```c
/*****************************************************************************************
 * HEIG-VD
 * Haute Ecole d'Ingenerie et de Gestion du Canton de Vaud
 * School of Business and Engineering in Canton de Vaud
 *****************************************************************************************
 * REDS Institute
 * Reconfigurable Embedded Digital Systems
 *****************************************************************************************
 *
 * File                 : execptions.c
 * Author               : Sébastien Masle
 * Date                 : 16.02.2018
 *
 * Context              : SOCF class
 *
 *****************************************************************************************
 * Brief: defines exception vectors for the A9 processor
 *        provides code that sets the IRQ mode stack, and that dis/enables interrupts
 *        provides code that initializes the generic interrupt controller
 *
 *****************************************************************************************
 * Modifications :
 * Ver    Date        Engineer      Comments
 * 0.0    16.02.2018  SMS           Initial version.
 * 1.0    13.03.2020  Spinelli Isaia
 *
*****************************************************************************************/
#include <stdint.h>

#include "address_map_arm.h"
#include "defines.h"




// Référence : Exemple dans Using The ARM Generic

// Define the IRQ exception handler
void __attribute__ ((interrupt)) __cs3_isr_irq(void)
{
    /***********
     * Attention dans Qsys mettre sur flanc et non level !
     **********/

    // Read CPU Interface registers to determine which peripheral has caused an interrupt
    int interrupt_ID =*((int*) 0xFFFEC10C);

    // Handle the interrupt if it comes from the KEYs
    if (interrupt_ID == 72) {
        pushbutton_ISR();
    } else {
        while (1);                      // if unexpected, then stay here
    }

    // Clear interrupt from the CPU Interface
    *((int*) 0xFFFEC110) = interrupt_ID;

    return;
```

```c
}

// Define the remaining exception handlers
void __attribute__ ((interrupt)) __cs3_reset (void)
{
    while(1);
}

void __attribute__ ((interrupt)) __cs3_isr_undef (void)
{
    while(1);
}

void __attribute__ ((interrupt)) __cs3_isr_swi (void)
{
    while(1);
}

void __attribute__ ((interrupt)) __cs3_isr_pabort (void)
{
    while(1);
}

void __attribute__ ((interrupt)) __cs3_isr_dabort (void)
{
    while(1);
}

void __attribute__ ((interrupt)) __cs3_isr_fiq (void)
{
    while(1);
}

/*
 * Initialize the banked stack pointer register for IRQ mode
*/
void set_A9_IRQ_stack(void)
{
    uint32_t stack, mode;
    stack = A9_ONCHIP_END - 7;      // top of A9 onchip memory, aligned to 8 bytes
    /* change processor to IRQ mode with interrupts disabled */
    mode = INT_DISABLE | IRQ_MODE;
    asm("msr cpsr, %[ps]" : : [ps] "r" (mode));
    /* set banked stack pointer */
    asm("mov sp, %[ps]" : : [ps] "r" (stack));

    /* go back to SVC mode before executing subroutine return! */
    mode = INT_DISABLE | SVC_MODE;
    asm("msr cpsr, %[ps]" : : [ps] "r" (mode));
}

/*
 * Turn on interrupts in the ARM processor
*/
void enable_A9_interrupts(void)
{
    uint32_t status = SVC_MODE | INT_ENABLE;
    asm("msr cpsr, %[ps]" : : [ps]"r"(status));
}

/** Turn off interrupts in the ARM processor*/
void disable_A9_interrupts(void) {
    int status = 0b11010011;
    asm("msr cpsr, %[ps]" : : [ps]"r"(status));
}

void config_GIC (void) {
     // configure the FPGA KEYs interrupt (72)
    config_interrupt (72, 1);

    // Set Interrupt Priority Mask Register (ICCPMR). Enable all priorities
    *((int*) 0xFFFEC104) = 0xFFFF;

    // Set the enable in the CPU Interface Control Register (ICCICR)
    *((int*) 0xFFFEC100) = 1;

    // Set the enable in the Distributor Control Register (ICDDCR)
    *((int*) 0xFFFED000) = 1;
}

void config_KEYs (void) {
    volatile int*KEY_ptr = (int*) 0xFF200050;   // KEY base address

    *(KEY_ptr + 2) = 0xF;    // enable interrupts for all four KEYs

}

void config_interrupt (int N, int CPU_target) {
    int reg_offset, index, value, address;

    /*Configure the Interrupt Set-Enable Registers (ICDISERn).
     *reg_offset = (integer_div(N / 32)*4; value = 1 << (N mod 32)*/

    reg_offset = (N >> 3) & 0xFFFFFFFC;
    index = N & 0x1F;
    value = 0x1 << index;
    address = 0xFFFED100 + reg_offset;

    /*Using the address and value, set the appropriate bit*/
    *(int*)address |= value;

    /*Configure the Interrupt Processor Targets Register (ICDIPTRn)
     * reg_offset = integer_div(N / 4)*4; index = N mod 4*/
    reg_offset = (N & 0xFFFFFFFC);
    index = N & 0x3;
    address = 0xFFFED800 + reg_offset + index;

    /*Using the address and value, write to (only) the appropriate byte*/
    *(char*)address = (char) CPU_target;
}

```