**Работа №6**

**Блок батарейного домена. Использование часов реального времени.**

**Цель работы:** ознакомиться с модулем батарейного домена и особенностями использования часов реального времени на МК 1986BE92.

**Задачи:**

- изучить функциональный состав модуля MDR\_BKP;

- получить навыки программирования модуля BKP и часов реального времени;

- закрепить на практике особенности программирования выводов общего назначения микроконтроллера;

- закрепить на практике особенности использования системы прерываний микроконтроллера;

- закрепить навыки работы с дисплеем МЭЛТ 12864;

- познакомиться со структурой представления времени из библиотеки time.h.

- закрепить навыки работы с библиотеками Common Microcontroller Software Interface Standard (CMSIS).

**Используемое оборудование:**

1. Отладочная плата с микроконтроллером Миландр MDR1986BE92QI.
2. Комплект Программатора

2.1 Программатор JLINK (USB-JTAG);

2.2 Кабель USB 2.0 A – B;

2.3 Шина (20 проводников).

1. Источник питания 5В (или дополнительный USB-кабель)
2. Комплект Логического анализатора

4.1 Логический анализатор

4.2. Кабель USB 2.0A – mini-B

4.3 Шина 10 проводников.

5. Элемент питания CR2032.

**Используемая документация:**

1. Техническое описание на ядро Миландр MDR1986BE9x (файл «1\_Тех\_описание\_ядро\_1986ВЕ9X.pdf или по ссылке с сайта разработчика [https://ic.milandr.ru/upload/iblock/a33/ioaf9ygfmq1lbxfhd5aad0mukg3dc93s/1986%D0%92%D0%959X.pdf](https://ic.milandr.ru/upload/iblock/a33/ioaf9ygfmq1lbxfhd5aad0mukg3dc93s/1986ВЕ9X.pdf))

2. Выводы отладочной платы микроконтроллера 1986BE92QI (файл «2\_Выводы\_платы\_1986BE92QI.pdf или по ссылке\* с сайта разработчика <https://ic.milandr.ru/upload/iblock/8f6/8f67b8b736b3ec94edbbeb4777a9c4db.zip>)

При загрузке схемотехнической документации с сайта Milandr (информация о выводах платы), загружается zip архив, который содержит 1 или 2 файла со схемой размещения выводов микроконтроллера на плате и несколько схемотехнических файлов для разводки и печати платы.

**Теоретическая часть**

Блок батарейного домена предназначен для обеспечения функций часов реального времени и сохранения некоторого набора пользовательских данных при отключении основного источника питания. При снижении питания UCC в блоке SW происходит автоматическое переключение питания BDUCC c UCC на BUCC. Если на BUCC имеется отдельный источник питания (батарейка), то батарейный домен остается включенным и может выполнять свои функции.

UCC – Основное питание микросхемы (2.2В-3.6В).

BUCC – 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 2.0В. Переключение с батарейного питания на основное происходит автоматически спустя примерно 4 мс после превышения уровнем UCC порога в 2.0В.

BDUCC – результирующее питание блока батарейного домена после выбора между линиями UCC и BUCC.

