

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И ТЕХНОЛОГИЙ

КАФЕДРА КОМПЬЮТЕРНЫХ СИСТЕМ И ПРОГРАММНЫХ ТЕХНОЛОГИЙ

## **Отчет по лабораторному практикуму**

**Дисциплина: Аппаратные платформы встраиваемых систем**

Выполнили студенты гр.13541/1:

---

(подпись) Егоров Д.Е.

---

(подпись) Чупин Н.С.

Руководитель:

---

(подпись) Васильев А.Е.

Санкт-Петербург  
2019 г.

# Содержание

<b>1</b>	<b>Лабораторная работа №1 «IAR, CMSIS, SPL, GPIO»</b>	<b>2</b>
1.1	Цель работы . . . . .	2
1.2	Программа работы . . . . .	2
1.3	Алгоритм переключения светодиодов . . . . .	4
1.4	Ход работы . . . . .	4
1.5	Выводы . . . . .	7
<b>2</b>	<b>Лабораторная работа №2 «Системы тайминга и прерываний»</b>	<b>8</b>
2.1	Цель работы . . . . .	8
2.2	Программа работы . . . . .	8
2.3	Ход работы . . . . .	8

# Лабораторная работа №1 «IAR, CMSIS, SPL, GPIO»

## 1.1 Цель работы

Ознакомиться с интегрированной средой разработки IAR Embedded Workbench for ARM, а также функциями CMSIS и MDRSPL, получить навыки создания и отладки программного обеспечения для целевой платформы на примере разработки программ, взаимодействующих с портами ввода-вывода.

## 1.2 Программа работы

1. Создать проект-заготовку для последующих лабораторных работ. Листинг демонстрационной программы приведен ниже.
2. Подключить к проекту библиотеку CMSIS. Объяснить назначение и содержание файлов библиотеки. Объяснить назначение и содержание файла startup\_MDR32F9Qx.c
3. Подключить к проекту библиотеку MDR32F9Qx Standart Peripherals Library.
4. Настроить параметры отладчика для запуска демонстрационного примера на отладочной плате. Собрать проект, продемонстрировать его исполнение «по шагам».
5. Разработать программу, включающую светодиоды на плате при нажатии кнопок; алгоритм согласовать с руководителем.

Код демонстрационного примера приведен в листинге 1.1.

