Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический

университет имени В.Ф. Уткина»

Кафедра «ЭВМ»

Отчет о практической работе №6

«Блок батарейного домена. Использование часов реального времени»

по дисциплине

«Специализированные ЭВМ»

Выполнили:

Студенты группы 045

Вашкулатов Н.А.

Анохин В.А.

Проверил:

доц. каф. ЭВМ Устюков Д.И.

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

**Ход работы**

**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.

Проанализируйте работу контроллера.

**Код программы:**

#include "MDR32F9Qx\_port.h"

#include "MDR32F9Qx\_rst\_clk.h"

#include "MDR32Fx.h"

static PORT\_InitTypeDef LedInit;

static PORT\_InitTypeDef InputInit;

const int delay = 8000000;

const uint8\_t LED\_ZERO = PORT\_Pin\_0;

const uint8\_t LED\_ONE = PORT\_Pin\_1;

void Delay(int del)

{

for (int i=0; i<del; i++);

}

void LedConfig (void)

{

LedInit.PORT\_Pin = (PORT\_Pin\_0 | PORT\_Pin\_1);

LedInit.PORT\_FUNC = PORT\_FUNC\_PORT;

LedInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

LedInit.PORT\_OE = PORT\_OE\_OUT;

LedInit.PORT\_SPEED = PORT\_SPEED\_SLOW;

PORT\_Init(MDR\_PORTC, &LedInit);

}

void CPU\_Config(){

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

while (RST\_CLK\_HSEstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul1);

RST\_CLK\_CPU\_PLLcmd(ENABLE);

while (RST\_CLK\_CPU\_PLLstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

}

void changeLedStatus(uint8\_t led){

if(PORT\_ReadInputDataBit(MDR\_PORTC, led)){

PORT\_ResetBits(MDR\_PORTC, led);

} else {

PORT\_SetBits(MDR\_PORTC, led);

}

}

void IntConfig(){

NVIC\_EnableIRQ(SysTick\_IRQn);

SysTick\_Config(8000000);

\_\_enable\_irq();

}

void SysTick\_Handler(void){

changeLedStatus(LED\_ZERO);

}

int main(){

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

CPU\_Config();

LedConfig();

IntConfig();

uint32\_t count = 0;

while(1) { }

}

После запуска программы каждую секунду моргают два светодиода.

**Задание 6.2.** Добавьте процедуру инициализации модуля часов реального времени, поместите в нее следующие строки. Добавьте вызов полученной процедуры в основной программе (до бесконечного цикла).

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.

Проанализируйте работу контроллера, в том числе при отключении основного питания от платы.

**Код программы:**

#include "MDR32F9Qx\_port.h"

#include "MDR32F9Qx\_rst\_clk.h"

#include "MDR32Fx.h"

#include "MDR32F9Qx\_bkp.h"

#include "time.h"

#include "lcdstring.h"

#include "mlt\_lcd.h"

#include "font.h"

#include "stdio.h"

static PORT\_InitTypeDef LedInit;

static PORT\_InitTypeDef InputInit;

PORT\_InitTypeDef PortStruct;

const int delay = 8000000;

const uint8\_t LED\_ZERO = PORT\_Pin\_0;

const uint8\_t LED\_ONE = PORT\_Pin\_1;

void LCDPins (void)

{

PortStruct.PORT\_Pin = PORT\_Pin\_0|PORT\_Pin\_1|PORT\_Pin\_2|PORT\_Pin\_3|PORT\_Pin\_4|PORT\_Pin\_5;

PortStruct.PORT\_OE = PORT\_OE\_IN;

PortStruct.PORT\_FUNC = PORT\_FUNC\_PORT;

PortStruct.PORT\_SPEED = PORT\_SPEED\_SLOW;

PortStruct.PORT\_MODE = PORT\_MODE\_DIGITAL;

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

PortStruct.PORT\_Pin = PORT\_Pin\_2|PORT\_Pin\_3;

PORT\_Init(MDR\_PORTF,&PortStruct);

PortStruct.PORT\_OE = PORT\_OE\_OUT;

PortStruct.PORT\_Pin = PORT\_Pin\_7|PORT\_Pin\_8|PORT\_Pin\_9|PORT\_Pin\_10;

PORT\_Init(MDR\_PORTB,&PortStruct);

PortStruct.PORT\_Pin = PORT\_Pin\_0|PORT\_Pin\_1;