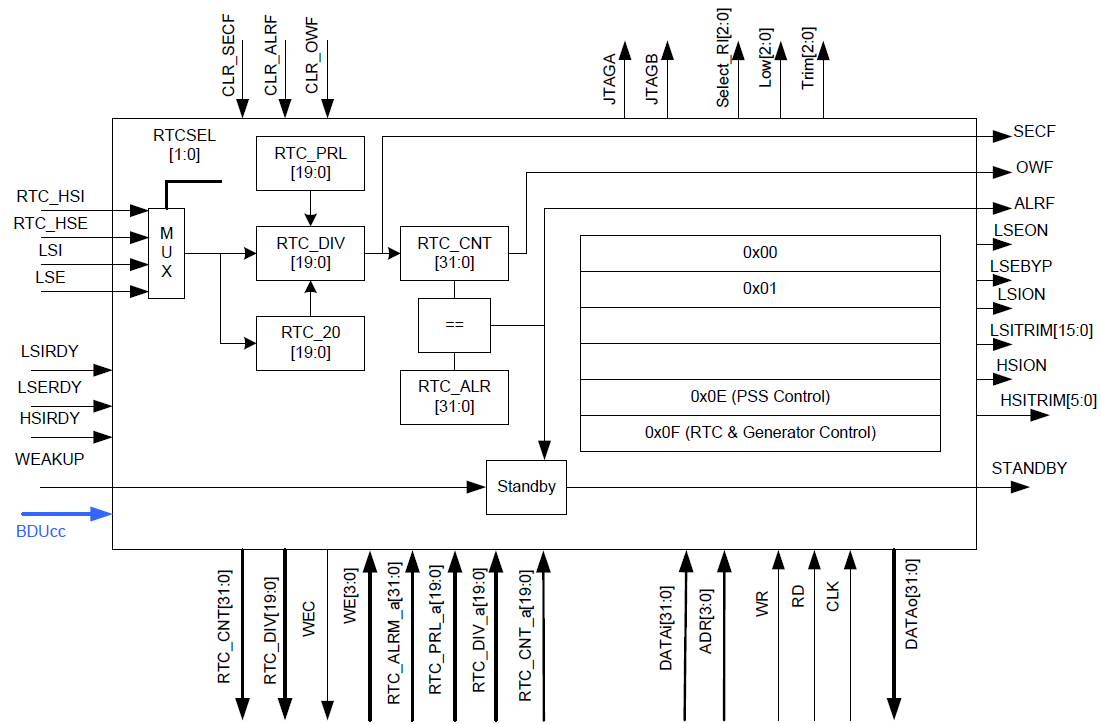


Рисунок 6.1 – Структурная схема блока батарейного домена и часов реального времени

**Регистры аварийного сохранения**

Батарейный домен имеет 16 встроенных 32-разрядных регистров аварийного сохранения. 14-й и 15-й регистры служат для хранения бит управления батарейным доменом, оставшиеся 14 [0:13] могут быть использованы разработчиком ПО для сохранения необходимых значений.

**Часы реального времени**

Часы реального времени (RTC – real time clock) позволяют вести отсчёт времени (секунд, минут, часов и т.д.) в кристалле, в том числе при отключении основного источника питания. Включение часов реального времени осуществляется битом RTCEN. В качестве источника тактовой частоты часов реального времени может выступать генератор LSI, или осциллятор LSE, или HSE, или HSI c дополнительным делителем до 256 (HSE и HSI формируются в блоке управления тактовыми частотами и могут быть выбраны только при наличии питания DUCC, LSI может быть выбран при наличии питания UCC, LSE может быть выбран при наличии UCC или BUCC). Выбор между источниками осуществляется битами RTCSEL. При возможном отключении основного источника питания UCC в качестве источника тактовой частоты должен использоваться осциллятор LSE, так как он также имеет питание BDUCC. Биты управления осциллятором LSE расположены в батарейном домене, и, таким образом, при отключении основного питания они не сбрасываются.

Для калибровки тактовой частоты используются биты CAL[6:0]. Значение CAL определяет, какое число тактов из 220 будет замаскировано. Таким образом, с помощью бит CAL[6:0] производится замедление хода часов. Изменение значения бит CAL может быть осуществлено в ходе работы часов реального времени.

Регистр RTC\_DIV выступает в роли 20-битного предварительного делителя входной тактовой частоты таким образом, чтобы на его выходе была тактовая частота в 1 Гц. Для задания коэффициента деления регистра RTC\_DIV используется регистр RTC\_PRL.

Регистр RTC\_CNT предназначен для отсчета времени в секундах и работает на выходной частоте делителя RTC\_DIV. Регистр RTC\_ALR предназначен для задания времени, при совпадении с которым вырабатывается флаг прерывания и пробуждения процессора. Таким образом, бит STANDBY, отключающий внутренний регулятор напряжения, автоматически сбрасывается при совпадении RTC\_CNT и RTC\_ALR. Бит STANDBY также может быть сброшен с помощью вывода WAKEUP.

