```c

#include "riscv_csr_encoding.h"
#include "mcu32_memory_map.h"
#include <power_manager.h>
#include <wdt.h>
#include <gpio.h>
#include <gpio_irq.h>
#include <epic.h>
#include "pad_config.h"

#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"

/* NDV [02.09.2024] */

#ifndef  DIG_TYPES_H
#define  DIG_TYPES_H

typedef  signed char         i8;
typedef  signed short        i16;
typedef  signed long         i32;
typedef  signed long long    i64;

typedef  unsigned char       u8;
typedef  unsigned short      u16;
typedef  unsigned long       u32;
typedef  unsigned long long  u64;

#endif /* DIG_TYPES_H */

/* NDV [03.09.2024] */

#include <timer32.h>
#include "scr1_csr_encoding.h"
#include "mcu32_memory_map.h"
#include <power_manager.h>
#include "pad_config.h"
#include <csr.h>
#include "uart_lib.h"
#include "xprintf.h"
#include "mik32_hal_rtc.h"
#include "mik32_hal_pcc.h"
#include "mik32_hal_dma.h"

#ifndef  K1948BK018_MACRO_H
#define  K1948BK018_MACRO_H

/* чтобы «протащить» через несколько макросов несколько аргументов как один аргумент */
#define  _(...)  __VA_ARGS__

/* превращает число в строку средствами препроцессора */
#ifndef  stringify
    #define  pro_stringify(a)  #a
    #define  stringify(a)      pro_stringify(a)
#endif

/* обёртка для блока _(<...>), защищённого мьютексом (FreeRTOS) */
#define  mutex_container(_mutex_, _block_)            \
             xSemaphoreTake(_mutex_, portMAX_DELAY);  \
                 _block_                              \
             xSemaphoreGive(_mutex_);

/* упрощённая работа с портами ввода/вывода */

#define  io_RCC_EN(_p_,_b_)  PM->CLK_APB_P_SET |= PM_CLOCK_APB_P_GPIO_##_p_##_M
#define  io_port(_p_,_b_)    (GPIO_##_p_)
#define  io_bit(_p_,_b_)     (_b_)
#define  io_bit_n(port_bit)  io_bit(port_bit)

#define  io_inp(port_bit)  do { io_RCC_EN(port_bit); io_port(port_bit)->DIRECTION_IN  = 1 << io_bit(port_bit); } while(0)
#define  io_out(port_bit)  do { io_RCC_EN(port_bit); io_port(port_bit)->DIRECTION_OUT = 1 << io_bit(port_bit); } while(0)
#define  io_set(port_bit)    io_port(port_bit)->SET   = 1 << io_bit(port_bit)
#define  io_clr(port_bit)    io_port(port_bit)->CLEAR = 1 << io_bit(port_bit)
#define  io_read(port_bit)  (io_port(port_bit)->STATE >> io_bit(port_bit)  &  1)
#define  io_SET_R(port_bit)  (&io_port(port_bit)->SET  )/* NDV [06.09.2024] */
#define  io_CLR_R(port_bit)  (&io_port(port_bit)->CLEAR)/* NDV [06.09.2024] */

#endif /* K1948BK018_MACRO_H */

/* ИСПОЛЬЗУЕМЫЕ ВЫВОДЫ */
   #define  USER_BTN  2,6
   #define  USER_LED  2,7
   #define  ASM_OUT   1,1

/* <...> */

#define  MILLISECONDS  configTICK_RATE_HZ/1000

extern unsigned long __TEXT_START__;

void ext_trap_handler(void)
{
    // <...>
}
```

```
 96
 97 extern void CPP_17_sample_1_fun();
 98
 99 volatile u8 global_flag = 0;
100
101 /* ЭТИ ДВА МАКРОСА НЕ ПОНАДОБИЛИСЬ */
102 #define  __push_3_macro()  \
103             "addi sp, sp, -12\n" /* Уменьшаем указатель стека на 12 байт (по 4 байта на каждый регистр) */  \
104             "sw   t0, 0(sp)\n"   /* Сохраняем x1 по адресу sp + 0 */  \
105             "sw   t1, 4(sp)\n"   /* Сохраняем x2 по адресу sp + 4 */  \
106             "sw   t2, 8(sp)\n"   /* Сохраняем x3 по адресу sp + 8 */
107 #define  __pop_3_macro()  \
108             "lw   t0, 0(sp)\n" /* Загружаем x1 с адреса sp + 0 */  \
109             "lw   t1, 4(sp)\n" /* Загружаем x2 с адреса sp + 4 */  \
110             "lw   t2, 8(sp)\n" /* Загружаем x3 с адреса sp + 8 */  \
111             "addi sp, sp, 12\n" /* Увеличиваем указатель стека на 12 байт, возвращая его в исходное положение */
112
113 static SemaphoreHandle_t xprintf_mutex = NULL;
114
115 static void test_task1(void *param)
116 {
117     u32 cycle_number = 0;
118     while (1)
119     {
120         if ( global_flag )
121         {
122             mutex_container ( xprintf_mutex,
123             _(
124                 io_clr(USER_LED);
125                 xprintf("TASK # 1 . LED_OFF : cycle number : %u" "\r\n", cycle_number  );
