_uart.h, которая описывает структуру UART\_InitTypeDef.

Структура UART\_InitTypeDef имеет следующие поля:

− UART\_BaudRate – скорость передачи данных;

− UART\_WordLength – длина слова в пакете;

− UART\_StopBits – количество стоп-битов;

− UART\_Parity – контроль четности;

− UART\_FIFOMode – определяет режим работы буфера FIFO: осуществлять передачу по нескольку байт или побайтно;

− UART\_HardwareFlowControl – включает/выключает аппаратный контроль потока.

При работе с UART-контроллером применяются функции записи данных UART\_SendData и чтения UART\_ReceiveData.

Пример программы:

#include "MDR32FxQI\_port.h" // Milandr::Drivers:PORT

#include "MDR32FxQI\_rst\_clk.h" // Milandr::Drivers:RST\_CLK

#include "MDR32FxQI\_config.h" // Milandr::Device:Startup

#include "MDR32FxQI\_timer.h" // Milandr::Drivers:TIMER

#include "MDR32FxQI\_uart.h" // Milandr::Drivers:UART

void timerInit(void) //инициализация таймера

{

TIMER\_CntInitTypeDef TIM1Config;

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_TIMER1, ENABLE);

TIMER\_CntStructInit(&TIM1Config);

TIMER\_BRGInit(MDR\_TIMER1, TIMER\_HCLKdiv1);

TIM1Config.TIMER\_Prescaler = 8000;

TIM1Config.TIMER\_Period = 10000;

TIMER\_CntInit(MDR\_TIMER1, &TIM1Config);

TIMER\_Cmd(MDR\_TIMER1, ENABLE);

}

void portInitUART(void) // Инициализация портов RX и TX с UART1

{

/\*1. Включить тактирование порта с RX, TX

2. Создать структуру для инициализации порта

3. Выбор функция порта – основная

4. Режим работы порта – цифровой

5. Скорость порта – максимально быстрая

6. Выбор пина с TX

7. Настройка пина на выход

8. Передача настроек порта из структуры

9. Выбор пина с RX

10. Настройка пина на вход

11. Передача настроек порта из структуры\*/

}

void UARTInit(void) // Инициализация UART

{

/\*1. Включить тактирование UART1

2. Создать структуру для инициализации UART

3. Делитель тактовой частоты UART1 = 1

4. Скорость передачи (Бодрейт) = 9600

5. Количество бит в посылке = 8

6. Количество стоп – бит = 1

7. Проверка четности – отсутствует

8. Работа FIFO – приемника отключена

9. Прием и передача данных разрешена (RXE | TXE)

10. Передача настроек UART1 из структуры

11. Включение UART1 \*/

}

void cpuInit(void)

{

portInitSA3();

timerInit();

portInitUART();

UARTInit();

}

int main(void)

{

cpuInit();

while(1)

{

}

}

Для нахождения пинов с RX и TX следует просмотреть таблицу 4 (UART\_TXD1, UART\_RXD1) либо принципиальную электрическую схему отладочной платы.

Для поиска методов и переменных, используемых при работе с УАПП, следует просмотреть библиотеку MDR32FxQI\_uart.h. Для работы с УАПП необходимо при создании проекта (либо уже в созданном проекте) подключить драйвер – UART во вкладке Manage Run-Time Environment.

Для обмена информацией между ПК и внешним терминалом воспользуемся свободно распространяемой программой Terminal v1.9b.

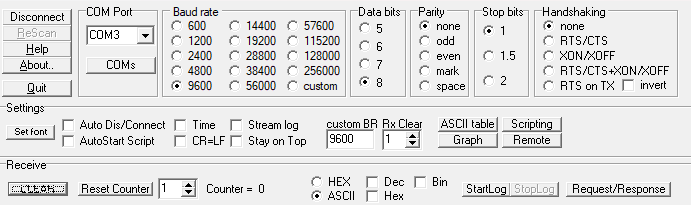


Рисунок 23 – Блок настроек терминала

Перед началом работы с терминалом необходимо настроить его. Для этого выставляем все значения с теми, что указали при инициализации UART (бодрейт, количество бит в пакете данных, проверка четности и т.п.). Далее выбираем COM – порт, к которому подключена отладочная плата и нажимаем «Connect». Если в списке отсутствует нужный COM – порт, то следует нажать на кнопку «Rescan». В поле настроек «Receive» можно выбрать способ отображения принимаемой информации (HEX, ASCII). В зависимости от того, какой тип информации будет передаваться на терминал (слова – ASCII, 16 - код – HEX), выбираем нужное значение. Все остальные настройки следует оставить по умолчанию.

Очистить поле приема информации можно кнопкой «CLEAR». Для передачи информации из терминала на отладочную плату в десятичном виде необходимо перед значением ставить символ «#», шестнадцатеричной информации – «$» (рисунок 24). В примере ниже происходит передача цифры «322» на отладочную плату и получение ФИО.