Листинг 1.1: Код демонстрационного примера

```
1 #include "MDR32Fx.h"
2 #include "MDR32F9Qx_config.h"
3 #include "MDR32F9Qx_port.h"
4 #include "MDR32F9Qx_rst_clk.h"
5
6 #define DELAY 500000
7
8 static void Delay( uint32_t delay );
9
10 void frq_init(void);
11 static void PeriphInit( void );
12 int check_btn(void);
13
14 void main(){
15     frq_init();
16     PeriphInit();
17     while(1){
18         PORT_SetBits(MDR_PORTC, PORT_Pin_0);
19         Delay( DELAY );
20         PORT_ResetBits(MDR_PORTC, PORT_Pin_0);
21         Delay( DELAY );
22     }
23 }
24
25 void frq_init(void)
26 {
27     MDR_RST_CLK->HS_CONTROL = 0x1; // Enable HSE oscillator
28     /* wait while HSE startup */
```

```

29 while (MDR_RST_CLK->CLOCK_STATUS == 0x00) __NOP();
30 MDR_RST_CLK->CPU_CLOCK = 0x102; // switch to HSE (8 MHz)
31 SystemCoreClockUpdate();
32 }
33 static void Delay( uint32_t delay ){
34     if (PORT_ReadInputDataBit(MDR_PORTB, PORT_Pin_5) == Bit_SET) delay *= 2;
35     if (PORT_ReadInputDataBit(MDR_PORTC, PORT_Pin_1) == Bit_SET) delay /= 2;
36     while( --delay ){
37         __NOP();
38     }
39 }
40 static void PeriphInit( void )
41 {
42     PORT_InitTypeDef PORT_InitStruct;
43
44     //Светодиоды
45     /* Включение тактирования порта C */
46     RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC, ENABLE);
47     /* Настройка порта C для вывода в дискретном режиме. */
48     PORT_InitStruct.PORT_OE      = PORT_OE_OUT;
49     PORT_InitStruct.PORT_FUNC     = PORT_FUNC_PORT;
50     PORT_InitStruct.PORT_MODE     = PORT_MODE_DIGITAL;
51     PORT_InitStruct.PORT_SPEED    = PORT_SPEED_SLOW;
52     PORT_InitStruct.PORT_PULL_UP  = PORT_PULL_UP_OFF;
53     PORT_InitStruct.PORT_PULL_DOWN = PORT_PULL_DOWN_OFF;
54     PORT_InitStruct.PORT_PD_SHM   = PORT_PD_SHM_OFF;
55     PORT_InitStruct.PORT_PD       = PORT_PD_DRIVER;
56     PORT_InitStruct.PORT_GFEN     = PORT_GFEN_OFF;
57     PORT_InitStruct.PORT_Pin      = PORT_Pin_0 | PORT_Pin_1;
58
59     PORT_Init(MDR_PORTC, &PORT_InitStruct);
60
61     //Кнопка SELECT
62     //Тактирование порта уже включено
63     /* Настройка порта C для входа в дискретном режиме. */
64     PORT_InitStruct.PORT_OE      = PORT_OE_IN;
65     PORT_InitStruct.PORT_FUNC     = PORT_FUNC_PORT;
66     PORT_InitStruct.PORT_MODE     = PORT_MODE_DIGITAL;
67     PORT_InitStruct.PORT_SPEED    = PORT_SPEED_SLOW;
68     PORT_InitStruct.PORT_PULL_UP  = PORT_PULL_UP_OFF;
69     PORT_InitStruct.PORT_PULL_DOWN = PORT_PULL_DOWN_OFF;
70     PORT_InitStruct.PORT_PD_SHM   = PORT_PD_SHM_OFF;
71     PORT_InitStruct.PORT_PD       = PORT_PD_DRIVER;
72     PORT_InitStruct.PORT_GFEN     = PORT_GFEN_OFF;
73     PORT_InitStruct.PORT_Pin      = PORT_Pin_2;
74
75     PORT_Init(MDR_PORTC, &PORT_InitStruct);
76
77     //Кнопки UP RIGHT
78     /* Включение тактирования порта B */
79     RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTB, ENABLE);
80     /* Настройка порта B для входа в дискретном режиме. */
81     PORT_InitStruct.PORT_OE      = PORT_OE_IN;
82     PORT_InitStruct.PORT_FUNC     = PORT_FUNC_PORT;
83     PORT_InitStruct.PORT_MODE     = PORT_MODE_DIGITAL;
84     PORT_InitStruct.PORT_SPEED    = PORT_SPEED_SLOW;
85     PORT_InitStruct.PORT_PULL_UP  = PORT_PULL_UP_OFF;
86     PORT_InitStruct.PORT_PULL_DOWN = PORT_PULL_DOWN_OFF;
87     PORT_InitStruct.PORT_PD_SHM   = PORT_PD_SHM_OFF;
88     PORT_InitStruct.PORT_PD       = PORT_PD_DRIVER;
89     PORT_InitStruct.PORT_GFEN     = PORT_GFEN_OFF;
90     PORT_InitStruct.PORT_Pin      = PORT_Pin_5 | PORT_Pin_6;
91
92     PORT_Init(MDR_PORTB, &PORT_InitStruct);
93
94     //Кнопки DOWN LEFT