**Описание регистров BKP**

**REG\_00 – REG\_0D** – пользовательские регистры аварийного сохранения 32-битных значений;

**REG\_0E** – аварийный регистр 14, управление подстройкой напряжения, загрузкой и отладкой программы.

**REG\_0F** – аварийный регистр 15, управление блокамиRTC, LSE, LSI, HSI

**RTC\_CNT –**  основной счётчик часов реального времени;

|  |  |
| --- | --- |
| Номер | [31:0] |
| Назначение | RTC\_CNT |
| Доступ | R/W |
| Сброс | 0 |

**RTC\_DIV** – регистр предварительного делителя счётчика RTC\_CNT

|  |  |  |
| --- | --- | --- |
| Номер | [31:20] | [19:0] |
| Назначение | – | RTC\_DIV |
| Доступ | – | R/W |
| Сброс | – | 0 |

**RTC\_PRL** – регистр основания счёта предварительного делителя

|  |  |  |
| --- | --- | --- |
| Номер | [31:20] | [19:0] |
| Назначение | – | RTC\_PRL |
| Доступ | – | R/W |
| Сброс | – | 0 |

**RTC\_ALRM** – регистр значения для сравнения основного счетчика и выработки сигнала ALRF

|  |  |
| --- | --- |
| Номер | [31:0] |
| Назначение | RTC\_ALRM |
| Доступ | R/W |
| Сброс | 0 |

**RTC\_CS –** регистр управления и состояния часов реального времени

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер | 31..7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Назначение | - | WEC | ALRF\_IE | SECF\_IE | OWF\_IE | ALRF | SECF | OWF |
| Доступ | U | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

WEC – Запись завершена:

0 – можно записывать в регистры RTC;

1 – идёт запись в регистры RTC, другие операции запрещены.

ALFR\_IE – Флаг разрешения прерывания по совпадению основного счётчика RTC\_CNT и регистра RTC\_ALRM:

0 – нет совпадения;

1 – есть совпадение.

SECF\_IE – разрешения прерывания по разрешению счета основного счетчика от счетчика предварительного деления:

0 – нет разрешения счёта;

1 – есть разрешение счёта.

OWF\_IE – Флаг разрешения прерывания по переполнению основного счётчика RTC\_ CNT:

0 – нет переполнения;

1 – было переполнение.

ALRF – Флаг совпадения основного счётчика и регистра RTC\_ALRM:

0 – нет совпадения;

1 – есть совпадение.

Сброс флага осуществляется записью 1.

SECF – Флаг разрешения счёта основного счётчика от счётчика предделителя.

0 – нет разрешения счёта;

1 – есть разрешение счёта.

Сброс флага осуществляется записью 1.

OWF – Флаг переполнения основного счётчика RTC\_CNT:

0 – нет переполнения;

1 – есть переполнение.

Сброс флага осуществляется записью 1.

**Функции блока MDR\_BKP\_RTC**

Сброс параметров блока

void BKP\_DeInit(void);

Включение и выключение JTAGA

void BKP\_JTAGA\_CMD(FunctionalState NewState);

Включение и выключение JTAGB

void BKP\_JTAGB\_CMD(FunctionalState NewState);

Выбор источника тактового сигнала для часов реального времени

void BKP\_RTCclkSource(uint32\_t RTC\_CLK);

Включение часов реального времени

void BKP\_RTC\_Enable(FunctionalState NewState);

Калибровка часов реального времени

void BKP\_RTC\_Calibration(uint32\_t RTC\_Calibration);

Сброс часов реального времени

void BKP\_RTC\_Reset(FunctionalState NewState);

Разрешение прерываний от часов реального времени

void BKP\_RTC\_ITConfig(uint32\_t RTC\_IT, FunctionalState NewState);

Чтение текущего значения счётчика часов реального времени

uint32\_t BKP\_RTC\_GetCounter(void);

Установка значения счётчика часов реального времени

void BKP\_RTC\_SetCounter(uint32\_t CounterValue);

Установка значения регистра ALRM значения сравнения основного счётчика для выработки сигнала ALRF

void BKP\_RTC\_SetAlarm(uint32\_t AlarmValue);

