CREATING THE CYCLONE ARCADE GAME USING AVR ASSEMBLY

Tanay Rangasamy: 2544162

School of Electrical and Information Engineering

Abstract: The project aims to recreate the popular cyclone arcade game using the Assembly language on an Arduino microcontroller. The main goals include creating the game with AVR Assembly utilizing specifically timers and interrupts. The game was successfully set up to meet most of the minimal requirements and constraints.

Keywords: ADC, Buttons, Flowchart, LEDs, Timer1, RGB

1.INTRODUCTION

The Assembly code uses timers and interrupts to create the game. Interrupts cause the program to stop what it is currently executing; the program then executes the instructions for the interrupt and then returns to what it was originally executing after the interrupt is completed.

1.1 Project Specifications and assumptions

Timers and interrupts are the primary specifications for the project. The functions, delays and active-polling are not to be used at any point in the project and the project must be coded specifically in AVR Assembly. The delay between LED blinks needs to be 1.5 seconds. The clock frequency of the Arduino Uno is 16MHz meaning to achieve a delay this large Timer1 will need to be used (Timer0 and Timer2 delays are too small as they are measured in microseconds) [2]. The ADC interrupt can be used to generate a random number for the indicator LED to be chosen by converting "noisy input signals" into a digital value [4].

1.2 Project Functionality

The game needs to consist of ten LEDs, push buttons and an RGB. The game begins when a push button is pressed (this same button needs to be able to pause and resume the game). When the game begins an LED turns on and is used as the indicator LED; the other LEDs begin lighting up in sequence with a delay of 1.5s between each blink. The player needs to be able to stop the current LED lit up with another push button and start a new round. The game should only have five rounds. After the five rounds are completed the RGB will display a color corresponding to the player's score.

1.3 Similar projects used for reference

The project used as reference to the beginning stages of this project is called "Cyclone" by GitHub user "Sambonic" [3]. This project utilizes the same basic components and is programmed using the Arduino IDE. The main differences however are the inclusion of various modes, the coding language (the reference project was coded in C/C++) and the incorporation of some other additional features.

2.CREATING THE GAME: The Assembly Code and the physical configuration

2.1 Flowchart for the design of the program

The ADC interrupt is primarily used to turn on a random LED and to keep that LED on. The Timer interrupt is used to control the LEDs blinking in sequence. The first button interrupt is used to start the game as well as pausing and resuming the game. The second button interrupt is used to track the score and eventually end the game. The project process is outlined in Figure 1.

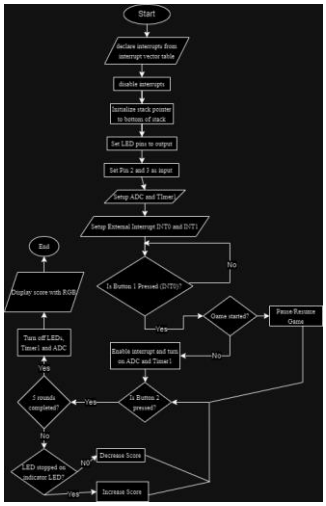

Figure 1: A flowchart of the project

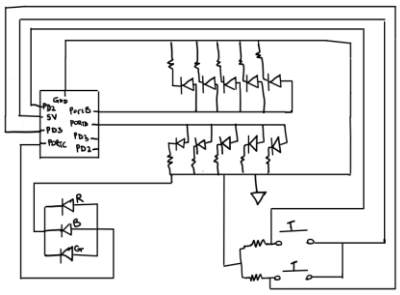

Figure 2: A schematic diagram of the project

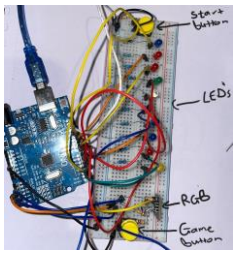

Figure 3: Image

Table 1: Configuration of components to the Microcontroller

Component	**Port used**	**Description**
LEDs	PORTB (8-12) and PORTD (1&4-7)	PORTB and PORTD are 8-bit I/O ports therefore it easier to assign numbers to these ports corresponding to their binary high/low form.
RGB LED	PORTC (A5, A4, A3)	Remaining unused port.
Button 1(Start button)	PD2 & 5V	PD2 is the pin for INT0. 5V connection needed to register input.
Button 2(Game button)	PD3 & 5V	PD3 is the pin for INT1. 5V connection needed to register input.

2.2 The LEDs

The project requires the LEDs to turn on sequentially. Therefore, when one LED is turned on the others must remain off excluding the indicator LED. These numbers are stored in two arrays allowing for outputs to PORTB and PORTD respectively. The arrays are then stored in the program memory. To access the arrays the high and low bytes of the addresses are stored in registers. The high byte corresponds to the array used and the low byte corresponds to the specific number to turn on the LED. A register incremented from zero to nine is added to the low byte of the Z register; indirect addressing is then used to get the value of the number in the memory address and output the number to the respective port (the low byte is sequentially updated to return the next number in the array thus allowing for the LEDs to sequentially turn on). Table 2 shows the arrays and their respective memory addresses (0x0100 is the beginning of the array storing the PORTD values and 0x0200 is the array storing the PORTB values).