```

```

95  /* Включение тактирования порта E */
96  RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTE, ENABLE);
97  /* Настройка порта E для входа в дискретном режиме. */
98  PORT_InitStruct.PORT_OE      = PORT_OE_IN;
99  PORT_InitStruct.PORT_FUNC    = PORT_FUNC_PORT;
100  PORT_InitStruct.PORT_MODE    = PORT_MODE_DIGITAL;
101  PORT_InitStruct.PORT_SPEED   = PORT_SPEED_SLOW;
102  PORT_InitStruct.PORT_PULL_UP = PORT_PULL_UP_OFF;
103  PORT_InitStruct.PORT_PULL_DOWN = PORT_PULL_DOWN_OFF;
104  PORT_InitStruct.PORT_PD_SHM  = PORT_PD_SHM_OFF;
105  PORT_InitStruct.PORT_PD      = PORT_PD_DRIVER;
106  PORT_InitStruct.PORT_GFEN    = PORT_GFEN_OFF;
107  PORT_InitStruct.PORT_Pin     = PORT_Pin_1 | PORT_Pin_3;
108  PORT_Init(MDR_PORTE, &PORT_InitStruct);
109
110 }

```

### 1.3 Алгоритм переключения светодиодов

По нажатию кнопки SELECT, двухразрядное двоичное число, отображаемое светодиодами должно инкрементироваться.

Конечный автомат состояний программы представлен на рисунке 1.1, где st0 – состояние ожидания прерывания, а при переходе в состояние st1 вызывается подпрограмма обработки этого прерывания.



Рис. 1.1: Схема конечного автомата программы

### 1.4 Ход работы

После настройки среды разработки IAR Embedded Workbench for ARM для работы с микросхемой Milandr, подключения необходимых библиотек и запуска демонстрационного проекта, код программы был запущен и протестирован на работоспособность. Затем были внесены изменения в соответствии с заданием преподавателя. Для этого был разработан конечный автомат, схема которого приведена выше. Код программы, разработанной в соответствии с индивидуальным заданием руководителя приведен в листинге 1.2.

Листинг 1.2: Код демонстрационного примера

```

1  /* Includes ----- */
2  #include "MDR32Fx.h"
3  #include "MDR32F9Qx_config.h"
4  #include "MDR32F9Qx_port.h"
5  #include "MDR32F9Qx_rst_clk.h"
6
7  /* Private typedef ----- */
8  /* Private define ----- */
9  #define DELAY_MIN 250000
10 #define DELAY_MAX 500000
11 #define step 10000
12 #define contact_bounce 1000
13 /* Private macro ----- */

```

```

14 /* Private variables ----- */
15 static volatile int btn_state = 0;
16 static int state = 0;
17 static int past_state = 0;
18 /* Private function prototypes ----- */
19 static void Delay( uint32_t delay );
20 static void PeriphInit( void );
21 static int Poll();
22 static void SwitchState();
23 static void LightUpLEDs();
24 /* Private functions ----- */
25
26
27 int main()
28 {
29     PeriphInit();
30     while(1)
31     {
32         if( Poll() ) //если состояние кнопки отжата => нажата
33         {
34             SwitchState(); //изменение состояние системы
35             LightUpLEDs(); //реакция на изменение состояния
36         }
37     }
38 }
39
40 static void LightUpLEDs()
41 {
42     switch (state)
43     {
44         case 0 :
45         {
46             //0 0
47             PORT_ResetBits( MDR_PORTC, PORT_Pin_0 );
48             PORT_ResetBits( MDR_PORTC, PORT_Pin_1 );
49             break;
50         }
51         case 1 :
52         {
53             //0 1
54             PORT_ResetBits( MDR_PORTC, PORT_Pin_0 );
55             PORT_SetBits( MDR_PORTC, PORT_Pin_1 );
56             break;
57         }
58         case 2 :
59         {
60             //1 0
61             PORT_SetBits( MDR_PORTC, PORT_Pin_0 );
62             PORT_ResetBits( MDR_PORTC, PORT_Pin_1 );
63             break;
64         }
65         case 3 :
66         {
67             //1 1
68             PORT_SetBits( MDR_PORTC, PORT_Pin_0 );
69             PORT_SetBits( MDR_PORTC, PORT_Pin_1 );
70             break;
71         }
72     }
73 }
74
75 static void SwitchState()
76 {
77     if (state == 3) state = 0;
78     else state++;
79 }

