REPORT

George Lalas AM: 1093406

Panagiotis Dimakopoulos AM: 1041734

Detailed Code Description:

The code that has been developed performs the following functions, which will be analyzed below: an elevator, which can move between three floors (0, 1, and 2), and depending on the direction of movement (upward or downward), an LED will turn on or off. Additionally, the current floor the elevator is on will be indicated at all times. If the elevator is on the top or bottom floor, no change will occur, while if both the up and down buttons are pressed simultaneously, an error LED will activate.

Initially, the first step in implementing the above functionalities is to utilize the appropriate libraries, define the pins, and establish certain constants that will be used in the code. The libraries included are <avr/io.h>, which contains definitions of the registers and ports for the microcontroller, <avr/interrupt.h> since the code includes interrupts, and <util/delay.h> because the delay function _delay_ms() will be used. Pins 5 and 6 of PORTF are used, and this choice was made because we need two switches: when the first one (pin5) is pressed, the elevator will move upward, and when the second one is pressed (pin6), the elevator will move downward. As for the LEDs that are activated depending on the movement and state of the elevator, they are located on PORTD, and four LEDs will be needed. The three floors, the four possible states the elevator can be in, as well as some flags, are represented as integers. The states of the elevator are the upward state, downward state, error state, and initial state. The initial state is the one where no changes occur, meaning the elevator has moved and is now stationary on a floor. Therefore, it has also been defined as the initialization state. Furthermore, based on the problem statement, the initial floor where the elevator is located is floor zero.

Next come the functions that implement the described functionality.
Function int main(void): In this function, the four mentioned LEDs are initially set as outputs and then initialized in the "off" state. This is done using the PORTD.DIR and PORTD.OUT commands respectively. Then, the pull-up is enabled so that when a button is not pressed, the pin has a defined state (high), avoiding uncertainties, by setting PORT_PULLUPEN_bm. Next, the edge on which the interrupt management unit will be triggered is defined, specifically detecting interrupts on both edges, using the setting PORT_ISC_BOTHEDGES_gc. The next step is enabling the interrupts with the command sei(). The main loop is infinite and is inside a while statement. This loop manages the different states the elevator may be in. More specifically, it may be in an upward or downward state, in which case the function changing_states() is called, or in an error state, so error_handling() is called. The last state is an idle state, where no changes

happen to the LEDs and the elevator remains stationary on a floor. This state has been added for better readability and clarity in the code.

Function ISR(PORTF_PORT_vect): This function is an interrupt service routine, which takes as an argument an address corresponding to an interrupt and is therefore responsible for handling interrupts. Since the interrupts originate from PORTF, the interrupt vector PORTF_PORT_vect is used. The first point worth noting is that depending on the button that triggered the interrupt (pin5 or pin6), the corresponding flag (up_button or down_button) is set to 1. The interrupt flag is then cleared to allow for a new interrupt detection. Specifically, when the interrupt is triggered by a change in the state of a switch, the INTFLAGS flag remains active until it is manually cleared; otherwise, a new interrupt detection cannot occur. Then, if-else if statements are used to handle all possible movement scenarios of the elevator. If both buttons are pressed simultaneously, resulting in both flags up_button and down_button being set to 1, the elevator moves into the error state, and both flags are reset to 0. If only the up_button flag is 1 and there is still room for the elevator to go up (i.e., it is not on the second floor), the state it enters is the upward state, and the flag is then reset. Furthermore, the purpose of the second else if is to cover the case when the elevator is already on the second floor and the up button is pressed again, in which case no change should occur. However, the up_button flag will already be 1, so it must be reset, since pressing the down button next would cause an unwanted error state. The same applies to the two subsequent else if conditions, with the difference now being that they concern the elevator's descent, and consequently the flag used is down_button, and the floor below which it cannot descend is zero.

Function void changing_states(void): This function implements the changes that occur depending on the state the elevator is in, i.e., the upward and downward states. When in the upward state, we check if the current floor is less than the highest floor we can reach – that is, the second. If already on the second floor, no change occurs. If not, the LED indicating the direction of movement, in this case upward movement, should light up using the command PORTD.OUTCLR = movement_led;. Then, we wait for 10 ms to simulate the delay required for the elevator to move. Next, the floor is increased by one, and depending on the new floor number, the corresponding LEDs will light up. If we are on the first floor, only one LED lights up with the command PORTD.OUTCLR = first_floor_led; while if we are on the second floor, both light up with the commands PORTD.OUTCLR = first_floor_led;, PORTD.OUTCLR = second_floor_led;. A similar process occurs for the downward state. However, now the LED indicating movement turns off with PORTD.OUT |= movement_led;, since we are moving downward, and the floor decreases by one. Another difference is that the floors the elevator can now reach are zero and one. On floor zero, both LEDs turn off using PORTD.OUT |= first_floor_led;, PORTD.OUT |= second_floor_led;, while on floor one, one of the two turns on, just like during the upward movement. Finally, the elevator's state returns to initial_state, meaning the state where no changes occur to the LEDs.

Function void error_handling(void): This function is used to handle the elevator's error state. When an error is detected, the error LED is activated – pin0, with the command PORTD.OUTCLR = error_led; and then a 10 ms delay is introduced until the error LED turns off with the command PORTD.OUT |= error_led;. Finally, the elevator's state

returns to initial_state, meaning the state where no changes occur to the LEDs. This error state function does not affect the other LEDs.

Flow Chart: