

|  |
| --- |
| **Course: Embedded Electronic Devices and Programming** |
| Laboratory work № 2 |
| “HANDLING INTERRUPT REQUESTS |

|  |  |
| --- | --- |
| Student name and surname: | Daniil Grammatikopulo |
| Student code: | 79101 |
| Group: | 4103BDA |
| Lecturer: | Artjoms Ivanov |

|  |  |
| --- | --- |
| Date performed: | 19.03.2023 |

Riga

2023

Table of Content

[TASK FOR LABORATORY WORK 3](#_Toc129893694)

[The report should have followed structure: 3](#_Toc129893695)

[Individual variant of the task 4](#_Toc129893696)

[Microcontroller I/O diagram 5](#_Toc129893697)

[ALGORITHM 6](#_Toc129893698)

[PROGRAM CODE 7](#_Toc129893699)

[CONCLUSION 8](#_Toc129893700)

TASK FOR LABORATORY WORK

The report should have followed structure:

* to develop of the microcontroller, I/O diagram.
* to create an algorithm of main function and an algorithm(-s) of handling interrupt requests for the ATMega48 microcontroller in accordance with an individual task.
* to create program code in Atmel Studio 7 environment.
* to debug the program in Atmel Studio 7 environment.
* to issue a report on laboratory work.
* to form time intervals, use functions that form delays.
* when debugging a program in the Atmel Studio 7 environment, the call of functions that generate delays must be commented out.
* the clock frequency of the ATMega48 microcontroller is 8 MHz.
* when the button is pressed on the input/output line of the port - logical "0", when the button is released on the input/output line of the port - logical "1".
* the LED is turned on by logical "0" on the I/O line of the port, and turned off by logical "1".

Individual variant of the task

(19) LEDs are connected to port B. Buttons are connected to port D. After pressing any button, a PCINT2 interrupt request is generated. After pressing the buttons SW1 - SW7, their numbers are remembered. Each even pressing of the SW0 button turns on the LEDs with the numbers of the previously pressed buttons, and each odd one turns off the LEDs, and the numbers of the pressed buttons are not stored after that.

Microcontroller I/O diagram

Изображение выглядит как диаграмма, схематичный

Автоматически созданное описание

**ATmega48 diagram 1**

ALGORITHM

Изображение выглядит как диаграмма

Автоматически созданное описание

**Algorithm diagram 1**

PROGRAM CODE

#define *F\_CPU* 8000000UL

#include <avr/interrupt.h>

// Define the macro to check if a button is pressed

#define ISPRESSED(n) (~PIND & ( 1 << n ))

// Declare global variables

volatile *uint16\_t* pressed\_buttons = 0xFF; // Initialize to all buttons not pressed

volatile *uint16\_t* press\_count = 0; // Keep track of button presses

// The main function

int main(void) {

// Set Port D as input and enable internal pull-up resistors

DDRD = 0x00;

PORTD = 0xFF;

// Set Port B as output and initialize all pins to high (LEDs off)

DDRB = 0xFF;

PORTB = 0xFF;

// Enable Pin Change Interrupt for Port D (PCINT16 to PCINT23)

PCICR |= (1 << PCIE2);

PCMSK2 = 0xFF; // Enable interrupt on all pins of Port D

sei(); // Enable global interrupts

while (1) {} // Infinite Loop

}

// Interrupt Service Routine for Pin Change Interrupt on Port D

ISR(PCINT2\_vect)

{

// Check if PD0 (button 1) is pressed

if(ISPRESSED(PD0)) {

press\_count++; // Increment button press counter

// If the button has been pressed an even number of times, turn on the LEDs according to the previous button state

if (press\_count % 2 == 0)

PORTB = pressed\_buttons;

else {

pressed\_buttons = 0xFF; // Forget all pressed buttons

PORTB = 0xFF; // Turn off all LEDs

}

}

// Check if any other button is pressed (PD1 to PD7)

if ((PIND & 0xFE) != 0xFE)

pressed\_buttons &= PIND; // Update the button state

}

CONCLUSION

In conclusion, this Lab involves the use of LEDs and buttons connected to ports B and D respectively, and the generation of PCINT2 interrupt requests upon button press. The program logic remembers the numbers of buttons SW1 - SW7 pressed and toggles the LEDs based on even/odd pressing of button SW0. Specifically, the even press of SW0 turns on the LEDs with previously pressed button numbers, while odd press of SW0 turns off the LEDs and clears previously pressed button numbers. Overall, this lab showcases the effective use of interrupt requests and logical programming in controlling LEDs with button inputs.

The use of interrupts for button inputs offers several advantages over polling-based methods. Interrupts allow the microcontroller to respond immediately to a button press without wasting processing power on continuously checking the status of the input pin. This approach improves system responsiveness and reduces power consumption. Additionally, interrupt-based input handling enables synchronization of various parts of the program and avoids potential issues with race conditions.

However, configuring interrupts can be more challenging than simple polling techniques, especially without proper documentation. Interrupt-based programming requires careful consideration of the interrupt service routine and its impact on the rest of the system. Nevertheless, with proper documentation and good programming practices, the benefits of using interrupts for button inputs outweigh the complexity of their configuration.

P.S work is also available on my GitHub: <https://github.com/Dast3X/ATmega48/tree/main/IO%20ports>