```

```

80
81 static int Poll() //опрос кнопки
82 {
83     btn_state = !PORT_ReadInputDataBit(MDR_PORTC, PORT_Pin_2);
84     if (btn_state != past_state)
85     {
86         past_state = btn_state;
87         if (btn_state)
88             return 1;
89     }
90     return 0;
91 }
92
93 static void Delay( uint32_t delay )
94 {
95     while( --delay )
96     {
97         __NOP();
98     }
99 }
100
101 static void PeriphInit( void )
102 {
103     PORT_InitTypeDef PORT_InitStruct;
104     /* Включение тактирования порта C */
105     RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC | RST_CLK_PCLK_PORTB | RST_CLK_PCLK_PORTE, ENABLE)
106     ;
107     /* Настройка порта C.0 для вывода в дискретном режиме. */
108     PORT_InitStruct.PORT_Pin = PORT_Pin_1 | PORT_Pin_0;
109     PORT_InitStruct.PORT_OE = PORT_OE_OUT;
110     PORT_InitStruct.PORT_FUNC = PORT_FUNC_PORT;
111     PORT_InitStruct.PORT_MODE = PORT_MODE_DIGITAL;
112     PORT_InitStruct.PORT_SPEED = PORT_SPEED_SLOW;
113     PORT_InitStruct.PORT_PULL_UP = PORT_PULL_UP_OFF;
114     PORT_InitStruct.PORT_PULL_DOWN = PORT_PULL_DOWN_OFF;
115     PORT_InitStruct.PORT_PD_SHM = PORT_PD_SHM_OFF;
116     PORT_InitStruct.PORT_PD = PORT_PD_DRIVER;
117     PORT_InitStruct.PORT_GFEN = PORT_GFEN_OFF;
118     PORT_Init(MDR_PORTC, &PORT_InitStruct);
119
120     PORT_InitStruct.PORT_Pin = PORT_Pin_2;
121     PORT_InitStruct.PORT_OE = PORT_OE_IN;
122     PORT_Init(MDR_PORTC, &PORT_InitStruct);
123     PORT_InitStruct.PORT_Pin = PORT_Pin_1;
124     PORT_Init(MDR_PORTC, &PORT_InitStruct);
125     PORT_InitStruct.PORT_Pin = PORT_Pin_5;
126     PORT_Init(MDR_PORTB, &PORT_InitStruct);
127 }
128
129 #if ( USE_ASSERT_INFO == 1 )
130 void assert_failed(uint32_t file_id, uint32_t line)
131 {
132     while (1)
133     {
134     }
135 }
136 #elif ( USE_ASSERT_INFO == 2 )
137 void assert_failed(uint32_t file_id, uint32_t line, const uint8_t* expr)
138 {
139     while (1)
140     {
141     }
142 }
143 #endif /* USE_ASSERT_INFO */
144

```

## 1.5 Выводы

По итогам лабораторной работы было произведено ознакомление с интегрированной средой разработки IAR Embedded Workbench for ARM, а также функциями CMSIS и MDRSPL. Также были получены навыки создания и отладки программного обеспечения для целевой платформы на примере разработки программ, взаимодействующих с портами ввода-вывода.

Была реализована система управления миганием светодиодов. Отличительной чертой данной реализации является конечный автомат который обрабатывает нажатие кнопки, и при помощи программно реализованного триггера переключает состояния системы, сравнивая текущее её состояние с сохраненным предыдущим.

Улучшение данной системы возможно путем использования обработчика прерываний. Это позволит оптимизировать работу системы, ввиду отсутствия лишней проверки на нажатие кнопки во время её работы.



