

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

1	Лабораторная работа №1 «IAR, CMSIS, SPL, GPIO»	2
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
2	Лабораторная работа №2 «Системы тайминга и прерываний»	8
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: Select a standard for the key pair

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_PORTD, 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_PORTD, &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

Код демонстрационного примера приведен в листинге 2.1.

Листинг 2.1: Код демонстрационного примера

```
1 /* Includes ----- */
2
3 #include "MDR32Fx.h"
4 #include "MDR32F9Qx_config.h"
5 #include "MDR32F9Qx_port.h"
6 #include "MDR32F9Qx_rst_clk.h"
7 #include "MDR32F9Qx_timer.h"
8
9 /* Private typedef ----- */
```

```

10 /* Private define ----- */
11 /* Private macro ----- */
12 /* Private variables ----- */
13 /* Private function prototypes ----- */
14 static void PeriphInit(void);
15 /* Private functions ----- */
16
17 int main()
18 {
19     PeriphInit();
20
21     while(1)
22     {
23     }
24 }
25
26 static void PeriphInit( void )
27 {
28     PORT_InitTypeDef PORT_InitStruct;
29     TIMER_CntInitTypeDef TIMER_CntInitStruct;
30
31     // -----
32     //  ENABLE RCC
33     // -----
34
35     RST_CLK_PCLKcmd( RST_CLK_PCLK_RST_CLK | RST_CLK_PCLK_PORTC
36                     | RST_CLK_PCLK_TIMER1 | RST_CLK_PCLK_TIMER2,
37                     ENABLE );
38
39     // -----
40     //  INIT PORTC
41     // -----
42
43     PORT_StructInit(&PORT_InitStruct);
44
45     PORT_InitStruct.PORT_Pin   = ( PORT_Pin_0 | PORT_Pin_1 );
46     PORT_InitStruct.PORT_OE    = PORT_OE_OUT;
47     PORT_InitStruct.PORT_FUNC  = PORT_FUNC_PORT;
48     PORT_InitStruct.PORT_MODE  = PORT_MODE_DIGITAL;
49     PORT_InitStruct.PORT_SPEED = PORT_SPEED_SLOW;
50     PORT_Init( MDR_PORTC, &PORT_InitStruct );
51
52     PORT_InitStruct.PORT_Pin   = PORT_Pin_2;
53     PORT_InitStruct.PORT_OE    = PORT_OE_IN;
54     PORT_Init( MDR_PORTC, &PORT_InitStruct );
55
56     // -----
57     //  INIT TIMER 1, 2
58     // -----
59
60     TIMER_CntStructInit(&TIMER_CntInitStruct);
61
62     TIMER_CntInitStruct.TIMER_Period = 100;
63     TIMER_CntInit( MDR_TIMER1, &TIMER_CntInitStruct );
64
65     TIMER_BRGInit( MDR_TIMER1, TIMER_HCLKdiv1);
66     TIMER_ITConfig( MDR_TIMER1, TIMER_STATUS_CNT_ARR, ENABLE );
67
68     TIMER_CntInitStruct.TIMER_Period = 40000;
69     TIMER_CntInit( MDR_TIMER2, &TIMER_CntInitStruct );
70
71     TIMER_BRGInit( MDR_TIMER2, TIMER_HCLKdiv1 );
72     TIMER_ITConfig( MDR_TIMER2, TIMER_STATUS_CNT_ARR, ENABLE );
73
74     // -----
75     //  INIT NVIC

```

```

76 // -----
77 NVIC_SetPriorityGrouping( 3 );
78 NVIC_SetPriority( Timer1_IRQn, 0 );
79 NVIC_SetPriority( Timer2_IRQn, 0 );
80 NVIC_EnableIRQ( Timer1_IRQn );
81 NVIC_EnableIRQ( Timer2_IRQn );
82
83 TIMER_Cmd( MDR_TIMER1, ENABLE );
84 TIMER_Cmd( MDR_TIMER2, ENABLE );
85 }
86
87
88
89
90 #if (USE_ASSERT_INFO == 1)
91 void assert_failed(uint32_t file_id, uint32_t line)
92 {
93     while (1)
94     {
95     }
96 }
97 #elif (USE_ASSERT_INFO == 2)
98 void assert_failed(uint32_t file_id, uint32_t line, const uint8_t* expr)
99 {
100     while (1)
101     {
102     }
103 }
104 #endif /* USE_ASSERT_INFO */
105
106 /* END OF FILE main.c */

```

Содержимое файла MDR32F9Qx_it.c приведено в листинге 2.2.

