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) Никитин А.Е.

(подпись) Баринов М.С.

Руководитель:

(подпись) Васильев А.Е.

Санкт-Петербург
2019 г.

Содержание

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

```
#include "MDR32Fx.h"
#include "MDR32F9Qx_config.h"
#include "MDR32F9Qx_port.h"
#include "MDR32F9Qx_rst_clk.h"

#define DELAY 500000

static void Delay( uint32_t delay );

void frq_init(void);
static void PeriphInit( void );
int check_btn(void);

void main(){
  frq_init();
  PeriphInit();
  while(1){
    PORT_SetBits(MDR_PORTC, PORT_Pin_0);
    Delay( DELAY );
    PORT_ResetBits(MDR_PORTC, PORT_Pin_0);
    Delay( DELAY );
  }
}

void frq_init(void)
{
  MDR_RST_CLK->HS_CONTROL = 0x1; // Enable HSE oscillator
  /* 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_PORTE,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 инкреметироваться.

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.

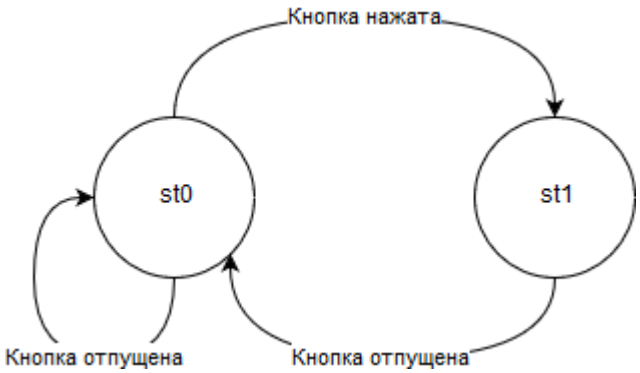

Рис. 1.1: Схема конечного автомата программы

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 -------------------------------------------------------------*/
```

```
/* Private variables ---------------------------------------------------------*/
static volatile int btn_state = 0;
static int state = 0;
static int past_state = 0;
/* Private function prototypes -----------------------------------------------*/
static void Delay( uint32_t delay );
static void PeriphInit( void );
static int Poll();
static void SwitchState();
static void LightUpLEDs();
/* Private functions ---------------------------------------------------------*/


int main()
{
    PeriphInit();
    while(1)
    {
        if(Poll()) //если состояние кнопки отжата => нажата
        {
            SwitchState(); //изменение состояние системы
            LightUpLEDs(); //реакция на изменение состояния
        }
    }
}

static void LightUpLEDs()
{
    switch (state)
    {
        case 0 :
        {
            //0 0
            PORT_ResetBits( MDR_PORTC, PORT_Pin_0 );
            PORT_ResetBits( MDR_PORTC, PORT_Pin_1 );
            break;
        }
        case 1 :
        {
            //0 1
            PORT_ResetBits( MDR_PORTC, PORT_Pin_0 );
            PORT_SetBits( MDR_PORTC, PORT_Pin_1 );
            break;
        }
        case 2 :
        {
            //1 0
            PORT_SetBits( MDR_PORTC, PORT_Pin_0 );
            PORT_ResetBits( MDR_PORTC, PORT_Pin_1 );
            break;
        }
        case 3 :
        {
            //1 1
            PORT_SetBits( MDR_PORTC, PORT_Pin_0 );
            PORT_SetBits( MDR_PORTC, PORT_Pin_1 );
            break;
        }
    }
}

static void SwitchState()
{
   if (state == 3) state = 0;
   else state++;
}
```

```
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     /* Настройка порта C.0 для вывода в дискретном режиме. */
107     PORT_InitStruct.PORT_Pin        = PORT_Pin_1 | PORT_Pin_0;
108     PORT_InitStruct.PORT_OE         = PORT_OE_OUT;
109     PORT_InitStruct.PORT_FUNC       = PORT_FUNC_PORT;
110     PORT_InitStruct.PORT_MODE       = PORT_MODE_DIGITAL;
111     PORT_InitStruct.PORT_SPEED      = PORT_SPEED_SLOW;
112     PORT_InitStruct.PORT_PULL_UP    = PORT_PULL_UP_OFF;
113     PORT_InitStruct.PORT_PULL_DOWN = PORT_PULL_DOWN_OFF;
114     PORT_InitStruct.PORT_PD_SHM     = PORT_PD_SHM_OFF;
115     PORT_InitStruct.PORT_PD         = PORT_PD_DRIVER;
116     PORT_InitStruct.PORT_GFEN       = PORT_GFEN_OFF;
117     PORT_Init(MDR_PORTC, &PORT_InitStruct);
118
119     PORT_InitStruct.PORT_Pin        = PORT_Pin_2;
120     PORT_InitStruct.PORT_OE         = PORT_OE_IN;
121     PORT_Init(MDR_PORTC, &PORT_InitStruct);
122     PORT_InitStruct.PORT_Pin        = PORT_Pin_1;
123     PORT_Init(MDR_PORTE, &PORT_InitStruct);
124     PORT_InitStruct.PORT_Pin        = PORT_Pin_5;
125     PORT_Init(MDR_PORTB, &PORT_InitStruct);
126
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
```

```
145 /* END OF FILE main.c */
```