# Лабораторная работа №2 «Системы тайминга и прерываний»

## 2.1 Цель работы

Развитие навыков разработки встраиваемых приложений реального времени.

## 2.2 Программа работы

1. Изучит листинг программы, приведенной ниже. Собрать на его основе проект и проанализировать его работу.
2. Настроить таймеры общего назначения 1 и 2 и обработчики запросов прерываний от них следующим образом:
  - (a) период счета таймера 2 много больше, чем период счета таймера 1;
  - (b) время обслуживания запроса прерывания от таймера 2 много больше, чем от таймера 1 (в обработчике запросов прерывания от таймера 2 организовать длительный 'пустой' цикл или иные продолжительные вычисления);
  - (c) в обработчике запроса прерывания от таймера 1 выполнять инверсию бита заданного порта; в обработчике запроса прерывания от таймера 2 при входе в обработчик выполнять установку другого бита порта, при выходе - его сброс;
  - (d) приоритеты прерываний установить равными.
3. Зафиксировать характерные осциллограммы и объяснить поведение системы.
4. Добавить к проекту возможность смены приоритетов прерываний таймеров по сигналу внешнего прерывания. Зафиксировать осциллограммы и объяснить поведение системы.
5. Разработать систему измерения частоты следования импульсов внешнего сигнала (в качестве источника использовать внешний генератор импульсов или ФИД).
6. Разработать простейший осциллограф: в заданном темпе регистрировать значения входного аналогового сигнала и отображать его на ЖКИ.
7. Разработать простейший генератор аналогового периодического сигнала.

## 2.3 Ход работы

В файле MDR32F9Qx\_it.c содержатся процедуры обработки прерываний. Особенность написания кода в данной лабораторной работе заключается в том, что основной цикл программы не содержит практически ни одного вызова: при запуске инициализируются таймеры-счетчики, а вся логика описана в процедурах обработки прерываний.

В листинге N приведена реализация настройки обработки прерываний таймеров-счетчиков 1, 2. Данный код относится к процедуре InitTimers() в файле main.c.

Непосредственно реализация обработки описана в файле MDR32F9Qx\_it.c и имеет следующий вид (Листинг N):

При данной конфигурации проект был собран и запущен на плате. Зафиксированная осциллограмма продемонстрирована на рисунке N.

На приведенном рисунке N мы видим, что время обслуживания запроса прерывания от таймера 2 много больше, чем от таймера 1, но поскольку приоритет прерывания первого таймера не выше второго, то во

### Листинг 2.1: Код настройки обработки прерываний

```

111 // Установка приоритетов для T1 и T2
112 NVIC_SetPriorityGrouping(3);
113 NVIC_SetPriority(Timer1_IRQn, 0);
114 NVIC_SetPriority(Timer2_IRQn, 0);
115 // Разрешение прерываний для T1 и T2
116 NVIC_EnableIRQ(Timer1_IRQn);
117 NVIC_EnableIRQ(Timer2_IRQn);
118
119 // Включение таймера 1 и 2
120 TIMER_Cmd(MDR_TIMER1, ENABLE);
121
122 // Установка периода для T1
123 TIMER_CntInitStruct.TIMER_Period = 31250;
124 // Установка периода для T2
125 TIMER_CntInitStruct.TIMER_Period = 62500;