Table 2: Table showing the respective LEDs "on" value and the memory addresses in the program memory

LED	One	Two	Three	Four	Five	Six	Seven	Eight	Nine	Ten
Output value	2	16	32	64	128	1	2	4	8	16
Memory Address	0x0100	0x0101	0x0102	0x0103	0x0104	0x0205	0x0206	0x0207	0x0208	0x0209

2.3 Using the ADC interrupt to select the indicator LED randomly (ADC Interrupt Vector)

The ADC interrupt uses noise voltage to generate a random value within the range of zero to nine; this value is then used to determine the indicator LED. The ADC is configured to have a reference voltage of 1.1V by loading a bit into the registers REFS0 and REFS1 respectively. The input channel ADC2 is selected by loading a bit into MUX1. The ADC then operates for one conversion only by loading a bit into the respective registers of ADCSRA generating a value in ADCL. The ADCL is however a 16-bit register and the value to reference an LED needs to be in the range of zero to nine. Therefore, an "AND" operation is used to remove the lower 8-bits of the ADCL, so an 8-bit binary number is produced which corresponds to a random LED. The ADC is then disabled.

2.4 Using Timer (Timer1 Compare Match A Interrupt Vector)

Timer1 is used to create the delay of 1.5s between each LED blink, as well as controlling the sequential blinking of the LEDs. Timer0 and Timer2 are of an 8-bit design while Timer1 is of a 16-bit design therefore Timer1 can achieve a delay as large as 1.5s. To achieve the specific delay CTC mode is selected for the timer. Equation 1 shows the calculation of the TOP value to be stored in OCR1A register which corresponds to the delay time. The prescaler value of 1024 is chosen as only this prescaler value can achieve a large enough clock frequency for the delay.

$$(1)\frac{Clockfreuency}{\mathrm{Pr}escaler} = \frac{16 \cdot 10^6}{1024} = 15625Hz = 4.2s \rightarrow 23437.5 = 1.5s \rightarrow 5B8D(Hexadecimal)$$

Once the delay completes, the ISR for the Timer1 increments the register tracking the LED numbers and the next LED turns on.

2.5 Buttons and RGB LED (INT0 and INT1 Vector)

Buttons are connected to PD2 and PD3 to allow external interrupts INT0 and INT1. The respective ISR for INT0 then begins the game by enabling the other global interrupts. If the game has begun then the button will stop the timer to prevent the LEDs from continuing to blink, another press resumes the LED blinks. The ISR for INT1 restarts the timer which also restarts the LED blinks. If the LED currently lit is the indicator LED, a register tracking the score increments. If the LED currently lit is not the indicator LED, the score decrements. The ISR for INT1 also increments a register tracking the rounds every time it is called. When the round register reaches five a function is called to end the game. This function clears all interrupts, turns off the ADC, turns off Timer1 and outputs 0 to all LEDs. The score is then checked, and a bit is sent to the corresponding pin of the RGB to display a color for the score.

3.ANALYSIS

The main function to make the LEDs blink is used in the ADC ISR to turn on a random LED; however, this interferes with the TImer1 ISR which also utilizes the LED blink function to sequentially blink the LEDs. The buttons were tested separately due to the issue with the LEDs and there was evident interference causing the buttons to not work properly. Since the rising edge is used in the interrupt sense control for both INT0 and INT1 a pull-down resistor is added in between the button and the ground terminal.

4.CONCLUSION

The project met most of the specifications and was a success. The buttons have input interference due to bouncing but they work. The sequential blinking of the LEDs which utilizes the Timer1 interrupt still needs adjustment however the delay was successfully achieved.

REFERENCES

[1]:Arxterra. (Year). 10 ATmega328P Interrupts. [Online]. Available at: https://www.arxterra.com/10-atmega328p-interrupts/ (Accessed: May 8, 2024).

[2] maxEmbedded. (2011). "AVRTimers–TIMER1." [online] maxEmbedded, June 28. Available at: https://maxembedded.wordpress.com/2011/06/28/avr-timers-timer1/ (Accessed: May 8, 2024).

[3] Sambonic. (November 2022). Cyclone. [Online]. Available at: https://github.com/Sambonic/Cyclone (Accessed: May 8, 2024).

[4] Silicon Labs. (n.d.). "EFM32 and EFR32 Platform Application Examples: platform_adc_rng." GitHub. Available at: https://github.com/SiliconLabs/platform_applications/blob/master/platform_adc_rng/README.md (Accessed: May 8, 2024).