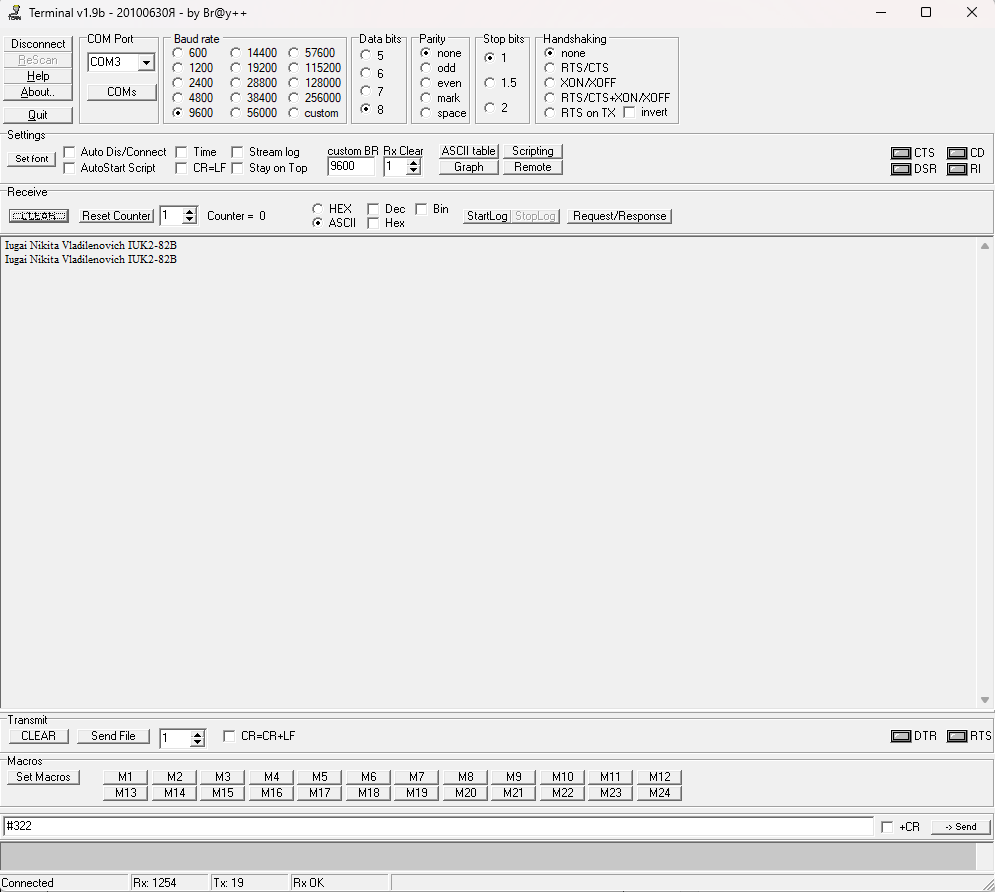


Рисунок 24 – Пример выполнения задания

Если при однократном нажатии кнопки на отладочной плате, на терминал информация поступает сразу несколько раз (рисунок 24), то это признак явления «дребезг контактов». В данной лабораторной работе этим можно пренебречь.

Таблица 8. Варианты заданий

|  |  |
| --- | --- |
| **№** | **Задание** |
| 1 | Настроить работу портов RX и TX |
| 2 | Инициализировать устройство UART |
| 3 | Вывести (по нажатию кнопки) в терминал ФИО и группу с помощью UART |
| 4 | С помощью команды в терминале включить/отключить индикацию светодиода на отладочной плате |

**Ход работы:**

1. Изучить теоретический материал

2. Создать и настроить новый проект в среде Keil uVision

3. Написать код программы согласно заданию

4. Загрузить программу во FLASH – память микроконтроллера с помощью UART - загрузчика

5. Настроить терминал и проверить правильность работы программы

**Требования к отчету:**

1. Цель работы.

2. Задание.

3. Описание хода выполнения работы

4. Листинг программы.

5. Выводы по результатам выполнения работы

**Вопросы для самоконтроля:**

1. Что представляет собой устройство УАПП?

2. Как настроить порты микроконтроллера для приема и передачи информации с помощью УАПП?

3. Какие программируемые параметры имеет УАПП?

4. Для чего используется терминал и как его настроить?

# **Рекомендованная литература**

1. Алалуев Р. В. Основы программирования 32-разрядных микроконтроллеров 1986ВЕ1QI компании «Миландр»: руководство к выполнению лабораторных работ / Р.В. Алалуев, В.М. Глаголев, А.А. Мосур, Л.Л. Владимиров. – М., 2017. – 128 с.: ил.

2. Благодаров А.В. Программирование микроконтроллеров на основе отечественных микросхем семейства 1986ВЕ1x разработки и производства компании «Миландр» / А.В. Благодаров, Л.Л. Владимиров. – М., 2016. – 242 с.: ил.

3. Васильев А.С. Основы программирования микроконтроллеров / А.С. Васильев, О.Ю. Лашманов, А.В. Пантюшин. – СПб: Университет ИТМО, 2016. – 95 с.

4. Пуговкин А.В. Методическое пособие по программированию микроконтроллеров: Учебно-методическое пособие / А.В. Пуговкин, И.А. Куан, Н.К. Ахметов, А.В. Бойченко. – Томск, 2016. – 69 с.

5. Строганова С.М. Методические указания к выполнению лабораторных работ по микроконтроллерам семейства 1986BE1Х компании «Миландр» для студентов специальности 27.03.04 «Управление в технических системах» / С.М. Строганова, Н.Н. Теодорович. – Королев, 2016. – 85 с.

6. Лабораторный практикум по микроконтроллерам семейства Cortex-M: Методическое пособие по проведению лабораторных работ на отладочных платах фирмы «Миландр» / сост.: В.Г. Рубанов, А.С. Кижук, Д.А. Бушуев, Е.Б. Карик, Е.П. Добринский. – Белгород: Изд-во БГТУ, 2016. – 61 с.