```

### Листинг 2.2: Алгоритм обработки прерываний

```

248 /*****
249 * Function Name : Timer1_IRQHandler
250 * Description   : This function handles Timer1 global interrupt request.
251 * Input        : None
252 * Output       : None
253 * Return       : None
254 *****/
255 void Timer1_IRQHandler(void)
256 {
257     TIMER_ClearFlag(MDR_TIMER1, TIMER_STATUS_CNT_ARR);
258
259     uint32_t rxtx = PORT_ReadInputData(MDR_PORTC);
260     uint32_t res = rxtx ^ PORT_Pin_0;
261     PORT_Write( MDR_PORTC, res );
262 }
263 /*****
264 * Function Name : Timer2_IRQHandler
265 * Description   : This function handles Timer2 global interrupt request.
266 * Input        : None
267 * Output       : None
268 * Return       : None
269 *****/
270 void Timer2_IRQHandler(void)
271 {
272     PORT_SetBits( MDR_PORTC, PORT_Pin_1 );
273     Delay(2000000);
274     TIMER_ClearFlag(MDR_TIMER2, TIMER_STATUS_CNT_ARR);
275     PORT_ResetBits( MDR_PORTC, PORT_Pin_1 );
276 }

```

время обработки прерывания таймера-счетчика 2, прерывание таймера-счетчика 1, хотя и возникает, но не обслуживается. С целью рассмотрения поведения системы при иных значениях приоритетности добавим в основной цикл файла `main.c` возможность смены приоритетов во время работы программы. При подобном подходе, в случае установки более высокого приоритета для обработки прерывания таймера-счетчика 1, последнее будет обработано даже в том случае, если в данный момент не закончилась более длительная по времени обработка прерывания таймера-счетчика 2.

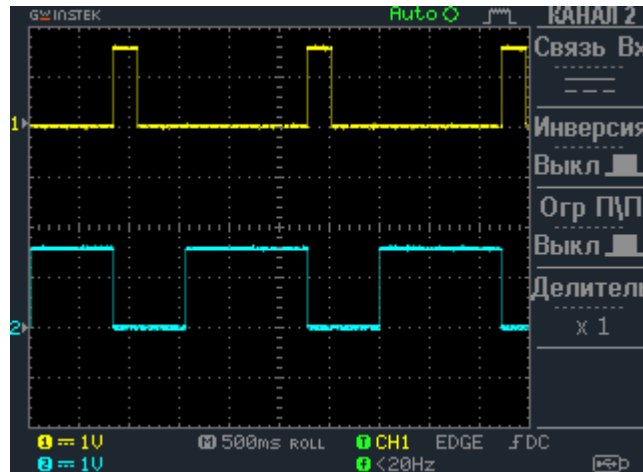


Рис. 2.1: Осциллограмма обработки прерываний при равных приоритетах обработки прерываний

Листинг 2.3: Код настройки обработки прерываний

```

32 while(1)
33 {
34     if (PORT_ReadInputDataBit(MDR_PORTE, PORT_Pin_1))
35     {
36         NVIC_SetPriority(Timer1_IRQn, 1);
37         NVIC_SetPriority(Timer2_IRQn, 0);
38     }
39     else
40     {
41         NVIC_SetPriority(Timer1_IRQn, 0);
42         NVIC_SetPriority(Timer2_IRQn, 1);
43     }
44 }

```

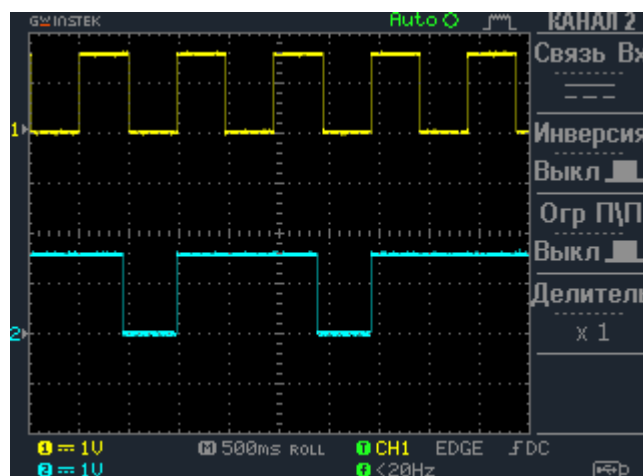


Рис. 2.2: Осциллограмма обработки прерываний при более высоком приоритете таймера-счетчика 1