Листинг 2.2: Листинг файла MDR32F9Qx_it.c

```

1 /**
2  *****
3  * @file    Examples/MDR32F9Q2_EVAL/UART/Interrupt/MDR32F9Qx_it.c
4  * @author  Milandr Application Team
5  * @version V1.2.0
6  * @date    04/07/2011
7  * @brief   Main Interrupt Service Routines.
8  *
9  *****
10 * <br><br>
11 *
12 * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS
13 * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE
14 * TIME. AS A RESULT, MILANDR SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT
15 * OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING
16 * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE
17 * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.
18 *
19 * <h2><center>&copy; COPYRIGHT 2011 Milandr</center></h2>
20 */
21
22 /* Includes ----- */
23 #include "MDR32F9Qx_it.h"
24 #include "MDR32F9Qx_port.h"
25 #include "MDR32F9Qx_timer.h"
26
27 /* Private typedef ----- */
28 /* Private define ----- */
29 /* Private macro ----- */
30 /* Private variables ----- */

```

```

31 /* Private function prototypes -----*/
32 /* Private functions -----*/
33
34 /******
35 * Function Name : NMI_Handler
36 * Description : This function handles NMI exception.
37 * Input : None
38 * Output : None
39 * Return : None
40 *****/
41 void NMI_Handler(void)
42 {
43 }
44 /******
45 * Function Name : HardFault_Handler
46 * Description : This function handles Hard Fault exception.
47 * Input : None
48 * Output : None
49 * Return : None
50 *****/
51 void HardFault_Handler(void)
52 {
53 /* Go to infinite loop when Hard Fault exception occurs */
54 while (1)
55 {
56 }
57 }
58 /******
59 * Function Name : MemManage_Handler
60 * Description : This function handles Memory Manage exception.
61 * Input : None
62 * Output : None
63 * Return : None
64 *****/
65 void MemManage_Handler(void)
66 {
67 /* Go to infinite loop when Memory Manage exception occurs */
68 while (1)
69 {
70 }
71 }
72 /******
73 * Function Name : BusFault_Handler
74 * Description : This function handles Bus Fault exception.
75 * Input : None
76 * Output : None
77 * Return : None
78 *****/
79 void BusFault_Handler(void)
80 {
81 /* Go to infinite loop when Bus Fault exception occurs */
82 while (1)
83 {
84 }
85 }
86 /******
87 * Function Name : UsageFault_Handler
88 * Description : This function handles Usage Fault exception.
89 * Input : None
90 * Output : None
91 * Return : None
92 *****/
93 void UsageFault_Handler(void)
94 {
95 /* Go to infinite loop when Usage Fault exception occurs */
96 while (1)

```

```

97  {
98  }
99  }
100 /******
101 * Function Name : SVC_Handler
102 * Description : This function handles SVCcall exception.
103 * Input : None
104 * Output : None
105 * Return : None
106 *****/
107 void SVC_Handler(void)
108 {
109 }
110 /******
111 * Function Name : DebugMon_Handler
112 * Description : This function handles Debug Monitor exception.
113 * Input : None
114 * Output : None
115 * Return : None
116 *****/
117 void DebugMon_Handler(void)
118 {
119 }
120 /******
121 * Function Name : PendSV_Handler
122 * Description : This function handles Debug PendSV exception.
123 * Input : None
124 * Output : None
125 * Return : None
126 *****/
127 void PendSV_Handler(void)
128 {
129 }
130 /******
131 * Function Name : SysTick_Handler
132 * Description : This function handles SysTick Handler.
133 * Input : None
134 * Output : None
135 * Return : None
136 *****/
137 void SysTick_Handler(void)
138 {
139 }
140 /******
141 * Function Name : CAN1_IRQHandler
142 * Description : This function handles CAN1 global interrupt request.
143 * Input : None
144 * Output : None
145 * Return : None
146 *****/
147 void CAN1_IRQHandler(void)
148 {
149 }
150 /******
151 * Function Name : CAN2_IRQHandler
152 * Description : This function handles CAN2 global interrupt request.
153 * Input : None
154 * Output : None
155 * Return : None
156 *****/
157 void CAN2_IRQHandler(void)
158 {
159 }
160 /******
161 * Function Name : USB_IRQHandler
162 * Description : This function handles USB global interrupt request.