126             ) )
127             vTaskDelay(1000 * MILLISECONDS);
128         }
129         if ( global_flag )
130         {
131             mutex_container ( xprintf_mutex,
132             _(
133                 io_set(USER_LED);
134                 xprintf("TASK # 1 . LED_ON  : cycle number : %u" "\r\n", cycle_number++);
135             ) )
136             vTaskDelay( 500 * MILLISECONDS);
137         }
138         u32 clr = io_CLR_R(ASM_OUT),
139             set = io_SET_R(ASM_OUT),
140             msk = 1 << io_bit_n(ASM_OUT);
141         mutex_container ( xprintf_mutex,
142         _(
143             xprintf("TASK # 1 . t0 = 0x%08X, t1 = 0x%08X, t2 = 0x%08X" "\r\n", clr, set, msk);
144         ) )
145         asm volatile
146         (
147             /// __push_3_macro() <--не_нужно
148                 "mv t0, %0\n" // Копируем значение переменной msk в регистр t0
149                 "mv t1, %1\n" // Копируем значение переменной set в регистр t1
150                 "mv t2, %2\n" // Копируем значение переменной clr в регистр t2
151                 "li t3, 12\n" // Счётчик цикла, значение 12
152                 "1:\n"
153                     "sw t2, 0(t1)\n" // Записываем значение t2 по адресу в t1
154                     "nop\n nop\n nop\n" // на осциллограмме можно увидеть длительность пустых операций
155                     "sw t2, 0(t0)\n" // Записываем значение t2 по адресу в t0
156                     "addi t3, t3, -1\n" // Уменьшаем счётчик
157                     "bnez t3, 1b\n" // Если t3 не равен нулю, продолжаем цикл
158             /// __pop_3_macro() <--не_нужно
159             : : "r" (clr), "r" (set), "r" (msk)
160         ) ;
161     }
162 }
163
164 static void test_task2(void *param)
165 {
166     u32 cycle_number = 0;
167     while(1)
168     {
169         global_flag = ! io_read(USER_BTN);
170         mutex_container ( xprintf_mutex,
171         _(
172            xprintf("TASK # 2 : cycle number : %u" "\r\n", cycle_number++);
173            if ( ! UART_IsRxFifoEmpty(UART_0) ) xprintf("TASK # 2 : you send the byte 0x%02X" "\r\n", UART_0->RXDATA);
174         ) )
175         vTaskDelay(300 * MILLISECONDS);
176         asm volatile
177         (
178             "nop\n" /* здесь мы старательно портим все регистры t ) */
179                 "li t0, 123456789\n"
180                 "li t1, 123789456\n"
181                 "li t2, 456789123\n"
182                 "li t3, 456123789\n"
183                 "li t4, 789456123\n"
184                 "li t5, 789123789\n"
185                 "li t6, 777777777\n"
186             "nop\n"
187         ) ;
188     }
189 }
190
```

```
191 #include "welcome.inc"
192
193 void main()
194 {
195     // interrupt vector init
196     write_csr(mtvec, &__TEXT_START__);
197
198     PM->CLK_APB_P_SET = PM_CLOCK_APB_P_GPIO_IRQ_M;
199     PM->CLK_APB_M_SET =   PM_CLOCK_APB_M_PAD_CONFIG_M
200                         | PM_CLOCK_APB_M_WU_M
201                         | PM_CLOCK_APB_M_PM_M
202                         | PM_CLOCK_APB_M_EPIC_M
203                         ;
204
205     io_inp(USER_BTN); // BTN init
206     io_out(USER_LED); // LED init
207     io_out(ASM_OUT);
208
209     /* NDV [04.09.2024] */
210        PM->CLK_AHB_SET |= PM_CLOCK_AHB_SPIFI_M
211                         | PM_CLOCK_AHB_DMA_M ;
212
213     /* VCP (terminal output) */
214        UART_Init ( UART_0, 3333/* соответствует 9600 бод ) */,
215                    UART_CONTROL1_TE_M | UART_CONTROL1_RE_M | UART_CONTROL1_M_8BIT_M, 0, 0 ) ;
216        xprintf("\r\n\r\n" "%s" "\r\n", welcome);
217
218     /* create mutex for xprintf */
219        xprintf_mutex = xSemaphoreCreateMutex();
220
221     /* create work threads */
222        xTaskCreate ( test_task1, "Task1", 1024, ( void * ) 1, tskIDLE_PRIORITY + 1 , NULL ) ;
223        xTaskCreate ( test_task2, "Task2",  512, ( void * ) 1, tskIDLE_PRIORITY + 1 , NULL ) ;
224
225     vTaskStartScheduler();
226 }
227
```