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. Зафискировать 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 осцилограмма 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

```
// Установка приоритетов для T1 и T2
  NVIC_SetPriorityGrouping(3);
  NVIC_SetPriority(Timer1_IRQn, 0);
  NVIC_SetPriority(Timer2_IRQn, 0);
  // Разрешение прерываний для T1 и T2
  NVIC_EnableIRQ(Timer1_IRQn);
  NVIC_EnableIRQ(Timer2_IRQn);

  // Включение таймера 1 и 2
  TIMER_Cmd(MDR_TIMER1, ENABLE);

  // Установка периода для T1
  TIMER_CntInitStruct.TIMER_Period = 31250;
  // Установка периода для T2
  TIMER_CntInitStruct.TIMER_Period = 62500;
```

Листинг 2.2: Алгоритм обработки прерываний

```
/*******************************************************************************
* Function Name  : Timer1_IRQHandler
* Description    : This function handles Timer1 global interrupt request.
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************/
void Timer1_IRQHandler(void)
{
  TIMER_ClearFlag(MDR_TIMER1, TIMER_STATUS_CNT_ARR);

  uint32_t rxtx = PORT_ReadInputData(MDR_PORTC);
  uint32_t res = rxtx ^ PORT_Pin_0;
  PORT_Write( MDR_PORTC, res );
}
/*******************************************************************************
* Function Name  : Timer2_IRQHandler
* Description    : This function handles Timer2 global interrupt request.
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************/
void Timer2_IRQHandler(void)
{
  PORT_SetBits( MDR_PORTC, PORT_Pin_1 );
  Delay(2000000);
  TIMER_ClearFlag(MDR_TIMER2, TIMER_STATUS_CNT_ARR);
  PORT_ResetBits( MDR_PORTC, PORT_Pin_1 );
}
```

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.

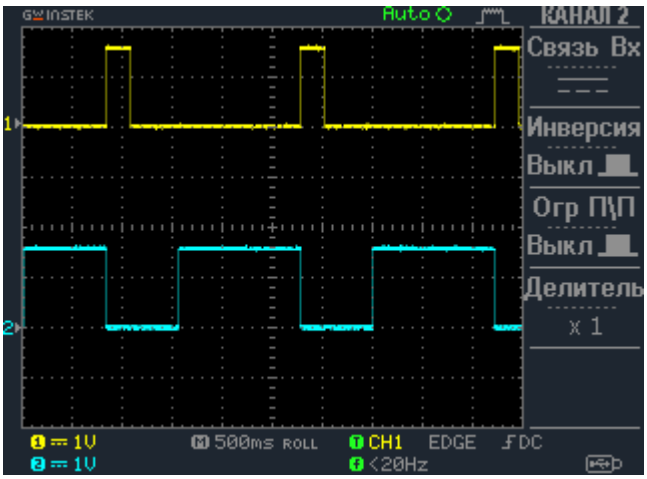

Рис. 2.1: Осцилограмма обработки прерываний при равных приоритетах обработки прерываний

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 }
```

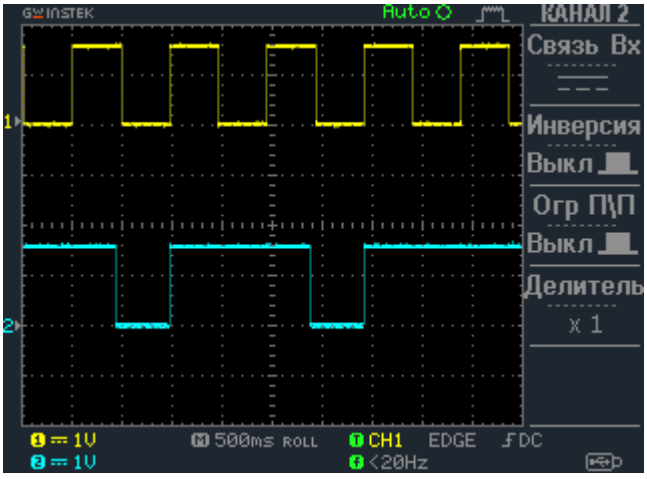

Рис. 2.2: Осцилограмма обработки прерываний при более высоком приоритете таймера-счетчика 1