```

```

163 * Input : None
164 * Output : None
165 * Return : None
166 *****/
167 void USB_IRQHandler(void)
168 {
169 }
170 /******
171 * Function Name : DMA_IRQHandler
172 * Description : This function handles DMA global interrupt request.
173 * Input : None
174 * Output : None
175 * Return : None
176 *****/
177 void DMA_IRQHandler(void)
178 {
179 }
180 /******
181 * Function Name : UART1_IRQHandler
182 * Description : This function handles UART1 global interrupt request.
183 * Input : None
184 * Output : None
185 * Return : None
186 *****/
187 void UART1_IRQHandler(void)
188 {
189 }
190 /******
191 * Function Name : UART2_IRQHandler
192 * Description : This function handles UART2 global interrupt request.
193 * Input : None
194 * Output : None
195 * Return : None
196 *****/
197 void UART2_IRQHandler(void)
198 {
199 }
200 /******
201 * Function Name : SSP1_IRQHandler
202 * Description : This function handles SSP1 global interrupt request.
203 * Input : None
204 * Output : None
205 * Return : None
206 *****/
207 void SSP1_IRQHandler(void)
208 {
209 }
210 /******
211 * Function Name : I2C_IRQHandler
212 * Description : This function handles I2C global interrupt request.
213 * Input : None
214 * Output : None
215 * Return : None
216 *****/
217 void I2C_IRQHandler(void)
218 {
219 }
220 /******
221 * Function Name : POWER_IRQHandler
222 * Description : This function handles POWER global interrupt request.
223 * Input : None
224 * Output : None
225 * Return : None
226 *****/
227 void POWER_IRQHandler(void)
228 {

```

```

229 }
230 /******
231 * Function Name : WWDG_IRQHandler
232 * Description   : This function handles WWDG global interrupt request.
233 * Input        : None
234 * Output       : None
235 * Return       : None
236 *****/
237 void WWDG_IRQHandler(void)
238 {
239 }
240
241 /******
242 * Function Name : Timer1_IRQHandler
243 * Description   : This function handles Timer1 global interrupt request.
244 * Input        : None
245 * Output       : None
246 * Return       : None
247 *****/
248 void Timer1_IRQHandler(void)
249 {
250     TIMER_ClearFlag( MDR_TIMER1, TIMER_STATUS_CNT_ARR );
251     MDR_PORTC->RXTX ^= PORT_Pin_0;
252 }
253 /******
254 * Function Name : Timer2_IRQHandler
255 * Description   : This function handles Timer2 global interrupt request.
256 * Input        : None
257 * Output       : None
258 * Return       : None
259 *****/
260 void Timer2_IRQHandler(void)
261 {
262     TIMER_ClearFlag( MDR_TIMER2, TIMER_STATUS_CNT_ARR );
263     MDR_PORTC->RXTX ^= PORT_Pin_1;
264 }
265 /******
266 * Function Name : Timer3_IRQHandler
267 * Description   : This function handles Timer3 global interrupt request.
268 * Input        : None
269 * Output       : None
270 * Return       : None
271 *****/
272 void Timer3_IRQHandler(void)
273 {
274 }
275 /******
276 * Function Name : ADC_IRQHandler
277 * Description   : This function handles ADC global interrupt request.
278 * Input        : None
279 * Output       : None
280 * Return       : None
281 *****/
282 void ADC_IRQHandler(void)
283 {
284 }
285 /******
286 * Function Name : COMPARATOR_IRQHandler
287 * Description   : This function handles COMPARATOR global interrupt request.
288 * Input        : None
289 * Output       : None
290 * Return       : None
291 *****/
292 void COMPARATOR_IRQHandler(void)
293 {
294 }

```

```

295 /* *****
296 * Function Name : SSP2_IRQHandler
297 * Description : This function handles SSP2 global interrupt request.
298 * Input : None
299 * Output : None
300 * Return : None
301 *****/
302 void SSP2_IRQHandler(void)
303 {
304 }
305 /* *****
306 * Function Name : BACKUP_IRQHandler
307 * Description : This function handles BACKUP global interrupt request.
308 * Input : None
309 * Output : None
310 * Return : None
311 *****/
312 void BACKUP_IRQHandler(void)
313 {
314 }
315 /* *****
316 * Function Name : EXT_INT1_IRQHandler
317 * Description : This function handles EXT_INT1 interrupt request.
318 * Input : None
319 * Output : None
320 * Return : None
321 *****/
322 void EXT_INT1_IRQHandler(void)
323 {
324 }
325 /* *****
326 * Function Name : EXT_INT2_IRQHandler
327 * Description : This function handles EXT_INT2 interrupt request.
328 * Input : None
329 * Output : None
330 * Return : None
331 *****/
332 void EXT_INT2_IRQHandler(void)
333 {
334 }
335 /* *****
336 * Function Name : EXT_INT3_IRQHandler
337 * Description : This function handles EXT_INT3 global interrupt request.
338 * requests.
339 * Input : None
340 * Output : None
341 * Return : None
342 *****/
343 void EXT_INT3_IRQHandler(void)
344 {
345 }
346 /* *****
347 * Function Name : EXT_INT4_IRQHandler
348 * Description : This function handles EXT_INT4 interrupt request.
349 * Input : None
350 * Output : None
351 * Return : None
352 *****/
353 void EXT_INT4_IRQHandler(void)
354 {
355 }
356
357 /* ***** (C) COPYRIGHT 2011 Milandr *****/
358
359 /* END OF FILE MDR32F9Qx_it.c */

```