PORT\_Init(MDR\_PORTC,&PortStruct);

}

void LCDStart(void)

{

uint8\_t status;

LcdInit();

do

{

status = ReadStatus(1);

}while ((status&0x80)!=0x00);

do

{

status = ReadStatus(2);

}while((status&0x80)!=0x00);

DispOn (1);

DispOn (2);

LcdClearChip(1);

LcdClearChip(2);

}

void setPixels(uint8\_t x, uint8\_t y, uint8\_t data){

uint8\_t chip = (x / 64) + 1;

uint8\_t page = y / 8;

uint8\_t address = x % 64;

SetPage(chip,page);

SetAdress(chip, address);

ReadData(chip);

uint8\_t mask = 1 << (y % 8);

uint8\_t cellData = ReadData(chip);

if (data) {

cellData |= mask;

} else {

cellData &= ~mask;

}

SetPage(chip,page);

SetAdress(chip, address);

WriteData(chip, cellData);

}

void LedConfig (void)

{

LedInit.PORT\_Pin = (PORT\_Pin\_0 | PORT\_Pin\_1);

LedInit.PORT\_FUNC = PORT\_FUNC\_PORT;

LedInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

LedInit.PORT\_OE = PORT\_OE\_OUT;

LedInit.PORT\_SPEED = PORT\_SPEED\_SLOW;

PORT\_Init(MDR\_PORTC, &LedInit);

}

void TimeConfig(void){

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);

}

void CPU\_Config(){

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

while (RST\_CLK\_HSEstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul1);

RST\_CLK\_CPU\_PLLcmd(ENABLE);

while (RST\_CLK\_CPU\_PLLstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

}

void changeLedStatus(uint8\_t led){

if(PORT\_ReadInputDataBit(MDR\_PORTC, led)){

PORT\_ResetBits(MDR\_PORTC, led);

} else {

PORT\_SetBits(MDR\_PORTC, led);

}

}

void IntConfig(){

NVIC\_EnableIRQ(SysTick\_IRQn);

SysTick\_Config(8000000);

\_\_enable\_irq();

}

void SysTick\_Handler(void){

struct tm \*tmstrct;

time\_t time;

char stroka[16];

MDR\_PORTC->RXTX^=0x2;

time = BKP\_RTC\_GetCounter();

tmstrct = localtime(&time);

//output string

strftime(stroka,16,"%H,%M,%S",tmstrct);

PrintString(stroka,7);

}

int main(){

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

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

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

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

CPU\_Config();

LedConfig();

LCDPins();

LCDStart();

TimeConfig();

IntConfig();

uint8\_t \*brigada[9] = {{cyr\_B},{cyr\_r} ,{cyr\_i},{cyr\_g}, {cyr\_a},{cyr\_d},{cyr\_a},{sym\_sp}, {dig\_4}};

LcdScrollString (brigada, 0, 9, 8);

while(1){ }

}

На рисунке 1 показан результат выполнения программы на микроконтроллере.

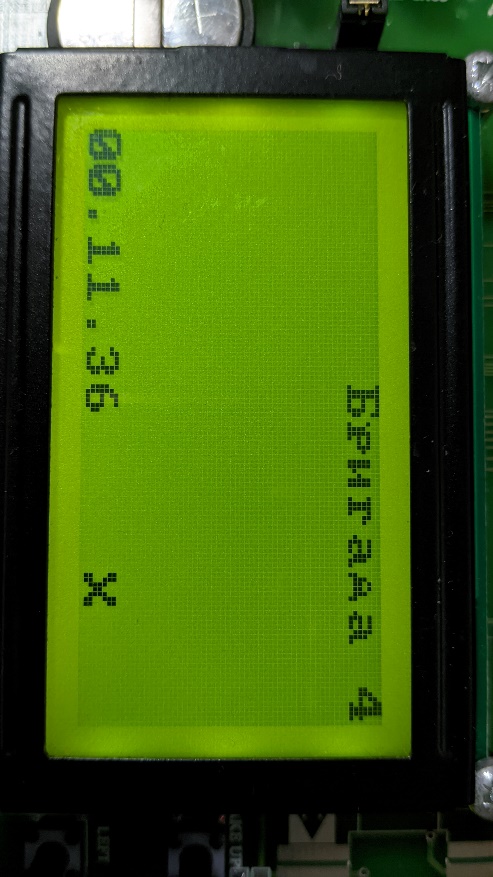


Рисунок 1 – Отображение времени на дисплее

**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.

Проанализируйте работу контроллера, в том числе при отключении основного питания от платы.