Чтение текущего состояния предделителя частоты

uint32\_t BKP\_RTC\_GetDivider(void);

Установка предделителя частоты

void BKP\_RTC\_SetPrescaler(uint32\_t PrescalerValue);

Ожидание завершения операций изменения значений в блоке BKP\_RTC

void BKP\_RTC\_WaitForUpdate(void);

Анализ флагов состояния модуля RTC

FlagStatus BKP\_RTC\_GetFlagStatus(uint32\_t RTC\_FLAG);

**Параметры блока MDR\_BKP\_RTC**

Описание регистров аварийного сохранения

BKP\_REG00

BKP\_REG01

BKP\_REG02

………

BKP\_REG0E

BKP\_REG0F

Обращение к регистру может быть выполнено следующим образом: MDR\_BKP->RTC\_CS

Параметры RTC\_CLK функции

void BKP\_RTCclkSource(uint32\_t RTC\_CLK);

BKP\_RTC\_LSIclk – использовать LSI для RTC

BKP\_RTC\_LSEclk– использовать LSE для RTC

BKP\_RTC\_HSIclk – использовать HSI для RTC

BKP\_RTC\_HSEclk – использовать HSE для RTC

Параметры RTC\_IT функции

void BKP\_RTC\_ITConfig(uint32\_t RTC\_IT, FunctionalState NewState);

BKP\_RTC\_IT\_ALRF прерывание по совпадению с ALRМ

BKP\_RTC\_IT\_SECF прерывание по изменению счёта

BKP\_RTC\_IT\_OWF прерывание по переполнению

Параметры RTC\_FLAG функции

FlagStatus BKP\_RTC\_GetFlagStatus(uint32\_t RTC\_FLAG);

BKP\_RTC\_FLAG\_WEC – готовность к записи данных

BKP\_RTC\_FLAG\_OWF - переполнение

BKP\_RTC\_FLAG\_ALRF – совпадение с ALRM

BKP\_RTC\_FLAG\_SECF – изменение счёта

**Практическая часть**

**Задание 6.1.** Создайте проект по алгоритму из работы 1.

Добавьте в функцию SystemInit цикл, позволяющий задержать начало выполнения основной программы процессором на 5 секунд (файл system\_MDR32F9Qx.c).

Правильно определите используемый JTAG-разъем в файле MDR32F9Qx\_config.h.

Опишите процедуру инициализации модуля RST\_CLK, обеспечивающую переход процессора на генератор HSE на частоте 8МГц.

Опишите процедуру инициализации выводов контроллера для работы со светодиодами (PC0).

Опишите инициализацию таймера SysTick и загрузите в него значение для формирования секундных задержек (учтите сформированную частоту в рамках задания, не забудьте подать тактовый сигнал на SysTick).

Опишите процедуру обработчика прерывания от системного таймера. В теле обработчика опишите процедуру изменения состояния светодиода на противоположное. Вход в обработчик будет осуществляться раз в секунду, поэтому не нужно сразу выполнять операции вывода 1 и 0 и использовать задержки.

В основной программе обеспечьте тактирование используемых портов, инициализацию портов, глобальное разрешение прерываний.

Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте работу контроллера. Зафиксируйте реакцию системы в отчёте.

**Задание 6.2.** В обработчик прерывания от системного таймера добавьте следующие строки

|  |
| --- |
| struct tm \*tmstrct;  time\_t time;  char stroka[16];  MDR\_PORTC->RXTX^=0x2;  time = BKP\_RTC\_GetCounter();  tmstrct = localtime(&time);  strftime(stroka,16,"%H,%M,%S",tmstrct);  PrintString (stroka,5); |

Добавьте процедуру инициализации модуля часов реального времени, поместите в нее следующие строки. Добавьте вызов полученной процедуры в основной программе (до бесконечного цикла).