**Код программы:**

void IntConfig(){

NVIC\_EnableIRQ(SysTick\_IRQn);

NVIC\_EnableIRQ(BACKUP\_IRQn);

SysTick\_Config(8000000);

\_\_enable\_irq();

}

void SysTick\_Handler(void){

struct tm \*tmstrct;

time\_t time;

char stroka[16];

MDR\_PORTC->RXTX^=0x2;

time = BKP\_RTC\_GetCounter();

tmstrct = localtime(&time);

//output string

strftime(stroka,16," %H,%M",tmstrct);

PrintString(stroka,6);

char stroka2[16];strftime(stroka2,16," %d,%m,%y",tmstrct);

PrintString(stroka2,7);

}

void BACKUP\_IRQHandler(void){

char intr[9];

sprintf(intr, "%s", "Interrupt");

PrintString(intr, 1);

}

int main(){

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

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

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

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

CPU\_Config();

LedConfig();

LCDPins();

LCDStart();

TimeConfig();

IntConfig();

BKP\_RTC\_ITConfig(BKP\_RTC\_IT\_ALRF, ENABLE);

BKP\_RTC\_SetAlarm(BKP\_RTC\_GetCounter() + 10);

uint8\_t \*brigada[9] = {{cyr\_B},{cyr\_r} ,{cyr\_i},{cyr\_g}, {cyr\_a},{cyr\_d},{cyr\_a},{sym\_sp}, {dig\_4}};

LcdScrollString (brigada, 0, 8, 8);

uint32\_t count = 0;

while(1) { }

}

На рисунках 2 – 3 показан результат выполнения программы на микроконтроллере. Через 10 секунд после запуска программы возникает прерывание. Отображаемые часы останавливаются. При перезагрузке часы на дисплее снова отсчитывают 10 секунд. Начальное значение времени имеет значение большее, чем на момент перезагрузки, т.к. внутренние часы продолжают идти.