|  |
| --- |
| RST\_CLK\_PCLKcmd (RST\_CLK\_PCLK\_BKP,ENABLE);  PORT\_InitTypeDef Port\_struct;  Port\_struct.PORT\_Pin = PORT\_Pin\_6;  Port\_struct.PORT\_OE = PORT\_OE\_IN;  Port\_struct.PORT\_MODE= PORT\_MODE\_ANALOG;  PORT\_Init(MDR\_PORTE, &Port\_struct);  RST\_CLK\_LSEconfig(RST\_CLK\_LSE\_ON);  while (RST\_CLK\_LSEstatus()!=SUCCESS);  BKP\_RTC\_WaitForUpdate();  BKP\_RTCclkSource(BKP\_RTC\_LSIclk);  BKP\_RTC\_WaitForUpdate();  BKP\_RTC\_SetPrescaler(32768);  BKP\_RTC\_WaitForUpdate();  BKP\_RTC\_Enable(ENABLE); |

Проанализируйте код и прокомментируйте все строки обработчика прерывания и процедуры инициализации BKP. Назначение использования вывода PORTE6 посмотрите в документе со схемотехническим описанием отладочной платы.

Через утилиту Manage Run-Time Environment добавьте библиотеку ВКР (Drivers→BKP).

Добавьте в проект инициализацию LCD дисплея в соответствии методикой, изложенной в практической работе 4 (4.1, 4.2).

Подключите библиотеки: #include "MDR32F9Qx\_bkp.h", #include "time.h", #include "lcdstring.h", #include "mlt\_lcd.h", #include "font.h", #include "stdio.h"

Выведите номер бригады в правой верхней части экрана.

Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте работу контроллера, в том числе при отключении основного питания от платы. Зафиксируйте реакцию системы в отчёте.

**Задание 6.3.**

Измените код из задания 6.2 таким образом, чтобы контроллер выводил на экран информацию в виде, представленном в таблице ниже.

|  |
| --- |
| Номер бригады  ЧЧ:ММ  ДД.ММ.ГГГГ |

Добавьте в процедуру инициализации модуля BKP\_RTC разрешение прерывания по совпадению с ALRM от модуля, а в регистр ALRM занесите значение, на минуту опережающее текущее состояние счетчика (перед сборкой и загрузкой в контроллер).

Разрешите прерывания от модуля MDR\_BKP.

Добавьте обработчик прерывания MDR\_BKP, в котором выведите на экран сообщение о наступлении прерывания.

Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте работу контроллера, в том числе при отключении основного питания от платы. Зафиксируйте реакцию системы в отчёте.

**Задание 6.4.** Измените код из задания 6.3 следующим образом:

- проинициализируйте любую подключенную к МК кнопку;

- в бесконечном цикле программы опишите инициализацию структуры tmstrct и инициализацию таймера RTC при нажатии на кнопку значениями, близкими к реальным.

Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте работу контроллера, в том числе при отключении основного питания от платы. Зафиксируйте реакцию системы в отчёте.

**Задание 6.5.** Измените код из задания 6.4 следующим образом:

Глобально объявите переменную uint32\_t. В обработчике прерывания от системного счётчика опишите операцию инкремента над данной переменной, её вывод на экран, а также сохранение в одном из регистров BKP. В основной программе контроллера на этапе инициализации обеспечьте загрузку значения из выбранного регистра BKP в созданную переменную.

Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте работу контроллера, в том числе при отключении основного питания от платы, затем при сбросе (кнопка RESET) или при повторной загрузке программы в контроллер. Зафиксируйте реакцию системы в отчёте.

**Вопросы для самоконтроля**

1. Что представляет из себя модуль BKP и какие функции выполняет?
2. Какие функции выполняют регистры BKP->0E и BKP->0f?
3. От какого генератора тактируются часы реального времени МК?
4. Чем UCC отличается от BUCC?
5. Опишите регистры часов реального времени.
6. Как должен быть настроен предделитель частоты часов реального времени для подсчёта секунд при тактировании от LSE и почему?
7. Какие прерывания может формировать модуль BKP\_RTC?
8. Какую разрядность имеет счётчик RTC?
9. Какими датами будет ограничено применение библиотеки time.h (например, в случае переполнения счётчика)?
10. К чему может привести чтение значения счётчика RTC в переменную со знаком?