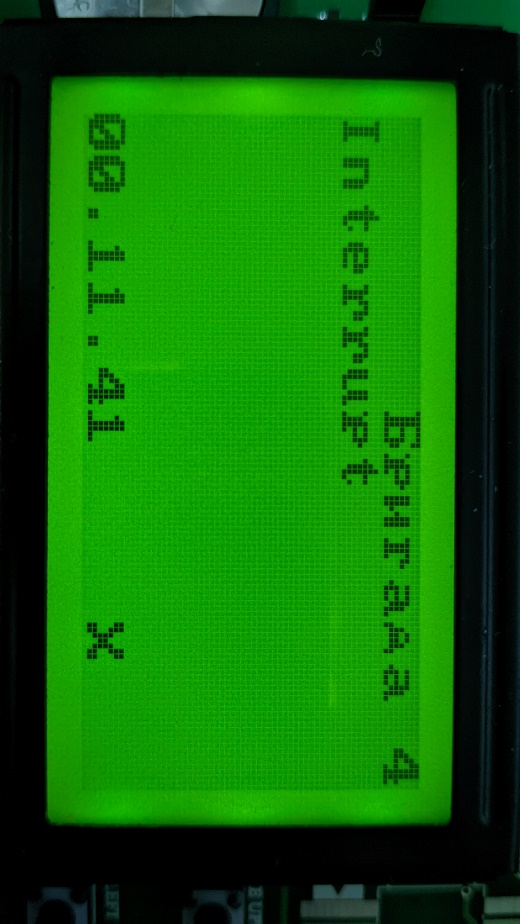


Рисунок 2 – Добавлено прерывание по таймеру

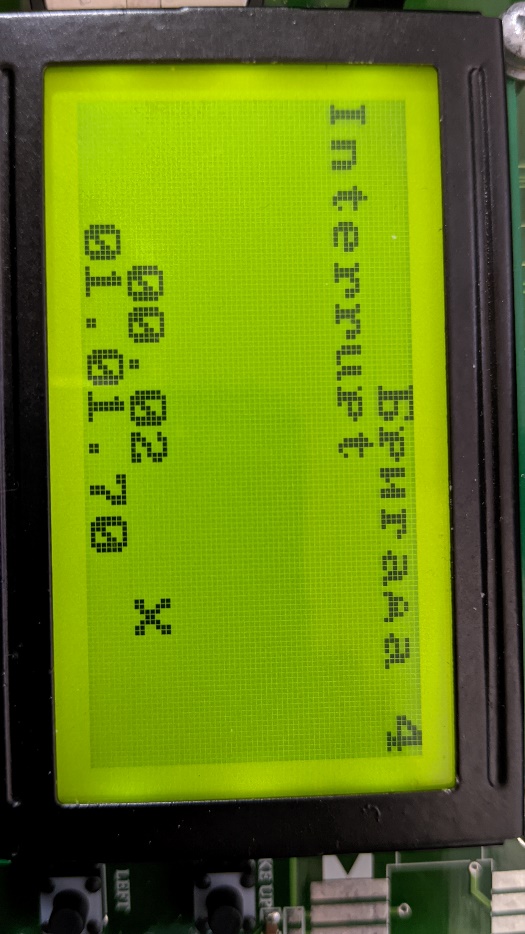


Рисунок 3 – Отображение даты и времени

**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.

Проанализируйте работу контроллера, в том числе при отключении основного питания от платы.

На рисунках 4 – 5 показан результат выполнения программы на микроконтроллере. Отсчет времени после первого запуска программы начинается с даты 01.01.70. После нажатия на кнопку устанавливается заданное нами время. При выключении и включении питания часы продолжают отсчет текущего времени.



Рисунок 4 – Отображение текущей даты и времени



Рисунок 5 – Сброс часов по нажатию кнопки

**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.

Проанализируйте работу контроллера, в том числе при отключении основного питания от платы, затем при сбросе (кнопка RESET) или при повторной загрузке программы в контроллер.

На рисунках 6 – 7 показан результат выполнения программы на микроконтроллере. После первого запуска программы отсчет переменной начинается с нуля. При выключении и включении питания переменная начинает отсчет со значения, которое было до выключения питания.



Рисунок 6 – Отображение счетчика

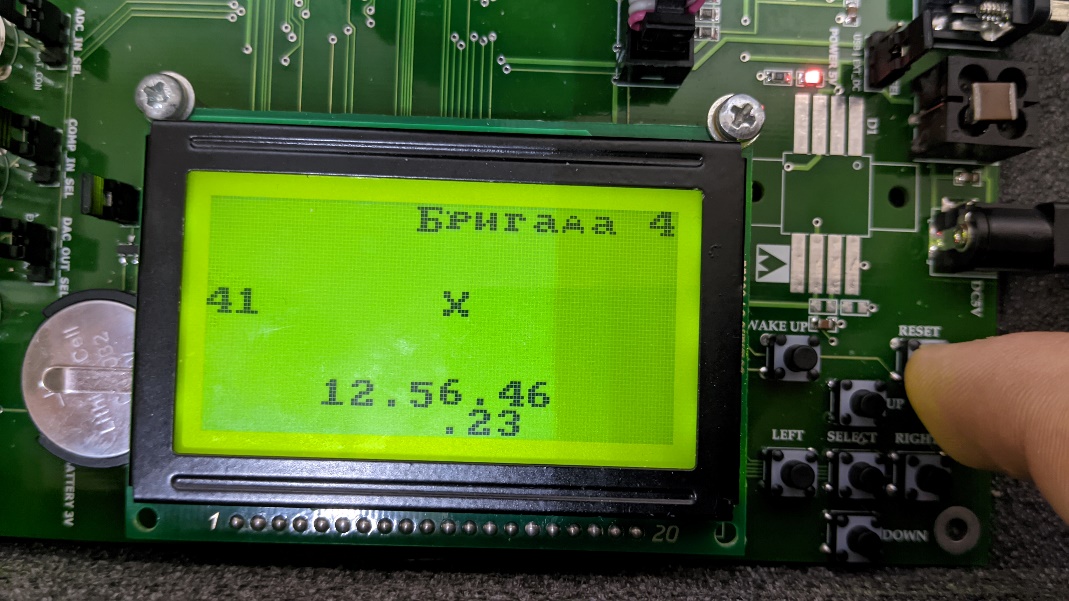


Рисунок 7 – Сохранение счетчика после перезагрузки

**Вывод**: в ходе выполнения работы были ознакомлены с модулем батарейного домена и особенностями использования часов реального времени на МК 1986BE92.